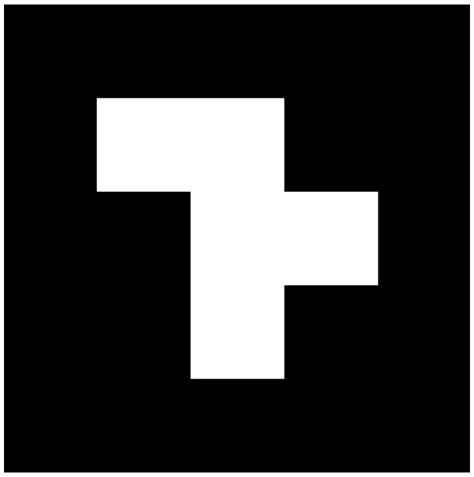
Robust Marker-Based Camera Solving with Automatic Calibration



Joshua Sideris
209649005
Date: April, 2014
Submitted for Course: CSE4471 – Introduction to Virtual Reality

Background

In cinematography, matchmoving (or camera solving) is the process of extracting the position and orientation of the camera. It is commonly used to assist in rendering computer-generated imagery (CGI) into the scene relative to other live-action objects.

Adding correctly-positioned CGI to a live action shot is trivial if the camera is motionless. One of the most widely known applications of using chroma keys for placing CGI is news networks using chroma key software to draw weather maps behind weather reporters. When camera motion is required, it is necessary to track the camera's movement.

Traditionally, matchmoving was (and still is) accomplished by manually selecting points of reference within a scene. Points are tracked by using pattern tracking, which is accomplished by using a sliding window to search for a "pattern area" within a "search area." The search area is a small box that is centered at the last known position of the pattern. The pattern area is a smaller box that slides within the search area for the highest possible correlation between the content of the pattern area in the current frame and the content in the previous frame. [1] Points can be selected manually in most video editing software, but there are also more advanced algorithms that are able to automatically decide on which points to track.

However, matchmoving via pattern tracking has several limitations: errors can accumulate over time as points shift or change shape, some scenes do not have points suitable for tracking, and all matchmoving is done after the film is shot (not in real-time).

More recently, camera tracking has been accomplished in film by the use of motion tracking markers (also known as glyphs for augmented reality or AR glyphs). These glyphs are usually monochromatic symbols (similar to bar codes), which are designed to be easily detected by a camera that uses basic computer vision techniques.

The markers are positioned in known coordinates on the ceiling. An upward-facing camera that is attached to the scene camera observes the markers. From their (the marker's) positions, it is possible to extract the position of the camera. Real-time placement of CGI is possible if one also knows the intrinsic properties of the scene camera (such as zoom, focal length, etc).

Introduction

Camera solving by using AR glyphs requires calibration. Each marker represents a reference to a position on a map, and these positions must be decided and known ahead of time. Therefore, a set must be methodically planned because it is very time consuming (and time is money in film production). For this reason, most filming that requires motion tracking markers is done in an specially-equipped studio.

The goal of this paper is to outline a method for camera solving that uses AR glyphs without the need for manual calibration and without sacrificing precision or accuracy (which reduces the cost of setup, and opens doors to the possibility of economical camera solving outside of a fully-equipped film studio).

[1] Tim Dobbert (2005), *Match Moving: The Invisible Art of Camera Tracking*, Sybex

Method Overview

Our setup consists of a camera facing a surface that is littered with attached AR glyphs. If the position of each glyph is known relative to the scene, it is possible to calculate the position of the camera by using the position and dimensions of any single glyph. Because (in our setup) the positions of the glyphs relative to the world or each other are unknown, it is not possible to map a glyph's position to a worldly one. However, it is possible to actively observe the position of every glyph relative to one another, and to get the position of the camera relative to the glyphs. This simplifies calibration; rather than having to know the position of each glyph relative to the scene, it is only required that a constraint be set to anchor the position and orientation of a glyph relative to the scene.

Whenever two or more glyphs are seen together at the same time, their positions relative to each other are recorded. This builds a network of nodes that allows us to apply the constraint between the anchored glyph and the scene to any other attached glyph.

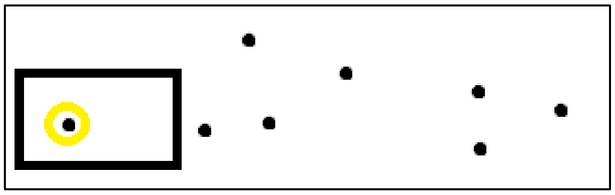


Figure 1 - Initial point observed through glyph camera set as anchor – constrained to the scene.

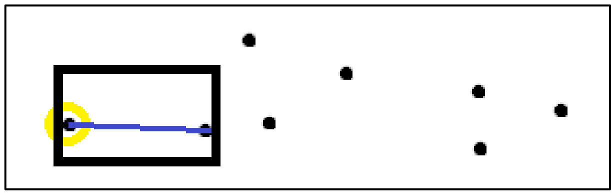


Figure 2 - Active links between two nodes as the glyph camera observes two nodes at the same time.

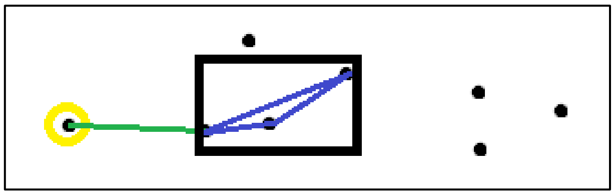


Figure 3 - Several glyphs seen at the same time, but the software remembers where the original constrained glyph is. This means that each new glyph is constrained to the real-world.

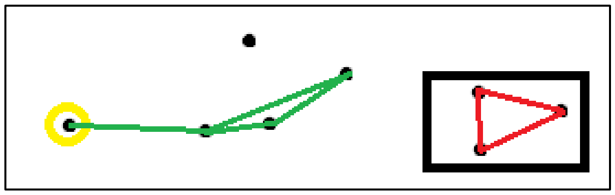


Figure 4 - If, for some reason, the glyph camera suddenly jumps and discovers a new cluster that does not have a path to the constrained node, the cluster is stored, unattached to the main cluster, and is unusable.

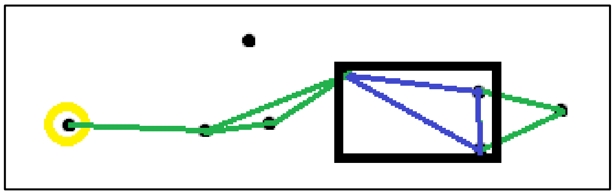


Figure 5 - The unattached cluster is usable if a link can be established to the main cluster. All unattached clusters should be resolved prior to filming.

The glyph configuration should be mapped out before filming starts. The anchor constraint can be set or reset at any time for any known glyph.

AR Glyphs

In order to understand how glyphs are used to generate maps, it is first important to understand what the glyphs are. There are many different types of AR glyphs available, and any number may be suitable for use in this project. In the current implementation, libraries from the open source "Glyph Recognition

Studio" project were used for glyph detection/recognition. Glyph recognition is performed on live video within this library, and frame by frame, potential glyph patterns are compared with glyphs from a pre-defined dictionary of distinct patterns (each representing a unique ID number).

These glyphs are square-shaped and have labeled corners. Patterns are designed so that the glyphs' orientations or identities are not ambiguous (for instance, rotating a glyph upside down or 90° to the left or right cannot result in the pattern being interpreted as being right-side up, or as a glyph with another ID). This functionality is provided by Glyph Recognition Studio, but we were required to define our own patterns.

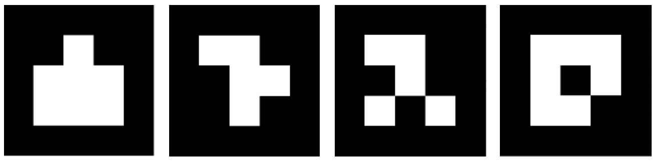


Figure 6 - Examples of AR glyphs.

Glyphs as Coordinate Systems

Glyphs represent their own coordinate systems (with the top-left corner as the origin, and the top and left sides as unit vectors along the axis).

The x and y lengths of both unit vectors (**A** and **B**) relative to the camera are found as $\mathbf{x_a}$, $\mathbf{y_a}$, $\mathbf{x_b}$, and $\mathbf{y_b}$, by finding the difference in position between glyph corners.

The distance in unit vectors between one glyph and the origin of another can be found (**a** and **b**). Thus, each glyph remembers the position of the origin of each other glyph relative to its own dimensions.

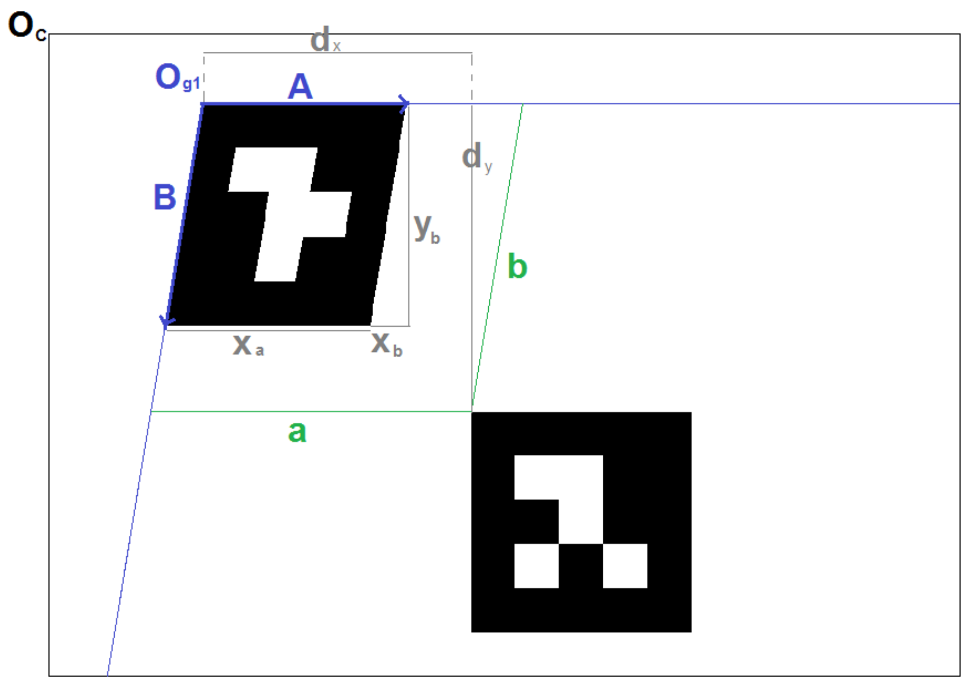


Figure 7 – Two glyphs featured in a camera frame. The origin of the lower glyph is measured in terms of lengths a and b (along unit vectors A and B, respectively).

It can be shown that the length **a** (in units of **A**) and length **b** (in units of **B**) are can be found as follows:

$$b = (x_a * d_y - y_a * d_x) / ((x_a * y_b) - (y_a * x_b))$$

$$a = (d_x - b * xb) / xa$$

Where x_a is the length of the unit vector A along the x axis of the camera frame, y_a is the same (but for the y axis). x_b and y_b are similar (except that they are for the unit vector B instead of A).

d_x is the difference between the origins of both glyphs ($x_2 - x_1$) on the camera coordinate system. d_y is the same for the y axis of the scene.

This technique allows us to create our map of glyphs, which must be in the same plane. From this, it's possible to do camera solving using by any visible glyph.

Camera Solving

Any single glyph can be used to solve for the position of the camera relative to that glyph, and if the current glyph is anchored to the scene, or connected directly or indirectly to another anchored glyph within its cluster, then we can find the position of the camera relative to the scene. The only two additional parameters that need to be configured are the side length of the glyphs (they must all be the same size) in the desired units of measurement, and the angle view of the camera.

Consider one axis of a camera N pixels wide, and suppose that an object with a known length **L** in cm along that axis is observed within a scene. That length might span across l pixels. If the field of view α (as an angle) of the camera is known, then the angle occupied by the object within the field of view is $\alpha' = \alpha \times l / N$, and the camera's distance from the object is $D = L / \sin(\alpha' / 2)$.

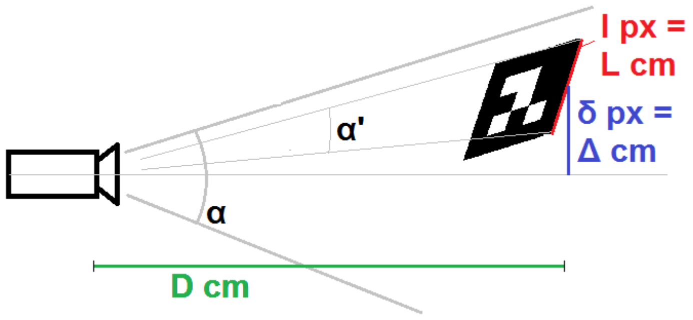


Figure 8 - Illustration of each measured or calculated value.

The camera will also have a horizontal displacement from the measured object. This time, the distance from the object to the camera's center in pixels is known (**δ**), but not the distance in real-world coordinates (**Δ**). Because the offset is measured at the same distance as the object, the pixel-to-distance ratio of the object's length can be used to find **Δ**.

$$\Delta = \delta \times L / l$$

The same procedure must be done in both the x and y axis of the camera. The length of the object used must be parallel to the image plane, which means that the length used might not actually lie along the edge of a glyph; rather, it may be the x or y component of the known distance **L** (as observed by the camera). Therefore, the corrected length of the object must be found.

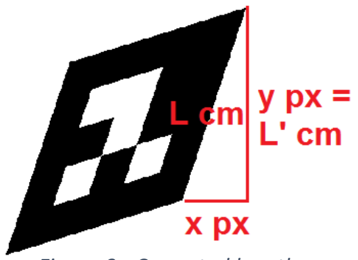


Figure 9 - Corrected length.

$$\mathbf{l}' = \frac{\mathbf{1}}{\sqrt{\mathbf{l}^2 - \boldsymbol{x}^2}}$$

$$\boldsymbol{L}' = \boldsymbol{L} \times \frac{\boldsymbol{l}'}{\boldsymbol{l}}$$

Note that all four corners of the glyph must be on the image plane parallel to the image plane in order for these transformations to work. The ability to calculate the position of the camera when glyphs are moving in and out of depth is outside of the scope of this project (but is a topic for future research).

Precision and Error

The following are sources of error:

- Quantization error.
- Malformed glyphs.
- Glyphs with varying depth from the camera.
- Moving glyphs.
- Camera noise, motion blur, vibrations, etc.
- Error accumulates as the camera moves farther away from the anchored node within the node cluster.

Quantization Error

Quantization error is affected by two parameters: camera resolution and the distance between camera and glyph. For example, at a distance of 4 m with a camera with 1080p resolution and a horizontal angle of view of 42.4°, the distance between pixels is:

$$d = \frac{4m \times 2 \times \sin\left(\frac{42.2°}{2}\right)}{1080} = 0.00527m$$

Assuming that the glyphs are correctly set up, this will be the primary source of error.

Measurement Smoothing

The software is constantly computing distances between pairs of glyphs. Due to quantization error (as well as other errors), the calculated position of the camera is sometimes jagged over time, which is a problem if we intend to use this solution in film production. To address the issue of small inconsistencies between glyphs, a smoothing algorithm was implemented. There are many algorithms that can be used for smoothing measurements. One is the Kalman Filter [2]. For the purposes of this project, a simpler algorithm was used in which the system's belief for a given value is moved towards new measurements by 10% each 60th of a second.

[2] Greg Welch & Gary Bishop (2001). An Introduction to the Kalman Filter, University of North Carolina at Chapel Hill