Autocalibration for Virtual Environments Tracking Hardware*

Stefan Gottschalk
Computer Science Department
University of North Carolina
Chapel Hill, NC
gottscha@cs.unc.edu

John F. Hughes
Computer Science Department
Brown University
Providence, RI
jfh@cs.brown.edu

Abstract

We describe two instances in which precise mechanical calibration of virtual environments equipment has been replaced by automated algorithmic calibration through software that encapsulates the hardware design and uses a goal-based approach to adjust calibration parameters. We describe a back-projection system for adjusting the assumed locations of beacons in a head-mounted display tracking system; the calculated errors in the navigation system are used to compute adjustments to the beacon positions to reduce such errors. In a second application, a piggyback head-tracking/hand-tracking system is calibrated by a similar reduction of computed errors.

CR Categories: I.3.m [Computer Graphics]: Miscellaneous; I.3.7 [Computer Graphics]: 3-dimensional Graphics and Realism — Virtual Reality; I.4.8 [Image Processing] Scene Analysis — Photometry

Additional Keywords: Virtual environments, tracking, autocalibration.

1 Introduction

A number of calibration issues for virtual environments (VE) hardware are approached with standard engineering techniques in which the accuracy of the calibration is directly dependent on the accuracy of the assemblies in the VE machinery. This approach is successful to a degree but has several drawbacks. First, it makes the machinery very sensitive to rough handling. Second, frequent realignment may be required, which may be time-consuming and may be necessary so frequently that extended use of the equipment becomes impossible. Third, modifications of the machinery become very difficult.

We therefore take a *goal-based* approach to these problems, applying methods learned in computer graphics to solve engineering problems. Instead of requiring precise calibration of parts, we ask the systems to autocalibrate, a notion that was inspired in part by the auto-assembling systems of Barzel and Barr [BB88] but which first appeared in Wang's dissertation [Wan90]. This allows us to write a program encoding the *design* of the system that uses the system's observations to adjust itself. Since realignment can sometimes actually be done while the machinery is in use, rather than in a separate calibration phase, the first and second problems above are reduced. And because the software that implements the autocalibration encodes the intent of the design, the mechanical design can be modified in parallel with software modification, helping to reduce the third problem. In this paper, we discuss two sample applications: calibration of a head-tracking system and of a piggyback hand tracker attached to the head-tracking unit.

We stress that the techniques here serve the general goal of head-tracking. The current interest in Virtual Reality, evidenced by the attention it has attracted in both the technical literature and the media, may well have led to unjustified expectations. There is a belief that "any day now" the technology will become available. But there are three substantial obstacles: (1) for comfort, the units need small, high-resolution displays; (2) graphics hardware must be capable of real-time, low-latency image generation; (3) a low-latency, high-accuracy system for head tracking in unprepared, possibly-noisy environments is necessary. We are addressing the third of these issues. There is as yet no tracking system that is lightweight and works in unprepared environments and in large spaces. As far as we know, no one has demonstrated a working head-tracking system for a room-sized environment (about 15' x 15'). The ceiling tracker described here is a start: the environment is large and expandable and the equipment, although heavy, is bearable. We envision an eventual system in which methods similar to those described here are used to calibrate the system's view of its environment. The algorithms may differ, but the principle — having the system model its sources of error and calibrate itself against them — will remain.

We wish to make one more point: the two examples presented in this paper give details of a general principle, and this general principle is applicable to cases other than the ones we describe. In short, as one designs a tracker (or other electro-mechanical assembly), one has the opportunity to leave some physical parameters fixed but unknown, and to then determine their exact values after construction. Doing this kind of post-construction calibration does, however, require that some aspects of the system be overdetermined. In the head-tracker example below, we could not have performed autocalibration if the tracker computed its position from just three LED beacons, since there would be no "error measure" as we computed the position — the equations would be exactly determined rather than overdetermined. Similarly, without multiple samples in the hand-tracking application, we could not determine the orientation matrix. So the principle is this: if one wishes to use autocalibration, the system

* This work was sponsored in part by grants from NSF, DARPA, IBM, NCR, Sun Microsystems, DEC, and HP.

must have a surplus of information and a way to measure whether this information is internally consistent. The cost of obtaining this surplus of information is a design tradeoff, and should be considered during the design rather than after.

2 Operation of the Ceiling Tracker

Most current head trackers achieve a large working volume at the expense of accuracy and precision. However, some virtual environments applications require a large working volume and some minimum tracking precision.

A team at UNC-CH has developed an optoelectronic tracking system capable of tracking head motion with precision of approximately 0.2 degrees orientation and 1 mm translation. (A description of this system and its design can be found in the references [WAB+92][WAB+90]). System accuracy has not been measured precisely, but has been found to be very adequate for the purposes of head-mounted display (HMD) applications. At present, the working volume is a 10' by 12' area, but the tracking area can in principle be expanded arbitrarily by adding LED-studded ceiling panels.

The method used by the optoelectronic tracker is conceptually similar to celestial navigation. A mariner observes the angles between some number of stars and the horizon, and then, knowing the stars' locations in the heavens, determines the vessel's position. Similarly, we observe a number of ceiling-mounted infrared LEDs, and knowing their positions, we compute the location (and orientation) of the head-tracking unit.

To be more precise, we have a helmet with cameras mounted on it. Some of the ceiling LEDs are rapidly flashed in a known sequence, and each one is possibly sighted by a camera. The choice of subset and sequence is not preset, but is determined "on the fly" as it is learned which LEDs are visible to which cameras. The cameras are lateral-effect photodiodes with lenses, and each can report the centroid of a spot of light that strikes its surface. The centroid's location is reported in image plane coordinates, x and y. We call these *photocoordinates* (see Figure 1).

The placements of the cameras on the helmet are known, as are the locations of the principal points of the lens systems and the placements of the photodiodes' image planes within the camera casings. Thus, when the camera reports the photocoordinates of LED image on its image plane, we can compute the line, in head space, along which the LED must lie. We call this line a *back-projection*, because it is the result of projecting the ray from the photodiode back through the lens system and outward.

Now, given several back-projections in head space, and given the true locations of the LEDs in world space, where must the head be in world space so as to cause the back-projections to pass through their respective LEDs? With three (sufficiently general) back-projections, an unique solution can be found. With more than three, we have an overdetermined system and we compute a best fit according to a least-squares criterion, using a method called "space-resection by collinearity" (abbreviated "CA" for "collinearity algorithm"). We briefly describe CA in Section 3.1; full details can be found elsewhere [AW91]. Several questions about this tracker design that are often raised are discussed in an Appendix.

3 Explaining the Problem

The current design uses an adjustable superstructure to support the ceiling panels. The adjustments are needed because any conceivable support structure would bend under the loading of the panels, giving an undesirable curvature to the ceiling's surface.

With the current design of 10' by 12' (30 2' by 2' panels), the leveling process requires about 90 minutes of operator time, with specialized equipment.

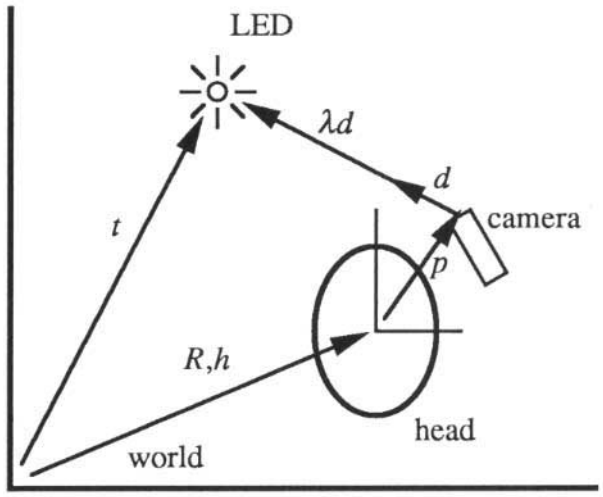

Figure 1: The geometry of the head-tracking system.

We plan to build another, larger ceiling without this superstructure. The panels will be of the same size, but will drop directly into the standard ceiling grid, replacing the acoustic tiles found in many buildings. This ceiling will be 18' by 30'; the expense of a comparable-size superstructure is prohibitive, and leveling time would be several hours.

Standard ceiling grids are by no means flat, and we have therefore developed the autocalibration technique described here to determine the location of the LEDs after the panels are installed. Before describing that technique, however, we give more details of the collinearity algorithm.

3.1 The Collinearity Algorithm

The collinearity algorithm (CA) works by observing many (typically 10 to 20) LEDs and then computing a best estimate of headmount position and orientation. When an LED shines onto a photodiode, the photodiode reports the centroid of the LED's image on its face. Since the algorithm knows the headmount geometry, it is able to compute, in head space, where the back-projection emerges and in what direction it is pointed. Somewhere along this back-projection lies the LED (see Figure 2).

Thus

$$R(p+\lambda d)+h=t, \quad \lambda>0 \tag{1}$$

where t the is location of the LED in world space, R is the matrix that takes vectors in head coordinates to world coordinates (i.e., R defines the *orientation* of the head-mount), h is the world-space coordinates of the origin of the head-mount coordinate system; and p and d are the basepoint and direction (unit vector) of the back-projection ray in head-coordinates; λ is the distance from the camera to the LED.

Equation 1 actually consists of three scalar equations, one for each of the x-, y-, and z-components. We can solve the z component for λ and substitute this into the x and y components. This eliminates λ and leaves us with two scalar equations in the unknowns R and h.

Many LEDs are seen at the same moment. Each of these generates two scalar equations. So each observation, which sights 12 to 20 LEDs, constructs a system of 24 to 40 equations in the unknowns R and h. CA seeks those values of R and h that minimize the residuals of these equations in the least-squares sense. These values are found by applying a multidimensional Newton's method.

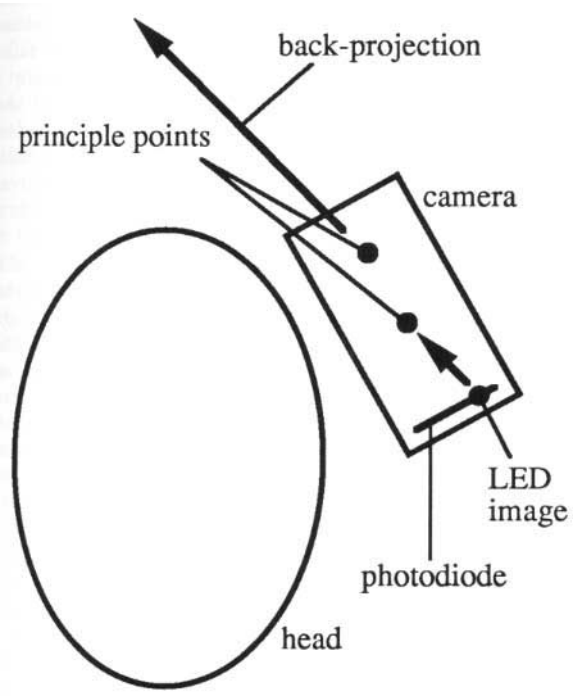

Figure 2: The back-projection ray from the camera towards the ceiling.

The method is most successful when given an initial guess very close to the optimal solution. In practice, this is easy to supply. The optical tracker typically provides updates every 12 to 20 milliseconds, and a person does not move far in that interval. Thus, for the initial guess, the algorithm merely uses the value of the previous update, which is guaranteed to be close.

4 Autocalibration: Rationale and Description

We have pointed out that it is very desirable to be able to construct the ceiling with loose tolerances, and be able to determine the locations of the LEDs afterward. CA does not depend upon any particular configuration of LED beacons — all places are alike to it. It does, however, require an exact knowledge of the locations of the LEDs, wherever they may be.

An "engineering" approach to achieving agreement between the physical geometry of the beacons and their software representation is prohibitively expensive. Therefore, we sought a way to determine the locations of the LEDs using existing hardware and some numerical processing.

The collinearity algorithm was derived from photgrammetric methods. Our LED calibration method, which makes use of CA as one step, was based primarily on influences from mathematics and computer graphics rather than the photgrammetry literature. We have since learned, however, that our approach has parallels in that literature, although we have found no exact analog. Nonethless, we strongly recommend that others working on optical tracking systems consult the photogrammetry literature [Sla80] for many ideas which, with slight modifications, may prove valuable in tracking.

We begin with an estimate of the beacon locations. We then take several thousand headmount observations (collecting 25,000 observations takes about 45 minutes) from a variety of positions, and use CA to "fit" the position of each observation to its beacon data. Of course, we know only approximately where the LEDs are, but fitting the headmount position to the beacon data allows some of the error in the beacon location estimates to cancel. The CA solution for the location of the headmount at each of these thousands of observations is likely to be rather bad: the sum of squares value will be large. To return to the marine analogy, it is as though the several circles of positions on the earth, each determined by a single star, failed to intersect at a single point, and instead intersected pairwise at several different points that surrounded a large region. The mariner estimates the vessel's position as somewhere at the center of the region, and begins to doubt the accuracy of the almanac's star locations.

After this initial set of observations, we derive the back-projections from each of these computed headmount locations, to yield "sightings" of the LEDs from roughly known positions. An LED sighted from several positions should be located at the intersection of the back-projections extending from those positions, but in general, the back-projections do not come together at a point, but tend instead to cluster in a particular region. We therefore adjust our estimate of each LED to be closer to this back-projection cluster. (The mariner, after several sets of inconsistent observations, decides to correct the almanac). This is the second step of our autocalibration.

After we adjust all the LEDs, the old observation positions are no longer optimal solutions in CA. So, we apply CA again to the observation positions, using the same data as before, but with the new beacon location estimates. (The mariner re-computes the vessel's position on each of the previous days, and now has circles of position that come closer to intersecting at single points). Thus we repeat the first step. We now continue, alternating between the two steps in this fashion, adjusting first one set of parameters and then the other, until we have settled to some configuration.

It seems surprising at first that this process converges at all; it is even more surprising to see how fast and how accurately it converges. We tested this by perturbing three of the ceiling panels as shown in Figure 4, and then running the algorithm. The average error-vector magnitudes for the first five full iterations were 13.1 mm, 4.7 mm, 3.2 mm, 2.5 mm, 2.2 mm, and 1.9 mm. After 20 iterations, which takes about two hours for 25,000 observations, the average error vector is down to 1.1 mm. Figure 5 is a computer-generated picture of the tracker ceiling. The beacons on the tilted panels are clearly visible.

The adjustment made to an LED's location depends on its relationship to the back-projections associated with it. A back-projection, in general, passes nearby the LED's estimated location. The vector drawn from the LED's estimated position to the back-projection's closest approach to that position is the *error vector* for that back-projection. A given LED has many back-projections, for each of which there is an associated error vector. We average these error vectors, and use this average as the adjustment to the LED's estimated position.

In a sense, each observation of an LED "votes" in the adjustment. An observation typically sees many LEDs, and cannot find a position from which to spear all its LEDs with its back-projections. The smallest adjustment possible for each LED that would completely satisfy an observation's collinearity conditions, would be an adjustment along the error vector. However, such an adjustment might conflict with the adjustment required by another observation.

The averaging is thus done as a compromise among the needs of the various observations that sight a given LED. It is possible to determine a new position for the LED that actually minimizes the sum of the squared lengths of the error vectors, but it is computationally expensive, and the averaging method works well and fast in practice.

4.1 Concerns About Noise: the Method in Practice

The autocalibration method was originally tried with simulated data so that it could be evaluated in the absence of noise and other complicating factors. It was found to be quite effective, providing

rapid convergence. Performance on real data was not nearly as good - for reasons we now discuss.

First, the photodiode readings are noisy. The photocoordinates have as much as 12 microns of uncertainty. If the LED is a meter away from the camera, which has a 50-mm lens, the back-projection will miss by more than .25 mm even if headmount's position and orientation are exactly correct.

Second, the system of equations produced by each observation assumes that the LEDs are sighted simultaneously, and this is not true in practice. The LEDs are sampled in sequence, and each sample may take as much as a millisecond. If the user's head is turning at the (reasonable) rate of 180 degrees per second, the LED is 1 m. away from the axis of rotation, and 20 LEDS are sampled for the observation, then in the 20 milliseconds of sampling, the back-projection to the first LED may have traveled 6 cm. This causes the system of equations given by the observation data to be inconsistent, so that it cannot be satisfied by any position and orientation. The fact that the equations cannot be satisfied implies that the back-projections are simply wrong, and hence will "pull on" the LEDs wherever the observation settles.

Third, acquiring the right spatial distribution of observations is surprisingly difficult. The LEDs in the corner of the ceiling are typically seen in many fewer observations than the ones in the center. And when the LEDs in the corner are seen, it tends to be from one direction. Naturally, an LED in the corner can be seen only from one octant: below ceiling height and beneath the ceiling. But diversity in the angles from which the LED is seen is helpful. If an LED is seen from within a narrow cone of positions, then the location of the back-projection cluster is more sensitive to the errors mentioned earlier: a slight distortion in the back-projections' placement tends to disperse the cluster, denying the LED a strong centering influence.

Three observations can be made about the first source of error. First, in addition to using superior photodiodes and electronics, the error can be reduced by using lenses of longer focal length. With longer focal lengths, the 12-micron error in the LED image location would translate to an even narrower error cone for the corresponding back-projection. The primary disadvantage of the resulting small fields of view is that they can slip between the LEDs and fail to see any at all. Second, one can allow the headmount to sit still, accumulating photocoordinates, and average them over time to distill a more accurate reading. Unfortunately, with thousands of observations required, data acquisition for calibration would be very time-consuming. Third and most important, however, sensor noise error is insignificant in comparison to the other two sources of error.

The second source of error comes from the motion of the headmount. Again, for calibration purposes, we could take data points only when the headmount is still. But this again would make data acquisition intolerably slow. In practice, we have found that moving the headmount slowly helps substantially in reducing this error. A better solution is to change the system of equations to take into account the headmount velocity, both linear and rotational. This would require a minimum of six LEDs per observation to obtain a fully-determined system, but typical counts are already 12 to 20 LEDs per observation. This is future work.

The third problem is being addressed by an graphics application that assists in data acquisition. A top view (map) of the ceiling is displayed on a nearby workstation, on which LEDs presently observed are marked. (This is needed because the LEDs emit infrared light, invisible to the naked eye.) The least-sampled LEDs are marked in a different color, allowing the operator to direct his efforts to sighting those LEDs. During the calibration process, in addition, certain LEDs are identified as having unusually large error vectors, meaning that their associated back-projections do not cluster tightly enough. A second run of data collection can be made, and special attention paid to these trouble spots.

In addition to the precautions and program assistance mentioned above, the calibration algorithm tests for high error vectors and culls out observations for which CA cannot find a satisfactory solution. (This is similar to computing robust statistics by eliminating outliers.) In this way, the algorithm is made somewhat more tolerant of operator mistakes or wild readings from the sensors (which are very rare).

Two features of the automated calibration method have not yet mentioned. First, the ceiling tracker is in frequent use. We can simply collect the observations *during use* and use these in an off-line calibration computation, so that we can keep the tracking system aligned without downtime. At present the system does not need frequent recalibration, and we do separate calibration runs, allowing us to collect only "good" data (i.e., data taken with slow head motion). Second, the entire algorithm is subject to a kind of systematic error: if we apply a rigid motion to our estimates of the beacon locations, CA converges exactly as well as before. This means that if one wishes to calibrate the system in absolute coordinates (relative to some frame of reference for the room in which the ceiling tracker sits), one may have to apply a rigid motion to the computed beacon positions so taht the estimated locations of a few key beacons are their actual positions as determined, for example, by measurements from the walls of the room.

5 Using a Headmounted Magnetic Tracker for Handtracking

Although the optical tracker gives satisfactory accuracy over a large working volume, its design does not lend itself to hand tracking for several reasons: the bulkiness of the cameras, the geometry of the situation (the user's body may obscure the hand's "view" of the ceiling, and the hand may not be held upright), and the dynamic range requirements on photodiode sensitivity (because of changing distances from the ceiling). We have found, however, that magnetic trackers [RBJ79] usually provide satisfactory performance within a small tracking volume, although in our environment they report significantly distorted position and orientation outside of a range of about five feet. Since one's hands never get farther than a few feet from one's head, we decided to place a magnetic source on the headmount and track hand motion from there.

Ultimately, however, we want to know the hand's location in the ceiling coordinate system. The optical tracker reports the head location in ceiling space, the magnetic source lies at some fixed location in head space, and the Polhemus tracking system reports the hand's location in source space. We compose the change-of-coordinate transformations among these three systems to get the hand's location in ceiling space.

Of course, the fixed location of the magnetic source within head space must be known before we can compose the transforms. As before, we have two choices: engineering, i.e., careful placement of the source on a precise rigid mount attached to the headframe, and autocalibration, in which we place the source approximately and then infer its position precisely using autocalibration. We chose the latter approach.

5.1 The Calibration Problem and Solution

We attach the magnetic source to the headmount with a rigid Plexiglas framework whose position is known within a few inches, and whose orientation is easy to measure within about 10 degrees. These are clearly not adequate measurements: if the hand is held 3' from the source, a 1 degree error in the measurement of the source's orientation would cause a 15 mm error in the computation of the hand's placement.

Our calibration approach is simple. We take simultaneous optical tracker and magnetic tracker readings, and use them to recover the placement of the source within head space. The

algorithm starts with a very approximate estimate of the source's placement, such as might be obtained by inspection.

We start by fixing the Polhemus sensor at some location in ceiling space. The exact location is not important — it need only stay still.

Now consider what *should* happen (if the system were calibrated properly) as the headmount moves about in the proximity of the sensor. We receive readings from the optical and magnetic trackers. The optical tracker produces the *Ceiling-from-Head* transform, and the magnetic tracker provides the *Source-from-Sensor* transform. If the *Head-from-Source* is correct, then the composition of these transforms, *Ceiling-from-Head* × *Head-from-Source* × *Source-from-Sensor*, should remain constant and should be the *Ceiling-from-Sensor* transform, which is constant (because the sensor is not moving) (see Figure 3).

If, for observation *i*, R_i is the reported *Ceiling-from-Head* transform, T_i is the reported *Source-from-Sensor* transform, S is the unknown but fixed *Head-from-Source* transform, and M is the unknown *Ceiling-from-Sensor* transform, then for any pair of reports from the trackers,

$$R_i S T_i = M,$$

provided the trackers are accurate. But if S is wrong, then as we walk around the room, the sensor's position and orientation (i.e., M), as computed by the transform composition, will drift, appearing to be in different places, depending on where we are standing.

After n readings from n different places, we have a system of n equations,

$$R_i S T_i = M_i, \quad i = 0 \ldots n-1,$$

where S is our (incorrect) estimate of *Head-from-Source*, and each M_i is computed as $R_i S T_i$. We seek the value of S that will make the M_i 's equal (i.e., the value of S that keeps our reports of the sensor positions and orientation constant).

Our estimate of S and the readings R_i and T_i give rise to many estimates of the sensor location M_i. We might get closer to the true value of M by taking some compromise among the M_i 's, say, by estimating that it is the average of the M_i 's. We actually bias this average slightly by averaging the matrix entries, and then performing the Gram-Schmidt process on the rotational part of the matrix. This averaging and orthonormalization step is likely to prompt objections, which we address below. For now, we continue with our description of the algorithm.

Let's call this resulting average transform Q. If we imagine that this is the correct value for the sensor location, then we can write the system

$$R_i S T_i = Q, \quad i = 0 \ldots n-1,$$

If Q really were the correct location, then we could take any one of the equations $R_i S T_i = Q$ and solve for S to recover that value, since the remaining transforms would be known. However, when we actually do this we find that we get different values for S. Why? Because Q is not correct — but it might be close. Solving for S gets us

$$S_i = R_i^{-1} Q T_i^{-1}, \quad i = 0 \ldots n-1,$$

each of which suggests a different value for S. We average these in exactly the way we did the M_i 's to arrive at a new estimate for S.

This completes one iteration of the algorithm. With our new estimate of S we go back and acquire new M_i 's, which we average to get Q, which we substitute back into the system so we can solve for the S_i's, which we average to get our new S. We iterate until the value of S stabilizes.

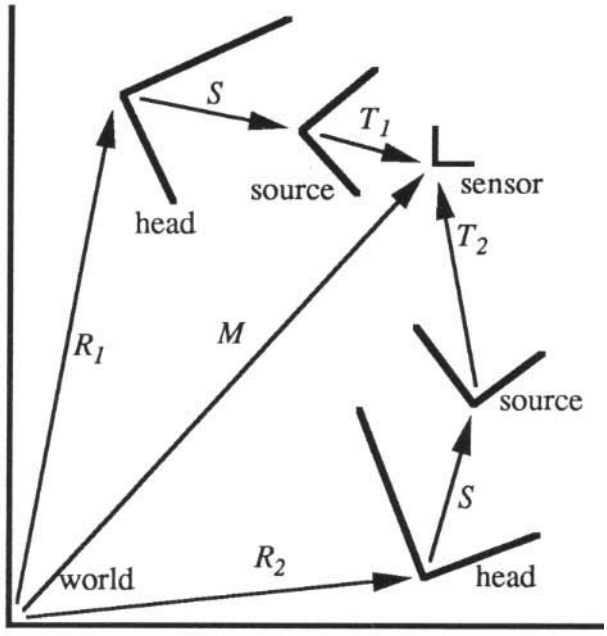

Figure 3: The geometry of the composite tracker during two different observations. The source is held fixed in the head-mounted-display coordinate system, and the sensor is fixed in the world coordinate system, but the relationship of the head to the world and of the sensor to the course change with each observation.

5.2 Justification for averaging matrices

Averaging makes sense for points in a linear space like a plane, but we are trying to use it in a nonlinear space (the set of 3 by 3 rotation matrices). But in general, the average of a set of points on a non-linear space like a sphere is almost always a point that is not on the sphere. Even so, if all the points are very close together on the sphere, this averaging yields a point that is near to a point on the sphere that one might call the "average." The reason is that the local geometry of the sphere is well approximated by any of the tangent planes within the local region, and so the sphere-based averaging is a close approximation to the tangent-plane averaging. Since the average of the sphere points does not lie on the sphere, however,to get a meaningful average we must project back onto the sphere. The critical properties of the projection map here are (1) it is continuous in a neighborhood of the sphere, and (2) for points already on the sphere, the projection is the identity. We now explain why the process we used in averaging matrices is analogous.

The set Q of 4 by 4 translation-and-rotation matrices is a subset of R^{16}; it is curved in much the same way that the sphere is a curved subset of R^3. We can average a collection of points on the object Q (i.e., several matrices), in much the same way as we averaged points on the sphere. Before this can make sense, though, we must honor the restriction that the points being averaged should be close to one another. And the same caveat applies: the R^{16}-average of a set of points in Q is not likely to lie in Q, and will need to be projected back to Q, which is what the Gram-Schmidt process does. Note, though, that the Gram-Schmidt process has the same properties as radial projection: it is a continuous function of the entries of the matrix (at least for matrices that are close to rotation matrices), and for a rotation matrix, the Gram-Schmidt process does nothing.

Still, there remains the question, "How small a region must the points be gathered in for averaging to make sense?" On the sphere, it certainly makes sense when all the points are contained in some hemisphere. For matrices, the averaging of the translational part is simply an average in a linear space, and needs no justification; for the rotational part, we believe (but have not proved formally) that the averaging process makes sense for any collection of (rotation) matrices A_i for which all the inner products $z_{ij} = trace(A_i A^t_j)$ are greater than 1/2. In practice, however, our matrices are all quite close to one another, and these inner products are large. Furthermore, the algorithm in practice is far more robust than we had expected. In a 2D simulation of the problem, for example, it takes some effort to give an initial estimate of the matrix S that makes the algorithm diverge.

5.3 Noise in the data, and the algorithm in practice

The accuracy of this method depends on the accuracy of the trackers providing the data. The optical and magnetic trackers, providing the R_is and T_i's are noisy. In general, no choice of *S* and *M* satisfies all the equations simultaneously. It is impossible to determine what the correct values are, and we can only hope to get an approximation to the correct *S*. Nonetheless, the error in the estimates of *S* and *M*, since they are based on multiple samples, should average out the random noise from the trackers. The *systematic* noise (e.g., one tracker always reports a slightly scaled *x*-coordinate) is not averaged out, but is also inherent in the system; if such systematic noise were too large, the system would be unusable in practice. Our experience is that the values of *S* and *M* converge quite rapidly to values that provide quite good hand-tracking.

There is one important observation about this instance of autocalibration: the sensor readings from which the calibration is done must be in fairly general position. In some cases, for example if the orientation of the headmount remains constant throughout the sampling process and the headmount is translated only along a single axis, then a little linear algebra shows that the estimates of *S* and *M* can all be identical but nonetheless be incorrect. But if the headmount is tilted and translated about all three axes during data gathering, and if multiple tilts and translations about each axis are included, then the equations will be sufficiently general to guarantee convergence (given a good enough initial estimate of *S*).

5.4 Remarks on the Method

One nice aspect of this method is that no exact measurements are needed. The location of the sensor somewhere in lab space may remain unknown. The position of the source in head space need only be estimated — and that is the only measurement necessary: the rest of the information is taken directly from the tracker sensors themselves.

The calibration procedure takes about 20 minutes in all: 5 minutes to put the sensor in place and gather data, and about 15 minutes (including graphical display of progress at each step) to settle on a value for *S*.

The number of equations and the "tightness" of the cluster of estimates can give a feel for the accuracy of the estimate. In averaging the S_i's, we can compute a residual for each, that is the magnitude of the deviation from the average S_i (deviation, here, being the difference in the translation components of the transforms). The angular deviation could be treated in precisely the same manner: the angle of rotation required to get from one transform's orientation to the other's. The root mean square of these residuals can be used as a reasonable metric for the "tightness" of the estimates of *S*. In a typical calibration run of 25 measurements, the RMS value of the deviations from the mean S_i was about 4.6 millimeters.

These residuals are not the same as the error in the result, although they are related. The more equations we use, the more likely the resulting transform is to be close to the actual one. This is somewhat like averaging a random variable — the variance can be very high, but the longer we average, the closer we are likely to get to the expected value.

6 Conclusion

We have described two applications of a goal-based approach to alignment of mechanical systems in VE tracking. In both cases, the automated calibration simplifies the construction of the systems, and makes it easier to modify the systems without extensive redesign of hardware or software. Note that the autocalibration system is designed to calibrate against a particular source of error, LED position error in the first case and Polhemus source location error in the second. Other sources of error in the system will confound the autocalibration process, so that if they are persistent enough, the autocalibration model should be revised to incorporate them as well. As the number of variables to be calibrated is increased, the number of observations must increase as well, of course, but in the head-tracking system, we have calibrated about 3000 variables successfully.

7 Acknowledgments

We would like to thank Al Barr for his initial involvement in the discussion of autocalibration, which helped to lead us away from the engineering approach and into the mathematical one. We also thank J.-F. Wang for having the idea of autocalibration for headtracking systems in the first place. Henry Fuchs' persistent demands for greater accuracy and bigger tracking spaces have provided a constant impetus. And we both owe a debt to our colleagues who have supported us in this project, particularly Ron Azuma and Russell Taylor.

References

[AW91] Ronald Azuma and Mark Ward. Space Resection by Collinearity: Mathematics behind the Optical Ceiling Head-Tracker. Technical Report TR91-048, UNC-Chapel Hill Department of Coputer Science, November 1991.

[BB88] Ronen Barzel and Alan H. Barr. A Modeling System Based on Dynamic Constraints. *Computer Graphics*, 22(4):179-188, August 1988.

[RBSJ79] F. H. Raab, E. B. Blood, T. O. Steiner, and H. R. Jones. Magnetic Position and Orientation Tracking System. *IEEE Transactions on Aerospace and Electronic Systems*, AES15(5):709-718, September 1979.

[Sla80] C. C. Slama, editor. *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, Va, fourth edition, 1980.

[WAB+90] J. F. Wang, R. Azuma, G. Bishop, V. Chi, J. Eyles, and H. Fuchs. Tracking a Head-Mounted Display in a Room-sized Environment with Head-Mounted Cameras. *Proc SPIE 1990 Technical Symposium on Optical Engineering and Photonics in Aerospace Sensing*, 1290, 1990.

[WAB+92] M. Ward. R. Azuma, R. Bennett, S. Gottschalk, and H. Fuchs. A Demonstrated Optical Tracker with Scalable Work Area for Head-Mounted Display

Systems. In *Proceedings of 1992 Symposium on Interactive 3D Graphics*, Cambridge, Mass., pages 43-52, March 1992.

[Wan90] Jih-Fang Wang. A Real-time Optical 6D Tracker for Head-mounted Display Systems. Technical Report TR90-011, UNC-Chapel Hill Department of Computer Science, March 1990.

Appendix: Head Tracker Design

Several issues concerning our current tracker design are often raised by those unfamiliar with it. First, why use exotic, expensive lateral-effect photodiodes instead of the highly developed, inexpensive CCD technologies?

The reason is timing. We want updates from the tracking system every 12 to 20 milliseconds. With CCDs, both the bandwidth required for data transfer and the image processing necessary per frame were prohibitive. We found it more feasible to digitize the voltages coming from the lateral-effect photodiodes (the only thing of interest after all in the image that a CCD camera would have seen) and transmit this comparatively low bandwidth signal.

Second, why use multiple cameras with narrow fields of view? Why not use a single camera with a wide-angle lens?

The problem here is the limited precision of the photocoordinates. We have observed that, in practice, the photocoordinates reported by the camera may be off by as much as 12 microns. A narrow field of view helps reduce this problem, but since CA requires disparate angles to operate effectively (otherwise the matrices involved tend to become ill-conditioned), this field-of-view requirement compels us to use multiple cameras.

Lastly, why put cameras on the head and LEDs on the ceiling, rather than vice versa, since the headmount would be much lighter with LEDs rather than cameras?

To explain our strategy, we call the cameras on the walls the "outside-looking-in" approach, and the cameras on the headmount the "inside-looking-out" approach. "Inside-looking-out" has three advantages over its counterpart: sensitivity to orientation, economical scalability, and energetics considerations.

Sensitivity to orientation is the ability to detect a head rotation. In the current system, a .5 degree turn of the head, for instance, causes a very significant change in the LEDs' coordinates on the photodiodes, regardless of their distances from the camera. By contrast, in the outside-looking-in approach this change in orientation would be almost imperceptible.

An *economically scalable system* is one in which the cost of increasing the working volume is low in terms of cost per unit tracking space. Because of the narrow field-of-view requirement on the cameras, the outside-looking-in approach (on a 30 ft^2 area) would need many cameras mounted on the walls. Covering the ceiling with LEDs is less expensive.

Energetics refers to how light energy is received from an LED. Quadrupling the distance between LED and camera, for instance, decreases the light energy received by a factor of 16. Furthermore, LEDs do not emit light uniformly in every direction: most of their power is emitted in the direction they face, and drops off with the angle away from their axis (depending on the packaging). If the cameras are wall-mounted, and the LEDs are head-mounted, then, as the user walks about, many LEDs may be oblique to the cameras, and the distances between user and cameras may vary a great deal. These two effects combine to make the range of signal strengths received by the cameras too wide.

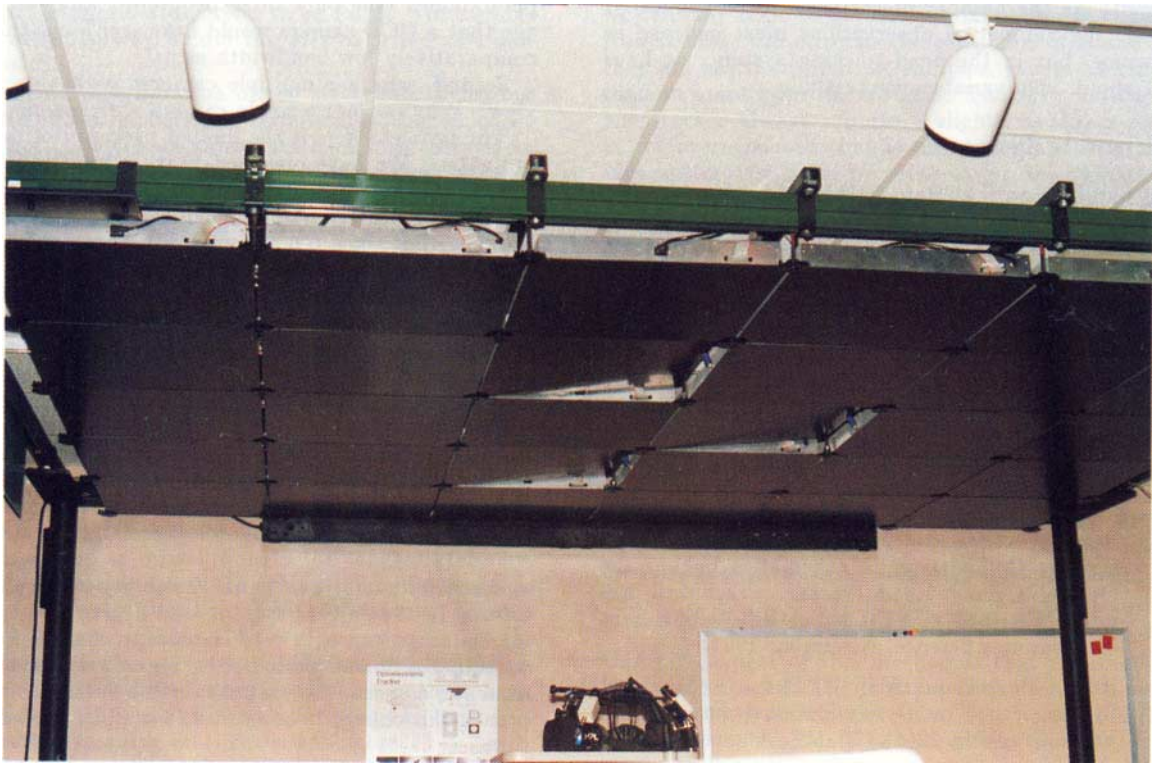

Figure 4: Tilting three panels in the ceiling to test autocalibration.

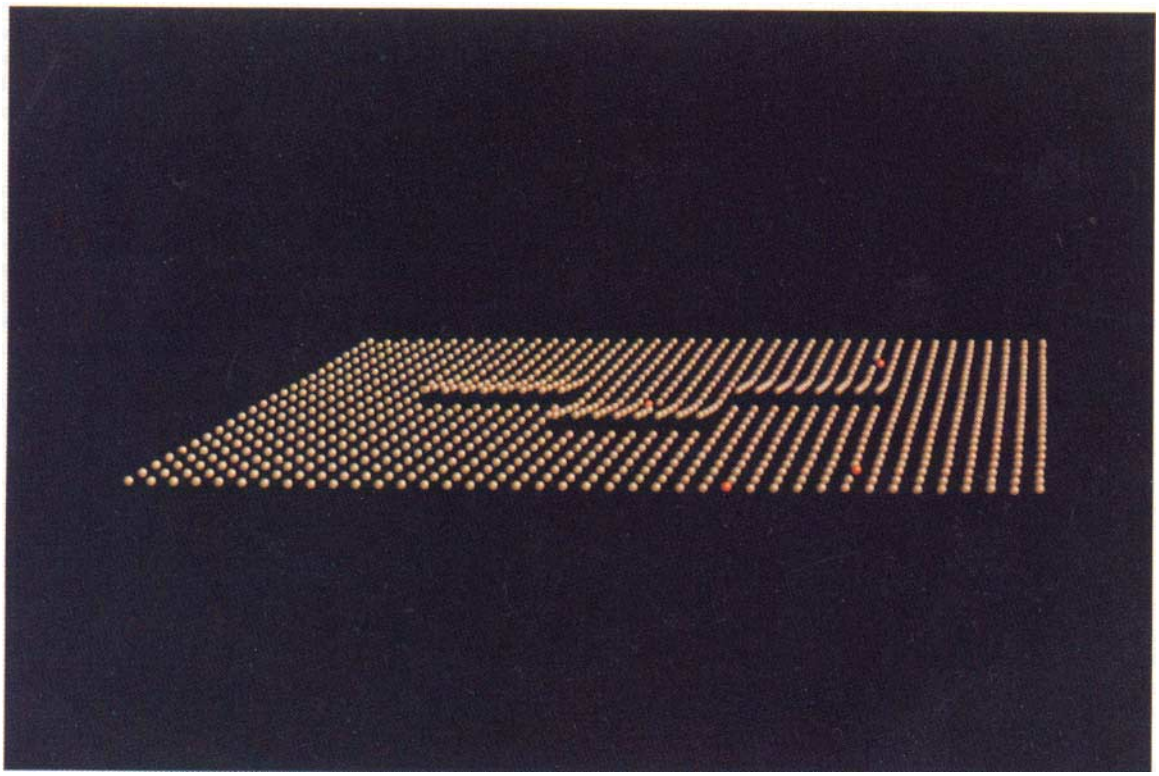

Figure 5: Computer display of calibrated beacon locations. The beacons shown in red were insufficiently sampled and could not be calibrated by the algorithm (see Section 4.1).