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_{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.)

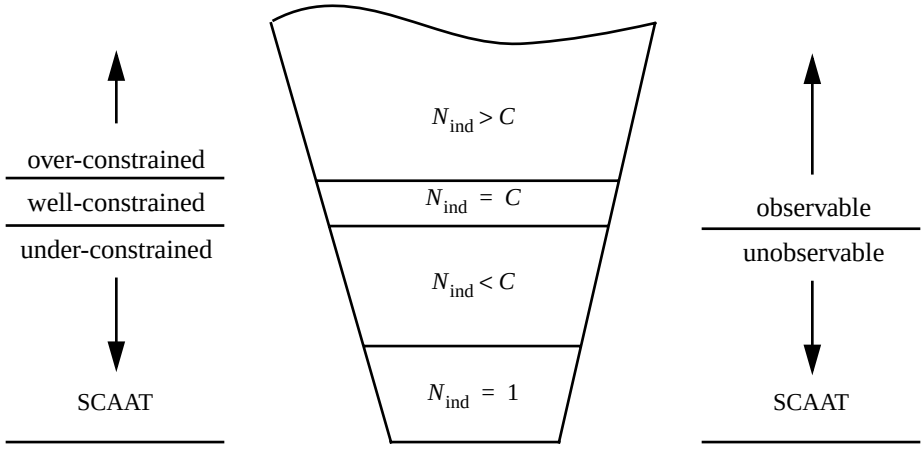

Figure 1: SCAAT and constraints on a system of simultaneous equations. C is the minimal number of independent simultaneous constraints necessary to uniquely determine a solution, N is the number of given constraints, and N_{ind} is the number of *independent* constraints that can be formed from the N. (For most systems of interest $C > 1$). The conventional approach is to ensure $N \geq N_{ind}$ and $N_{ind} \geq C$, 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 $N = N_{ind} = 1$ 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

$$t_e = Nt_m + t_c \ , \qquad r_e = \frac{1}{t_e} = \frac{1}{Nt_m + t_c} \ . \tag{1}$$

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 $10 \le N \le 20$ 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.

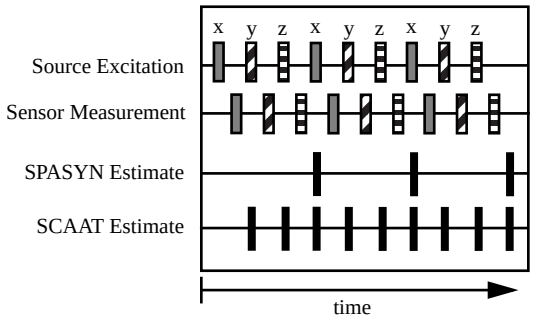

Figure 2: A timing diagram comparing the SPASYN (Polhemus Navigation Systems) magnetic position and orientation tracking system with a hypothetical SCAAT implementation.

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
β^-	=	matrix/vector *prediction* (super minus)
β^T	=	matrix/vector transpose (super T)
α_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 δt we model the target's dynamic motion with the following linear difference equation:

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

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 (ϕ, θ, ψ) 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 (ϕ, θ, ψ) 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

$$\vec{x} = \left[x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}\ \phi\ \theta\ \psi\ \dot{\phi}\ \dot{\theta}\ \dot{\psi}\right]^T \tag{3}$$

and the four-element external orientation quaternion

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

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 δt. 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 δt, 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 δt 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* σ 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 σ) and source becomes available at time t:

a. Compute the time δt 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*.

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

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

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

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

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

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.