Comp. 4900D: Assignment #2
Due: Tuesday March 1, 2010

1) The goal of the first questions is to implement some code that performs calibration using the method described in the book; by first computing a projection matrix, and then decomposing that matrix to find the extrinsic and intrinsic parameters. On the web site I have given you a program, written in C that uses OpenCV, called assign2_shell.c. This program takes ten 3d points, and projects them using the given camera matrix, rotation matrix and translation vector. Your goal is to write the two routines that are missing, which are `computeprojectionmatrix` and `decomposeprojectionmatrix`. The first routine computes the projection matrix using the method described in Section 6.3.1 of the book, and the second uses the method in Section 6.3.2 to decompose the projection matrix into a camera matrix, rotation matrix and translation vector. It should be the case that the computed camera matrix, rotation matrix and translation vector are the same as the original versions that were used to create the projected points. The program assign2_shell.c is on the web site, and I will also e-mail it to you. You hand in your program source and executable and the resulting output file assign2-out created by running the program. **8 marks**
2) Two kinds of line detection are implemented in OpenCV, the Hough Transform and the probabilistic Hough transform. Assume that there are **n** feature points in an image and that it takes O(**k**) time to increment the accumulator array for each feature point when computing the Hough transform. What is the running time of the Hough transform in big O notation, that is O(…). Justify your answer. **1/2 mark**
3) The probabilistic Hough transform uses random sampling instead of an accumulator array. In this approach the number of random samples **r,** is not specified in the OpenCV call, but is an important hidden parameter. If there are **n** feature points in an image then what is the running time of the probabilistic Hough transform in big O notation. **1/2 mark**
4) If we wished to find an ellipse using the Hough Transform, which of the two approaches is most practical (Standard HT, or probabilistic HT). Justify your answer. **1 mark**

Below is the output from my program, your output should look similar. Your projection matrix should be the same up to a scale factor, and the computed R, T, and K should be the same as the original R, T, and K as is the case below.

Rotation matrix
0.902701 0.051530 0.427171
0.182987 0.852568 -0.489535
-0.389418 0.520070 0.760184

Translation vector
10.000000 15.000000 20.000000

Camera Calibration
-1000.000000 0.000000 0.000000
0.000000 -2000.000000 0.000000
0.000000 0.000000 1.000000

Object point 0 x 0.125100 y 56.358500 z 19.330400
Object point 1 x 80.874100 y 58.500900 z 47.987300
Object point 2 x 35.029100 y 89.596200 z 82.284000
Object point 3 x 74.660500 y 17.410800 z 85.894300
Object point 4 x 71.050100 y 51.353500 z 30.399500
Object point 5 x 1.498500 y 9.140300 z 36.445200
Object point 6 x 14.731300 y 16.589900 z 98.852500
Object point 7 x 44.569200 y 11.908300 z 0.466900
Object point 8 x 0.891100 y 37.788000 z 53.166300
Object point 9 x 57.118400 y 60.176400 z 60.716600

Image point 0 x -332.640564 y -1676.437988
Image point 1 x -1922.371338 y -2027.917847
Image point 2 x -704.610962 y -995.889526
Image point 3 x -1761.512817 y -44.656986
Image point 4 x -2129.829590 y -2700.072021
Image point 5 x -528.037964 y -201.473557
Image point 6 x -677.086243 y 337.669006
Image point 7 x -5553.260254 y -7197.459961
Image point 8 x -444.832092 y -535.692322
Image point 9 x -1204.620728 y -1250.743286

Computed Projection matrix
0.028475 0.001625 0.013475
0.011544 0.053787 -0.030884
0.000012 -0.000016 -0.000024

Computed Rotation matrix
0.902701 0.051530 0.427171
0.182987 0.852568 -0.489535
-0.389418 0.520070 0.760184

Computed Translation vector
9.999999 15.000001 20.000000

Computed Camera Calibration
-1000.000061 0.000000 0.000018
0.000000 -2000.000000 0.000020
0.000000 0.000000 1.000000