

CS 223b: Introduction to Computer Vision

Assignment 2: Camera Calibration

Due date: **Monday, January 26th 23:59 PST**

You may work in **teams of up to 3 persons**

Submission via email with subject “Assignment 2: [NAMES]” to cs223b09@gmail.com, where [NAMES] is a comma separated list of the full names of everyone in your group.

1 Understanding Camera Calibration(10 pts)

- When the extrinsics are the same, how will the distance of the pixel coordinate from the optical center change for a given point when you double the focal length?
- What is aspect ratio?
- In 3D space, how many degrees of freedom does a rotation matrix have?
- Given 10 known points for a calibration object with a distortion-free camera, how many images are required to calibrate the camera?
- Where is the origin for a camera’s coordinate system?

2 Calibration Software Using OpenCV(10 pts)

2.1 Software Installation (0 pts)

OpenCV is free, widely used software in the computer vision community, and it will be useful to learn OpenCV for your project. For this assignment, you will start from installing OpenCV.

- Install a C++ compiler. MS Visual Studio 6, for example, is available in the Stanford CS Department Software Library, Gates Room 161.
- Install the Intel OpenCV Library from <http://sourceforge.net/projects/opencvlibrary/>.
- Compile and run the sample project from `OpenCV/samples/c/cvsample.dsp` to test that your machine is set up properly. It should open two windows showing a demo. The OpenCV Wiki (<http://opencv.willowgarage.com/wiki>) might be useful to find out bugs or missing dependancies.

2.2 Extracting Corners(5 pts)

- For calibration, OpenCV implements the [Zhang 99] algorithm. This algorithm inputs correspondences between 2D image points and 3D scene points over a number of images and outputs the intrinsic camera matrix as well as the radial distortion coefficients k_1 and k_2 . It also computes the rotation and translation of the scene towards the camera for every image, but this information is not needed for calibration.

The scene points of the correspondence have to be in the same plane and to obtain good results, this plane should be different for every calibration image.

In practice, an easy way to achieve this is to print a chessboard pattern onto a piece of paper, attach this paper to a rigid surface and take the corners between the chessboard fields¹ as the correspondence points.

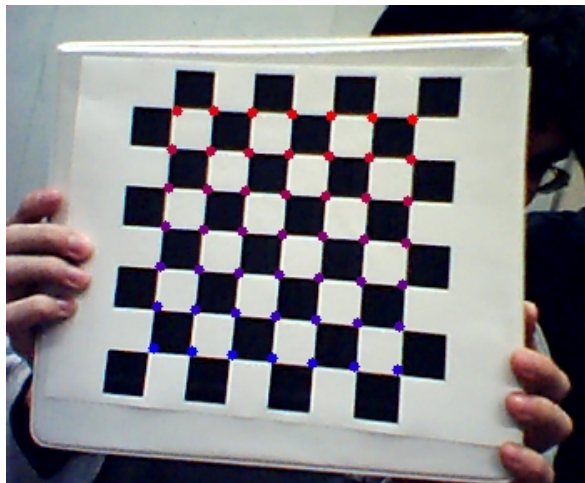
- Luckily, OpenCV has a routine to extract the corner points automatically. Write a program using OpenCV to open a series of images and extract corner points using the images at <http://cs223b.stanford.edu/assign2>.

In OpenCV, you will find the following functions interesting:

- `cvCreateImage` initializes an image data structure.
- `cvNamedWindow` creates a window.
- `cvShowImage` displays an image in a window.
- `cvLoadImage` and `cvSaveImage` load and save image files.
- `cvFindChessBoardCornerGuesses` and `cvFindCornerSubPix` to find chessboard corners and refine these positions automatically.
- `cvDrawChessBoardCorners` to render the detected chessboard corners.

A complete function reference can be found at [OpenCV/docs/ref](http://opencv.org/docs/ref).

- Draw circles around the corner points found for `image00.bmp` ~ `image11.bmp` and submit the images. An example for `image00.bmp` should look like this:



¹that is, use only the corners on the *inside* of the chessboard

2.3 Camera Calibration (5 pts)

- Create the set of correspondence points for each of your images. For the 3D scene coordinates, use (0,0,0) as coordinates for the top left corner and increment 1 (the size of each grid is 1 inch \times 1 inch) for x and y directions. Since we have seven such points for both horizontally and vertically, you will have (m,n,0) where $0 \leq m < 7$, and $0 \leq n < 7$
- Use the function `cvCalibrateCamera` to calibrate your camera using your correspondences. Add the intrinsic camera matrix and the lens distortion parameters to your solution email. Format floats using `printf("%.1f")`.

2.4 Extra Credit: Calibrate Your Own Cell Phone Camera

By acquiring images of a calibration pattern, and going over the same procedure described above, you can calibrate any cameras at hand. What is the calibration parameters for your own cell phone camera?

- Print a chessboard, and capture 10+ images. Be sure that the captured images have enough perspective projection to increase the chance of getting the correct intrinsics.
- Using the same procedure, find the chessboard corners. Specify your cell phone, and submit images with corners found.
- Calibrate your cell phone camera. As before, include the intrinsic camera matrix and the lens distortion parameters in `printf("%.1f")` format.

3 Submitting

When finished, submit any sourcecode and a .doc or .pdf file containing all output images. You should have:

- short answers for Sec. 1.
- C++ code used to find and draw the chessboard corners and calibrate the camera.
- `image00.bmp` \sim `image11.bmp` with a circle overlaid at the location of corners.
- intrinsic camera matrix and the lens distortion parameters of the camera that captured provided images (in inches)

And if you are willing to work on the extra credit question in Sec. 2.4, be sure to include:

- (optional) your cell phone model
- (optional) images of an calibration object captured from your cell phone camera with corners found

- (optional) the intrinsic camera matrix and the lens distortion parameters of your cell phone camera (in inches)

Submission should be via an e-mail to cs223b09@gmail.com with the subject “Assignment 2: [NAMES]”.

References

[Zhang 99] Z. Zhang: *Flexible Camera Calibration by Viewing a Plane from Unknown Orientations*. International Conference on Computer Vision (ICCV'99). Corfu, Greece, September 1999, pp. 666-673.