LAB 2: LENS DISTORTION

Task 1 (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 2 (25) Undistort the images

The second task is to 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. Meshgrid is a useful tool to do domain transformations. You create two matrices with the values of the coordinates at each coordinate of the image. Then you can do operations on them, and use them as the indexes for sampling. Note that in the case of an homography, you might try to access coordinates outside the sampled image.

When sampling, the coordinates will be “between pixels”. Use bilinear interpolation to get a better image. https://en.wikipedia.org/wiki/Bilinear_interpolation. Compare with nearest neighbour sampling.
If you have used a loop, this is simple to add. If you used a meshgrid, this would be more tedious, but you can simply use ndimage.map_coordinates. Notice that you have other types of interpolations with higher order polynomia

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

Choose one (or more) of the following.

Task 3 (50) Lense distortion Parameter Estimation

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 the

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

form above and solve it with LeastSquares.

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

Task 3 (50) Camera calibration with OpenCV

Follow this tutorial on camera calibration with OpenCV. Print the chess board and

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_calibration/py_calibration.html

Task 3 (100) Camera calibration with OpenCV

Implement a version of Zhang et al. This involves detecting the corners using openCV (or use the detected points from the data files), then estimate the homographies between the model and the image. You can also use openCV for this, or your code from Computational Photography. Then use this homographies to set the equations in the paper, section 3, solve the equations and recover the parameters for K, R and t for each camera. This should be close to the results given in the dataset, but not identical.

If you are familiar with non-linear optimisation, you can follow the paper to refine the solution and iterate with radial distortion (3.3)