Viewpoint Calibration With Respect To A Tracker

Kok-Lim Low

Department of Computer Science
University of North Carolina at Chapel Hill
Email: lowk@cs.unc.edu

March 18, 2002

1 INTRODUCTION

In many *virtual reality* (VR) and *augmented reality* (AR) applications, view-dependent images of the virtual worlds are often displayed for the users. These images can be displayed on computer monitors, head-mounted displays, or projected by projectors onto some surfaces. In order to produce *perspective-correct* images as viewed by a user, the positions of the user's eyes must be known when generating the images. A way to get the positions of the eyes (viewpoints) is to use a tracking device.

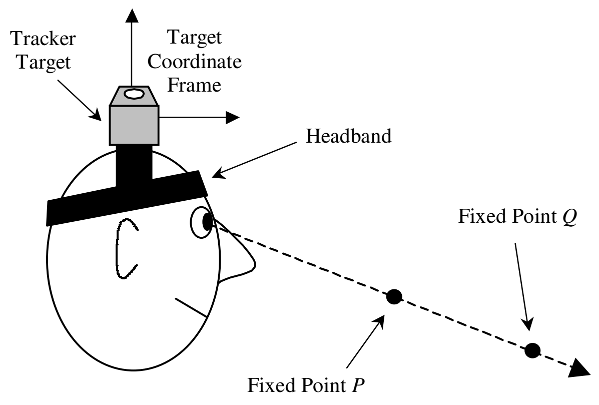

***Figure 1**: A tracker target is rigidly mounted on a user's head. Our objective is to find the positions of the user's eyes with respect to the target coordinate frame. P and Q are two fixed points in the physical space and their positions with respect to the tracker coordinate frame are known. This diagram is not drawn to scale.*

A tracking system typically consists of a *tracker* and one or more *tracker targets*. The tracker is installed in a fixed location in the physical space, and defines a *tracker coordinate frame*. For the purpose of measuring the positions of the eyes, a tracker target is usually rigidly attached to the user's head (see Figure 1). When the tracker target moves together with the user's head, the tracking system constantly keeps track of the target's *pose* (position and orientation)[†] with respect to the tracker coordinate frame. Each pose tells how the target's local coordinate frame (*target coordinate frame*) is positioned relative to the tracker coordinate frame. Note that the eyes' positions are always constant with respect to the target coordinate frame, therefore, to determine the eyes' positions in the tracker coordinate frame, we can first determine the constant positions of the eyes with respect to the target coordinate frame. Then, we can always use the target's current pose in the tracker coordinate frame to express the eyes' positions with respect to the tracker coordinate frame.

This article explains a method to calibrate the viewpoints with respect to a tracker—to find the eyes' positions in the target coordinate frame.

2 VIEWPOINT CALIBRATION

The calibration of each viewpoint consists of two steps. The first step involves some physical procedures to collect the necessary data from the tracking system. These data represent at least two lines in the target coordinate frame, and these lines are supposed to intersect at the viewpoint. They are then fed to the second step to compute an estimate of the viewpoint's position.

Since the calibration steps are exactly the same for both eyes, from here onwards, we will describe the calibration of only the *right* eye.

2.1 Data Collection

We want to collect data of at least two lines in the target coordinate frame that are supposed to intersect at the right eye. We begin by physically marking two points in the space within the tracking range of the tracker. Let these two points be *P* and *Q* (see Figure 1), and they are preferably more than 6 feet apart. One of the two points, say *P*, should be at about eye level, and there should be a clear line of sight from *P* to *Q*.

Next, we measure the positions of *P* and *Q* with respect to the tracker coordinate frame. This usually can be done by using a pointing device that is attached with a tracker target. Let $\mathbf{p}_T$ and $\mathbf{q}_T$ be the positions of *P* and *Q* with respect to the tracker coordinate frame.

Next, we attach a tracker target firmly to the user's head. The user then positions his right eye near point *P* (preferably within a foot) and tries to line-up points *P* and *Q*, as shown in Figure 1. When *P* and *Q* are lined-up, the pose of the target with respect to the tracker coordinate frame is recorded. At this very moment, *P* and *Q* actually form a line that passes through the right viewpoint. Since we already know $\mathbf{p}_T$ and $\mathbf{q}_T$ (the positions of *P* and *Q* with respect to the tracker coordinate frame), and we also know the pose of the target at that moment, we can now express the positions of *P* and *Q* in the target coordinate frame, as $\mathbf{p}_1$ and $\mathbf{q}_1$, respectively. More specifically, we use the target's pose to transform $\mathbf{p}_T$ and $\mathbf{q}_T$ to get $\mathbf{p}_1$ and $\mathbf{q}_1$, respectively. Even though, a moment later, the user might move his right eye away from the line formed by *P* and *Q*, $\mathbf{p}_1$ and $\mathbf{q}_1$ still remain the same and still form a line that passes through the right eye in the target coordinate frame. You can imagine that the line passing through $\mathbf{p}_1$ and $\mathbf{q}_1$ has become rigidly "attached" to the target coordinate

[†] A 6-DOF (degrees of freedom) tracking system provides both position and orientation of the target, while a 3-DOF tracking system usually provides only the position.

frame, and will always pass through the right eye no matter how the user moves. Up to here, we say we have captured the first *line of sight* passing through $\mathbf{p}_1$ and $\mathbf{q}_1$.

In order to find the position of the right eye in the target coordinate frame, we need to capture at least another line of sight, so that its intersection with the first can be used to determine the viewpoint's position. To do this, the user is asked to line-up P and Q again, but with a different head orientation from the first time. This requires the user to roll his right eyeball to another direction[‡]. The new pose of the tracker target at that moment is recorded and used to transformed $\mathbf{p}_T$ and $\mathbf{q}_T$ into the target coordinate frame, as $\mathbf{p}_2$ and $\mathbf{q}_2$, respectively. $\mathbf{p}_2$ and $\mathbf{q}_2$ represent the second line of sight.

We can repeat the above procedure to capture additional lines of sight. In practice, because of measurement errors, these captured lines of sight might not pass through the viewpoint exactly, and they might not intersect one another at all. Additional lines help to improve the accuracy of the estimate of the viewpoint's position.

2.2 Solution Computation

In this section, we will look at two ways to estimate the viewpoint's position in the target coordinate frame. The first way considers only two lines of sight, and the second way can accommodate the more general case of two or more lines of sight.

2.2.1 Two Lines of Sight

When only two lines of sight are captured, the basic idea to find the viewpoint's position is to solve for the intersection of the two lines. However, measurement errors can produce two lines of sight that do not intersect with each other at all. In this case, we can look for the points on the lines of sight at which the two lines are closest. Two lines of sight, l_1 and l_2, are shown in Figure 2. The two lines are closest to each other at points $\mathbf{r}_1$ and $\mathbf{r}_2$ on l_1 and l_2, respectively. We will use the midpoint, $\mathbf{m}$, between $\mathbf{r}_1$ and $\mathbf{r}_2$ as an estimate for the viewpoint position.

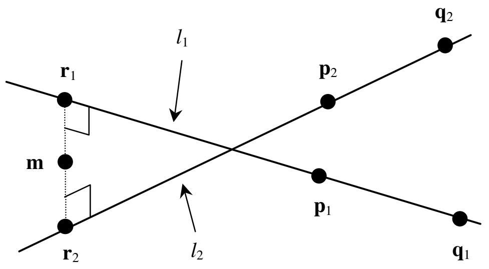

***Figure 2**: Estimating the intersection of two lines of sight. The two lines of sight do not actually intersect, so we find the points on the lines at which the two lines are closest, and use the midpoint between these two points as an estimate of the viewpoint's position. This diagram is not drawn to scale.*

To find $\mathbf{m}$, we have to first find $\mathbf{r}_1$ and $\mathbf{r}_2$. We start by letting $\mathbf{u}_1$ be the vector from $\mathbf{q}_1$ to $\mathbf{p}_1$, and $\mathbf{u}_2$ be the vector from $\mathbf{q}_2$ to $\mathbf{p}_2$:

$$\begin{aligned}\mathbf{u}_1 &= \mathbf{p}_1 - \mathbf{q}_1\\ \mathbf{u}_2 &= \mathbf{p}_2 - \mathbf{q}_2\end{aligned} \tag{1}$$

Let t_1 and t_2 be two scalars. We can express $\mathbf{r}_1$ and $\mathbf{r}_2$ as follows:

$$\begin{aligned}\mathbf{r}_1 &= \mathbf{q}_1 + t_1\mathbf{u}_1\\ \mathbf{r}_2 &= \mathbf{q}_2 + t_2\mathbf{u}_2\end{aligned} \tag{2}$$

Our objective is to solve for t_1 and t_2, so that we can compute $\mathbf{r}_1$ and $\mathbf{r}_2$. Observing that the line passing through $\mathbf{r}_1$ and $\mathbf{r}_2$ is perpendicular to both l_1 and l_2, we can construct the following two constraints:

$$\begin{aligned}(\mathbf{r}_2 - \mathbf{r}_1) \bullet \mathbf{u}_1 &= 0\\ (\mathbf{r}_2 - \mathbf{r}_1) \bullet \mathbf{u}_2 &= 0\end{aligned} \tag{3}$$

where $\bullet$ is the dot product operator.

By substituting (2) into (3), we get the following equations:

$$\begin{aligned}t_1(\mathbf{u}_1 \bullet \mathbf{u}_1) - t_2(\mathbf{u}_1 \bullet \mathbf{u}_2) &= \mathbf{q}_2 \bullet \mathbf{u}_1 - \mathbf{q}_1 \bullet \mathbf{u}_1\\ t_1(\mathbf{u}_1 \bullet \mathbf{u}_2) - t_2(\mathbf{u}_2 \bullet \mathbf{u}_2) &= \mathbf{q}_2 \bullet \mathbf{u}_2 - \mathbf{q}_1 \bullet \mathbf{u}_2\end{aligned} \tag{4}$$

After solving the simultaneous linear equations in (4) for t_1 and t_2, we can compute $\mathbf{r}_1$ and $\mathbf{r}_2$ in (2). The midpoint between them is just

$$\mathbf{m} = \tfrac{1}{2}(\mathbf{r}_1 + \mathbf{r}_2)\,. \tag{5}$$

Therefore, the position $\mathbf{m}$ is the estimate of the right viewpoint's position in the target coordinate frame.

2.2.2 n Lines of Sight

In this case, we will compute the point that has the shortest total distance to all the n lines of sight. First, let l_i be the ith line of sight, passing through the points $\mathbf{p}_i$ and $\mathbf{q}_i$, where $1 \le i \le n$. We further let $\mathbf{p}_i = (p_{ix}, p_{iy}, p_{iz})^T$, $\mathbf{q}_i = (q_{ix}, q_{iy}, q_{iz})^T$, and $\mathbf{u}_i = (u_{ix}, u_{iy}, u_{iz})^T = \mathbf{p}_i - \mathbf{q}_i$. Let $\mathbf{m} = (m_x, m_y, m_z)^T$ be the point that has the shortest total distance to all the n lines of sight.

Suppose all the lines of sight intersect *exactly* at the common point $\mathbf{m}$, then the following is true for all $1 \le i \le n$:

$$\mathbf{m} + t_i\mathbf{u}_i = \mathbf{q}_i \tag{6}$$

where each t_i is some scalar whose value is yet to be determined.

By combining (6) for all $1 \le i \le n$, we can write them in the form of $\mathbf{Ax} = \mathbf{b}$ as

$$\begin{pmatrix}1&0&0&u_{1x}&0&\cdots&0\\0&1&0&u_{1y}&0&\cdots&0\\0&0&1&u_{1z}&0&\cdots&0\\1&0&0&0&u_{2x}&\cdots&0\\0&1&0&0&u_{2y}&\cdots&0\\0&0&1&0&u_{2z}&\cdots&0\\\vdots&\vdots&\vdots&\vdots&\vdots&\ddots&\vdots\\1&0&0&0&0&\cdots&u_{nx}\\0&1&0&0&0&\cdots&u_{ny}\\0&0&1&0&0&\cdots&u_{nz}\end{pmatrix}\begin{pmatrix}m_x\\m_y\\m_z\\t_1\\t_2\\\vdots\\t_n\end{pmatrix}=\begin{pmatrix}q_{1x}\\q_{1y}\\q_{1z}\\q_{2x}\\q_{2y}\\q_{2z}\\\vdots\\q_{nx}\\q_{ny}\\q_{nz}\end{pmatrix} \tag{7}$$

[‡] We are assuming that the position of the *center of projection* of the eye does not change significantly when the eyeball rolls.

where **A** is the $3n \times (n + 3)$ matrix, **x** is the $(n + 3) \times 1$ column vector, and **b** is the $3n \times 1$ column vector.

In practice, because of errors in the measurements, $\mathbf{Ax} = \mathbf{b}$ is almost always an inconsistent system, i.e. **b** is not in the *range* of **A**. Generally, the columns of **A** are independent, therefore **A** has a *rank* of $n + 3$. So, the *least squares solution* of $\mathbf{Ax} = \mathbf{b}$ is just

$$\bar{\mathbf{x}} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}\,.$$

$\mathbf{A}^{\mathrm{T}}\mathbf{A}$ is invertible because it is a $(n + 3) \times (n + 3)$ matrix and it has the same rank as **A**. For more information about linear least squares solution, see [1].

The last step of the calibration is just to extract m_x, m_y, and m_z from $\bar{\mathbf{x}}$. $\mathbf{m} = (m_x, m_y, m_z)^{\mathrm{T}}$ is the estimate of the right viewpoint's position with respect to the target coordinate frame.

Using MATLAB [2], the least squares solution $\bar{\mathbf{x}}$ can be computed as follows:

```
x = A \ b
```

provided `A` and `b` have already been set up as in (7).

REFERENCES

[1] Gilbert Strang. *Linear Algebra and Its Applications, Third Edition* (1988). International Thomson Publishing.

[2] MATLAB. See http://www.mathworks.com/