LAB 1: CAMERA MATRIX AND CALIBRATION

Task 1 (25) Get Projection Matrix P

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 for moving from 3D world to 2D camera coordinates is:

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

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \cong \begin{pmatrix} u * s \\ v * s \\ s \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

get a matrix that is a **scaled equivalent** of the following:

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

The first task for you is to write the least squares regression to solve for M given the corresponding normalized points. You have to write the code to set up the linear system of equations Ax =0, solve for the unknown entries of M, and reshape it into the estimated projection matrix. Use the homogeneous method solved with SVD.

You have the equations and the Matlab code to solve it in the Lecture slides.

Task 2 (25) Compute reprojection error

To validate that you've found a reasonable projection matrix, write a function which computes the total "residual" between the projected 2d location of each 3d point and the actual location of that point in the 2d image. The residual is just the Euclidean distance (square root of the sum of squared differences in u and v). This should be quite very small.

Check your the solution with the not-normalized points and with the normalized points. What is the error? Why do you think this happens?

Task 3 (25) Get K, R, T and C from P

Once you have an accurate projection matrix M, it is possible to tease it apart into the more familiar and more useful matrix K of intrinsic parameters and matrix [R | T] of extrinsic parameters. Find a function for RQ decomposition (in python you can use scipy.linalg.rq). Then Find T by multiplying the last column of P by the inverse of K. Find C as explained in class.

There is a nice article about camera matrices back and forth, you can have a look.

http://ksimek.github.io/2012/08/22/extrinsic/

Task 4 (25) Visualize camera positions and orientations

Plot the axis and points so you can visualize your results in 3D to confirm that your estimations are reasonable.

Use C to plot a three axis marker representing the camera.

R is an orthonormal matrix, and each of its rows represent an axis of the camera orientation in world coordinates (each column is the axis of the world origin in camera coordinates). Plot a short 3d line in each direction.

You can also plot the 3D points.