COMP 558: Assignment 2
Available: Thursday, February 19th, 2015
Due Date: Thursday, March 12th, 2015

Notes: You must use **MATLAB** package for all of your implementations for this assignment which is currently installed on Unix machines in the CS labs. Be creative (but careful) in presenting your results and in performing the numerical comparisons. We expect everyone to submit original work. You are not allowed to use built-in functions from the Computer Vision toolbox. We will look over the code you produce. Use the mycourses discussion board to post questions.

Question I:

Consider the eight point algorithm we have discussed in class (Section 7.3.5). On Morteza's page (`http://www.cim.mcgill.ca/~morteza/COMP558-Winter2015.php`), there are two pairs of images, which are stereo pairs of two different 3D scenes. Implement Algorithm **EIGHT-POINT** (page 156) to estimate the fundamental matrix F for each of the two stereo pairs. To do so, for each stereo pair, you will have to manually identify n corresponding pixel pairs. Pay attention to the procedure outlined at the end of Section 7.3.5 to avoid numerical instabilities. Explain your working, and examine how "stable" the entries of F are, as the pixel pairs are changed, and as the value of n changes.

Implement algorithm **EPIPOLES-LOCATION** (section 7.3.6, page 157) and compute the locations of the epipoles in each image, for both stereo pairs. Your output for this question should be the two fundamental matrices, the location of the epipole for each image, as well as a discussion of your results. Also hand in your source code for the two algorithms.

Hint: Computing SVD is trivial in Matlab.

Question II:

Now assume that the matrices of "intrinsic" parameters of the two images in each pair are:

$$M_r = M_l = \begin{bmatrix} -f/s_x & 0 & o_x \\ 0 & -f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix},$$

where the pixels sizes values and focal lengths are presented on Morteza's webpage. You can assume that the image center (the principal point) (o_x, o_y) is actually located in the center of the image. Implement Algorithm **TRIANG** (section 7.4.1, page 162) and use it

to recover the depth of several corresponding pixel pairs chosen manually, in each stereo pair. For each stereo pair, choose your corresponding pixel pairs so that they belong to at least 5 different objects, so that you can recover at least 5 different depth levels.

Indicate the resulting depth on the input images, for each pixel pair that you used. Do that in a concise and meaningful way. For example, you can try to use color coding for depth, or some other scheme that to show depth. One idea could be to pick some points that lie on outline of a polygonal face so that the reconstruction could then be showed as a polygon in depth.

Hand in your source code, as well as a discussion of your work and your results. If you find that it is easier to compute depth for some pixel pairs than for others, or for one stereo pair than for the other, you should include a discussion of why this might be the case. However, your discussion should not be limited to just this issue.

Question III:

You will now use the code developed above to reconstruct a 3D scene from the given stereo pairs or from a stereo pair of your choice. Consider a scene with an interesting object (or objects) with surface texture to allow for dense correspondences to be computed. You can obtain a stereo pair by taking two photographs of the scene with a digital camera, where the second photograph is taken from a slightly different vantage point (a small translation and a rotation) than the first. Now carry out the following steps:

1. Using your implementation in Question I, estimate the fundamental matrix for your stereo pair. As before, you can do this by using a certain number of fixed correspondences, determined by hand.

2. Write a simple procedure to compute matching locations between corresponding points in the left-right image pairs. For example, you can do this as follows: for each pixel in the left image obtain its match by selecting the pixel in the right image (searching over a neighborhood of nearby locations) that minimizes the sum of squared differences in intensity within a window. For this choose a window size and a search neighborhood that seems appropriate for your scene. Visualize your correspondence results to show that they are meaningful. Note: for clarity do not show correspondences at every pixel, but rather, for a subset of pixels obtained by subsampling the original left-right image pair.

3. Making reasonable assumptions about the focal length, the pixel spacing and the camera center (you can treat these as inputs to the **TRIANG** algorithm implemented in Question 2) and the dense set of correspondences found above, try to reconstruct the underlying 3D scene. You can use any visualization tools available in Matlab to show the reconstructed (3D) scene points. Discuss your results.