*Space resection by collinearity*
*Mathematics behind the optical ceiling head-tracker*

Ronald Azuma Mark Ward
University of North Carolina at Chapel Hill
November 1991

## 0.0 Abstract

At SIGGRAPH '91, UNC Chapel Hill demonstrated an electro-optical tracking system for Head-Mounted Displays that can track a user inside a room-sized volume. The mathematics that the system uses to compute the position and orientation of the user's head is based on a photogrammetric technique called *space resection by collinearity*. This paper gives a detailed description of this technique and its behavior in our working system.

## 1.0 Introduction

### 1.1 The optical ceiling head-tracker

UNC Chapel Hill is actively conducting research in Head-Mounted Displays (HMDs). An HMD consists of a pair of video displays mounted in front of the user's eyes, a tracking system, and an image generation system. The tracker senses the position and orientation of the user's head and relays those measurements to the image generation system, which in turn updates the images on the video displays. As the user moves and tilts his head, the HMD changes the images based on his head position and orientation, generating the illusion of being inside a "virtual world."

At SIGGRAPH '91, UNC demonstrated an electro-optical system that can track a user's head inside a room-sized volume [Ward92]. It consists of a head-mounted unit, a 10' by 12' ceiling, and supporting hardware and software. The head-mounted unit has four camera-like devices, which we will call *photodiode units*, aimed toward the ceiling. Each photodiode unit contains a flat 10x10 $mm^2$ detector that is sensitive to infrared light. Implanted in the ceiling are 960 infrared Light Emitting Diodes (LEDs). Based upon the known locations of the LEDs, the projected images of the LEDs on the photodiode detectors, and the fixed geometry of the photodiode units on the head-mounted unit, we can compute the position and orientation of the user's head.

*Figure 1: Conceptual drawing of the electro-optical tracking system*

*Figure 2: Head-mounted frame with four photodiode units, surrounded by supporting hardware*

*Figure 3: User wearing HMD under ceiling*

### 1.2 Space resection by collinearity

The problem of a camera viewing a set of beacons, where the beacon locations are known but the camera position is not, falls in the domain of photogrammetry. It should then be no surprise that we apply results from this field to our particular problem. In particular, we use a method called *space resection by collinearity*. We must stress that collinearity is **not** a new technique; it appears in introductory photogrammetry textbooks, such as [Wolf83]. What we present here is our adaptation of this technique to our problem, in enough detail to enable the reader to write code that implements this routine.

A quick overview of the remainder of the paper:



## 2.0 Description

### 2.1 Overview

How can we recover the position and orientation of the user's head from observations of beacon positions? We start in Section 2.2 by first defining the coordinate systems and relationships we need. Then in Section 2.3 we describe the known geometrical relationships. For example, we know:

- the 3D locations of all the LED beacons in the ceiling
- the images that the photodiode detectors see
- the relative position and orientation of each photodiode unit with respect to the others (because the photodiode units are fixed in position on the head-mounted frame)

With these known parameters and some basic geometry, we form the collinearity condition equations: a set of equations that must be true, given the true values of all the parameters. We know all the parameters except the true position and orientation of the user's head (which we are trying to find) and several scale factors (which we do not care about). By doing some algebra in Section 2.4, we produce another set of equations that removes the scale factors from the system. Then in Section 2.5 we apply Taylor's theorem to generate a linear approximation of our system. We can use these linearized equations to generate an iterative solution to our problem.

The Taylor expansion requires that we take partial derivatives of all of the expressions in the equations in 2.5; some of these derivatives are fairly messy, so we save them for Section 2.6.

### 2.2 Definitions

This section defines the coordinate systems, points, vectors, and transformations that we will need to express the geometric relationships in our system.

We have three classes of coordinate systems: one World space, one Head space, and a Photodiode space for each photodiode unit on the user's head. All are right-handed coordinate systems.

World space is the global coordinate system, aligned with the ceiling, and centered at one corner.

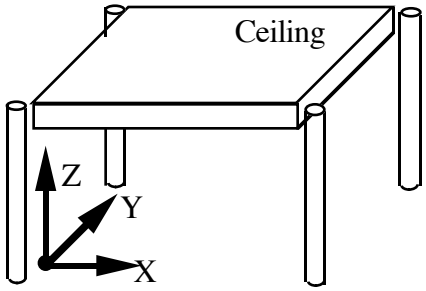

*Figure 4: World space*

Head space is a local coordinate system for the cluster of photodiode units mounted on the user's head.

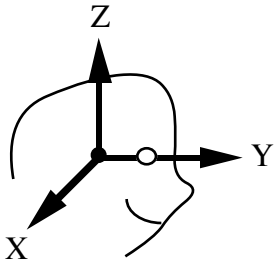

*Figure 5: Head space*

Photodiode space is a local coordinate system aligned with a particular photodiode unit. Each photodiode unit consists of a frame, a lens, a square surface that detects light, and supporting electronics for the detector. Each photodiode unit has its own associated Photodiode space, with its origin at the center of the detector.

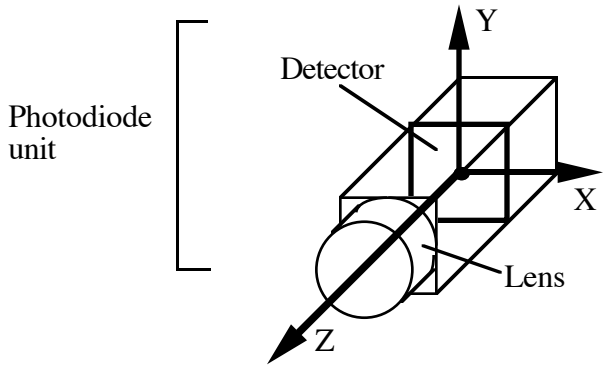

*Figure 6: Photodiode space*

All three types of coordinate spaces are shown in the following overall view:

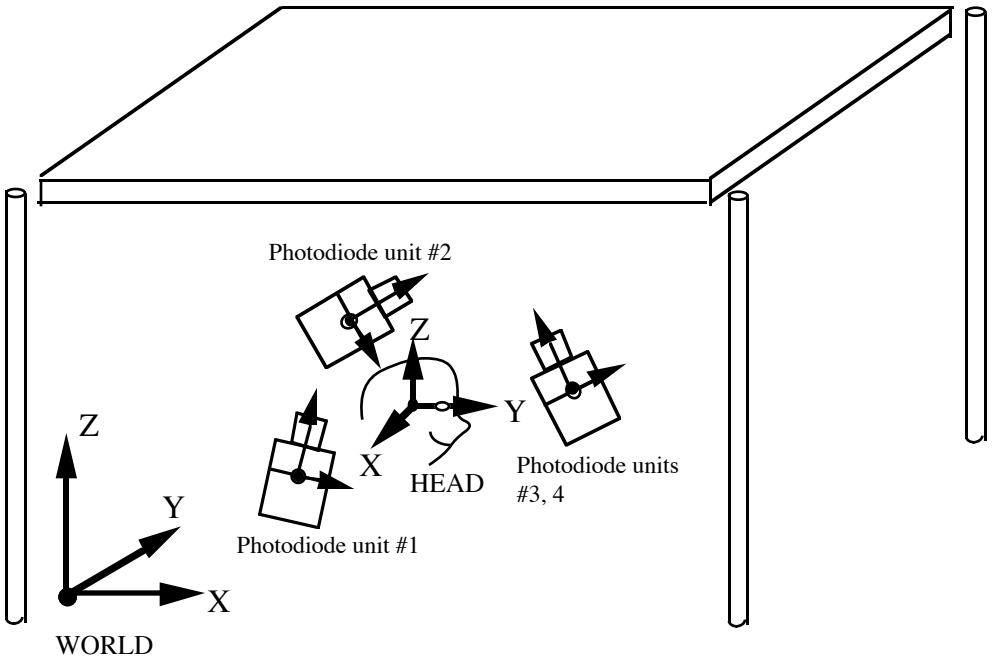


*Figure 7: Overall view of all three coordinate spaces*

We need to establish relationships among these spaces. We can express points and vectors in all three classes of coordinate systems, and we change representations from one to another by performing a rotation, followed by a translation. (A scale factor is not required because all spaces use the same measurement units.) Rotation is performed by multiplying the coordinates of one representation by a 3x3 matrix (see Section 6.1). We define the following rotation matrices:

$\mathbf{M}_i$ = 3x3 matrix that rotates Photodiode space #i coordinates to Head space coordinates
$\mathbf{M}$ = 3x3 matrix that rotates Head space coordinates to World space coordinates

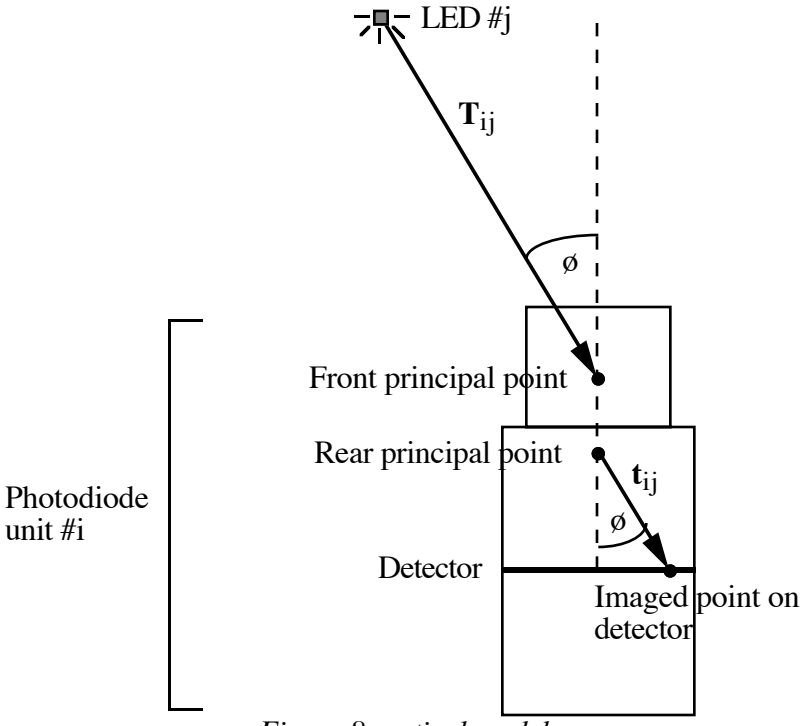


*Figure 8: optical model*

We use a simple optical model for our photodiode unit (see Figure 8). Light from an LED enters at a *front principal point* and leaves at a *rear principal point*, creating an *imaged point* on the

photodiode. The angle of entry to the front principal point is the same as the angle of exit from the rear principal point.

Now we can now define the vectors and points of interest, segregated by what space they are in:

*In Photodiode space:*

$[x_{ij}, y_{ij}, 0]$ = coordinates of an imaged point on detector for photodiode #i and LED #j

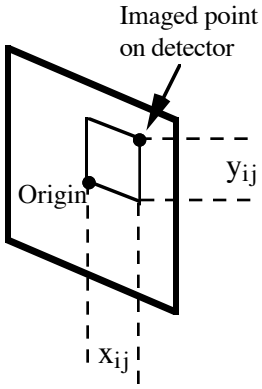


*Figure 9: imaged point on photodiode detector*

*In Head space:*

$\mathbf{t}_{ij}$ = vector from the rear principal point to the $[x_{ij}, y_{ij}, 0]$ imaged point for photodiode unit #i and LED #j (see Figure 8)

$\mathbf{d}_i$ = vector from the origin of the Head coordinate system to the center of the detector of photodiode unit #i

$\mathbf{e}_i$ = vector from the origin of the Head coordinate system to the rear principal point of photodiode unit #i

$\mathbf{f}_i$ = vector from the origin of the Head coordinate system to the front principal point of photodiode unit #i

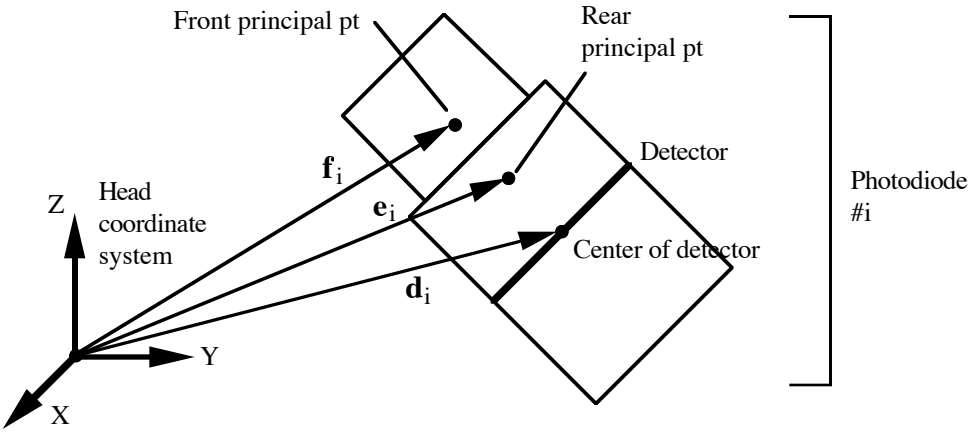


*Figure 10: vectors d, e, f*

*In World space:*

$X_0, Y_0, Z_0$ = location of the origin of the Head coordinate system

$X_j, Y_j, Z_j$ = location of LED #j

$\mathbf{T}_{ij}$ = vector from LED #j to the front principal point of photodiode unit #i

Note that knowing $\mathbf{M}$, $X_0$, $Y_0$, and $Z_0$ is equivalent to knowing the position and orientation of the user's head. Thus, these are the variables that we will eventually solve for.

## 2.3 Geometry

Now that we have defined the various vectors and points that we need, let's see what kind of geometrical relationships we can establish among them.

Collinearity, reduced to its essense, is nothing more than similar triangles. Look at Figure 8 again. Collinearity expresses the observation that the vector from the LED to the front principal point ($\mathbf{T}_{ij}$) and the vector from the rear principal point to the imaged point on the photodiode's surface ($\mathbf{t}_{ij}$) differ only by a scale factor. That is, if the two vectors were placed at the same start point, they would be collinear. In equations:

$$\exists\, \lambda \in \text{Reals}: \quad \mathbf{T}_{ij} = \lambda\, \mathbf{M}\, \mathbf{t}_{ij} \tag{1}$$

where $\lambda$ is the scale factor
and $\mathbf{M}$ transforms $\mathbf{t}_{ij}$ from Head space coordinates to World space coordinates

Now we expand this basic relationship by generating expressions for $\mathbf{T}_{ij}$ and $\mathbf{t}_{ij}$ in terms of the other vectors already defined, then substituting those back into equation (1).

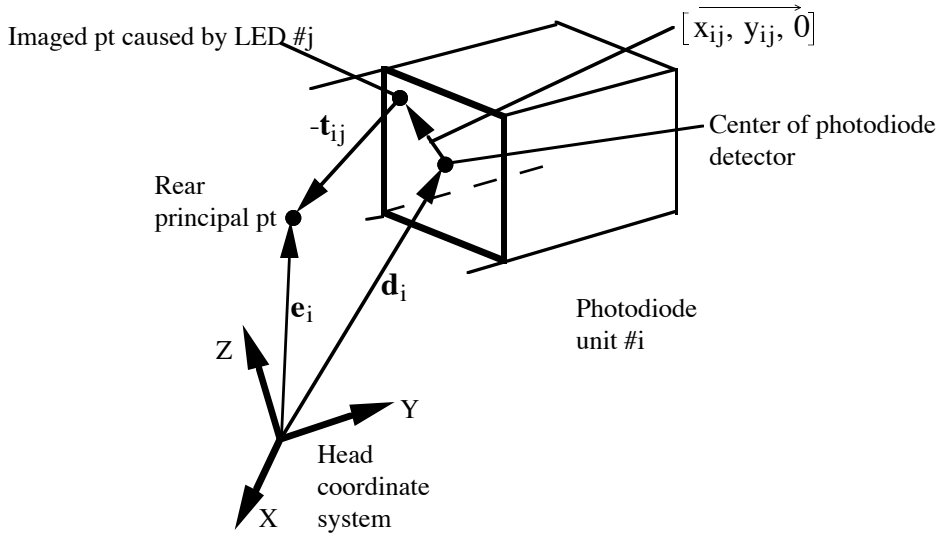


*Figure 11: expressing $\mathbf{t}_{ij}$ in terms of other vectors*

First, we write an expression for $\mathbf{t}_{ij}$ (see Figure 11). Starting at the origin of the Head coordinate system, we can reach the rear principal point by two paths. First, we can follow $\mathbf{d}_i$ to the center of the detector in photodiode unit #i. From the center we move to the imaged point projected onto the photodiode by LED #j, then take $-\mathbf{t}_{ij}$ to the rear principal point of the lens. But we can also reach the rear principal point by starting at the origin of the Head coordinate system and following vector $\mathbf{e}_i$. Expressing this algebraically we get:

$$\mathbf{d}_i + \mathbf{M}_i \begin{bmatrix} x_{ij} \\ y_{ij} \\ 0 \end{bmatrix} - \mathbf{t}_{ij} = \mathbf{e}_i$$

The matrix $\mathbf{M}_i$ is applied to $[x_{ij}, y_{ij}, 0]$ so that all vectors are expressed in Head space coordinates. Rewriting this expression yields:

$$\mathbf{t}_{ij} = \mathbf{d}_i - \mathbf{e}_i + \mathbf{M}_i \begin{bmatrix} x_{ij} \\ y_{ij} \\ 0 \end{bmatrix} \tag{2}$$

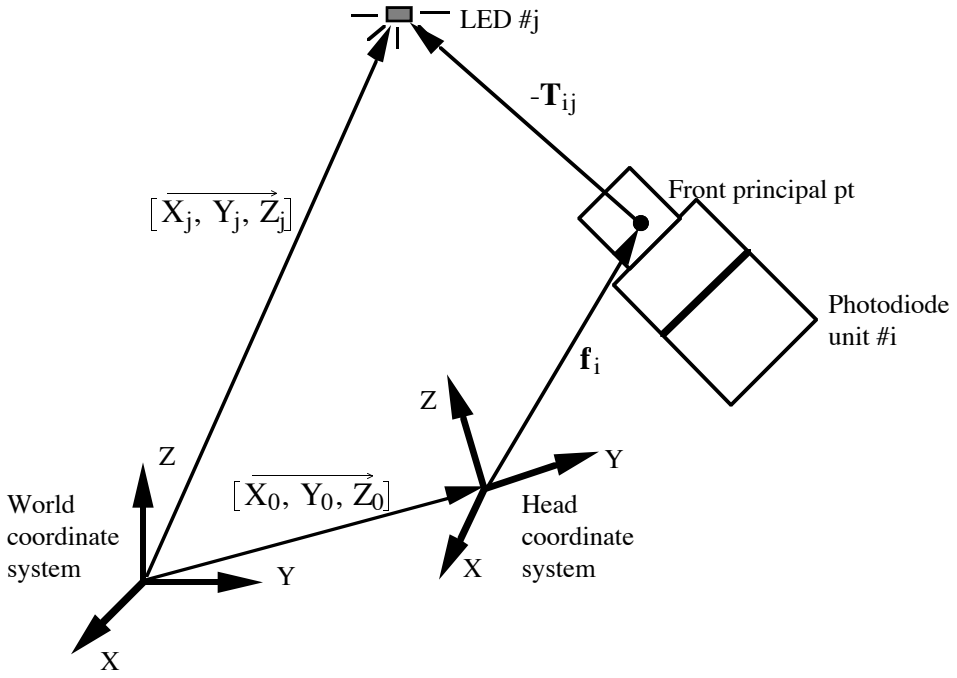


*Figure 12: expressing $\boldsymbol{T}_{ij}$ in terms of other vectors*

Now let's work on $\mathbf{T}_{ij}$ (see Figure 12). This time we start at the origin of the World coordinate system and work our way to LED #j. We can reach this by two paths. The direct path is to follow the vector represented by $[X_j, Y_j, Z_j]$, the known coordinates of the LED in World space. Alternately, we can take a more indirect route by first following $[X_0, Y_0, Z_0]$ to the center of the Head coordinate system, then following vector $\mathbf{f}_i$ to the front principal point, and finally taking vector $-\mathbf{T}_{ij}$ to LED #j. The algebraic expression that equates these two paths is:

$$\begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \mathbf{M}\,\mathbf{f}_i - \mathbf{T}_{ij} = \begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix}$$

where matrix $\mathbf{M}$ is applied to vector $\mathbf{f}_i$ to express everything in World space coordinates. If we rewrite the expression in terms of $\mathbf{T}_{ij}$, we get:

$$\mathbf{T}_{ij} = \begin{bmatrix} X_0 - X_j \\ Y_0 - Y_j \\ Z_0 - Z_j \end{bmatrix} + \mathbf{M}\,\mathbf{f}_i \tag{3}$$

Now we substitute both (2) and (3) into equation (1), yielding

$$\begin{bmatrix} X_0 - X_j \\ Y_0 - Y_j \\ Z_0 - Z_j \end{bmatrix} + \mathbf{M}\,\mathbf{f}_i = \lambda\,\mathbf{M}\left(\mathbf{d}_i - \mathbf{e}_i + \mathbf{M}_i \begin{bmatrix} x_{ij} \\ y_{ij} \\ 0 \end{bmatrix}\right) \tag{4}$$

If we rewrite (4) slightly, we get the *collinearity condition equation* $c_{ij}$, for photodiode #i and LED #j:

$$c_{ij}: \quad \begin{bmatrix} X_0 - X_j \\ Y_0 - Y_j \\ Z_0 - Z_j \end{bmatrix} + \mathbf{M}\,\mathbf{f}_i + \lambda\,\mathbf{M}\left(\mathbf{e}_i - \mathbf{d}_i - \mathbf{M}_i \begin{bmatrix} x_{ij} \\ y_{ij} \\ 0 \end{bmatrix}\right) = \vec{0} \tag{5}$$

## 2.4 Equation setup

Let's look at equation (5). We can divide the variables in the equation into known and unknown components:

*Known:*

| | |
|---|---|
| $X_j$, $Y_j$, $Z_j$ | (all LED locations in the ceiling are fixed and known) |
| $x_{ij}$, $y_{ij}$ | (locations of imaged points are read from the photodiode detectors) |
| $\mathbf{d}_i$, $\mathbf{e}_i$, $\mathbf{f}_i$, $\mathbf{M}_i$ | (photodiode unit locations are fixed and known with respect to the head) |

*Unknown:*

| | |
|---|---|
| $X_0$, $Y_0$, $Z_0$ | (head position) |
| $\mathbf{M}$ | (head orientation) |
| $\lambda$ | (scale factor) |

Also note that equation (5) really contains three equations: one each for X, Y, and Z components. But to extract these three equations, we need to replace the **M**'s by the 3x3 rotation matrices they represent and multiply everything out so $c_{ij}$ becomes one massive equality between 3x1 vectors.

What do rotation matrices look like? In Section 6.1, we define a rotation matrix **R** to be:

$$\mathbf{R} = \begin{bmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{bmatrix}$$

where the nine components r11, r12… r33 are functions of three rotation parameters $\omega$, $\alpha$, and $\kappa$. See Section 6.1 for the definitions of these functions. Using Euler angles to represent rotation opens the possibility of *gimbal lock*, which is also discussed in Section 6.1. In practice, we do not see gimbal lock because positions that can cause it are positions that do not have point the photodiode units toward the ceiling, so we lose tracking at those positions.

We will be using two types of rotation matrices, **M** and $\mathbf{M}_i$. To distinguish between the components of **M** and the components of $\mathbf{M}_i$, we will use the following notation:

$r11_M$ is the r11 term for matrix **M**
$r11_{M_i}$ is the r11 term for matrix $\mathbf{M}_i$

Now let's look at what we have:

- When photodiode unit #i sees LED #j, we can write a $c_{ij}$ to express the geometry of that event
- Each $c_{ij}$ is really composed of three separate equations
- 6+N total unknowns: 3 of position, 3 of orientation, and N scale factors, where N is the number of LEDs that we see

Note that we have N unknown scale factors because each $c_{ij}$ generates a different scale factor. In contrast, the 6 variables of position and orientation that we are trying to find are the same in every $c_{ij}$. We don't need the scale factors, and since they are potentially numerous, it is worthwhile to eliminate them. We can do this by rewriting the three equations in $c_{ij}$ so that all components with the scale factor lie on one side of the equals sign and all components without the scale factor lie on the other side. Then we divide the first and second equations by the third, eliminating the scale factor and the third equation. This should not lead to a division by zero because (Z0 - Zj) is always non-zero in our system, and because it is impossible to make the r31, r32 and r33 terms simultaneously zero. Our situation then becomes:

- When photodiode unit #i sees LED #j, it generates a $c_{ij}$ which in turn yields two independent equations
- Six total unknowns: 3 of position and 3 of orientation

For a linear system involving 6 unknowns, we need at least 6 independent equations to find a unique solution. Since each LED that we see produces 2 equations, our photodiodes must see at least 3 LEDs or we will not have enough information to determine the position and orientation of the user's head. (Our equations are actually nonlinear, but we make a linear approximation later in Section 2.5).

Now let's actually do the math that we just described. First, some notation. We need to break vectors and equations up into their X, Y and Z components, so we use the following notation:

$$c_{ij} = \begin{bmatrix} c_{ij}\langle x\rangle \\ c_{ij}\langle y\rangle \\ c_{ij}\langle z\rangle \end{bmatrix} \qquad \mathbf{d}_i = \begin{bmatrix} d_i\langle x\rangle \\ d_i\langle y\rangle \\ d_i\langle z\rangle \end{bmatrix}$$

Now if you expand the **M**'s into their matrix representations and do the matrix multiplication, you get the following three equations of $c_{ij}$:

$$\begin{aligned} c_{ij}\langle x\rangle: \quad & (X_0 - X_j) + r11_M\left(f_i\langle x\rangle + \lambda(e_i\langle x\rangle - d_i\langle x\rangle - J)\right) \\ & + r12_M\left(f_i\langle y\rangle + \lambda(e_i\langle y\rangle - d_i\langle y\rangle - H)\right) \\ & + r13_M\left(f_i\langle z\rangle + \lambda(e_i\langle z\rangle - d_i\langle z\rangle - L)\right) = 0 \end{aligned}$$

$$\begin{aligned} c_{ij}\langle y\rangle: \quad & (Y_0 - Y_j) + r21_M\left(f_i\langle x\rangle + \lambda(e_i\langle x\rangle - d_i\langle x\rangle - J)\right) \\ & + r22_M\left(f_i\langle y\rangle + \lambda(e_i\langle y\rangle - d_i\langle y\rangle - H)\right) \\ & + r23_M\left(f_i\langle z\rangle + \lambda(e_i\langle z\rangle - d_i\langle z\rangle - L)\right) = 0 \end{aligned}$$

$$\begin{aligned} c_{ij}\langle z\rangle: \quad & (Z_0 - Z_j) + r31_M\left(f_i\langle x\rangle + \lambda(e_i\langle x\rangle - d_i\langle x\rangle - J)\right) \\ & + r32_M\left(f_i\langle y\rangle + \lambda(e_i\langle y\rangle - d_i\langle y\rangle - H)\right) \\ & + r33_M\left(f_i\langle z\rangle + \lambda(e_i\langle z\rangle - d_i\langle z\rangle - L)\right) = 0 \end{aligned}$$

where

$$\begin{aligned} J &= r11_{M_i}\, x_{ij} + r12_{M_i}\, y_{ij} \\ H &= r21_{M_i}\, x_{ij} + r22_{M_i}\, y_{ij} \\ L &= r31_{M_i}\, x_{ij} + r32_{M_i}\, y_{ij} \end{aligned}$$

Now if we rewrite $c_{ij}\langle x\rangle$, $c_{ij}\langle y\rangle$, and $c_{ij}\langle z\rangle$ so that all terms with the scale factor are on one side and all terms without the scale factor are on the other side, the result is:

$$\begin{aligned} c_{ij}\langle x\rangle: \quad & \begin{matrix} (X_0 - X_j) + r11_M\, f_i\langle x\rangle \\ + r12_M\, f_i\langle y\rangle \\ + r13_M\, f_i\langle z\rangle \end{matrix} = -\lambda \begin{pmatrix} r11_M(e_i\langle x\rangle - d_i\langle x\rangle - J) \\ + r12_M(e_i\langle y\rangle - d_i\langle y\rangle - H) \\ + r13_M(e_i\langle z\rangle - d_i\langle z\rangle - L) \end{pmatrix} \\ c_{ij}\langle y\rangle: \quad & \begin{matrix} (Y_0 - Y_j) + r21_M\, f_i\langle x\rangle \\ + r22_M\, f_i\langle y\rangle \\ + r23_M\, f_i\langle z\rangle \end{matrix} = -\lambda \begin{pmatrix} r21_M(e_i\langle x\rangle - d_i\langle x\rangle - J) \\ + r22_M(e_i\langle y\rangle - d_i\langle y\rangle - H) \\ + r23_M(e_i\langle z\rangle - d_i\langle z\rangle - L) \end{pmatrix} \\ c_{ij}\langle z\rangle: \quad & \begin{matrix} (Z_0 - Z_j) + r31_M\, f_i\langle x\rangle \\ + r32_M\, f_i\langle y\rangle \\ + r33_M\, f_i\langle z\rangle \end{matrix} = -\lambda \begin{pmatrix} r31_M(e_i\langle x\rangle - d_i\langle x\rangle - J) \\ + r32_M(e_i\langle y\rangle - d_i\langle y\rangle - H) \\ + r33_M(e_i\langle z\rangle - d_i\langle z\rangle - L) \end{pmatrix} \end{aligned}$$

Finally, we divide $c_{ij}\langle x\rangle$ and $c_{ij}\langle y\rangle$ by the third equation, $c_{ij}\langle z\rangle$, to eliminate the scale factor. This produces two independent equations, which we express as $G_{ij}\langle 1\rangle = 0$ and $G_{ij}\langle 2\rangle = 0$:

$$G_{ij}\langle 1\rangle = \frac{\begin{pmatrix}(X_0 - X_j) + m11_M\, f_i\langle x\rangle \\ + m12_M\, f_i\langle y\rangle \\ + m13_M\, f_i\langle z\rangle\end{pmatrix}}{\begin{pmatrix}(Z_0 - Z_j) + m31_M\, f_i\langle x\rangle \\ + m32_M\, f_i\langle y\rangle \\ + m33_M\, f_i\langle z\rangle\end{pmatrix}} - \frac{\begin{pmatrix}m11_M\,(e_i\langle x\rangle - d_i\langle x\rangle - J) \\ + m12_M\,(e_i\langle y\rangle - d_i\langle y\rangle - H) \\ + m13_M\,(e_i\langle z\rangle - d_i\langle z\rangle - L)\end{pmatrix}}{\begin{pmatrix}m31_M\,(e_i\langle x\rangle - d_i\langle x\rangle - J) \\ + m32_M\,(e_i\langle y\rangle - d_i\langle y\rangle - H) \\ + m33_M\,(e_i\langle z\rangle - d_i\langle z\rangle - L)\end{pmatrix}} = 0 \qquad (6)$$

$$G_{ij}\langle 2\rangle = \frac{\begin{pmatrix}(Y_0 - Y_j) + m21_M\, f_i\langle x\rangle \\ + m22_M\, f_i\langle y\rangle \\ + m23_M\, f_i\langle z\rangle\end{pmatrix}}{\begin{pmatrix}(Z_0 - Z_j) + m31_M\, f_i\langle x\rangle \\ + m32_M\, f_i\langle y\rangle \\ + m33_M\, f_i\langle z\rangle\end{pmatrix}} - \frac{\begin{pmatrix}m21_M\,(e_i\langle x\rangle - d_i\langle x\rangle - J) \\ + m22_M\,(e_i\langle y\rangle - d_i\langle y\rangle - H) \\ + m23_M\,(e_i\langle z\rangle - d_i\langle z\rangle - L)\end{pmatrix}}{\begin{pmatrix}m31_M\,(e_i\langle x\rangle - d_i\langle x\rangle - J) \\ + m32_M\,(e_i\langle y\rangle - d_i\langle y\rangle - H) \\ + m33_M\,(e_i\langle z\rangle - d_i\langle z\rangle - L)\end{pmatrix}} = 0 \qquad (7)$$

## 2.5 Iterative solution

Equations (6) and (7) express a relationship that is geometrically true. We know that if the correct values are plugged in for all the values in $G_{ij}$, then they will indeed equal zero. We know all of the values except the six variables representing position and orientation. So what can equations (6) and (7) tell us about those missing six variables?

One approach is to first make an initial guess of the position and orientation, then rewrite equations (6) and (7) in such a way that when this guess is plugged in, it provides correction factors that we can add to our six guesses to get a more accurate guess. We can then plug in this new guess and calculate new correction factors. This cycle repeats until the correction factors become so small that we converge to a solution.

The requirement of needing an initial guess of the user's position and orientation is actually quite reasonable in our particular application. We usually have a very good initial guess available: the last known position and orientation. Since our system can track the user's head at rates between 20 and 100 Hz, the last known position and orientation will almost certainly be very close to the current position and orientation. The only times when we run into trouble is when we have no idea where the user is, such as at system startup or when the user orients his head so that too few photodiode units can see the ceiling. We discuss recovering from such degenerate situations in Section 3.5.

To get these correction factors, we linearize equations (6) and (7) by using Taylor's theorem to get the first-order linear approximation:

$$0 = (G_{ij}\langle 1\rangle)_{PO} + \left(\frac{\partial G_{ij}\langle 1\rangle}{\partial X_0}\right)_{PO} dX_0 + \left(\frac{\partial G_{ij}\langle 1\rangle}{\partial Y_0}\right)_{PO} dY_0 + \left(\frac{\partial G_{ij}\langle 1\rangle}{\partial Z_0}\right)_{PO} dZ_0 + \left(\frac{\partial G_{ij}\langle 1\rangle}{\partial \omega}\right)_{PO} d\omega + \left(\frac{\partial G_{ij}\langle 1\rangle}{\partial \alpha}\right)_{PO} d\alpha + \left(\frac{\partial G_{ij}\langle 1\rangle}{\partial \kappa}\right)_{PO} d\kappa$$

$$0 = (G_{ij}\langle 2\rangle)_{PO} + \left(\frac{\partial G_{ij}\langle 2\rangle}{\partial X_0}\right)_{PO} dX_0 + \left(\frac{\partial G_{ij}\langle 2\rangle}{\partial Y_0}\right)_{PO} dY_0 + \left(\frac{\partial G_{ij}\langle 2\rangle}{\partial Z_0}\right)_{PO} dZ_0 + \left(\frac{\partial G_{ij}\langle 2\rangle}{\partial \omega}\right)_{PO} d\omega + \left(\frac{\partial G_{ij}\langle 2\rangle}{\partial \alpha}\right)_{PO} d\alpha + \left(\frac{\partial G_{ij}\langle 2\rangle}{\partial \kappa}\right)_{PO} d\kappa$$

where: PO stands for the guess of the user's position and orientation,
$(G_{ij}\langle 1\rangle)_{PO}$ is the value of $G_{ij}$ given a PO,

$\left(\frac{\partial G_{ij}\langle 1\rangle}{\partial \omega}\right)_{PO}$ is the value of the partial derivative given a PO, etc.

Every LED that our system sees generates two such equations. Say that we see N LEDs. Then we can write the 2N generated equations in matrix form as follows:

$$\underset{2Nx1\ vector}{\mathbf{G}_0} + \underset{2Nx6\ matrix}{\partial \mathbf{G}} * \underset{6x1\ vector}{\mathbf{D}} = \underset{2Nx1\ vector}{\vec{0}} \qquad (8)$$

where the vectors and matrices are defined as follows:

$$\overrightarrow{\mathbf{G}_0} = \begin{bmatrix} (G_{i1,j1}\langle 1\rangle)_{PO} \\ (G_{i1,j1}\langle 2\rangle)_{PO} \\ \vdots \\ (G_{iN,jN}\langle 2\rangle)_{PO} \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} dX_0 \\ dY_0 \\ dZ_0 \\ d\omega \\ d\alpha \\ d\kappa \end{bmatrix}$$

$$\partial \mathbf{G} = \begin{bmatrix} \left(\frac{\partial G_{i1,j1}\langle 1\rangle}{\partial X_0}\right)_{PO} & \left(\frac{\partial G_{i1,j1}\langle 1\rangle}{\partial Y_0}\right)_{PO} & \left(\frac{\partial G_{i1,j1}\langle 1\rangle}{\partial Z_0}\right)_{PO} & \left(\frac{\partial G_{i1,j1}\langle 1\rangle}{\partial \omega}\right)_{PO} & \left(\frac{\partial G_{i1,j1}\langle 1\rangle}{\partial \alpha}\right)_{PO} & \left(\frac{\partial G_{i1,j1}\langle 1\rangle}{\partial \kappa}\right)_{PO} \\ \left(\frac{\partial G_{i1,j1}\langle 2\rangle}{\partial X_0}\right)_{PO} & \left(\frac{\partial G_{i1,j1}\langle 2\rangle}{\partial Y_0}\right)_{PO} & \left(\frac{\partial G_{i1,j1}\langle 2\rangle}{\partial Z_0}\right)_{PO} & \left(\frac{\partial G_{i1,j1}\langle 2\rangle}{\partial \omega}\right)_{PO} & \left(\frac{\partial G_{i1,j1}\langle 2\rangle}{\partial \alpha}\right)_{PO} & \left(\frac{\partial G_{i1,j1}\langle 2\rangle}{\partial \kappa}\right)_{PO} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \left(\frac{\partial G_{iN,jN}\langle 2\rangle}{\partial X_0}\right)_{PO} & \left(\frac{\partial G_{iN,jN}\langle 2\rangle}{\partial Y_0}\right)_{PO} & \left(\frac{\partial G_{iN,jN}\langle 2\rangle}{\partial Z_0}\right)_{PO} & \left(\frac{\partial G_{iN,jN}\langle 2\rangle}{\partial \omega}\right)_{PO} & \left(\frac{\partial G_{iN,jN}\langle 2\rangle}{\partial \alpha}\right)_{PO} & \left(\frac{\partial G_{iN,jN}\langle 2\rangle}{\partial \kappa}\right)_{PO} \end{bmatrix}$$

where i1, j1 = Photodiode unit #i1 and LED #j1
i2, j2 = Photodiode unit #i2 and LED #j2
etc.

**D** is what we want to find, because it holds the correction factors that we will use to modify our guess of the position and orientation of the head. So how do we rewrite equation (8) to extract **D**? We now list several ways to recover **D**:

We need to see at least three LEDs to generate six equations, or our system will be underdetermined. If we see exactly three LEDs, then $\partial \mathbf{G}$ becomes a 6x6 matrix and we can invert it to get:

$$\underset{6x1}{\mathbf{D}} = \underset{6x6}{\partial \mathbf{G}^{-1}} * \underset{6x1}{-\mathbf{G}_0} \qquad (9)$$

But if we see more than three LEDs, our system becomes overdetermined and we can use a least squares approach by taking the pseudo-inverse of $\partial \mathbf{G}$:

$$\underset{6x1}{\mathbf{D}} \approx \underset{6x2N}{\left[\partial \mathbf{G}^T\right.} * \underset{2Nx6}{\left.\partial \mathbf{G}\right]^{-1}} * \underset{6x2N}{\partial \mathbf{G}^T} * \underset{2Nx1}{-\mathbf{G}_0} \qquad (10a)$$

We can also use weighted least squares by introducing a new matrix **P** that weighs the values from each LED measurement differently. LEDs that are far away from our head generate more noisy measurements than LEDs that are close, because of the reduced signal/noise ratio and because of the limited resolution of our photodiodes. Although we don't currently do this, we could create a **P** to reduce the contribution of LEDs far from the user's head. The formula to extract **D** then becomes:

$$\mathbf{D} \approx \left[\partial\mathbf{G}^T * \mathbf{P} * \partial\mathbf{G}\right]^{-1} * \partial\mathbf{G}^T * \mathbf{P} * -\mathbf{G}_0 \qquad (10b)$$

where **P** is a 2Nx2N diagonal matrix of the form:

$$\mathbf{P} = \begin{bmatrix} p_1 & & & \\ & p_1 & & \text{zeroes} \\ & & \ddots & \\ & & & p_N \\ \text{zeroes} & & & \\ & & & p_N \end{bmatrix}$$

and $p_1$ to $p_N$ are the weights for each LED measurement.

Finally, we can use *singular value decomposition* to extract **D**. This is currently our method of choice because it handles ill-conditioned $\partial\mathbf{G}$ matrices better than the other methods can. For a description of how this method works, please see Section 6.2.

We can now outline the main loop of collinearity in psuedocode:

```
Generate initial guess of head position and orientation
repeat
{
        With our current guess, compute G0 and ∂G
        Compute D using equation (9), (10a), (10b), or singular value decomposition
        Add the correction factors in D to our current guess to generate a new guess
}
until ((magnitude of D is small enough) or (counter > iteration_limit))

if (counter > iteration_limit)
        convergence failed
else
        computed position and orientation = our guess
```

Some implementation notes:

- Currently we set iteration_limit to 10, because collinearity tends to converge quickly when it can find a solution (see Section 3.3).
- What does it mean for **D** to be "small enough?" We currently use the following criteria:

  A) $\sqrt{\partial X_0^2 + \partial Y_0^2 + \partial Z_0^2} < 0.1$ mm and

  B) $\sqrt{\partial\omega^2 + \partial\alpha^2 + \partial\kappa^2} < 0.1$ degrees

  because these limits provide adequate accuracy while still permitting convergence within a reasonable number of iterations

## 2.6 Partial derivatives

To compute the terms in the matrix $\partial\mathbf{G}$, we need to find the partial derivatives of $G_{ij}\langle 1\rangle$ and $G_{ij}\langle 2\rangle$ with respect to the three variables of position and the three of orientation. Since some of these are messy, we have saved them for this last section.

What should the partial derivatives look like? Let's look at the $G_{ij}$ expressions in equations (6) and (7). Notice that the only places where the position variables $X_0$, $Y_0$ and $Z_0$ occur are in the first fraction, and the variables have no coefficients associated with them. This should make the partial derivatives with respect to position fairly easy to do. In contrast, the three variables of orientation are "hidden" inside rotation terms r11… r33. These terms occur in all of the numerators and denominators in $G_{ij}$. Therefore, the partial derivatives with respect to orientation may be fairly complicated, although with shorthand notation we can reduce their complexity.

Let's work out the partial derivatives with respect to position first. To make the results a bit cleaner, we define the following expressions to use as shorthand:

$$
\begin{aligned}
A1 &= (X_0 - X_j) + r11_M\, f_i\langle x\rangle + r12_M\, f_i\langle y\rangle + r13_M\, f_i\langle z\rangle \\
A3 &= (Y_0 - Y_j) + r21_M\, f_i\langle x\rangle + r22_M\, f_i\langle y\rangle + r23_M\, f_i\langle z\rangle \\
B1 &= (Z_0 - Z_j) + r31_M\, f_i\langle x\rangle + r32_M\, f_i\langle y\rangle + r33_M\, f_i\langle z\rangle \\
A2 &= r11_M\,(e_i\langle x\rangle - d_i\langle x\rangle - J) \\
&\quad + r12_M\,(e_i\langle y\rangle - d_i\langle y\rangle - H) \\
&\quad + r13_M\,(e_i\langle z\rangle - d_i\langle z\rangle - L) \\
A4 &= r21_M\,(e_i\langle x\rangle - d_i\langle x\rangle - J) \\
&\quad + r22_M\,(e_i\langle y\rangle - d_i\langle y\rangle - H) \\
&\quad + r23_M\,(e_i\langle z\rangle - d_i\langle z\rangle - L) \\
B2 &= r31_M\,(e_i\langle x\rangle - d_i\langle x\rangle - J) \\
&\quad + r32_M\,(e_i\langle y\rangle - d_i\langle y\rangle - H) \\
&\quad + r33_M\,(e_i\langle z\rangle - d_i\langle z\rangle - L)
\end{aligned}
$$

With these definitions, equations (6) and (7) can be rewritten as:

$$G_{ij}\langle 1\rangle = \frac{A1}{B1} - \frac{A2}{B2} = 0$$

$$G_{ij}\langle 2\rangle = \frac{A3}{B1} - \frac{A4}{B2} = 0$$

And now we can list the partial derivatives of $G_{ij}$ with respect to $X_0$, $Y_0$, and $Z_0$:

$$\frac{\partial G_{ij}\langle 1\rangle}{\partial X_0} = \frac{1}{B1} \qquad \frac{\partial G_{ij}\langle 2\rangle}{\partial X_0} = 0$$

$$\frac{\partial G_{ij}\langle 1\rangle}{\partial Y_0} = 0 \qquad \frac{\partial G_{ij}\langle 2\rangle}{\partial Y_0} = \frac{1}{B1}$$

$$\frac{\partial G_{ij}\langle 1\rangle}{\partial Z_0} = -\frac{A1}{B1^2} \qquad \frac{\partial G_{ij}\langle 2\rangle}{\partial Z_0} = -\frac{A3}{B1^2}$$

We're halfway done. All we have left are the partial derivatives of $G_{ij}$ with respect to orientation. First, note that all of the rotation terms r11… r33 (defined in Section 6.1) depend on orientation, so we first calculate the partial derivatives of those, since we will need them later:

$$\frac{\partial r11}{\partial \omega} = 0 \qquad \frac{\partial r12}{\partial \omega} = \sin\kappa \sin\omega + \cos\kappa \sin\alpha \cos\omega$$
$$\frac{\partial r11}{\partial \alpha} = -\cos\kappa \sin\alpha \qquad \frac{\partial r12}{\partial \alpha} = \cos\kappa \cos\alpha \sin\omega$$
$$\frac{\partial r11}{\partial \kappa} = -\sin\kappa \cos\alpha \qquad \frac{\partial r12}{\partial \kappa} = -\cos\kappa \cos\omega - \sin\kappa \sin\alpha \sin\omega$$

$$\frac{\partial r13}{\partial \omega} = \sin\kappa \cos\omega - \sin\omega \cos\kappa \sin\alpha \qquad \frac{\partial r21}{\partial \omega} = 0$$
$$\frac{\partial r13}{\partial \alpha} = \cos\omega \cos\kappa \cos\alpha \qquad \frac{\partial r21}{\partial \alpha} = -\sin\kappa \sin\alpha$$
$$\frac{\partial r13}{\partial \kappa} = \cos\kappa \sin\omega - \cos\omega \sin\kappa \sin\alpha \qquad \frac{\partial r21}{\partial \kappa} = \cos\kappa \cos\alpha$$

$$\frac{\partial r22}{\partial \omega} = -\cos\kappa \sin\omega + \cos\omega \sin\kappa \sin\alpha \qquad \frac{\partial r23}{\partial \omega} = -\cos\omega \cos\kappa - \sin\omega \sin\kappa \sin\alpha$$
$$\frac{\partial r22}{\partial \alpha} = \sin\omega \sin\kappa \cos\alpha \qquad \frac{\partial r23}{\partial \alpha} = \cos\omega \sin\kappa \cos\alpha$$
$$\frac{\partial r22}{\partial \kappa} = -\sin\kappa \cos\omega + \sin\omega \cos\kappa \sin\alpha \qquad \frac{\partial r23}{\partial \kappa} = \sin\omega \sin\kappa + \cos\omega \cos\kappa \sin\alpha$$

$$\frac{\partial r31}{\partial \omega} = 0 \qquad \frac{\partial r32}{\partial \omega} = \cos\omega \cos\alpha \qquad \frac{\partial r33}{\partial \omega} = -\sin\omega \cos\alpha$$
$$\frac{\partial r31}{\partial \alpha} = -\cos\alpha \qquad \frac{\partial r32}{\partial \alpha} = -\sin\omega \sin\alpha \qquad \frac{\partial r33}{\partial \alpha} = -\cos\omega \sin\alpha$$
$$\frac{\partial r31}{\partial \kappa} = 0 \qquad \frac{\partial r32}{\partial \kappa} = 0 \qquad \frac{\partial r33}{\partial \kappa} = 0$$

Next, we define a few more shorthand expressions:

$$E_i\langle x\rangle = e_i\langle x\rangle - d_i\langle x\rangle - J$$
$$E_i\langle y\rangle = e_i\langle y\rangle - d_i\langle y\rangle - H$$
$$E_i\langle z\rangle = e_i\langle z\rangle - d_i\langle z\rangle - L$$

And finally, we can write the partial derivatives with respect to orientation, expressed in terms of the partials of r11… r33 that we previously calculated:

$$\frac{\partial G_{ij}\langle 1\rangle}{\partial \omega} = \frac{B1\left(\frac{\partial r12}{\partial \omega} f_i\langle y\rangle + \frac{\partial r13}{\partial \omega} f_i\langle z\rangle\right) - A1\left(\frac{\partial r32}{\partial \omega} f_i\langle y\rangle + \frac{\partial r33}{\partial \omega} f_i\langle z\rangle\right)}{B1^2} - \frac{B2\left(\frac{\partial r12}{\partial \omega} E_i\langle y\rangle + \frac{\partial r13}{\partial \omega} E_i\langle z\rangle\right) - A2\left(\frac{\partial r32}{\partial \omega} E_i\langle y\rangle + \frac{\partial r33}{\partial \omega} E_i\langle z\rangle\right)}{B2^2}$$

$$\frac{\partial G_{ij}\langle 1\rangle}{\partial \alpha} = \frac{B1\left(\frac{\partial r11}{\partial \alpha} f_i\langle x\rangle + \frac{\partial r12}{\partial \alpha} f_i\langle y\rangle + \frac{\partial r13}{\partial \alpha} f_i\langle z\rangle\right) - A1\left(\frac{\partial r31}{\partial \alpha} f_i\langle x\rangle + \frac{\partial r32}{\partial \alpha} f_i\langle y\rangle + \frac{\partial r33}{\partial \alpha} f_i\langle z\rangle\right)}{B1^2} - \frac{B2\left(\frac{\partial r11}{\partial \alpha} E_i\langle x\rangle + \frac{\partial r12}{\partial \alpha} E_i\langle y\rangle + \frac{\partial r13}{\partial \alpha} E_i\langle z\rangle\right) - A2\left(\frac{\partial r31}{\partial \alpha} E_i\langle x\rangle + \frac{\partial r32}{\partial \alpha} E_i\langle y\rangle + \frac{\partial r33}{\partial \alpha} E_i\langle z\rangle\right)}{B2^2}$$

$$\frac{\partial G_{ij}\langle 1\rangle}{\partial \kappa} = \frac{\frac{\partial r11}{\partial \kappa} f_i\langle x\rangle + \frac{\partial r12}{\partial \kappa} f_i\langle y\rangle + \frac{\partial r13}{\partial \kappa} f_i\langle z\rangle}{B1} - \frac{\frac{\partial r11}{\partial \kappa} E_i\langle x\rangle + \frac{\partial r12}{\partial \kappa} E_i\langle y\rangle + \frac{\partial r13}{\partial \kappa} E_i\langle z\rangle}{B2}$$

$$\frac{\partial G_{ij}\langle 2\rangle}{\partial \omega} = \frac{B1\left(\frac{\partial r22}{\partial \omega} f_i\langle y\rangle + \frac{\partial r23}{\partial \omega} f_i\langle z\rangle\right) - A3\left(\frac{\partial r32}{\partial \omega} f_i\langle y\rangle + \frac{\partial r33}{\partial \omega} f_i\langle z\rangle\right)}{B1^2} - \frac{B2\left(\frac{\partial r22}{\partial \omega} E_i\langle y\rangle + \frac{\partial r23}{\partial \omega} E_i\langle z\rangle\right) - A4\left(\frac{\partial r32}{\partial \omega} E_i\langle y\rangle + \frac{\partial r33}{\partial \omega} E_i\langle z\rangle\right)}{B2^2}$$

$$\frac{\partial G_{ij}\langle 2\rangle}{\partial \alpha} = \frac{B1\left(\frac{\partial r21}{\partial \alpha} f_i\langle x\rangle + \frac{\partial r22}{\partial \alpha} f_i\langle y\rangle + \frac{\partial r23}{\partial \alpha} f_i\langle z\rangle\right) - A3\left(\frac{\partial r31}{\partial \alpha} f_i\langle x\rangle + \frac{\partial r32}{\partial \alpha} f_i\langle y\rangle + \frac{\partial r33}{\partial \alpha} f_i\langle z\rangle\right)}{B1^2} - \frac{B2\left(\frac{\partial r21}{\partial \alpha} E_i\langle x\rangle + \frac{\partial r22}{\partial \alpha} E_i\langle y\rangle + \frac{\partial r23}{\partial \alpha} E_i\langle z\rangle\right) - A4\left(\frac{\partial r31}{\partial \alpha} E_i\langle x\rangle + \frac{\partial r32}{\partial \alpha} E_i\langle y\rangle + \frac{\partial r33}{\partial \alpha} E_i\langle z\rangle\right)}{B2^2}$$

$$\frac{\partial G_{ij}\langle 2\rangle}{\partial \kappa} = \frac{\frac{\partial r21}{\partial \kappa} f_i\langle x\rangle + \frac{\partial r22}{\partial \kappa} f_i\langle y\rangle + \frac{\partial r23}{\partial \kappa} f_i\langle z\rangle}{B1} - \frac{\frac{\partial r21}{\partial \kappa} E_i\langle x\rangle + \frac{\partial r22}{\partial \kappa} E_i\langle y\rangle + \frac{\partial r23}{\partial \kappa} E_i\langle z\rangle}{B2}$$

## 3.0 Behavior

In this section we describe some empirical observations of how collinearity behaves in our system.

### 3.1 Number of LEDs to use

One can write a program that implements collinearity as we just described it, place it in a simulator, and discover that it will converge to an exact solution given only three LEDs. In a real system, however, many potential sources of error exist that can degrade our solution. These include:

- Errors in the positions of the LEDs
- Lens distortion
- Nonlinear response of the photodiode unit detector
- Misplacement of a detector within its photodiode unit
- Photodiode unit detector resolution limits
- Analog to Digital conversion resolution limits
- Noise in the analog signals
- Errors in the positions and orientations of the photodiode units on the head unit

Collinearity has the nice feature of being able to reduce some of these errors by using more than three LEDs to generate a solution. As shown in equations (10a) and (10b), collinearity uses the extra information to get the best possible solution in a least squares sense. Because of this fact, we usually like to run on 12 to 14 LEDs in our real working system.

Using more than three LEDs is not a panacea. From simulation runs we have found that it can help reduce certain types of errors, but it cannot compensate for other types. Examples include:

*Errors that can be reduced:*

- Resolution limits
- Random errors in the positions of the LEDs

*Errors that are not reduced:*

- Nonlinear response of the photodiode detector
- Lens distortion
- Systematic errors in the positions of the LEDs

3.2 Photodiode unit configuration

In theory, we can track using any 3 LEDs that we can find. In practice, these 3 LEDs should be widely separated from each other, because of the limited resolution of our photodiode detectors and other system errors. Tracking using widely-separated LEDs gives more "geometric strength" to the system of equations that we generate and yields a better solution. We can try to build some intuition about this by comparing it to the problem of triangulation.

The accuracy of triangulation depends heavily on how physically separated the three points are. Say that an earthquake occurs in Santa Cruz, CA, and we have three seismographs at Eureka, Los Angeles, and Sacramento. Since those three detectors are widely separated, they should be able to locate the epicenter of the earthquake accurately.

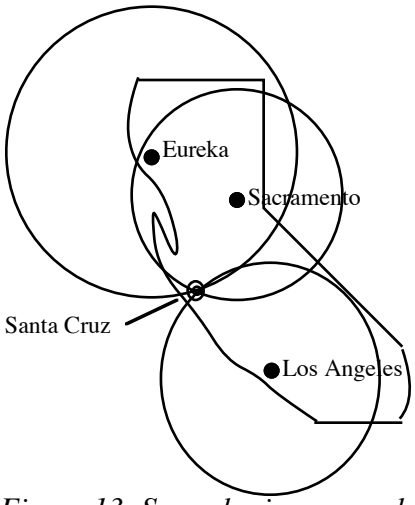


*Figure 13: Spread seisomographs*

Now take the same earthquake but place the three seismographs at UCLA, USC, and Pepperdine. Because those three detectors are close to each other, it will be much harder to accurately locate the epicenter by triangulation.

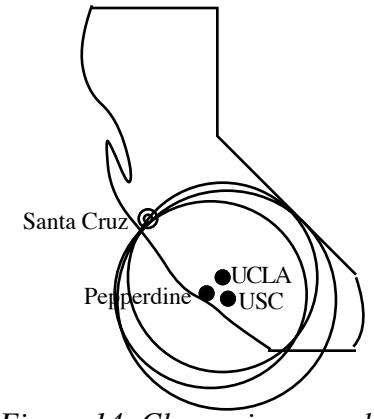


*Figure 14: Close seismographs*

To improve the solution, we need to add a seismograph that is far away from the Los Angeles region. Adding another seismograph near Los Angeles (at Cal Tech, perhaps) will not help nearly as much as adding one at, say, Fresno. Analogously, we discovered that in practice we usually cannot converge on a good solution based on the LEDs seen by only one photodiode unit, because our lenses have narrow fields of view. The LEDs that one photodiode unit sees are simply too close together. We need some distance between our LEDs to get a good solution. Therefore, we need to have two or three photodiode units viewing LEDs or we may not get a reasonable solution.

### 3.3 Convergence properties

How many iterations does it take for collinearity to converge on a solution? This depends on how close our intial guess of the user's position and orientation is to the true value. Under normal conditions, our guess should be very close, because our system tracks at rates up to 100 Hz and the user cannot move his head very far between two consecutive samples. In these cases, collinearity converges within one or two iterations.

If the guess is far away from the true value, collinearity will take many more iterations to converge to a solution, or it may not converge at all. If the guess is close enough to converge, then collinearity typically reaches a solution within six or seven iterations.

How close does our guess have to be to get collinearity to converge? This depends on the geometry of the system and how much error exists. We do not have an analytical statement that expresses how close the guess has to be, but we ran some experiments and generated the following rules of thumb for our actual system:

- Position should be within 6 feet
- $\omega$ should be within 30 degrees
- $\alpha$ should be within 30 degrees
- $\kappa$ should be within 45 degrees

where rotation parameters $\omega$, $\alpha$, $\kappa$ are as defined in Section 6.1.

### 3.4 Computation time

Collinearity is a computationally-intensive routine that demands a fast floating-point number cruncher if it is to run in realtime. If the system sees N LEDs, then at each iteration collinearity computes the values in the 2Nx1 matrix $\mathbf{G}_0$ and the 6x2N matrix $\partial\mathbf{G}$, then does the matrix operations required to extract $\mathbf{D}$, which includes inverting a 6x6 matrix. It may take a few iterations to reach a solution. And we would like to generate positions and orientations at a rate of 60 Hz or more.

Because of the amount of floating-point power required, we currently run collinearity on an Intel i860. Although we have not completely optimized our routine for the i860, we have found it yields satisfactory performance. A single iteration of collinearity, using 12 LEDs, usually takes about 6 milliseconds on the i860.

### 3.5 Generating initial guesses

Collinearity requires an initial guess of the user's position and orientation. Normally we have a very good guess: the last known position and orientation. But sometimes we do not know where the user is. This occurs at startup or when we lose track of the user because he tilts his head so far that the an insufficient number of photodiode units no longer face the ceiling. This usually means one or fewer photodiode units, although occasionally even two are insufficient. Under these degenerate conditions, we must generate initial guesses.

We do this by looking up a series of guesses stored in a table. In Section 3.3 we described how close a guess needs to be to the true value for collinearity to converge to a correct solution. Based on these empirical results, we space our guesses along each dimension as follows:

ω: 30 degree increments from 60º to -60º, for a total of 5 steps
α: 30 degree increments from 30º to -30º, for a total of 3 steps
κ: 45 degree increments across 360º, for a total of 8 steps

We use only one guess for position. It is directly underneath the center of our ceiling, with height Z in World space set to 5' 8" (a typical height for a user).

Therefore, our table has a total of 120 guesses (5*3*8*1 along the four dimensions). When we lose track of the user's position, we get a guess out of the table and insert that into collinearity. If collinearity converges to a reasonable solution, then we have found the user. Otherwise, we get another guess out of the table and try again. We keep circling through the table until collinearity converges to a proper solution, reestablishing our lock on the user's location.

## 4.0 Acknowledgements

Carney Clegg optimized some of the collinearity code to run on a Skybolt i860 board. Mark Mine and Stefan Gottschalk reviewed drafts of this report. Special thanks go to Dr. John F. Hughes of Brown University for providing detailed and constructive criticism of both the wording and the mathematics. This work was partially supported by ONR grant #N00014-86-K-0680, DARPA grant #DAEA18-90-C-0044, and a Pogue Fellowship.

## 5.0 References

[Foley90] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes. *Computer Graphics: Principles and Practice, Second Edition.* Addison-Wesley, USA, 1990.

[Press88] William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling. *Numerical Recipies in C.* Cambridge University Press, USA, 1988.

[Shoe89] Ken Shoemake, "Animating Rotation with Quaternion Curves," SIGGRAPH '89 course notes #23 (Math for SIGGRAPH)

[Ward92] Mark Ward, Ronald Azuma, Robert Bennett, Stefan Gottschalk, Henry Fuchs, "A demonstrated optical tracker with scalable work area for head-mounted display systems," to appear in *Proceedings of 1992 Symposium on Interactive 3D Graphics*, Cambridge, MA 1992

[Wolf83] Paul R. Wolf. *Elements of Photogrammetry with Air Photo Interpretation and Remote Sensing.* McGraw-Hill, USA 1983.

## 6.0 Appendices

### 6.1 Rotation matrix

For general information about rotation matrices, see any introductory computer graphics text, such as [Foley90].

We can represent any 3D rotation as a 3x3 matrix **R** that operates on the 3x1 coordinates of a point **P**, where

$$\mathbf{R} = \begin{bmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{bmatrix} \quad \mathbf{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

To rotate **P** to a new point **P'**, we set **P'** = **R P**.

Although **R** has nine terms, they are not all independent. We reduce this to three parameters by taking an Euler angle approach, saying that this rotation is the combination of three separate rotations around the three axes. This is simple and is the approach photogrammetrists usually take. We arbitrarily choose these three parameters as follows:

$\omega$ = angle of rotation around the X-axis
$\alpha$ = angle of rotation around the Y-axis
$\kappa$ = angle of rotation around the Z-axis

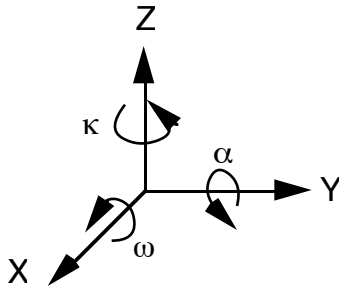

*Figure 15: Rotation parameters*

Rotations are not commutative, so we arbitrarily define our rotations to occur in the following order: first rotate the point around the X-axis, then around the Y-axis, and finally the Z-axis. We also define our coordinate system to be right-handed. With this we can define our matrix **R**:

$$\mathbf{R} = \underset{\text{Rotation around Z-axis}}{\begin{bmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix}} \underset{\text{Rotation around Y-axis}}{\begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}} \underset{\text{Rotation around X-axis}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{bmatrix}}$$

Multiplying the three individual rotations yields our desired rotation matrix:

$$\mathbf{R} = \begin{bmatrix} \cos\kappa\cos\alpha & -\sin\kappa\cos\omega + \cos\kappa\sin\alpha\sin\omega & \sin\kappa\sin\omega + \cos\omega\cos\kappa\sin\alpha \\ \sin\kappa\cos\alpha & \cos\kappa\cos\omega + \sin\omega\sin\kappa\sin\alpha & -\sin\omega\cos\kappa + \cos\omega\sin\kappa\sin\alpha \\ -\sin\alpha & \sin\omega\cos\alpha & \cos\omega\cos\alpha \end{bmatrix}$$

Therefore, the terms are:

$r11 = \cos\kappa\cos\alpha$

$r12 = -\sin\kappa\cos\omega + \cos\kappa\sin\alpha\sin\omega$

$r13 = \sin\kappa\sin\omega + \cos\omega\cos\kappa\sin\alpha$

$r21 = \sin\kappa\cos\alpha$

$r22 = \cos\kappa\cos\omega + \sin\omega\sin\kappa\sin\alpha$

$r23 = -\sin\omega\cos\kappa + \cos\omega\sin\kappa\sin\alpha$

$r31 = -\sin\alpha$

$r32 = \sin\omega\cos\alpha$

$r33 = \cos\omega\cos\alpha$

However, this approach suffers the possibility of *gimbal lock*, which means that in certain positions, you lose one degree of rotational freedom. For example, if $\alpha$ is 90 degrees, then both $\omega$ and $\kappa$ end up rotating around the Z-axis. However, our four photodiode units are not arranged in a way that allows a user to reach a gimbal lock orientation and still keep enough photodiode units aimed toward the ceiling to maintain tracking.

However, a larger ceiling or more photodiode units on the user's head will increase the orientation range to the point where we will have to seriously worry about gimbal lock. The first step is to change the order of the rotations so that the gimbal lock orienation occurs when the user tries to tilt 90 degrees along the X-axis (roll), which is the least likely to occur head rotation. In the unlikely case that even that orientation is achievable, more drastic measures are called for. One solution is to treat all nine r11..r33 terms as variables, subject to six constraints (that the 3 column vectors that form a rotation matrix are orthonormal). We get an initial guess for all nine variables, change our system to produce deltas for all nine variables, and use the Gram-Schmidt process at each iteration to enforce the six constraints. Another possibility is to use quaternions (see [Shoe89]), which do not suffer from gimbal lock. Rewriting all of the math to use quaternions is a nontrivial process, however.

## 6.2 Singular value decomposition

When solving the linear system expressed in equation (8), we can run into trouble when the matrix $\partial\mathbf{G}$ becomes singular or nearly so. To address this problem, we currently use the method of *singular value decomposition* (SVD) to solve for **D**. This method warns you when $\partial\mathbf{G}$ is becoming ill-conditioned and generates better solutions than direct methods can when $\partial\mathbf{G}$ is ill-conditioned.

When can $\partial\mathbf{G}$ become ill-conditioned? In theory this should not happen. As long as we can see three LEDs, no matter where they are, we should have enough information to converge on a correct solution. But as discussed in Section 3.2, resolution limits and system errors prevent this from being the case in a real working system. $\partial\mathbf{G}$ usually becomes ill-behaved if only one photodiode sees LEDs, occasionally if only two photodiodes see LEDs, and almost never if three or more photodiodes see LEDs.

When $\partial\mathbf{G}$ becomes ill-conditioned, it may be difficult or impossible to invert it because of the limited resolution of floating-point numbers. If we blindly feed our matrix into an inversion routine, we could get division by zero errors. We now provide an overview of how the SVD routines are applied in our system, but do not go into the details about how they work. For that, please see [Press88]. We use the routines almost exactly as listed in that reference.

SVD is based on a theorem from linear algebra: any mxn matrix **A** can be factored into three matrices **U**, **W**, and $\mathbf{V}^T$ :

$$\underset{m\times n}{\mathbf{A}} = \underset{m\times n}{\mathbf{U}} \quad \underset{n\times n}{\mathbf{W}} \quad \underset{n\times n}{\mathbf{V}^T} \tag{11}$$

where **W** is a diagonal matrix

$$\mathbf{W} = \begin{bmatrix} w_1 & & \\ & \ddots & \\ & & w_n \end{bmatrix}$$

and the columns of **U** and $\mathbf{V}^T$ are orthonormal; that is, they satisfy the following relationship:

$$\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}$$

The diagonal entries $w_1, w_2, \ldots w_n$ of matrix **W** tell how *singular* **A** is. Small values $w_j$ mean that **A** is ill-conditioned. We can measure this by computing the *condition number* of the matrix, which is defined as:

$$\text{condition number} = \frac{\underset{\text{all } j}{\text{MAX}}\, w_j}{\underset{\text{all } j}{\text{MIN}}\, w_j}$$

If the condition number is too large, we may not be able to invert the matrix. This limit is determined by the precision of the floating-point numbers. For single precision, we usually check for the following:

$$\frac{1}{\text{condition number}} \leq 10^{-6}$$

When this equation is satisfied, it states that the contributions from the smallest $w_j$ are so small that their values are corrupted by round-off errors, due to the limited precision of our floating-point numbers. They may end up "pulling" our solution far away from any reasonable answer. Under this situation, you can often get a better approximate solution by setting the smaller $w_j$ to zero, removing their contribution to the solution and relying only on the $w_j$ that are considered valid. We set a $w_j$ to zero if it meets the following criterion:

$$\text{set } w_j \text{ to zero if } w_j \leq \left(\underset{\text{all } i}{\text{MAX}} w_i\right) * 10^{-6}$$

After zeroing some of the $w_j$'s, we get a new matrix **W'** that in turn generates a new matrix **A'** by equation (11). Then for the standard linear system

$$\mathbf{A'\ x} = \mathbf{b}$$

we can solve for **x** directly, or alternately use a backsubstitution routine listed in [Press88].

To apply these routines to our particular problem, we note that our linear system (8) can be rewritten as:

$$\partial\mathbf{G} * \mathbf{D} = -\mathbf{G}_0$$

so $\partial\mathbf{G}$ plays the role of matrix **A**, to which we apply SVD.