3D-Image Calibration – *Mathematica* Documentation

Figure 1: Real World Coordinate Axes at the Army-Navy Game

The goal of this series of papers is to analyze images, like the one shown in Figure 1, taken with digital cameras at sporting events and in laboratory settings. We are interested in recognizable features in such images – for example, a football or a particular point on the goal posts at one end of the field. We need to work with two coordinate systems.

- A three-dimensional real world coordinate system. In Figure 1 we use the center of the field as the origin. The x-axis runs along the length of the field midway between the sidelines and the y-axis runs along the 50 yard line. The z-axis represents height above the field. Since football fields are crowned, we use the height around the goal lines and sidelines as the zero point on the z-axis. Thus, the origin is actually a bit below the playing surface at the center of the field. We use yards as units. We use (x, y, z) to denote real world coordinates.
- A two-dimensional image coordinate system. We use pixels as units. The origin is in the center of the image. The x-axis runs horizontally and the y-axis runs vertically. We use (s, t) to denote image coordinates.

The key to analyzing the three-dimensional real world from two-dimensional digital images is the ability to convert back-and-forth between three-dimensional real world coordinates and image coordinates. The key to this conversion is a set of calibration data, or the camera vectors, that describe how each image was taken. Basically there are four things we need to know about how a digital image was taken.

- Where the camera was. More specifically, we need to know the three-dimensional real world coordinates of the optical center of the camera lens.
- The direction in which the camera was pointing.
- A magnification factor based the focal length of the lens, the size of the camera sensor, and the dimensions (in pixels) of the image.
- How the camera was rotated about the line that is perpendicular to the plane of the camera's sensor and runs through the optical center of the lens.

This involves seven real parameters. The first three parameters describe the location of the optical center of the lens. The next two parameters describe the direction in which the camera was pointing. The next parameter is determined by the focal length of the lens, the size of the camera sensor, and the dimensions (in pixels) of the image. The final parameter is determined by how the camera was rotated about the line that is perpendicular to the plane of the camera's sensor and runs through the optical center of the lens. These seven parameters determine four vectors that are used for converting back-and-forth between three-dimensional real world coordinates and two-dimensional pixel coordinates on the image.

- A vector $\vec{q}$ that describes in real world coordinates the location of the optical center of the camera lens.
- A vector $\vec{p}$ whose direction indicates the direction in which the camera was pointing and whose length is determined by the focal length of the lens, the dimensions of the camera's sensor, and the dimensions (in pixels) of the image.
- A unit vector $\vec{u}$ that points in the positive x-direction of the image on the camera sensor. This vector is perpendicular to $\vec{p}$.
- A unit vector $\vec{v}$ that points in the positive y-direction of the image on the camera sensor. This vector is perpendicular to $\vec{p}$ and $\vec{u}$.

We use these vectors to convert back-and-forth between the three-dimensional real world coordinates of a point (feature) and its two-dimensional image coordinates. The translation from real world coordinates to image coordinates is given by

$$\begin{aligned}
\vec{h} &= \text{proj}_{\vec{p}}\,(\vec{a}-\vec{q}) = \left(\frac{1}{||\vec{p}||^2}\right)((\vec{a}-\vec{q})\cdot\vec{p})\vec{p} \\
s &= \left(\frac{||\vec{p}||}{||\vec{h}||}\right)((\vec{a}-\vec{q}-\vec{h})\cdot\vec{u}) \\
t &= \left(\frac{||\vec{p}||}{||\vec{h}||}\right)((\vec{a}-\vec{q}-\vec{h})\cdot\vec{v})
\end{aligned}$$

The mathematics and geometry behind this flurry of formulas is discussed in the paper **The Mathematics of Working with Digital 3D Imagery**.

In order to determine the camera vectors for a particular image we need data – the real world coordinates and the image coordinates (in pixels) of each feature in a set of recognizable features that appear in the digital image. Collecting this data is done using the program **3D-Image-Marker**. The mechanics of collecting this data is described in the documentation for that program. The basic idea is to choose a set of recognizable features whose real-world coordinates are known – for example, in Figure 1 we would use points where the yard lines meet the sidelines. As one example, the point toward the bottom left of this image where the goal line meets the left sideline has three-dimensional real world coordinates $(50, -26.6667, 0)$. (The width of a football field is 160 feet.) The user inputs the three-dimensional real world coordinates for each feature and then clicks on that feature in the image to determine its image coordinates. The data consists of a set of five coordinates for each of n features:

$$\{(x_1, y_1, z_1, s_1, t_1), (x_2, y_2, z_2, s_2, t_2), \ldots (x_n, y_n, z_m s_n, t_n)\}$$

where the first three coordinates are real world coordinates and the last two are image coordinates.

The process of determining the camera vectors from this data is called **calibration**. You should read this documentation with the *Mathematica* notebook **AN76.nb** open on your computer. We will calibrate the image shown in Figure 2. This is identical to Figure 1 except that the original image did not have the axes superimposed.

Cell 1 of the *Mathematica* notebook **AN76.nb** contains calibration data made for this image using the program **3D-Image-Marker**. This cell also specifies the width (in pixels) of the digital image. The user normally pastes data collected using **3D-Image-Marker** into this cell, changes the value of the variable **imageWidth**, and then executes the cell. You can just execute the cell since that data and the value of **imageWidth** are already in the cell. After executing this cell, you should see a plot showing the image coordinates of these data points.

Figure 2: A Photograph Taken at the Army-Navy Game

Cell 2 defines the procedures that convert from real world coordinates to image coordinates based on possible values for the calibration data. You should just execute this cell.

We measure the goodness-of-fit for the possible values for calibration data in the usual way. For each data point $\vec{a}_i = (x_i, y_i, z_i, s_i, t_i), i = 1, 2, \ldots n$ we compute the predicted image coordinates $\bar{s}_i$ and $\bar{t}_i$. Then we find the sum of the squared errors

$$\mathcal{E} = \sum_{i=1}^{n} ((\bar{s}_i - s_i)^2 + (\bar{t}_i - t_i)^2).$$

We eventually calibrate the image by minimizing $\mathcal{E}$.

The first step in this process is finding rough estimates for the camera vectors. This process begins in **Cell 3** of the *Mathematica* notebook **AN76.nb**.

- The vector $\vec{q}$, representing the location of the optical center of the lens is estimated based on intuitive knowledge of where the camera was. This can be a fairly rough

estimate. You can also get a rough estimate of $\vec{q}$ by looking at the image. Normally the user enters this estimate by changing the values of the variables **initialQX**, **initialQY**, and **initialQZ** but this notebook already contains estimates.

- The direction of the vector $\vec{p}$ that is perpendicular to the sensor and passes through the optical center of the lens is estimated based on an estimate of the real world coordinates of the point in the center of the image. These coordinates are called **targetX**, **targetY**, and **targetZ** in the *Mathematica* notebook. Normally the user enters values for **targetX**, **targetY**, and **targetZ**. but this notebook already contains estimates. The length of the vector $\vec{p}$ is estimated based on the focal length of the lens and the width (in pixels) of the image. The width of the image in pixels was already specified in **Cell 1**. In this cell the user normally enters an estimate for the (35mm equivalent) focal length of the lens as the value of the variable **lensFocalLength**. This value is already in this notebook. So all you have to do is execute this cell.

- So far we have estimated six of the seven parameters. The seventh parameter, θ, representing how the camera was rotated about the line that is perpendicular to the plane of the camera's sensor and runs through the optical center of the lens, is more difficult to estimate. **Cell 3** produces a graph of the sum of squared errors, $\mathcal{E}$, as a function of this parameter. The user should note the rough value of the parameter θ that minimizes this function. In our example the plot looks like Figure 3 and we see that the minimum value of the SSE occurs at roughly $\theta \approx 0.2$. The user then uses this value as the value of the variable **initialTheta** in **Cell 4**

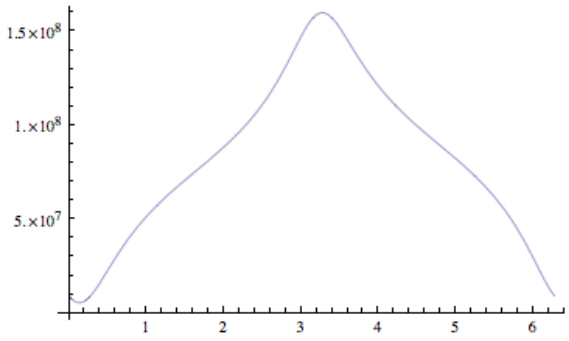

Figure 3: SSE as a Function of θ

- **Cell 4** calculates the values of the corresponding camera vectors and plots the image coordinates of the data points (in blue) and the predicted image coordinates (in red) for the same data points based on the estimated camera vectors. This plot gives you

a rough idea of how good the rough estimates are. Remember we only need rough estimates.

Now we are ready to estimate the camera vectors.

- **Cell 5** uses the *Mathematica* procedure **FindMinimum** to find values of the seven parameters that minimize $\mathcal{E}$. It then computes the corresponding camera vectors and draws a graph showing the image coordinates of the data points (in blue) and the predicted image coordinates (in red) for the same data points based on the new camera vectors. It also prints a table showing the actual image coordinates of the data points and the predicted image coordinates.

- **Cell 6** generates an xml file with the camera vectors for use in the program **3D-Analysis**. You will need to edit the first line of this cell to save the file in a convenient place. This file is not perfect. The user needs to open it in a text editor and do some editing. The first step is to remove all the quote marks by doing a find-and-replace replacing quote marks by the empty string. Then the user needs to replace single quote marks by double quote marks. Be sure to save the edited xml file.

- **Cell 7** generates a text file with the camera vectors for use in the spreadsheet **2D-Analysis**. You will need to edit the first line of this cell to save the file in a convenient place. This file is not perfect. The user needs to open it in a text editor and do some editing. Remove all the quote marks by doing a find-and-replace replacing quote marks by the empty string. Be sure to save the edited txt file.

In summary, here is the user input to this process.

- The calibration data. The user produces this using the program **3D Image-Marker** and cuts-and-pastes it into the first cell of this notebook.

- The width of the image in pixels. The user enters this in the first cell of the notebook as the value of the variable **ImageWidth**.

- Estimates for the coordinates **initialQX**, **initialQY**, and **initialQZ** that describe in real world coordinates the location of the optical center of the lens.

- Estimates for the real world coordinates **targetX**, **targetY**, and **targetZ** of the point at the center of the image. If the camera was pointing generally toward the origin of the real world coordinates then these are all zero.

- The (35mm equivalent) focal length of the lens. This is the value of the variable **lensFocalLength**.

- An estimate for the parameter θ that represents how the camera was rotated about the line that is perpendicular to the plane of the camera's sensor and runs through the optical center of the lens. This estimate is based on a graph of the function $\mathcal{E}$ for $0 \leq \theta \leq 2\pi$. This is the value of the variable **initialTheta**.
- Finally, the user must edit the first line of **Cell 6** or **Cell 7** to save the xml file in a convenient place and edit the xml file using a text editor as described above.