CSE 483 / ECE 483: Mobile Robotics

Assignment 2: Monocular Camera Calibration

General Instructions:

1. Read this document very carefully! We have discussed many complicacies that arise in the process of calibration.
2. The sole purpose of this assignment is to equip the students with implementation skills. It is not enough to know the theory, one needs to implement it to gain a better understanding of the topic.
3. Start well in advance. This assignment, unlike the previous one, may actually take fairly long.
4. In this assignment we also have question/s marked [BONUS], which in addition to helping you out with some extra marks, also provide an opportunity to go one step further in the implementation. We recommend the students to attempt them. However, remember they are bonus and failing to attempt them will NOT affect your assignment marks.
5. **Plagiarism** is strictly prohibited. Any instance of plagiarism will resort to a score of zero over this assignment (irrespective of the scale and nature of plagiarism).

Deliverables:

A main folder with two sub-folders: *code* and *images* (the constituents of these folders are discussed below). Name the main folder with your *roll number*. Zip this folder and upload it.

1. Place all your codes in the *code* folder. It should contain a **main_script.m** file which when run should generate the results, meaning it should print out the K matrix.
2. If you are also attempting the BONUS questions, then your code should also display appropriate plots.
3. Place all the images in the *images* folder. Your code should be reading images from this folder so make sure you use appropriate path.
4. The main folder should also contain a **report** (in pdf form) that briefly describes the implementation. The results and plots must be presented in the report. Note that it should be a very short write-up.

Some warm-up:

Camera calibration is the process of computing a camera's internal properties such as focal length, lens distortion coefficients, etc.. These properties are intrinsic to the camera, which once estimated, do not change. However, different calibration softwares (or different trials) might produce slightly different results; these slight variations are due to noise in the imaging system.

In computer/robotics vision, it is always advisable to work with calibrated cameras as it is a one-shot process and it makes things much easier there onwards. For e.g., consider the case of estimating the rotation (R) and translation (t) between two monocular camera positions. If you do not know K, you will have to estimate F (fundamental matrix) using the point correspondences and it is not easy to decompose F into R and t because F is composed of K, R, and t. On the other hand if you knew K, we can estimate E (essential matrix) and decompose it into R and t as E is only composed of R and t. Well we do have methods which perform automatic calibration on the fly, meaning you don't have to calibrate the camera a priori, these techniques are fairly complex. So, the bottom line is, if you can, always work with a calibrated camera.

Estimate K matrix:

In this **assignment** you will be estimating the K matrix using the widely used Zhang's method(*Zhang. Z, "A flexible new technique for camera calibration", PAMI, 2000*) covered in this course (calibration slide: 53-81). This method relies on planar checkerboards to easily establish 3D-2D correspondences and then solves for the planar homographies to factor out the K matrix.

The estimated K matrix will be of the form shown below:

$$K = \begin{pmatrix} fx_{pixels} & 0 & cx_{pixels} \\ 0 & fy_{pixels} & cy_{pixels} \\ 0 & 0 & 1 \end{pmatrix}$$

The subscript *pixels* means that the estimated quantities are in units of pixels and not in meters (or in any other metric units). For well manufactured (expensive) cameras, fx_pixels and fy_pixels are generally same. Also, cx_pixels and cy_pixels are equal to the image center coordinates. But many a times we do not use such good cameras and as a result we expect the above statements to be invalid.

To complete this assignment you will have to do the following:

1. Extract the images and save them in the *images* folder of your main deliverable folder. Although we have provided 5 images with this assignment, you only require a minimum of 3 checkerboard images. We have provided more images so that if you want, you can use all of them to construct a over determined system of homogeneous equations.
2. Read the images and extract checkerboard corners. You can either select these points manually (MATLAB's `imtool` can be used for this) or use the `detectCheckerboardPoints()` function in MATLAB.
3. If you manually extract the corners then you should note down the correspondences. However, if you choose to use the `detectCheckerboardPoints()` then the corner points are always output in the same order meaning the correspondence is automatically solved.
4. Using the method discussed in class, first compute the three or more homographies, depending upon how many images you use. (Since the checkerboard is a planar scene, the 3 x 4 projection matrix reduces to a 3 x 3 homography matrix (H)).
5. Use the computed homographies to construct a linear system and solve for K using a sequence of SVD and Cholesky decompositions as mentioned in class. (see **Note** section below).
6. [BONUS] Factor out R and t of each checkerboard w.r.t. camera and plot the checkerboard points (or just a plane, of the size of checkerboard, in camera coordinates for each checkerboard). Hint: This is easy, as once you know K, you can just apply inverse of K to H matrices of each checkerboard and what you will be left with is matrices comprising of 2 columns of rotation and a translation vector as the third column. But remember that rotation matrices are 3×3, so how to estimate the last column is left to you.)
7. [BONUS] Think of a suitable way to evaluate how accurate your intrinsics are. (Hint: look at how people usually evaluate intrinsics by means of reprojection error computation. Can something similar be done in this case).

Note: When you compute the B matrix (as explained in class or showed in the slides), it is possible that due to noise your B matrix will not be *Cholesky* decomposition compatible. Which means that the B matrix you estimate is not **Symmetric Positive Definite**. To tackle this problem you can adopt the following procedure:

1. Enforce symmetricity on the B matrix by doing $B' = (B + B^T)/2$ and check for the positive definiteness.
2. If previous step fails, then to enforce positive-definiteness of the B' matrix(i.e. to make eigenvalues positive) perform eigenvalue decomposition. If minimum of the three eigenvalues is negative, goto step 3 else if the minimum is zero goto to step 4.
3. Negative minimum eigenvalue:
   (a) Perform eigenvalue decomposition on the B' matrix and if the minimum eigenvalue is negative(say λ') find λ^2.
   (b) Add a compensation factor $C = \lambda^2 * I$ to B' such that $B' = B' + C$.
   (c) If B' is not positive definite, Goto step 2
4. Zero minimum eigenvalue:
   (a) Add a very small compensation factor δ(say 10^{-12})* I to B' i.e. $B' = B' + \delta * I$ and check again.
   (b) If B' is not positive definite, Goto step 2