CMSC 733, Computer Processing of Pictorial Information
Homework 1: AutoCalib!
Due on: 11:59:59PM on Wednesday, Feb 24 2017

Prof. Yiannis Aloimonos,
Nitin J. Sanket

February 10, 2017

The aim of this project is to estimate the elusive intrinsic parameters of the camera. In particular, we will estimate the focal lengths f_x, f_y, the principal points c_x, c_y and the radial distortion parameters k_1 and k_2. Note that we are neglecting skew s and we are using a two parameter distortion model. This model is not very good for ultra-wide angle lenses. We will be following the pipeline given in Ref. [1]. The next few sections will talk about what parts of code are to be filled by you and the functionality of each part of the code.

1 Data

The Zhang's paper relies on a calibration target to estimate camera intrinsic parameters. The calibration target used can be found in the file `checkerboardPattern.pdf`. This was printed on an A4 paper and the size of each square was 21.5mm. Note that the Y axis has odd number of squares and X axis has even number of squares. It is a general practice to neglect the outer squares (extreme square on each side and in both directions). Thirteen images taken from a Google Pixel XL phone with focus locked are given in the folder `./Code/Imgs`.

2 Initial parameter Estimation

We are trying to get a good initial estimate of the parameters so that we can feed it into the non-linear optimizer. We will define the parameters we are using in the code next.
x denotes the image points, X denotes the world points (points on the checkerboard), k_s denotes the radial distortion parameters, K denotes the camera calibration matrix, R and t represent the rotation matrix and the translation of the camera in the world frame.

2.1 Solving for approximate K or camera intrinsic matrix

Refer to Ref. [1] section 3.1 for a solution of parameters in K. As a side note, the explanation for each part of the code is provided in the comments. The main code is in `test.m`. For this Subsection you need to modify the `EstimateK_linear.m` function. The corners are detected by a built-in MATLAB function which needs the Computer Vision toolbox. You need to run these codes on the Server if you do not have the toolbox. More details about the camera calibrator tool in MATLAB can be found here: https://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html. Note that you are not allowed to use any other functions apart from that provided in the starter code.

2.2 Estimate approximate R and t or camera extrinsics

Refer to Ref. [1] section 3.1 for details on how to estimate R and t. Note that the author mentions a method to convert a normal matrix to a rotation matrix in Appendix C, this can be neglected most of the times. You need to fill in the `EstimateRt_linear.m` function for this part.

2.3 Approximate Distortion k_c

Because we assumed that the camera has minimal distortion we can assume that $k_c = [0, 0]^T$.

2.4 Non-linear Geometric Error Minimization

We have the initial estimates of K, R, t and , k_s, we want to minimize the geometric error defined as given below

$$\sum_{i=1}^{N}\sum_{j=1}^{M} \|x_{i,j} - \hat{x}_{i,j}\left(K, R_i, t_i, X_j, k_s\right)\|$$

Here $x_{i,j}$ and $\hat{x}_{i,j}$ are an inhomogeneous representation. You need to complete two functions for this part, `GeoError.m` and `MinGeoError.m`.

`GeoError.m` function evaluates the geometric error. In the function `[error, f] = GeoError(x, X, ks, K, Rs, ts)`, `error` is the geometric error and `f` is a vectorized form of a $2 \times N \times n$ matrix (here `n` is the number of corners in the checkerboard and `N` is the number of the calibration images) which has $x_{i,j,1} - \hat{x}_{i,j,1}$ or $x_{i,j,2} - \hat{x}_{i,j,2}$ as its elements. `x` is the 2D points in a $n \times 2 \times N$ matrix and `X` is the 3D points in a $n \times 2$ matrix. `ks` is the radial distortion parameter and is a 2×1 matrix and `K` is the 3×3 calibration matrix. `Rs` is a set of rotation matrices of size $3 \times 3 \times N$ and `ts` is a set of translation vectors or size $3 \times 1 \times N$. Note that this function can be called by a MATLAB built-in optimizer, `lsqnonlin()`.

`MinGeoError.m` function minimizes the geometric error by using the Levenberg-Marquardt solver (Amazing solver, a worthy read). You can use the built-in MATLAB optimizer `lsqnonlin()` for this.

Note that, the two parameter distrotion model is described next (you need this for the

`GeoError.m` function, even though we are neglecting this for the initial estimates we need to have this in the error function so that our non-linear optimizer can actually estimate it). Let $[a, b]^T$ be the inhomogeneous representation of $[R, t][X, 1]^T$. Let $r^2 = a^2 + b^2$.
Also, $[u_{ideal}, v_{ideal}]^T$ is an inhomogeneous representation of $K[R, t][X, 1]^T$.
Here $K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$, $[u_{img}, v_{img}]^T$ is the corresponding 2D point x. Now, the correction for the two-parameter distortion model can be written as

$$x_{corr} = x_{ideal} + x_{ideal}(k_1 r^2 + k_2 r^4)$$

More details about this can be found in Section 3.3 of Ref. [1].

2.5 Starter Code

All the starter code is given in the `Code` folder.

2.6 Submission Guidelines

Submit your codes (`.m` files) with the naming convention
`YourDirectoryID_hw1.zip` onto ELMS/Canvas (**Please compress it to .zip and no other format**). Your `DirectoryID` is the username to your UMD e-mail ID. If your email ID is `ABCD@terpmail.umd.edu` or `ABCD@umd.edu`, your `DirectoryID` is `ABCD`. Your zip file should have the following things:

- Folder named `Code` with all your code.
- Typeset a report in LaTeXusing the IEEETran format given to you in `Draft` folder. The output file should be (**pdf and pdf ONLY**). Describe the pipeline with a lot of images, intermediate outputs (include filter visualization), your implementation details and any observations in detail with appropriate references for both sections.
- `Rt.png` image generated by the `Evaluate.m` function.
- Report your reprojection error and post a snapshot of your result in the report. Your error should be around 0.5px.
- A `Readme.txt` file on how to run your code if it is not as simple as running `demo.m`.

If your code does not comply with the above guidelines, you'll be given **ZERO** credit.

3 Allowed Matlab functions

All general MATLAB functions except calibration functions.

4 Collaboration Policy

You are restricted to discuss the ideas with at most two other people. But the code you turn-in should be your own and if you **DO USE** (try not it and it is not permitted) other external codes/codes from other students - do cite them. For other honor code refer to the CMSC733 Spring 2017 website here https://www.cs.umd.edu/class/spring2017/cmsc733/.

Acknowledgements

This fun project was inspired from 'Machine Perception' (CIS 580) course of University of Pennsylvania (https://fling.seas.upenn.edu/~cis580/wiki/index.php?title=Homeworks_Spring_2016).

Thanks to Siddharth Mysore from University of Pennsylvania for help with some of the source codes.

DON'T FORGET TO HAVE FUN AND PLAY AROUND WITH IMAGES!.

References

[1] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.