**CS635: Assignment #0 – Camera Calibration**

**Out:** January 22, 2009  
**Due:** February 3, 2007, 8:59am

**Objective**

The objective of this assignment is for you to calibrate the parameters of a digital camera and then to use the camera to capture at least two pictures and reconstruct a small 3D point cloud. The theory of camera calibration is simple but in practice it is much harder than it seems. Precision is very important: small errors can lead to big calibration errors.

You are to base your work off the Tsai camera model and to use an optimization package to do the calibration work. You may use reasonable initial guesses to start off the optimization but ultimately you have to let the optimizer do the work. The reconstruction of the small 3D point cloud is just to verify your calibration is accurate/useful.

This assignment requires some mental planning work, some lab work, and some programming. The whole assignment is not that much work and mostly requires you to understand the process. I do recommend STARTING EARLY! (i.e. -- you will not be able to complete the assignment if you start the day before its due --).

**Detailed Description**

**Step 0 – Internal Parameters Calibration**

In this assignment, you will calibrate a digital CCD camera. If you have a digital camera of your own, you can use it. If you do not have one, or prefer not to use it, I can lend you a camera for some hours so that you can take the necessary pictures (which at the minimum are just two pictures).

To perform the calibration, you will need a calibration pad. In the lab, there is a large calibration pad. **PLEASE TREAT IT CAREFULLY.** The pad is flat and the checkerboard pattern is regularly spaced at intervals of 4 inches (or 101.6mm). When you take your pictures, make sure the images have some perspective foreshortening (i.e., the camera image plane is not perpendicular to the calibration pad plane).

You will need a minimum of one picture to internally calibrate the camera. In the picture, you will see checkerboard patterns. You can easily compute the 3D position of the corners. Use the mouse to click on the corners; this gives you their 2D projections. Now you have many 3D to 2D correspondences \((x_i, y_i)\) and you can calibrate the camera model as described in class.

The minimum set of internal camera parameters is:

a) Focal length \(f\)
b) Image plane center \((c_x, c_y)\)  
c) Horizontal scale, to compensate for any non-squareness of the pixels \(s_x\)  
d) Radial lens distortion correction factor \(k\)

Note: Depending on your camera \(s_x\) and/or \(k\) might be small.
The optimization will take as input \((f, c_x, c_y, s_x, k)\) set to initial guesses and produce the same values, but optimized as output. The observations \((x_i, y_i)\) and their 3D equivalents are used as data for the optimization.

**NOTE:** To perform this optimization and all those needed for this assignment, I recommend you use Levenberg-Marquardt’s nonlinear least squares method (lmdif). An implementation and sample interface code is available on the course website. The implementation is from the Numerical Recipes in C code. The file lmdif.c is “C” code but the interface sampleopt.C is a “C++” program. You may use other implementations if you prefer.

**Step 2 – External Parameters Calibration**

Once your camera is internally calibrated, you must take two pictures of an object of your choosing and estimate the external camera pose of each picture. You may in fact choose the same images as for internal calibration, but remember in theory the internal parameters are fixed and thus do not vary from picture to picture.

For external parameter calibration, take a picture so that the object is in view and the calibration pad is in view as well. Similar to internal calibration, obtain a corresponded set of 2D-3D feature points. Use these features points to calibrate the external camera parameters, namely the 3D position of the camera’s center of projection and its 3D orientation (total of 6 numbers).

Your optimization will take as input guesses for the pose and return the optimized pose. To verify that the pose estimation is correct, you can “reproject” the points you think you saw using the calibrated camera model and they should perfectly fall on top of the clicked points.

**Step 3 – 3D Point Cloud Reconstruction**

The last step is to reconstruct a few points (at least 20) on the object of your choosing. In this case, you already know the camera’s internal and external parameters. You must choose 20+ corresponded features points from the two camera views. You can then compute the 3D position of the object points using simple ray-ray intersection.

**Optional (extra credit): Step 4 - Subpixel Accuracy Estimation**

Once the above is complete (or simultaneously), a next step is to implement a subpixel accuracy estimation method for corresponded points. If you zoom in closely on the checkerboard pattern, you will see that the corners are not perfectly discrete. Thus, you must pick a local model to use for them and implement a refinement scheme so that your initial guess is “improved” to subpixel accuracy: think of a local model that works well as per what you see in the image and what you think you should see. You may also assume your initial guess is more or less correct, but the result should be it “snaps” to the right place. This optimization function takes as input the correspondence feature point \((x_i^r, y_i^r)\) and outputs a refined correspondence feature point \((x_i, y_i)\).

**Step 4 – Rendering**

In theory, you can perform the entire assignment using just two images from two viewpoints observing an object of your choosing. Assuming this, below are the minimum presentation requirements (if you have more images, no problem, just provide the equivalent functionality)
I recommend you write the program in OpenGL/GLUT/GLUI for effective reuse in next assignments. For this assignment, please demo a program that can show both of your captured images (one a time, or both simultaneously) and via a user interface (e.g., GLUI) can draw on top of the image the following:
- All user clicked 2D feature points
- All subpixel accurate refined 2D feature points (if implemented)
- A reprojection of the points used for external pose estimation
- A reprojection of the 20+ reconstructed object points
- The numeric values of the internal and external camera parameters (using printf’s is fine)

**Grading/Demonstration**

Your demonstration will consist of you showing me your program in short demo session to be arranged (in my office). On or before the due date, please provide me with a CD or a zip file with a single directory called “<your-name>-asgn0” containing:

- Windows PC Executable
- Data files (i.e., images)
- Other necessary files, DLLs, etc…

During your demo session, I will use the provided CD/zip-file to grade your program. Your grade will be influenced by how well your particular camera/object is reconstructed, by the presentation and usability of your program, and by how well you complete the assignment requirements.

In this assignment, you may collaborate only to help with the mechanics of the assignment (e.g., using the pad, your camera, etc). *Everybody must take their own pictures!* Practically speaking, this means that nobody should have the same pictures or calibration results.

**If you have questions, please come see me ASAP – do not wait until the last moment.**

Have fun and good luck!