

CS590G: Assignment #1 – Camera Calibration

Out: September 14, 2004

Due: October 5, 2004

Objective

This objective of this assignment is to calibrate a (digital) camera so as to be able to capture images of objects from known locations and with a known camera model. In the next assignment, the calibrated camera will be used to acquire images of an object and show reconstructed views.

Camera calibration was explained in class. In this assignment, you do not need to implement the calibration software from scratch but rather will use an existing implementation of the Tsai camera model. You will need to gather the calibration data, perform the calibration, and verify the results. 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!** Some of the requirements of this assignment are such to help you with the next assignment and potentially your mini-project.

Detailed Description

In this assignment, you will calibrate a digital CCD camera. I have provided a camera and a calibration pad for you in CS-G56. You may use this setup (camera + calibration pad) or you may use your own digital camera and my calibration pad, or you may use your own camera and your own calibration pad.

There are multiple steps to the assignment. You will probably want to build upon your assignment #0 program. I will explain each in turn. At the end, I provide some sections providing some specific details. There is some flexibility as to how you may do this assignment but everybody will have to satisfy the assignment requirements by giving me a brief individual demo of their system. Extra credit items are also listed below.

Step 0 – Choosing your setup (planning work)

The first step is you must decide which camera, calibration pad, and preliminary object you wish to use. I have provided equipment in CS-G56.

Camera: the default camera is in fact a compact high-resolution digital video camera (TREAT IT CAREFULLY!). You can use standard Firewire applications to use it; in particular a program “Flycap.exe” is installed on the computer. The images will be 1024x768 pixels (at 30 fps) – currently, there is a temporary camera that is at higher-resolution but a slower frame rate (1600x1200 at 14 fps). If you have a digital camera of your own (e.g., for taking pictures), this would provide an easy way to acquire higher-resolution images in the comfort of your own home. Depending on your mini-project, a

video camera or a picture camera might be preferred – you don't have to worry about it at this point.

Calibration pad: the default calibration pad provides you with an easy way of measuring the location of 3D points. Exact details of the calibration pad are in a later section. Basically, the provided pad allows you to manually measure the location of a 3D point relative to an origin and to easily “see” the same 3D point from a captured image. You may construct a more sophisticated or larger pad at home if you like – the key component here is accuracy.

Object: you may choose whatever object you wish to reconstruct. Some objects are harder than others. You will not be reconstructing the object in *this* assignment, so you can change the object later, but you will be calibrating your camera and practicing with image capture. If the object is too detailed, it will be hard to reconstruct. If the object is too simple, it will look too boring. If the object is too small, it will not look good up close (e.g., not enough pixels). If the object is too large, it may occlude the calibration pad and make it hard to calibrate the camera. You'll have to figure out a balance of these factors that suits your goals.

Step 1 – Measuring the 3D world (lab work)

Once your physical setup is complete (e.g., camera, calibration pad, object), you should place the object more or less centered on the calibration pad and take some pictures. You can store the pictures as PPM, JPEG, or whatever format you like. Either before, or after, you will also need to measure the location of a set of 3D points visible from the captured image. Use the calibration pad, the colored points, a measuring device (e.g., a ruler), and assignment #0 to obtain tuples of the form (x, y, z, u, v) where (x,y,z) is the 3D location of a projected image point (u, v) – remember: the more accurate your measurements the better!

You may use as many calibration points as you wish, the more the better. For this assignment, the minimum number of 3D calibration points per image is 11 (eleven) – they should be distributed throughout the calibration pad, covering as much of the field of view as possible.

Step 2 – Calibration (programming work)

In this step, you use the tuples obtained above to calibrate the camera. This calibration will compute a set of internal camera parameters and a set of external camera parameters. As we saw in class, the external parameters determine “where” the camera is and the internal camera parameters fit the Tsai camera model to the actual camera. In theory, the internal parameters will be the same for all images taken by the same camera.

For this assignment, you are required to capture (at least) two images of the object and calibration pad; these images will be called image #0 and #1. You will use the calibration software to perform a full calibration of image #0 (internal and external parameters). For

image #1, you will use the internal camera parameters of image #0, and only perform external camera calibration for image #1. In essence, what you have done is calibrated the camera using one image and then done camera pose estimation for the second image. The computed camera positions and orientations should be “plausible”, meaning their position and orientation seems reasonable – this simple verification will help you with the assignment and with future work.

You can think of this step as creating a single program that given an image, a set of tuples and an optional set of internal camera parameters, computes (or updates) the camera parameters.

Step 3 – Verification (cross-your-fingers work)

To verify the camera calibration, you must draw the *projected calibration points*. This means, starting with the 3D position of the previously measured calibration points, project them onto the image plane using the 3D->2D camera projection (provided by the library). Highlight these projected calibration points by drawing, for instance, a small box (e.g., 5x5 or 10x10 pixels) around them in the image (e.g., use `glRect()`). If all computations are performed well, the rectangle drawn by `glRect()` should appear almost perfectly centered around the actual calibration point on the image – if not verify all your calculations and do again; you might have to use a better or different set of calibration points or images, depending on how well a job you did with the previous set. --The quality of your calibration *does* affect your grade.--

Deliverable (i.e., what do you give me):

You must email me a zip file of the source code, executable(s), and relevant (previously captured) images by the assignment deadline. I will then schedule a very brief “demo” session with each of you and you will show me your system in action. You may either create a single program to do all the functionality or create multiple small programs – the assignment as described above can easily be done in one program. The following is the minimum you must show me:

- a) For each of the captured images, show me a window displaying the image. In the same window, draw small boxes (e.g., using `glRect()`) around the calibration data points and draw boxes in a different color around the **projected** calibration data points. The boxes should be about 5x5 or 10x10 pixels in size.
- b) Allow the user (e.g., me) to view and change any of the camera parameters. This can be done by editing a text file, an interactive prompt, or a GUI. Remember: initial internal camera parameters for image #0 and image #1 should be the same - - the external camera parameters will differ. If the user changes the camera parameters (which I will), the program should recompute the **projected** calibration data points and show them in the window at the new position.

NOTE: (a) and (b) can easily be the same program and based on assignment #0.

EXTRA CREDIT: the trickiest part of this portion of the capture-model-render pipeline is accuracy. Extra credit goes to whatever technique(s) you perform to improve accuracy. For instance, you can use a large number of datapoints and/or you can capture a larger number of images and average/extrapolate a better set of internal camera parameters. Another option is computing the calibration data points with subpixel accuracy. Since the calibration data points do not/cannot project to an infinitesimal point, you can improve accuracy by fitting a circle to each calibration data point and using the center as the pixel coordinate. Others improvements exist as well...

Capturing Images on Escher01 in CS-G56

In CS-G56, a Firewire video camera is attached to the PC Escher01 and placed on the calibration pad next to the computer. Using the installed program “Flycap.exe”, you can see video sequences and extract individual images. For this assignment, you will only need to capture a few images, so just use the application and pause the video when you see a picture you like, then press “Ctrl-Shift-PrtSc” to save the screen to the clipboard. Afterwards, paste the clipboard into Paint (or another image program), crop the image, and save out .BMP or .JPG image. In the next assignment, you will be dealing with the camera grabbing software via some example software (BTW, it will be quite straightforward). For your convenience, I am providing a simple command line program JPEGTOPNM to convert JPEG to PPM. This program takes JPEG from stdin and outputs PPM to stdout. The webpage is:

<http://www.cs.purdue.edu/~aliaga/cs590g/software/jpegtopnm.zip>.

PLEASE BE EXTREMELY CAREFUL WITH THE CAMERA AND CALIBRATION PAD!!! THE CAMERA IS TIED DOWN WITH A CABLE. THE CALIBRATION PAD IS STABLE BUT NOT BULLETPROOF – DO NOT SPILL COFFEE OR COKE ON IT PLEASE!

Please be considerate to other classmates using the equipment – share...

Making measurements using the calibration pad

The calibration pad contains a grid of boxes. All grid lines are spaced by 10 centimeters. The most logical origin to use is the far corner of the calibration pad – thus calculate your measurements using that as the origin and the use the xyz coordinate-frame indicated on the pad. To measure an indicated calibration point, count the number of steps to it from the origin along each axis. More accurate measurements can be obtained by using a ruler. You can use a level and right-angle if you want to be even fancier.

Tsai Software

The Tsai software is located in

<http://www.cs.purdue.edu/~aliaga/cs590g/software/tsai.zip>

It is currently compiled under Win32. To compile, use Cygwin and make a library makefile containing all *.ch files (except for tsai_demo.c) or make a MSVC library project and add all *.Ch files to the project (except for tsai_demo.c).

The file tsai_demo.c is an example top-level calibration program using non-coplanar points (as will be the general case with this assignment). Take a look at it. You should also look at tsai.h – the main interface to the Tsai calibration software. The Tsai library does rather complicated stuff but you really only need to look at tsai_demo.c and tsai.h. That program reads tuples from stdin (e.g., test.cd) and outputs the camera parameters to stdout (note: the Tsai class does support reading/writing to a file). It also projects 3D points to 2D pixels. You could modify tsai_demo and use the program as a system call or you could incorporate the functionality into your program and link with the Tsai code (preferred method, because of flexibility).

Briefly the most relevant Tsai object class methods are:

```
noncoplanar_calibration_with_full_optimization()
    Call this function to calibrate all parameters

noncoplanar_extrinsic_parameter_estimation()
    Call this function to calibrate external parameters

world_coord_to_image_coord()
    Call this function to project from 3D to 2D
```

You will also have to setup the Tsai.cp public variables according to your camera and image resolution – it's pretty straightforward. If you have questions about this, just see me.

The Tsai calibration library works best if the world origin is not near the middle of the field of view and is not near the y-axis of the camera coordinate frame. One way to ensure this is to temporarily offset the calibration data points by 1000 cms (e.g., very far away), calibrate, and then undo the offset.

Furthermore documentation about the Tsai software is found in several doc/*.txt files within the directory.

Collaboration

In this assignment, you may collaborate only to help with the mechanics of the assignment (e.g., using Flycap, using the camera, and to a very minor extent figuring out the API for the Tsai software). *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 – you're on your way to capturing a 3D model!