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.

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

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 distance (square root of the sum of squared differences in u and v). This should be very small.

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

Task 2 (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.

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

Task 3 (25) Lens Distortion

Now that you are familiar with camera matrices, we will check how to apply lens distortion. In the task 3 directory you have some images of a calibration board. These images (CalibIm1-5.gif) were used for calibrating the camera using the method of [Zhang et al. 2000]

http://research.microsoft.com/en-us/um/people/zhang/Papers/TR98-71.pdf **(also attached)**

The model plane contains a pattern of 8x8 squares, so there are 256 corners. The size of the pattern is 17cm x 17cm. The 2D coordinates (in inches) of these points are available in Model.txt. (We assume the plane is at Z=0.)

The result of the calibration in Calib.txt. You also have the result of un-distorting the images using these parameters for the first two images UndistortIm1-2.gif. Finally, you have the corners detected in the Calibration Images in data1-5.txt

In this task you should take the calibration matrices in Calib.txt and apply them to the points in Model.txt. Draw this points on the top of the Undistorted Images, they should project perfectly. You can also Draw the points on the Calibration Images to see the difference due to the distortion.

Finally, apply the distortion to this points and projected on the Calibration Images. Draw these points on the top of the Calibration images to see that it works.

You can find a more detailed description of the process and the data here.
http://research.microsoft.com/en-us/um/people/zhang/calib/

Task 4 (25) Undistort the images

The final task is correct the distortion of the image. You should produce images similar to UndistortIm1-2.gif for all the images. For efficiency in python, you might want to use numpy.meshgrid to do the warping. For details you can check the section “3.3 Dealing with radial distortion” in the paper.

Extras

You can make your own estimation of the lens distortion parameters using the perfect projection of the Model points and the measured ones in data1-5.txt You should set a system of equations with

$$\begin{bmatrix} (u-u_0)(x^2+y^2) & (u-u_0)(x^2+y^2)^2 \\ (v-v_0)(x^2+y^2) & (v-v_0)(x^2+y^2)^2 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \breve{u}-u \\ \breve{v}-v \end{bmatrix} .$$

the form above and solve it with LeastSquares.

For details you can check the section “3.3 Dealing with radial distortion” in the paper.

Deliverables

Code and images. You will demo it for marking during the next lab.

A good approach is to create a git repository and share it with me. Write some documentation about what you didn’t understand or have doubts, your sources of information, if you got some help, etc. You can do this in a Jupyter notebook, in the code, or in a separate file.