# SCAAT: Incremental Tracking with Incomplete Information

Greg Welch and Gary Bishop

University of North Carolina at Chapel Hill[†]

## Abstract

We present a promising new mathematical method for tracking a user's pose (position and orientation) for interactive computer graphics. The method, which is applicable to a wide variety of both commercial and experimental systems, improves accuracy by properly assimilating sequential observations, filtering sensor measurements, and by concurrently autocalibrating source and sensor devices. It facilitates user motion prediction, multisensor data fusion, and higher report rates with lower latency than previous methods.

Tracking systems determine the user's pose by measuring signals from low-level hardware sensors. For reasons of physics and economics, most systems make multiple sequential measurements which are then combined to produce a single tracker report. For example, commercial magnetic trackers using the SPASYN (*Space Synchro*) system sequentially measure three magnetic vectors and then combine them mathematically to produce a report of the sensor pose.

Our new approach produces tracker reports as each new low-level sensor measurement is made rather than waiting to form a complete collection of observations. Because single observations under-constrain the mathematical solution, we refer to our approach as single-constraint-at-a-time or SCAAT tracking. The key is that the single observations provide some information about the user's state, and thus can be used to incrementally improve a previous estimate. We recursively apply this principle, incorporating new sensor data as soon as it is measured. With this approach we are able to generate estimates more frequently, with less latency, and with improved accuracy. We present results from both an actual implementation, and from extensive simulations.

**CR Categories and Subject Descriptors**: I.3.7 [Computer Graphics] Three-Dimensional Graphics and Realism—Virtual reality; I.4.4 [Image Processing] Restoration—Kalman filtering; I.4.8 [Image Processing] Scene Analysis—Sensor fusion; G.0 [Mathematics of Computing] General—Numerical Analysis, Probability and Statistics, Mathematical Software.

**Additional Key Words and Phrases**: virtual environments tracking, feature tracking, calibration, autocalibration, delay, latency, sensor fusion, Kalman filter.

† CB 3175, Sitterson Hall, Chapel Hill, NC, 27599-3175
welch@cs.unc.edu, http://www.cs.unc.edu/~welch
gb@cs.unc.edu, http://www.cs.unc.edu/~gb

## 1 INTRODUCTION

The method we present requires, we believe, a fundamental change in the way people think about estimating a set of unknowns in general, and tracking for virtual environments in particular. Most of us have the preconceived notion that to estimate a set of unknowns we need as many constraints as there are degrees of freedom at any particular instant in time. What we present instead is a method to constrain the unknowns *over time*, continually refining an estimate for the solution, a *single constraint at a time*.

For applications in which the constraints are provided by real-time observations of physical devices, e.g. through measurements of sensors or visual sightings of landmarks, the SCAAT method isolates the effects of error in individual measurements. This isolation can provide improved filtering as well as the ability to individually calibrate the respective devices or landmarks concurrently and continually while tracking. The method facilitates user motion prediction, multisensor or multiple modality data fusion, and in systems where the constraints can only be determined sequentially, it provides estimates at a higher rate and with lower latency than multiple-constraint (batch) approaches.

With respect to tracking for virtual environments, we are currently using the SCAAT method with a new version of the UNC wide-area optoelectronic tracking system (section 4). The method could also be used by developers of commercial tracking systems to improve their existing systems or it could be employed by end-users to improve custom multiple modality hybrid systems. With respect to the more general problem of estimating a set of unknowns that are related by some set of mathematical constraints, one could use the method to trade estimate quality for computation time. For example one could incorporate individual constraints, one at a time, stopping when the uncertainty in the solution reached an acceptable level.

## 1.1 Incomplete Information

The idea that one might build a tracking system that generates a new estimate with each individual sensor measurement or *observation* is a very interesting one. After all, individual observations usually provide only partial information about a user’s complete state (pose), i.e. they are “incomplete” observations. For example, for a camera observing landmarks in a scene, only limited information is obtained from observations of any single landmark. In terms of control theory, a system designed to operate with only such incomplete measurements is characterized as *unobservable* because the user state cannot be observed (determined) from the measurements.

The notion of observability can also be described in terms of constraints on the unknown parameters of the system being estimated, e.g. constraints on the unknown elements of the system state. Given a particular system, and the corresponding set of unknowns that are to be estimated, let $C$ be defined as the minimal number of independent simultaneous constraints necessary to uniquely determine a solution, let $N$ be the number actually used to generate a new estimate, and let $N_{\text{ind}}$ be the number of *independent* constraints that can be formed from the $N$ constraints. For any $N \geq N_{\text{ind}}$ constraints, if $N_{\text{ind}} = C$ the problem is *well constrained*, if $N_{\text{ind}} > C$ it is *over constrained*, and if $N_{\text{ind}} < C$ it is *under-constrained.* (See Figure 1.)

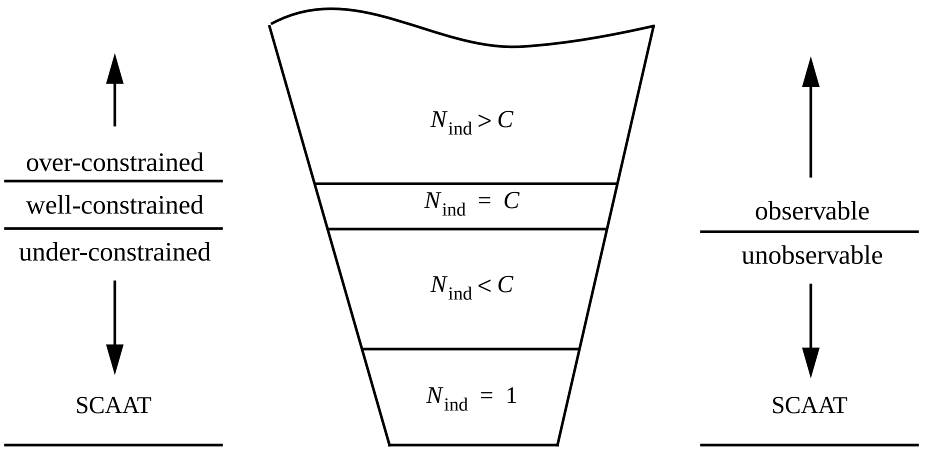


Figure 1: SCAAT and constraints on a system of simultaneous equations. $C$ is the minimal number of independent simultaneous constraints necessary to uniquely determine a solution, $N$ is the number of given constraints, and $N_{\text{ind}}$ is the number of *independent* constraints that can be formed from the $N$. (For most systems of interest $C > 1$). The conventional approach is to ensure $N \geq N_{\text{ind}}$ and $N_{\text{ind}} \geq C$, i.e. to use enough measurements to well-constrain or even over-constrain the estimate. The SCAAT approach is to employ the smallest number of constraints available at any one time, generally $N = N_{\text{ind}} = 1$ constraint. From this viewpoint, each SCAAT estimate is severely under-constrained.

## 1.2 Landmark Tracking

Consider for example a system in which a single camera is used to observe known scene points to determine the camera position and orientation. In this case, the constraints provided by the observations are multi-dimensional: 2D image coordinates of 3D scene points. Given the internal camera parameters, a set of four known coplanar scene points, and the corresponding image coordinates, the camera position and orientation can be uniquely determined in closed-form [16]. In other words if $N = C = 4$ constraints (2D image points) are used to estimate the camera position and orientation, the system is completely observable. On the other hand, if $N < C$ then there are multiple solutions. For example with only $N = 3$ non-collinear points, there are up to 4 solutions. Even worse, with $N = 2$ or $N = 1$ points, there are infinite combinations of position and orientation that could result in the same camera images.

In general, for closed-form tracking approaches, a well or over-constrained system with $N \geq C$ is observable, an under-constrained system with $N < C$ is not. Therefore, if the individual observations provide only partial information, i.e. the measurements provide insufficient constraints, then multiple devices or landmarks must be excited and (or) sensed prior to estimating a solution. Sometimes the necessary observations can be obtained simultaneously, and sometimes they can not. Magnetic trackers such as those made by Polhemus and Ascension perform three *sequential* source excitations, each in conjunction with a complete sensor unit observation. And while a camera can indeed observe multiple landmarks simultaneously in a single image, the image processing to identify and locate the individual landmarks must be done sequentially for a single CPU system. If the landmarks can move independently over time, for example if they are artificial marks placed on the skin of an ultrasound patient for the purpose of landmark-based tracking [41], batch processing of the landmarks can reduce the effectiveness of the system. A SCAAT implementation might grab an image, extract a *single* landmark, update the estimates of both the camera *and* landmark positions, and then throw-away the image. In this way estimates are generated faster and with the most recent landmark configurations.

## 1.3 Putting the Pieces Together

Given a tracker that uses multiple constraints that are each individually incomplete, a *measurement model* for any one of incomplete constraints would be characterized as *locally unobservable*. Such a system must incorporate a sufficient set of these incomplete constraints so that the resulting overall system is observable. The corresponding aggregate measurement model can then be characterized as *globally observable*. Global observability can be obtained over *space* or over *time*. The SCAAT method adopts the latter scheme, even in some cases where the former is possible.

## 2 MOTIVATION

## 2.1 The Simultaneity Assumption

Several well-known virtual environment tracking systems collect position and orientation constraints (sensor measurements) sequentially. For example, tracking systems developed by Polhemus and Ascension depend on sensing a sequence of variously polarized electromagnetic waves or fields. A system that facilitated simultaneous polarized excitations would be very difficult if not impossible to implement. Similarly both the original UNC optoelectronic tracking system and the newer HiBall version are designed to observe only one ceiling-mounted LED at a time. Based on the available literature [25,27,37] these systems currently assume (mathematically) that their sequential observations were collected simultaneously. We refer to this as the *simultaneity assumption*. If the target remains motionless this assumption introduces no error. However if the target is moving, the violation of the assumption introduces error.

To put things into perspective, consider that typical arm and wrist motion can occur in as little as 1/2 second, with typical “fast” wrist tangential motion occurring at 3 meters/second [1]. For the current versions of the above systems such motion corresponds to approximately 2 to 6 centimeters of translation *throughout* the sequence of measurements required for a single estimate. For systems that attempt sub-millimeter accuracies, even slow motion occurring during a sequence of sequential measurements impacts the accuracy of the estimates.

The error introduced by violation of the simultaneity assumption is of greatest concern perhaps when attempting any form of system *autocalibration*. Gottschalk and Hughes note that motion during their autocalibration procedure must be severely restricted in order to avoid such errors [19]. Consider that for a multiple-measurement system with 30 milliseconds total measurement time, motion would have to be restricted to approximately 1.5 centimeters/second to confine the translation (throughout a measurement sequence) to 0.5 millimeters. For complete autocalibration of a large (wide-area) tracking system, this restriction results in lengthy specialized sessions.

## 2.2 Device Isolation & Autocalibration

Knowledge about source and sensor imperfections can be used to improve the accuracy of tracking systems. While intrinsic sensor parameters can often be determined off-line, e.g. by the manufacturer, this is generally not the case for extrinsic parameters. For example it can be difficult to determine the exact geometric relationship between the various sensors of a hybrid system. Consider that the coordinate system of a magnetic sensor is located at some unknown location inside the sensor unit. Similarly the precise geometric relationship between visible landmarks used in a vision-based system is often difficult to determine. Even worse, landmark positions can change over time as, for example, a patient's skin deforms with pressure from an ultrasound probe. In general, goals such as flexibility, ease of use, and lower cost, make the notion of self-calibration or *autocalibration* attractive.

The general idea for autocalibration is not new. See for example [19,45]. However, because the SCAAT method *isolates* the measurements provided by each sensor or modality, the method provides a new and elegant means to autocalibrate concurrently while tracking. Because the SCAAT method isolates the individual measurements, or measurement dimensions, individual source and sensor imperfections are more easily identified and dealt with. Furthermore, because the simultaneity assumption is avoided, the motion restrictions discussed in section 2.1 would be removed, and autocalibration could be performed *while concurrently tracking a target*.

The isolation enforced by the SCAAT approach can improve results even if the constraints are obtained simultaneously through multidimensional measurements. An intuitive explanation is that if the elements (dimensions) are corrupted by independent noise, then incorporating the elements independently can offer improved filtering over a batch or ensemble estimation scheme.

## 2.3 Temporal Improvements

Per Shannon's sampling theorem [24] the measurement or *sampling* frequency should be at least twice the true target motion bandwidth, or an estimator may track an alias of the true motion. Given that common arm and head motion bandwidth specifications range from 2 to 20 Hz [13,14,36], the *sampling* rate should ideally be greater than 40 Hz. Furthermore, the *estimate* rate should be as high as possible so that normally-distributed white estimate error can be discriminated from any non-white error that might be observed during times of significant target dynamics, and so estimates will always reflect the most recent user motion.

In addition to increasing the estimate rate, we want to reduce the latency associated with generating an improved estimate, thus reducing the overall latency between target motion and visual feedback in virtual environment systems [34]. If too high, such latency can impair adaptation and the illusion of presence [22], and can cause motion discomfort or sickness. Increased latency also contributes to problems with head-mounted display registration [23] and with motion prediction [4,15,29]. Finally, post-rendering image deflection techniques are sometimes employed in an attempt to address latency variability in the rendering pipeline [32,39]. Such methods are most effective when they have access to (or generate) accurate motion predictions and low-latency tracker updates. With accurate prediction the best possible position and orientation information can be used to render a preliminary image. With fast tracker updates there is higher probability that when the preliminary image is ready for final deflection, recent user motion has been detected and incorporated into the deflection.

With these requirements in mind, let us examine the effect of the measurements on the estimate latency and rate. Let $t_m$ be the time needed to determine one constraint, e.g. to measure a sensor or extract a scene landmark, let $N$ be the number of (sequential) constraints used to compute a complete estimate, and let $t_c$ be the time needed to actually compute that estimate. Then the estimate latency $t_e$ and rate $r_e$ are

$$\begin{aligned} t_e &= Nt_m + t_c \ , \\ r_e &= \frac{1}{t_e} = \frac{1}{Nt_m + t_c} \ . \end{aligned} \qquad (1)$$

As the number of constraints $N$ increases, equation (1) shows how the estimate latency and rate increase and decrease respectively. For example the Polhemus Fastrak, which uses the SPASYN (*Space Synchro*) method for determining relative position and orientation, employs $N = 3$ sequential electromagnetic excitations and measurements per estimate [25,27,37], the original University of North Carolina (UNC) optoelectronic tracking system sequentially observed $10 \leq N \leq 20$ beacons per estimate [3,44], and the current UNC hybrid landmark-magnetic tracking system extracts (from a camera image) and then incorporates $N = 4$ landmarks per update. The SCAAT method seeks to improve the latencies and data rates of such systems by updating the current estimate with each new (individual) constraint, i.e. by fixing $N$ at 1. In other words, it increases the estimate rate to approximately the rate that individual constraints can be obtained and likewise decreases the estimate latency to approximately the time required to obtain a single constraint, e.g. to perform a single measurement of a single sensor, or to extract a single landmark.

Figure 2 illustrates the increased data rate with a timing diagram that compares the SPASYN (Polhemus Navigation Systems) magnetic position and orientation tracking system with a hypothetical SCAAT implementation. In contrast to the SPASYN system, a SCAAT implementation would generate a new estimate after sensing each *individual* excitation vector rather than waiting for a complete pattern.

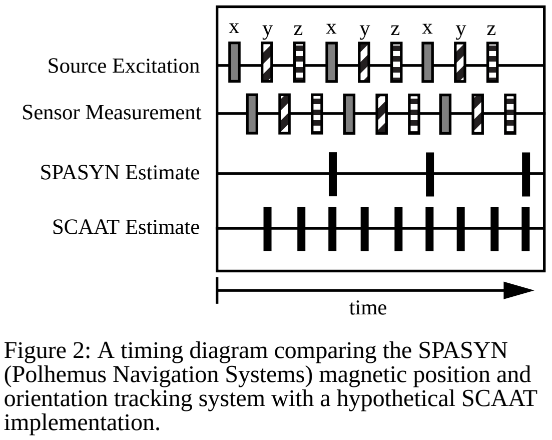


Figure 2: A timing diagram comparing the SPASYN (Polhemus Navigation Systems) magnetic position and orientation tracking system with a hypothetical SCAAT implementation.

## 2.4 Data Fusion & Hybrid Systems

The Kalman filter [26] has been widely used for data fusion. For example in navigation systems [17,30], virtual environment tracking systems [5,12,14], and in 3D scene modeling [20,42]. However the SCAAT method represents a new approach to Kalman filter based *multi-sensor data fusion*. Because constraints are intentionally incorporated one at a time, one can pick and choose which ones to add, and when to add them. This means that information from different sensors or modalities can be woven together in a common, flexible, and expeditious fashion. Furthermore, one can use the approach to ensure that each estimate is computed from the most recently obtained constraint.

Consider for a moment the UNC hybrid landmark-magnetic presented at SIGGRAPH 96 [41]. This system uses an off-the-shelf Ascension magnetic tracking system along with a vision-based landmark recognition system to achieve superior synthetic and real image registration for augmented reality assisted medical procedures. The vision-based component attempts to identify and locate multiple known landmarks in a single image before applying a correction to the magnetic readings. A SCAAT implementation would instead identify and locate only one landmark per update, using a new image (frame) each time. Not only would this approach increase the frequency of landmark-based correction (given the necessary image processing) but it would offer the added benefit that unlike the implementation presented in [41], no special processing would be needed for the cases where the number of visible landmarks falls below the number $C$ necessary to determine a complete position and orientation solution. The SCAAT implementation would simply cycle through any available landmarks, one at a time. Even with only one visible landmark the method would continue to operate as usual, using the information provided by the landmark sighting to refine the estimate where possible, while increasing the uncertainty where not.

## 3 METHOD

The SCAAT method employs a *Kalman filter* (KF) in an unusual fashion. The Kalman filter is a mathematical procedure that provides an efficient computational (recursive) method for the least-squares estimation of a linear system. It does so in a *predictor-corrector* fashion, predicting short-term (since the last estimate) changes in the state using a *dynamic model*, and then correcting them with a measurement and a corresponding *measurement model*. The *extended* Kalman filter (EKF) is a variation of the Kalman filter that supports estimation of *nonlinear* systems, e.g. 3D position and orientation tracking systems. A basic introduction to the Kalman filter can be found in Chapter 1 of [31], while a more complete introductory discussion can be found in [40], which also contains some interesting historical narrative. More extensive references can be found in [7,18,24,28,31,46].

The Kalman filter has been employed previously for virtual environment tracking estimation and prediction. For example see [2,5,12,14,42], and most recently [32]. In each of these cases however the filter was applied directly and only to the 6D pose estimates delivered by the off-the-shelf tracker. The SCAAT approach could be applied to either a hybrid system using off-the-shelf and/or custom trackers, or it could be employed by tracker developers to improve the existing systems for the end-user graphics community.

In this section we describe the method in a manner that does not imply a specific tracking system. (In section 3.4 we present experimental results of a specific implementation, a SCAAT wide-area optoelectronic tracking system.) In section 3.1 we describe the method for tracking, and in section 3.2 we describe one possible method for concurrent autocalibration.

Throughout we use the following conventions.

| | | |
|---|---|---|
| $x$ | = | scalar (lower case) |
| $\vec{x}$ | = | general vector (lower case, arrow) indexed as $\vec{x}[r]$ |
| $\hat{x}$ | = | filter estimate vector (lower case, hat) |
| $A$ | = | matrix (capital letters) indexed as $A[r, c]$ |
| $A^{-1}$ | = | matrix inverse |
| $I$ | = | the identity matrix |
| $\beta^{-}$ | = | matrix/vector *prediction* (super minus) |
| $\beta^{T}$ | = | matrix/vector transpose (super T) |
| $\alpha_i$ | = | matrix/vector/scalar identifier (subscript) |
| $E\{\bullet\}$ | = | mathematical expectation |

## 3.1 Tracking

### 3.1.1 Main Tracker Filter

The use of a Kalman filter requires a mathematical (state-space) model for the dynamics of the process to be estimated, the target motion in this case. While several possible dynamic models and associated state configurations are possible, we have found a simple *position-velocity* model to suffice for the dynamics of our applications. In fact we use this same form of model, with different parameters, for all six of the position and orientation components $(x, y, z, \phi, \theta, \psi)$. Discussion of some other potential models and the associated trade-offs can be found in [7] pp. 415-420. Because our implementation is discrete with inter sample time $\delta t$ we model the target's dynamic motion with the following linear difference equation:

$$\vec{x}(t + \delta t) = A(\delta t)\vec{x}(t) + \vec{w}(\delta t) . \qquad (2)$$

In the standard model corresponding to equation (2), the $n$ dimensional Kalman filter *state vector* $\vec{x}(t)$ would completely describe the target position and orientation at any time $t$. In practice we use a method similar to [2,6] and maintain the complete target orientation externally to the Kalman filter in order to avoid the nonlinearities associated with orientation computations. In the internal state vector $\vec{x}(t)$ we maintain the target position as the Cartesian coordinates $(x, y, z)$, and the *incremental* orientation as small rotations $(\phi, \theta, \psi)$ about the $(x, y, z)$ axis. Externally we maintain the target orientation as the *external quaternion* $\vec{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z))$. (See [9] for discussion of quaternions.) At each filter update step, the incremental orientations $(\phi, \theta, \psi)$ are factored into the external quaternion $\vec{\alpha}$, and then zeroed as shown below. Thus the incremental orientations are linearized for the EKF, centered about zero. We maintain the derivatives of the target position and orientation internally, in the state vector $\vec{x}(t)$. We maintain the angular velocities internally because the angular velocities behave like orthogonal vectors and do not exhibit the nonlinearities of the angles themselves. The target state is then represented by the $n = 12$ element internal state vector

$$\vec{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \qquad (3)$$

and the four-element external orientation quaternion

$$\vec{\alpha} = (\alpha_w, (\alpha_x, \alpha_y, \alpha_z)) , \qquad (4)$$

where the time designations have been omitted for clarity.

The $n \times n$ *state transition matrix* $A(\delta t)$ in (2) projects the state forward from time $t$ to time $t + \delta t$. For our linear model, the matrix implements the relationships

$$\begin{aligned} x(t+\delta t) &= x(t) + \dot{x}(t)\delta t \\ \dot{x}(t+\delta t) &= \dot{x}(t) \end{aligned} \tag{5}$$

and likewise for the remaining elements of (3).

The $n \times 1$ *process noise vector* $\vec{w}(\delta t)$ in (2) is a normally-distributed zero-mean sequence that represents the uncertainty in the target state over any time interval $\delta t$. The corresponding $n \times n$ *process noise covariance matrix* is given by

$$E\{\vec{w}(\delta t)\vec{w}^T(\delta t + \varepsilon)\} = \begin{cases} Q(\delta t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \tag{6}$$

Because our implementation is discrete with inter sample time $\delta t$, we can use the transfer function method illustrated by [7] pp. 221-222 to compute a *sampled* process noise covariance matrix. (Because the associated random processes are presumed to be time stationary, we present the process noise covariance matrix as a function of the inter-sample duration $\delta t$ only.) The non-zero elements of $Q(\delta t)$ are given by

$$\begin{aligned} Q(\delta t)[i,i] &= \vec{\eta}[i]\frac{(\delta t)^3}{3} \\ Q(\delta t)[i,j] = Q(\delta t)[j,i] &= \vec{\eta}[i]\frac{(\delta t)^2}{2} \\ Q(\delta t)[j,j] &= \vec{\eta}[i](\delta t) \end{aligned} \tag{7}$$

for each pair

$$(i,j) \in \{(x,\dot{x}), (y,\dot{y}), (z,\dot{z}), (\phi,\dot{\phi}), (\theta,\dot{\theta}), (\psi,\dot{\psi})\}.$$

The $\vec{\eta}[i]$ in (7) are the *correlation kernels* of the (assumed constant) noise sources presumed to be driving the dynamic model. We determined a set of values using Powell's method, and then used these in both simulation and our real implementation. The values can be "tuned" for different dynamics, though we have found that the tracker works well over a broad range of values.

The use of a Kalman filter requires not only a dynamic model as described above, but also a *measurement model* for each available type of measurement. The measurement model is used to predict the ideal noise-free response of each sensor and source pair, given the filter's current estimate of the target state as in equations (3) and (4).

*It is the nature of the measurement models and indeed the actual sensor measurements that distinguishes a SCAAT Kalman filter from a well-constrained one.*

For each sensor *type* $\sigma$ we define the $m_\sigma \times 1$ *measurement vector* $\vec{z}_\sigma(t)$ and corresponding *measurement function* $\vec{h}_\sigma(\bullet)$ such that

$$\vec{z}_{\sigma,t} = \vec{h}_\sigma(\vec{x}(t), \vec{b}_t, \vec{c}_t) + \vec{v}_\sigma(t). \tag{8}$$

Note that in the "purest" SCAAT implementation $m_\sigma = 1$ and the measurements are incorporated as single scalar values. However if it is not possible or necessary to isolate the measurements, e.g. to perform autocalibration, then multi-dimensional measurements can be incorporated also. Guidelines presented in [47] lead to the following heuristic for choosing the SCAAT Kalman filter measurement elements (constraints):

*During each SCAAT Kalman filter measurement update one should observe a single sensor and source pair only.*

For example, to incorporate magnetic tracker data as an end-user, $m_\sigma = 7$ for the three position and four orientation (quaternion) elements, while if the manufacturer were to use the SCAAT implementation, $m_\sigma = 3$ for each 3-axis electromagnetic response to a single excitation. For an image-based landmark tracker such as [41] the measurement function would, given estimates of the camera pose and a single landmark location, transform the landmark into camera space and then project it onto the camera image plane. In this case $m_\sigma = 2$ for the 2D image coordinates of the landmark.

The $m_\sigma \times 1$ *measurement noise vector* $\vec{v}_\sigma(t)$ in (8) is a normally-distributed zero-mean sequence that represents any random error (e.g. electrical noise) in the measurement. This parameter can be determined from component design specifications, and (or) confirmed by off-line measurement. For our simulations we did both. The corresponding $m_\sigma \times m_\sigma$ *measurement noise covariance matrix* is given by

$$E\{\vec{v}_\sigma(t)\vec{v}_\sigma^T(t+\varepsilon)\} = \begin{cases} R_\sigma(t), & \varepsilon = 0 \\ 0, & \varepsilon \neq 0 \end{cases}. \tag{9}$$

For each measurement function $\vec{h}_\sigma(\bullet)$ we determine the corresponding Jacobian function

$$H_\sigma(\vec{x}(t), \vec{b}_t, \vec{c}_t)[i,j] \equiv \frac{\partial}{\partial \vec{x}[j]}\vec{h}_\sigma(\vec{x}(t), \vec{b}_t, \vec{c}_t)[i], \tag{10}$$

where $1 \le i \le m_\sigma$ and $1 \le j \le n$. Finally, we note the use of the standard (Kalman filter) $n \times n$ *error covariance matrix* $P(t)$ which maintains the covariance of the error in the estimated state.

### 3.1.2 Tracking Algorithm

Given an initial state estimate $\hat{x}(0)$ and error covariance estimate $P(0)$, the SCAAT algorithm proceeds similarly to a conventional EKF, cycling through the following steps whenever a discrete measurement $\vec{z}_{\sigma,t}$ from some sensor (type $\sigma$) and source becomes available at time $t$:

a. Compute the time $\delta t$ since the previous estimate.

b. Predict the state and error covariance.

$$\begin{aligned} \hat{x}^- &= A(\delta t)\hat{x}(t-\delta t) \\ P^- &= A(\delta t)P(t-\delta t)A^T(\delta t) + Q(\delta t) \end{aligned} \tag{11}$$

c. Predict the measurement and compute the corresponding Jacobian.

$$\begin{aligned} \hat{z} &= \vec{h}_\sigma(\hat{x}^-, \vec{b}_t, \vec{c}_t) \\ H &= H_\sigma(\hat{x}^-, \vec{b}_t, \vec{c}_t) \end{aligned} \tag{12}$$

d. Compute the *Kalman gain*.

$$K = P^- H^T (H P^- H^T + R_\sigma(t))^{-1} \tag{13}$$

e. Compute the *residual* between the actual sensor measurement $\vec{z}_{\sigma,t}$ and the predicted measurement from (12).

$$\overrightarrow{\Delta z} = \vec{z}_{\sigma,t} - \hat{z} \tag{14}$$

f. Correct the predicted tracker state estimate and error covariance from (11).

$$\begin{aligned} \hat{x}(t) &= \hat{x}^- + K\overrightarrow{\Delta z} \\ P(t) &= (I - KH)P^- \end{aligned} \tag{15}$$

g. Update the external orientation of equation (4) per the change indicated by the $(\phi, \theta, \psi)$ elements of the state.*

$$\begin{aligned}\Delta\hat{\alpha} &= \text{quaternion}(\hat{x}[\phi], \hat{x}[\theta], \hat{x}[\psi]) \\ \hat{\alpha} &= \hat{\alpha} \otimes \Delta\hat{\alpha}\end{aligned} \tag{16}$$

h. Zero the orientation elements of the state vector.

$$\hat{x}[\phi] = \hat{x}[\theta] = \hat{x}[\psi] = 0 \tag{17}$$

The equations (11)-(17) may seem computationally complex, however they can be performed quite efficiently. The computations can be optimized to eliminate operations on matrix and vector elements that are known to be zero. For example, the elements of the Jacobian $H$ in (12) that correspond to the velocities in the state $\hat{x}(t)$ will always be zero. In addition, the matrix inverted in the computation of $K$ in (13) is of rank $m_\sigma$ ($2 \times 2$ for our example in section 3.4) which is smaller for a SCAAT filter than for a corresponding conventional EKF implementation. Finally, the increased data rate allows the use of the *small angle approximations* $\sin(\theta) = \theta$ and $\cos(\theta) = 1$ in $\vec{h}_\sigma(\bullet)$ and $H_\sigma(\bullet)$. The total *per estimate* computation time can therefore actually be less than that of a corresponding conventional implementation. (We are able to execute the SCAAT filter computations, with the autocalibration computations discussed in the next section, in approximately $100 \mu s$ on a 200 MHz PC-compatible computer.)

### 3.1.3 Discussion

The key to the SCAAT method is the number of constraints provided by the measurement vector and measurement function in equation (8). For the 3D-tracking problem being solved, a unique solution requires $C = 6$ non-degenerate constraints to resolve six degrees of freedom. Because individual sensor measurements typically provide less than six constraints, conventional implementations usually construct a complete measurement vector

$$\vec{z}_t = \left[\vec{z}^T_{\sigma_1, t_1} \ \ldots \ \vec{z}^T_{\sigma_N, t_N}\right]^T$$

from some group of $N \geq C$ individual sensor measurements over time $t_1 \ldots t_N$, and *then* proceed to compute an estimate. Or a particular implementation may operate in a *moving-window* fashion, combining the most recent measurement with the $N-1$ previous measurements, possibly implementing a form of a finite-impulse-response filter. In any case, for such well-constrained systems complete observability is obtained at each step of the filter. Systems that collect measurements sequentially in this way inherently violate the simultaneity assumption, as well as increase the time $\delta t$ between estimates.

In contrast, the SCAAT method blends individual measurements that each provide incomplete constraints into a complete state estimate. The EKF inherently provides the means for this blending, no matter how complete the information content of each individual measurement $\vec{z}_{\sigma, t}$. The EKF accomplishes this through the Kalman gain $K$ which is computed in (13). The Kalman gain, which is used to adjust the state and the error covariance in (15), is optimal in the sense that it minimizes the error covariance if certain conditions are met. Note that the inversion in (13) forms a ratio that reflects the relative uncertainties of the state and the measurement. Note too that the ratio is affected by the use of the measurement function Jacobian $H$. Because the Jacobian reflects the rate of change of each measurement with respect to the current state, it indicates a direction in state space along which a measurement could *possibly* affect the state. Because the gain is recomputed at each step with the appropriate measurement function and associated Jacobian, it inherently reflects the amount and direction of information provided by the individual constraint.

## 3.2 Autocalibration

The method we use for autocalibration involves *augmenting* the *main tracker filter* presented in section 3.1 to effectively implement a distinct *device filter*, a Kalman filter, for each source or sensor to be calibrated. (We use the word "device" here to include for example scene landmarks which can be thought of as passive sources, and cameras which are indeed sensors.) In general, any constant device-related parameters used by a measurement function $\vec{h}_\sigma(\bullet)$ from (8) are candidates for this autocalibration method. We assume that the parameters to be estimated are contained in the device parameter vectors $\vec{b}_t$ and $\vec{c}_t$, and we also present the case where both the source and sensor are to be calibrated since omission of one or the other is trivial. We note the following new convention.

$$\widehat{\alpha} = \text{augmented matrix/vector (wide hat)}$$

### 3.2.1 Device Filters

For each device (source, sensor, landmark, etc.) we create a distinct device filter as follows. Let $\vec{\pi}$ represent the corresponding device parameter vector and $n_\pi = \text{length}(\vec{\pi})$.

a. Allocate an $n_\pi \times 1$ state vector $\hat{x}_\pi$ for the device, initialize with the best *a priori* device parameter estimates, e.g. from design.

b. Allocate an $n_\pi \times n_\pi$ noise covariance matrix $Q_\pi(\delta t)$, initialize with the expected parameter variances.

c. Allocate an $n_\pi \times n_\pi$ error covariance matrix $P_\pi(t)$, initialize to indicate the level of confidence in the *a priori* device parameter estimates from (a) above.

### 3.2.2 Revised Tracking Algorithm

The algorithm for tracking with concurrent autocalibration is the same as that presented in section 3.1, with the following exceptions. After step (a) in the original algorithm, we form augmented versions of the state vector

$$\widehat{x}(t-\delta t) = \left[\hat{x}^T(t-\delta t) \ \ \hat{x}^T_{b,t}(t-\delta t) \ \ \hat{x}^T_{c,t}(t-\delta t)\right]^T, \tag{18}$$

the error covariance matrix

$$\widehat{P}(t-\delta t) = \begin{bmatrix} P(t-\delta t) & 0 & 0 \\ 0 & P_{b,t}(t-\delta t) & 0 \\ 0 & 0 & P_{c,t}(t-\delta t) \end{bmatrix}, \tag{19}$$

the state transition matrix

$$\widehat{A}(\delta t) = \begin{bmatrix} A(\delta t) & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}, \tag{20}$$

and the process noise matrix

$$\widehat{Q}(\delta t) = \begin{bmatrix} Q(\delta t) & 0 & 0 \\ 0 & Q_{b,t}(\delta t) & 0 \\ 0 & 0 & Q_{c,t}(\delta t) \end{bmatrix}. \tag{21}$$

* The operation $\alpha \otimes \Delta\alpha$ is used to indicate a quaternion multiply [9].

We then follow steps (b)-(h) from the original algorithm, making the appropriate substitutions of (18)-(21), and noting that the measurement and Jacobian functions used in step (c) have become $\grave{h}_\sigma(\widehat{x}(t))$ and $H_\sigma(\widehat{x}(t))$ because the estimates of parameters $\grave{b}_t$ and $\grave{c}_t$ ($\hat{x}_{b,t}$ and $\hat{x}_{c,t}$) are now contained in the augmented state vector $\widehat{x}$ per (18). After step (h) we finish by extracting and saving the device filter portions of the augmented state vector and error covariance matrix

$$\begin{aligned} \hat{x}_{b,t}(t) &= \widehat{x}(t)[i \ldots j] \\ P_{b,t}(t) &= \widehat{P}(t)[i \ldots j, i \ldots j] \\ \hat{x}_{c,t}(t) &= \widehat{x}(t)[k \ldots l] \\ P_{c,t}(t) &= \widehat{P}(t)[k \ldots l, k \ldots l] \end{aligned} \tag{22}$$

where

$$\begin{aligned} i &= n+1 \\ j &= n+n_b \\ k &= n+n_b+1 \\ l &= n+n_b+n_c \end{aligned}$$

and $n$, $n_b$, and $n_c$ are the dimensions of the state vectors for the main tracker filter, the source filter, and the sensor filter (respectively). We leave the main tracker filter state vector and error covariance matrix in their augmented counterparts, while we swap the device filter components in and out with each estimate. The result is that individual device filters are updated less frequently than the main tracker filter. The more a device is used, the more it is calibrated. If a device is never used, it is never calibrated.

With respect to added time complexity, the computations can again be optimized to eliminate operations on matrix and vector elements that are known to be zero: those places mentioned in section 3.1, and see (19)-(21). Also note that the size of and thus time for the matrix inversion in (13) has not changed. With respect to added space complexity, the autocalibration method requires storing a separate state vector and covariance matrix for each device—a fixed amount of (generally small) space per device. For example, consider autocalibrating the beacon (LED) positions for an optical tracking system with 3,000 beacons. For each beacon one would need 3 words for the beacon state (its 3D position), $3 \times 3 = 9$ words for the noise covariance matrix, and $3 \times 3 = 9$ words for the error covariance matrix. Assuming 8 bytes per word, this is only $3{,}000 \times 8 \times (3+9+9) = 504{,}000$ bytes.

#### 3.2.3 Discussion

The ability to simultaneously estimate two dependent sets of unknowns (the target and device states) is made possible by several factors. First, the dynamics of the two sets are very different as would be reflected in the process noise matrices. We assume the target is undergoing some random (constant) acceleration, reflected in the noise parameter $\eta$ of $Q(\delta t)$ in (6). Conversely, we assume the device parameters are constant, and so the elements of $Q_\pi(\delta t)$ for a source or sensor simply reflect any allowed variances in the corresponding parameters: usually zero or extremely small. In addition, while the target is expected to be moving, the filter expects the motion between any two estimations to closely correspond to the velocity estimates in the state (3). If the tracker estimate rate is high enough, poorly estimated device parameters will result in what appears to be almost instantaneous target motion. The increased rate of the SCAAT method allows such motion to be recognized as unlikely, and attributed to poorly estimated device parameters.

### 3.3 Stability

Because the SCAAT method uses individual measurements with insufficient information, one might be concerned about the potential for instability or divergence. A linear system is said to be stable if its response to any input tends to a finite steady value after the input is removed [24]. For the Kalman filter in general this is certainly not a new concern, and there are standard requirements and corresponding tests that ensure or detect stability (see [18], p. 132):

a. The filter must be uniformly completely observable,

b. the dynamic and measurement noise matrices in equations (6) and (9) must be bounded from above and below, and

c. the dynamic behavior represented by $A(\delta t)$ in equation (2) must be bounded from above.

As it turns out, these conditions and their standard tests are equally applicable to a SCAAT implementation. For the SCAAT method the conditions mean that the user dynamics between estimates must be bounded, the measurement noise must be bounded, one must incorporate a sufficient set of non-degenerate constraints *over time*. In particular, the constraints must be incorporated in less than 1/2 the time of the user motion time-constant in order to avoid tracking an alias of the true motion. In general these conditions are easily met for systems and circumstances that would otherwise be stable with a multiple-constraint implementation. A complete stability analysis is beyond the scope of this paper, and is presented in [47].

### 3.4 Measurement Ordering

Beyond a simple round-robin approach, one might envision a measurement scheduling algorithm that makes better use of the available resources. In doing so one would like to be able to monitor and control uncertainty in the state vector. By periodically observing the eigenvalues and eigenvectors of the error covariance matrix $P(t)$, one can determine the directions in state-space along which more or less information is needed [21]. This approach can be used to monitor the stability of the tracker, and to guide the source/sensor ordering.

## 4 EXPERIMENTS

We are using the SCAAT approach in the current version of the UNC wide-area optoelectronic tracking system known as the *HiBall tracker*. The *HiBall*, shown below in Figure 3, incorporates six optical sensors and six lenses with infrared filters into one golf ball sized sensing unit that can be worn on a user's head or hand. The principal mechanical component of the HiBall, the senor housing unit, was fabricated by researchers at the University of Utah using their $\alpha 1$ modeling environment.

Because the HiBall sensors and lenses share a common transparent space in the center of the housing, a single sensor can actually sense light through more than one lens. By making use of all of these *views* we end up with effectively 26 "cameras". These cameras are then used to observe ceiling-mounted light-emitting diodes (LEDs) to track the position and orientation of the HiBall. This inside-looking-out approach was first used with the previous UNC optoelectronic tracking system [44] which spanned most of the user's head and weighed approximately ten pounds, not including a backpack containing some electronics. In contrast, the HiBall sensing unit is the size of a golf ball and weighs only five ounces, *including* the electronics. The combination of reduced weight, smaller packaging, and the new SCAAT algorithm results in a very ergonomic, fast, and accurate system.

In this section we present results from both simulations performed during the design and development of the HiBall, and

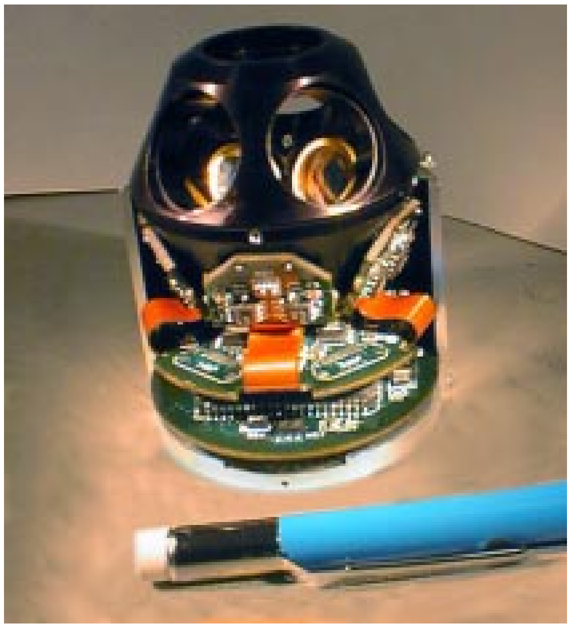


Figure 3: The HiBall is shown here with the internal circuitry exposed and the lenses removed. The sensors, which can be seen through the lens openings, are mounted on PC boards that fold-up into the HiBall upon assembly. The mechanical pencil at the bottom conveys an indication of the relative size of the unit.

preliminary results from the actual implementation. The simulations are useful because we have control over the "truth" and can perform controlled experiments. The results from the actual implementation serve to demonstrate actual operation and to provide some support for our accuracy and stability claims.

With respect to the SCAAT implementation, the tracker *sensors* are the HiBall cameras and the tracker *sources* are the ceiling-mounted 2D array of approximately 3000 electronic beacons (LEDs). The cameras provide a single 2D measurement vector, i.e. a 2D constraint, which is the $(u, v)$ image coordinates of the beacon as seen by the camera. So for this example, $m_\sigma = 2$ and $\vec{z}_\sigma = [u, v]^T$. The measurement function $\vec{h}_\sigma(\bullet)$ transforms the beacon into camera coordinates and then projects it onto the camera's image plane in order to predict the camera response.

For the simulations we generated individual measurement events (a single beacon activation followed by a single camera reading) at a rate of 1000 Hz, and corrupted the measurements using the noise models detailed in [8]. We tested components of our real system in a laboratory and found the noise models in [8] to be reasonably accurate, if not pessimistic. We also perturbed the 3D beacon positions prior to simulations with a normally-distributed noise source with approximately 1.7 millimeters standard deviation. We controlled all random number generators to facilitate method comparisons with common random sequences.

To evaluate the filter performance we needed some reference data. Our solution was to collect motion data from *real-user* sessions with a conventional tracking system, and then to filter the data to remove high frequency noise. We then *defined* this data to be the "truth". We did this for seven such sessions.

The simulator operated by sampling the truth data, choosing one beacon and camera (round-robin from within the set of valid combinations), computing the corresponding camera measurement vector $\vec{z}_{\sigma,t}$, and then adding some measurement noise. The (noisy) measurement vector, the camera parameter vector $\vec{c}_t$ (position and orientation in user coordinates), and the beacon parameter vector $\vec{b}_t$ (position in world coordinates) were then sent to the tracker.

For the tracking algorithm, we simulated both the SCAAT method (section 3.1, modified per section 3.2 for autocalibration) and several multiple-constraint methods, including the Collinearity method [3] and several variations of moving window (finite impulse response) methods. For each of the methods we varied the measurement noise, the measurement frequency, and the beacon position error. For the multiple constraint methods we also varied the number of constraints (beacon observations) per estimate $N$. In each case the respective estimates were compared with the truth data set for performance evaluation.

## 4.1 Tracker Filter

The 12 element state vector $\hat{x}(t)$ for the main tracker filter contained the elements shown in (3). Each of the 3000 beacon filters was allocated a 3 element state vector

$$\hat{x}_b = \begin{bmatrix} x_b \ y_b \ z_b \end{bmatrix}^T$$

where $(x_b, y_b, z_b)$ represents the beacon's estimated position in cartesian (world) coordinates. The $12 \times 12$ state transition matrix for the main tracker filter was formed as discussed section 3.1, and for each beacon filter it was the $3 \times 3$ identity matrix. The $12 \times 12$ process noise matrix for the main tracker was computed using (7), using elements of $\vec{\eta}$ that were determined off-line using Powell's method and a variety of real motion data. For each beacon filter we used an identical noise covariance matrix

$$Q_b(\delta t)[i, j] = \begin{cases} \eta_b & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

for $1 \le i, j \le 3$, with beacon position variance $\eta_b$ also determined off-line. (See [47] for the complete details.) At each estimate step, the *augmented* 15 element state vector, $15 \times 15$ process noise matrix, $15 \times 15$ state transition matrix, and $15 \times 15$ error covariance matrix all resembled (18)-(21) (without the camera parameter components). The measurement noise model was distance dependent (beacon light falls-off with distance) so $R_\sigma(t)$ from (9) was computed prior to step (d), by using a beacon distance estimate (obtained from the user and beacon positions in the predicted state $\widehat{x}^-$) to project a distance-dependent electrical variance onto the camera.

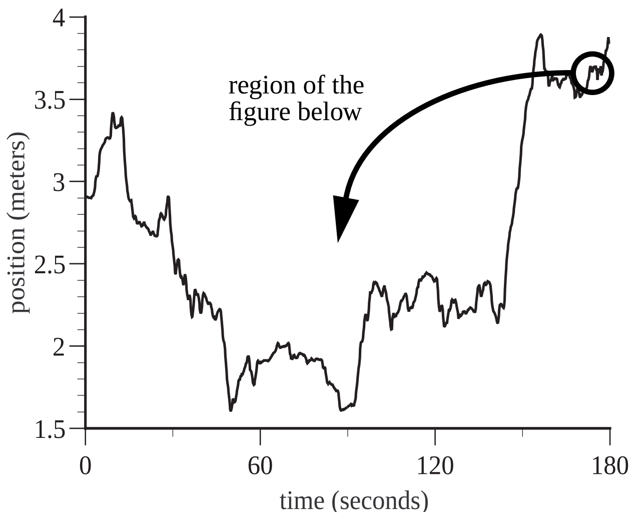
## 4.2 Initialization

The position and orientation elements of the main tracker state were initialized with the true user position and orientation, and the velocities were initialized to zero. The 3000 beacon filter state vectors were initialized with (potentially erroneous) beacon position estimates. The main tracker error covariance matrix was initialized to the null matrix. All beacon filter error covariance matrices were initialized to

$$P_b(0)[i, j] = \begin{cases} (0.001)^2 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

for $1 \le i, j \le 3$, to reflect 1 millimeter of uncertainty in the initial beacon positions.

While for the presented simulations we initialized the filter state with the true user pose information, we also performed (but will not show here) simulations in which the state elements were initialized to arbitrary values, e.g. all zeros. It is a testament to the stability of the method that in most cases the filter completely converged in under a tenth of a second, i.e. with fewer than 100 measurements. (In a few cases the camera was facing away from the beacon, a condition not handled by our simulator.)



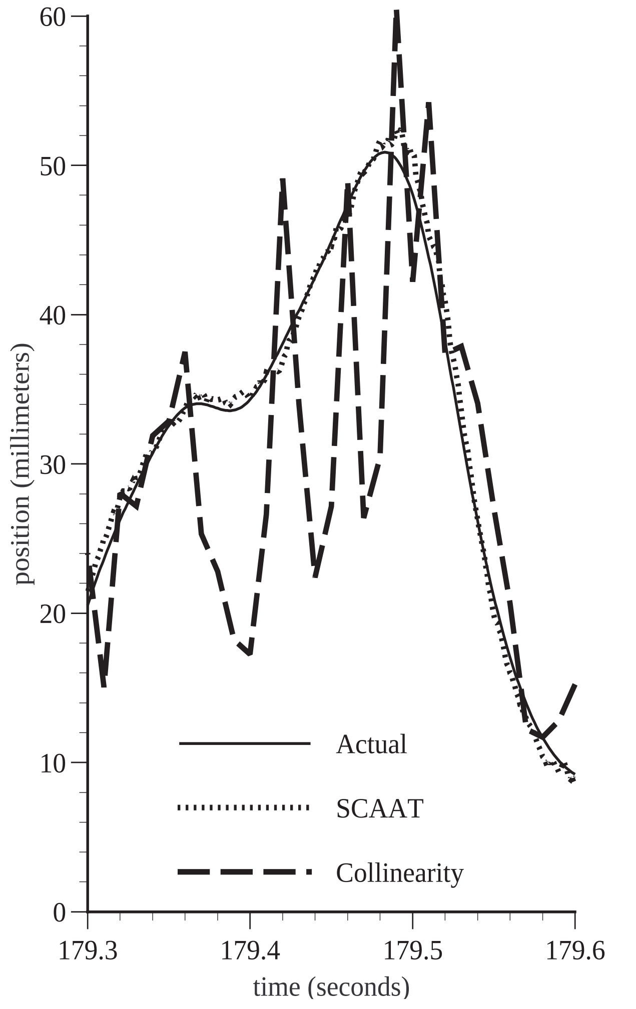


Figure 4: The upper plot depicts the entire 3 minutes of *x*-axis position data from user motion data set 'a' of sets 'a'-'f'. The lower plot shows a close-up of a short portion of the simulation. Collinearity here used $N = 10$ beacons per observation, hence its lower estimate rate. On the other hand, notice that the SCAAT estimates and the actual (truth) data are almost indistinguishable.

## 4.3 Simulation Results

We present here only comparisons of the SCAAT method with the Collinearity method, the "conventional approach" mentioned in the accompanying video. More extensive simulation results can be found in [47], including tests for stability under "cold starts" and periodic loss of data. All error measures reflect the RMS position error for a set of three imaginary points located approximately at arms length. This approach combines both position and orientation error into a metric that is related to the error a user would encounter in [HMD] screen space.

Figure 4 contains two related plots. The upper plot shows the entire three minutes (180 seconds) of the *x*-axis position for the first of seven data sets, data set 'a'. The lower plot shows a close-up of a particular segment of 300 milliseconds near the end. Notice that the Collinearity estimates appear very jerky. This is partially a result of the lower estimate rate, it is using $N = 10$ beacon observations to compute an estimate, and partially due to the method's inability to deal with the erroneous beacon position data. In contrast, the SCAAT method hugs the actual motion track, appearing both smooth and accurate. This is partly a result of the higher update rate (10 times Collinearity here), and partly the effects of Kalman filtering, but mostly the accuracy is a result of the SCAAT autocalibration scheme. With the autocalibration turned on, the initially erroneous beacon positions are being refined at the same high rate that user pose estimates are generated.

Figure 5 shows progressively improving estimates as the number of beacons $N$ is reduced from 15 (Collinearity) down to 1 (SCAAT), and a clear improvement in the accuracy when autocalibration is on. Consider for a moment that the motion prediction work of Azuma and Bishop [4] was based on jerky Collinearity estimates similar to those in Figure 4. The smooth and accurate SCAAT estimation should provide a much better basis for motion prediction, which could in turn provide a more effective means for addressing other system latencies such as those in the rendering pipeline. The improved accuracy should also improve post-rendering warping or image deflection [32,39].

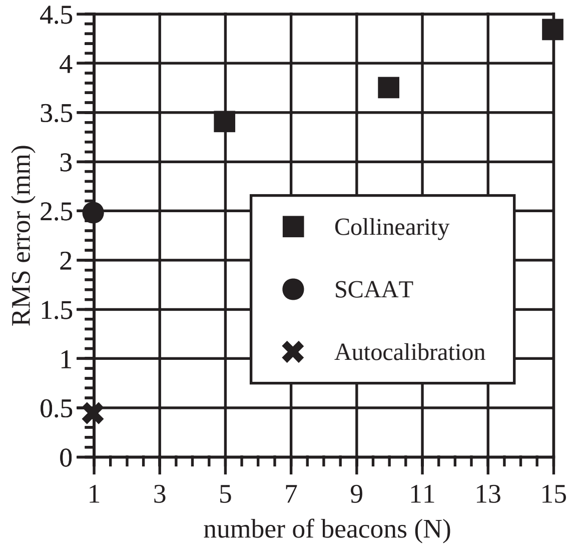


Figure 5: As the number of beacons $N$ is reduced from 15 to 5, the Collinearity results improve slightly. (The Collinearity algorithm generally becomes unstable with $N \leq 4$.) The SCAAT results, with $N = 1$ beacons, are better, and especially good once autocalibration is turned on.

As further evidence of the smoothing offered by the SCAAT approach, Figure 6 presents an error spectra comparison between a Collinearity implementation with $N = 10$, and a SCAAT implementation with and without autocalibration. Even without autocalibration the SCAAT output has significantly less noise than collinearity, and with autocalibration it is better by more than a factor of 10. These reductions in noise are clearly visible to the HMD user as a reduction in the amount of jitter in virtual-world objects.

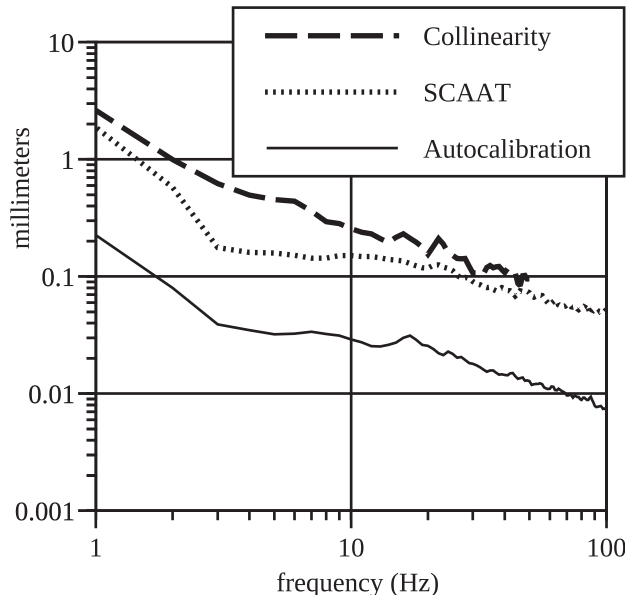


Figure 6: Here we show an error spectra comparison for the Collinearity method with $N = 10$ beacons, and the SCAAT method with and without autocalibration.

Figure 7 provides results for all seven of the real-user motion data sets. Again the Collinearity implementations observe $N = 10$ beacons per estimate, while the SCAAT implementations observe only $N = 1$. Because the beacon positions were being autocalibrated during the SCAAT run, we repeated each run, the second time using the beacon position estimation results from the first simulation. The more beacons are sighted during tracking, the better they are located. The second-pass simulation results are identified with the dagger (†) in Figure 7.

Figure 8 presents results that support the claim that the beacon location estimates are actually improving during tracking with autocalibration, as opposed to simply shifting to reduce spectral noise. Note that in the case of data set ‘d’, the beacon error was reduced nearly 60%.

Finally, we simulated using the SCAAT approach with tracking hardware that allowed truly simultaneous observations of beacons. For the Collinearity and other multiple-constraint methods we simply used the methods as usual, except that we passed them truly simultaneous measurements. For the SCAAT method we took the $N$ simultaneous observations, and simply processed them one at a time with $\delta t = 0$. (See equation (2).) We were, at first, surprised to see that even under these ideal circumstances the SCAAT implementation could perform better, even significantly better than a multiple-constraint method with simultaneous constraints. The reason seems to be autocalibration. Even though the multiple-constraint methods were “helped” by the truly simultaneous observations, the SCAAT method still had the advantage in that it could still autocalibrate the beacons more effectively that any multiple-constraint method. This again arises from the method’s inherent isolation of individual observations.

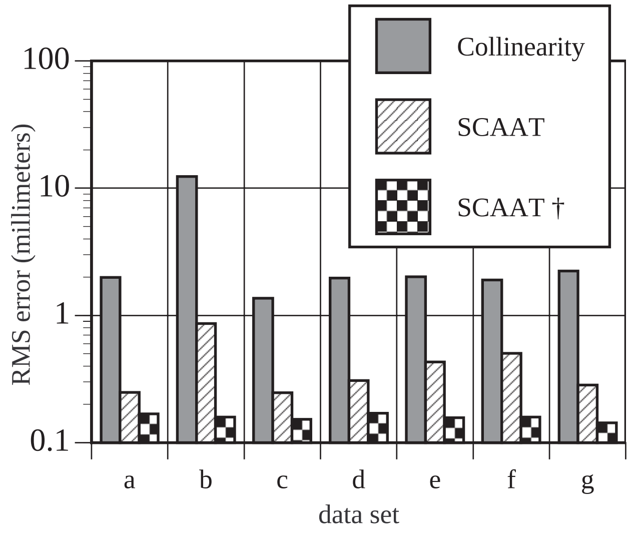


Figure 7: RMS error results for simulations of all seven real user motion data sets. The † symbol indicates a second pass through the motion data set, this time using the already autocalibrated beacons.

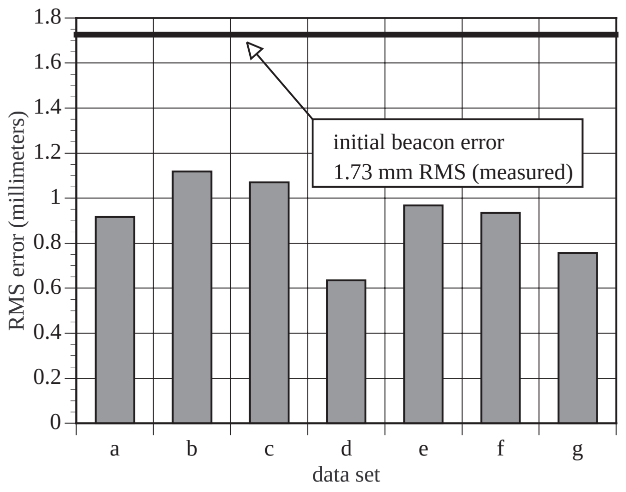


Figure 8: Autocalibration in action. Here we show the final beacon position error for runs through each of the seven user motion data sets.

## 4.4 Real Results

We have demonstrated the SCAAT algorithm with the HiBall tracker, a head-mounted display, and a real application. However, at the time of the submission of this publication we have yet to perform extensive optimization and analysis. As such we present here only limited, albeit compelling results.

The SCAAT code runs on a 200 MHz PC-compatible computer with a custom interface board. With unoptimized code, the system generates new estimates at approximately 700 Hz. We expect the optimized version to operate at over 1000 Hz. Out of the approximately 1.4 millisecond period, the unoptimized SCAAT code takes approximately 100 microseconds and sampling of the sensors takes approximately 200 microseconds. The remaining time is spent on overhead including a significant amount of unoptimized code to choose an LED and to gather results.

In one experiment we set the HiBall on a flat surface under the ceiling beacons and collected several minutes worth of data. Given that the platform was relatively stable, we believe that the deviation of the estimates provides an indication of the noise in the system. Also, because the HiBall was not moving, we were able to observe the progressive effects of the autocalibration. The standard deviation of the position estimates for the first 15 seconds is shown in Figure 9. With autocalibration off, the estimates deviate approximately 6.0 millimeters in translation and 0.25 degrees in orientation (not shown). With autocalibration on, notice in Figure 9 how the deviation decreases with time, settling at approximately 0.3 millimeters in translation and 0.01 degrees in orientation (again not shown).

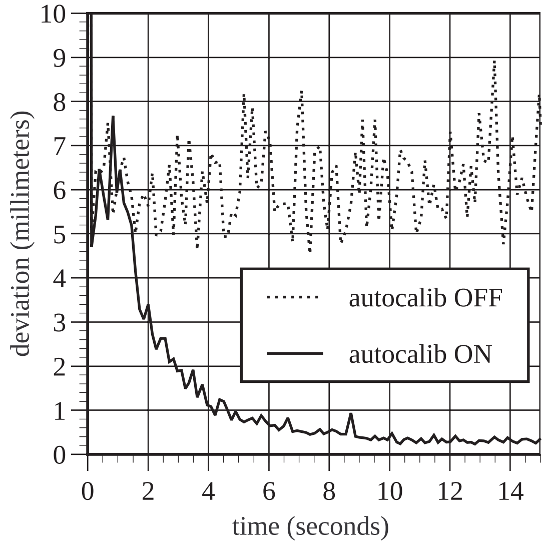


Figure 9: SCAAT position (only) estimate deviation for a Hiball sitting still on a flat surface, with and without autocalibration.

In another experiment we mounted the HiBall on a calibrated translation rail of length one meter, and slid (by hand) the HiBall from one end to the other and then back again. The disagreement between the HiBall and the calibrated position on the rail was less than 0.5 millimeters. The deviation of the measured track from co-linearity was 0.9 millimeters. Because the tolerances of our simple test fixture are of similar magnitude, we are unable to draw conclusions about how much of this disagreement should be attributed to error in the tracking system.

## 5 CONCLUSIONS

Stepping back from the details of the SCAAT method, we see an interesting relationship: Because the method generates estimates with individual measurements, it not only avoids the simultaneity assumption but it operates faster; by operating faster, it decreases the elapsed time since the previous state estimate; the more recent the previous estimate, the better the prediction in (12); the better the prediction, the more likely we can discriminate bad measurements; if we can discriminate bad measurements, we can autocalibrate the measurement devices; and if we can calibrate the measurement devices, we can improve the individual measurements, thus improving predictions, etc. In other words, the faster, the better.

Looking more closely, it is amazing that such a tracker can function at all. Consider for example the system presented in section 4. Any single beacon sighting offers so few constraints—the user could be theoretically *anywhere*. Similarly, knowledge about where the user was a moment ago is only an indicator of where the user *might* be now. But used together, these two sources of information can offer more constraints than either alone. With a Kalman filter we can extract the information from the previous state and a new (individual) measurement, and blend them to form a better estimate than would be possible using either alone.

The SCAAT method is accurate, stable, fast, and flexible, and we believe it can be used to improve the performance of a wide variety of commercial and custom tracking systems.

## Acknowledgements

We would like to thank the tracker team at UNC, in particular Vernon Chi, Steve Brumback, Kurtis Keller, Pawan Kumar, and Phillip Winston. This work was supported by DARPA/ETO contract no. DABT 63-93-C-0048, “Enabling Technologies and Application Demonstrations for Synthetic Environments”, Principle Investigators Frederick P. Brooks Jr. and Henry Fuchs (University of North Carolina at Chapel Hill), and by the National Science Foundation Cooperative Agreement no. ASC-8920219: “Science and Technology Center for Computer Graphics and Scientific Visualization”, Center Director Andy van Dam (Brown University). Principle Investigators Andy van Dam, Al Barr (California Institute of Technology), Don Greenberg (Cornell University), Henry Fuchs (University of North Carolina at Chapel Hill), Rich Riesenfeld (University of Utah).

## References

[1] C.G. Atkeson and J.M. Hollerbach. 1985. “Kinematic features of unrestrained vertical arm movements,” Journal of Neuroscience, 5:2318-2330.

[2] Ali Azarbayejani and Alex Pentland. June 1995. “Recursive Estimation of Motion, Structure, and Focal Length,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, June 1995, 17(6).

[3] Ronald Azuma and Mark Ward. 1991. “Space-Resection by Collinearity: Mathematics Behind the Optical Ceiling Head-Tracker,” UNC Chapel Hill Department of Computer Science technical report TR 91-048 (November 1991).

[4] Ronald Azuma and Gary Bishop. 1994. “Improving Static and Dynamic Registration in an Optical See-Through HMD,” SIGGRAPH 94 Conference Proceedings, Annual Conference Series, pp. 197-204, ACM SIGGRAPH, Addison Wesley, July 1994. ISBN 0-201-60795-6

[5] Ronald Azuma. 1995. “Predictive Tracking for Augmented Reality,” Ph.D. dissertation, University of North Carolina at Chapel Hill, TR95-007.

[6] Ted J. Broida and Rama Chellappa. 1986. “Estimation of object motion parameters from noisy images,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, January 1986, 8(1), pp. 90-99.

[7] R. G. Brown and P. Y. C. Hwang. 1992. *Introduction to Random Signals and Applied Kalman Filtering, 2nd Edition*, John Wiley & Sons, Inc.

[8] Vernon L. Chi. 1995. “Noise Model and Performance Analysis of Outward-looking Optical Trackers Using Lateral Effect Photo Diodes,” University of North Carolina, Department of Computer Science, TR 95-012 (April 3, 1995)

[9] Jack C.K. Chou. 1992. “Quaternion Kinematic and Dynamic Differential Equations,” IEEE Transactions on Robotics and Automation, Vol. 8, No. 1, pp. 53-64.

[10] J. L. Crowley and Y. Demazeau. 1993. “Principles and Techniques for Sensor Data Fusion,” *Signal Processing (EURASIP)* Vol. 32. pp. 5-27.

[11] J. J. Deyst and C. F. Price. 1968. “Conditions for Asymptotic Stability of the Discrete Minimum-Variance Linear Estimator,” *IEEE Transactions on Automatic Control*, December, 1968.

[12] S. Emura and S. Tachi. 1994. “Sensor Fusion based Measurement of Human Head Motion,” *Proceedings 3rd IEEE International Workshop on Robot and Human Communication, RO-MAN’94 NAGOYA* (Nagoya University, Nagoya, Japan).

[13] P. Fischer, R. Daniel and K. Siva. 1990. “Specification and Design of Input Devices for Teleoperation,” *Proceedings of the IEEE Conference on Robotics and Automation* (Cincinnati, OH), pp. 540-545.

[14] Eric Foxlin. 1993. “Inertial Head Tracking,” Master’s Thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

[15] M. Friedman, T. Starner, and A. Pentland. 1992. “Synchronization in Virtual Realities,” *Presence: Teleoperators and Virtual Environments*, 1:139-144.

[16] S. Ganapathy. November 1984. “Camera Location Determination Problem,” AT&T Bell Laboratories Technical Memorandum, 11358-841102-20-TM.

[17] G. J. Geier, P. V. W. Loomis and A. Cabak. 1987. “Guidance Simulation and Test Support for Differential GPS (Global Positioning System) Flight Experiment,” National Aeronautics and Space Administration (Washington, DC) NAS 1.26:177471.

[18] A. Gelb. 1974. *Applied Optimal Estimation*, MIT Press, Cambridge, MA.

[19] Stefan Gottschalk and John F. Hughes. 1993. “Autocalibration for Virtual Environments Tracking Hardware,” Proceedings of ACM SIGGRAPH 93 (Anaheim, CA, 1993), Computer Graphics, Annual Conference Series.

[20] A Robert De Saint Vincent Grandjean. 1989. “3-D Modeling of Indoor Scenes by Fusion of Noisy Range and Stereo Data,” *IEEE International Conference on Robotics and Automation* (Scottsdale, AZ), 2:681-687.

[21] F. C. Ham and R. G. Brown. 1983. “Observability, Eigenvalues, and Kalman Filtering,” *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-19, No. 2, pp. 269-273.

[22] R. Held and N. Durlach. 1987. *Telepresence, Time Delay, and Adaptation*. NASA Conference Publication 10023.

[23] Richard L. Holloway. 1995. “Registration Errors in Augmented Reality Systems,” Ph.D. dissertation, The University of North Carolina at Chapel Hill, TR95-016.

[24] O. L. R. Jacobs. 1993. *Introduction to Control Theory, 2nd Edition*. Oxford University Press.

[25] Roy S. Kalawsky. 1993. *The Science of Virtual Reality and Virtual Environments*, Addison-Wesley Publishers.

[26] R. E. Kalman. 1960. “A New Approach to Linear Filtering and Prediction Problems,” Transaction of the ASME—Journal of Basic Engineering, pp. 35-45 (March 1960).

[27] J. B. Kuipers. 1980 “SPASYN—An Electromagnetic Relative Position and Orientation Tracking System,” *IEEE Transactions on Instrumentation and Measurement*, Vol. IM-29, No. 4, pp. 462-466.

[28] Richard Lewis. 1986. *Optimal Estimation with an Introduction to Stochastic Control Theory*, John Wiley & Sons, Inc.

[29] J. Liang, C. Shaw and M. Green. 1991. “On Temporal-spatial Realism in the Virtual Reality Environment,” *Fourth Annual Symposium on User Interface Software and Technology*, pp. 19-25.

[30] R. Mahmoud, O. Loffeld and K. Hartmann. 1994. “Multisensor Data Fusion for Automated Guided Vehicles,” *Proceedings of SPIE - The International Society for Optical Engineering*, Vol. 2247, pp. 85-96.

[31] Peter S. Maybeck. 1979. *Stochastic Models, Estimation, and Control, Volume 1*, Academic Press, Inc.

[32] Thomas Mazuryk and Michael Gervautz. 1995. “Two-Step Prediction and Image Deflection for Exact Head Tracking in Virtual Environments,” *EUROGRAPHICS ‘95*, Vol. 14, No. 3, pp. 30-41.

[33] K. Meyer, H. Applewhite and F. Biocca. 1992. A Survey of Position Trackers. *Presence*, a publication of the *Center for Research in Journalism and Mass Communication*, The University of North Carolina at Chapel Hill.

[34] Mark Mine. 1993. “Characterization of End-to-End Delays in Head-Mounted Display Systems,” The University of North Carolina at Chapel Hill, TR93-001.

[35] National Research Council. 1994. “Virtual Reality, Scientific and Technological Challenges,” National Academy Press (Washington, DC).

[36] P.D. Neilson. 1972. “Speed of Response or Bandwidth of Voluntary System Controlling Elbow Position in Intact Man,” *Medical and Biological Engineering*, 10:450-459.

[37] F. H. Raab, E. B. Blood, T. O. Steiner, and H. R. Jones. 1979. “Magnetic Position and Orientation Tracking System,” *IEEE Transactions on Aerospace and Electronic Systems*, Vol. AES-15, 709-718.

[38] Selspot Technical Specifications, Selcom Laser Measurements, obtained from Innovision Systems, Inc. (Warren, MI).

[39] Richard H. Y. So and Michael J. Griffin. July-August 1992. “Compensating Lags in Head-Coupled Displays Using Head Position Prediction and Image Deflection,” *AIAA Journal of Aircraft*, Vol. 29, No. 6, pp. 1064-1068

[40] H. W. Sorenson. 1970. “Least-Squares estimation: from Gauss to Kalman,” *IEEE Spectrum*, Vol. 7, pp. 63-68, July 1970.

[41] Andrei State, Gentaro Hirota, David T. Chen, Bill Garrett, Mark Livingston. 1996. “Superior Augmented Reality Registration by Integrating Landmark Tracking and Magnetic Tracking,” SIGGRAPH 96 Conference Proceedings, Annual Conference Series, ACM SIGGRAPH, Addison Wesley, August 1996.

[42] J. V. L. Van Pabst and Paul F. C. Krekel. “Multi Sensor Data Fusion of Points, Line Segments and Surface Segments in 3D Space,” TNO Physics and Electronics Laboratory, The Hague, The Netherlands. [cited 19 November 1995]. Available from http://www.bart.nl/~lawick/index.html.

[43] J. Wang, R. Azuma, G. Bishop, V. Chi, J. Eyles, and H. Fuchs. 1990. “Tracking a head-mounted display in a room-sized environment with head-mounted cameras,” *Proceeding: SPIE'90 Technical Symposium on Optical Engineering & Photonics in Aerospace Sensing* (Orlando, FL).

[44] Mark Ward, Ronald Azuma, Robert Bennett, Stefan Gottschalk, and Henry Fuchs. 1992. “A Demonstrated Optical Tracker With Scalable Work Area for Head-Mounted Display Systems,” *Proceedings of 1992 Symposium on Interactive 3D Graphics* (Cambridge, MA, 29 March - 1 April 1992), pp. 43-52.

[45] Wefald, K.M., and McClary, C.R. “Autocalibration of a laser gyro strapdown inertial reference/navigation system,” *IEEE PLANS '84*. Position Location and Navigation Symposium Record.

[46] Greg Welch and Gary Bishop. 1995. “An Introduction to the Kalman Filter,” University of North Carolina, Department of Computer Science, TR 95-041.

[47] Greg Welch, 1996. “SCAAT: Incremental Tracking with Incomplete Information,” University of North Carolina at Chapel Hill, doctoral dissertation, TR 96-051.

[48] H. J. Woltring. 1974. “*New possibilities for human motion studies by real-time light spot position measurement*,” Biotelemetry, Vol. 1.