Measurement Sample Time Optimization for Human Motion Tracking/Capture Systems

Greg Welch*
The University *of* North Carolina *at* Chapel Hill

B. Danette Allen†
NASA Langley Research Center and
The University *of* North Carolina *at* Chapel Hill

Adrian Ilie‡
The University *of* North Carolina *at* Chapel Hill

Gary Bishop§
The University *of* North Carolina *at* Chapel Hill

Abstract

Many human motion tracking systems average, integrate, or correlate device samples over some non-zero period of time in order to produce a single system-level measurement. This is done to reduce the effects of device noise, or because one has no choice with some devices. The problem is that the target (user) is likely to be moving throughout the sample time, and this movement will in effect introduce additional "noise" (uncertainty) into the measurements.

In this paper we introduce a method to optimize the device sampling time at the measurement level. The central idea is to determine the sampling time that maximizes device noise filtering while minimizing the impact of the expected target motion over the interval. We present the theory behind the approach, and both simulated and real-world examples.

Keywords: Tracking, human motion, optimization, noise.

Index Terms:
I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism — Virtual reality— [I.4.4]: Image Processing and Computer Vision—Restoration — Kalman filtering I.4.8 [Image Processing and Computer Vision]: Scene Analysis — Sensor fusion— [I.4.8]: Image Processing and Computer Vision—Scene Analysis — Motion I.4.8 [Image Processing and Computer Vision]: Scene Analysis — Tracking— [G.3]: Probability and Statistics— Stochastic processes

1 Introduction

For any given human motion tracking system there is an *inherent pose uncertainty* associated with the hardware design, independent of any particular tracking algorithm. This uncertainty comes from such things as poor signal, under-constrained measurements, measurement noise, and "competition" from dynamic human motion. (If the users would only sit still!) In short, the hardware design matters, as it determines an effective bound on how well one can resolve the pose of a target throughout the volume. No algorithm can overcome a loss of signal resulting from poor hardware choices.

In some circumstances, for example if the hardware design is relatively homogeneous, and the design choices are correspondingly limited, one can attempt to *automatically* optimize the hardware design as in [8, 15, 11] and [9, 10]. As the design becomes more complex, such optimizations become more or completely intractable. An alternative approach in these cases is to provide a human with an appropriate indication of the "cost" of a design, and then allow them to vary the parameters in an attempt to minimize the cost. This *intelligence amplification* approach [2, 3] has been used to evaluate specific designs [12] and is the basis for design tools like Pandora [17], an interactive software simulator for human-in-the-loop optimization of multi-camera setups.

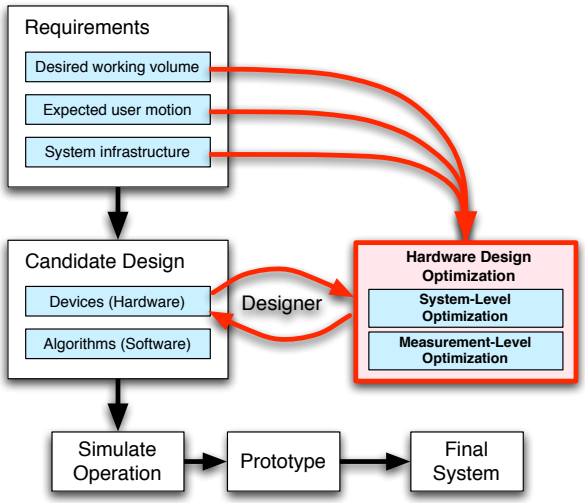

Figure 1: The tracking system design process with hardware optimization at the system and measurement level.

In [1] we introduced a more general method for evaluating, comparing, and interactively optimizing the expected performance of tracking and motion capture systems. Given a candidate hardware design and some model for the expected user motion, we compute and interactively visualize estimates for the asymptotic (steady-state) pose uncertainty throughout the working volume. (Asymptotic uncertainty is one possible measure of the inherent uncertainty for a particular design.) As with systems for the simulation and visualization of fluid or air flow, the idea is to modify the hardware design, looking for "hot" and "cold" spots in uncertainty, unusual shapes or variations in uncertainty, or uncertainty *sensitivities* to the hardware configuration. As illustrated in Figure 1, this *hardware design optimization* process is intended to augment traditional design and simulation practices, offering another tool to achive the best possible hardware design.

As indicated in the red box on the right in Figure 1, one can consider the hardware design optimization at two levels. In particular we think of the design optimization described above and addressed in [1] as a *system*-level optimization. At this level one is primarily concerned with the choice of sensors/sources, the number and arrangement of the devices, and the device sampling *rates*, all in the context of the expected user motion.

*e-mail: welch@cs.unc.edu
†e-mail: bdallen@cs.unc.edu
‡e-mail:adyilie@cs.unc.edu
§e-mail: gb@cs.unc.edu

In this paper we introduce the notion of a *measurement*-level optimization. In particular we introduce the idea that in addition to an optimal device sample *rate* there is an optimal sample *duration*, for systems that sample devices over some non-zero time interval. For example, because cameras integrate light over a non-zero shutter time, estimating camera motion or dynamic scene structure using feature or color matching always involves a tradeoff between maximizing the signal (dynamic range or contrast) and minimizing the motion-induced noise (blur). If the shutter time is too short, the uncertainty in the measurements will be greater than necessary. If the shutter time is too long, the measurements will be corrupted by scene or camera motion. In this paper we present a structured way of thinking about such tradeoffs, an approach for determining the optimal sample time, and some experimental results.

2 State-Space Models

In order to trade off electrical measurement noise against motion-induced measurement noise (e.g., blur in a camera) it is useful to characterize each in a common mathematical framework. In [1] we described the *system*-level optimization using state-space notation, and we do so again here for the *measurement*-level optimization.

Without loss of generality, let us assume a 3D tracking system. In this case the *state* would likely include the 3D position, and perhaps also the corresponding derivatives with respect to time (3D velocity and 3D acceleration). If we consider only position and velocity, the 6D state would be

$$\bar{x} = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^{\mathsf{T}}. \tag{1}$$

Given any point in this state space one can use a mathematical *motion model* to predict how the user will move over some small time interval. Similarly one can use some form of a *measurement model* to predict what will be measured by each hardware device. Both of these models can be structured with deterministic and random components, and we do so in the following sections.

2.1 Motion Models

There are at least two ways to model the expected user motion: using stochastic models with deterministic and random components, or using deterministic (only) classical mechanics. We begin by discussing the former, and touch on the latter at the end of the section.

Motion models have traditionally been described in terms of a physical parameter that can be modeled as a constant over time. This basically indicates the "order" of the model—the number of terms in a Taylor series describing the motion. Examples of such models include [7] *constant position* ($\dot{x} = 0$), *constant velocity* ($\dot{v} = 0$), and *constant acceleration* ($\dot{a} = 0$). See Figure 2.

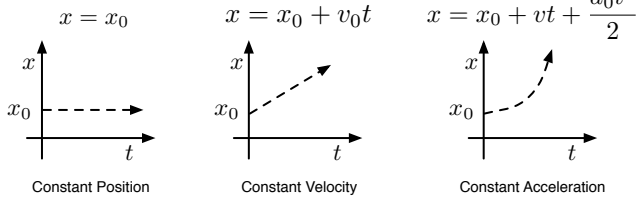

Figure 2: Traditional Motion Models

In a typical stochastic model the "constant" term (e.g., x_0, v_0, a_0) is replaced with an integrated random noise process, i.e. a random walk. So in the "constant" velocity case shown in Figure 2, the position would instead be modeled as $x = x_0 + vt$, where $v = \int a$, and $a \sim \mathrm{N}(0,q)$.[1] If we incorporate this random component into each type of constant motion model, we end up with what are typically called Position (P), Position-Velocity (PV) and Position-Velocity-Acceleration (PVA) motion models. Figure 3 shows the resulting stochastic models, using integrals to relate the position x with its temporal derivatives and the "driving" noise source $\mathrm{N}(0,q)$.

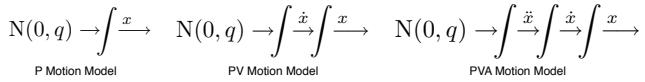

Figure 3: Stochastic Motion Models

Each of the above forms of stochastic motion models can be used to describe a different type of (actual) user motion. For example, a P-motion model might be an appropriate choice for situations where the user is almost still most of time, and the velocity can be effectively (although not perfectly) modeled by random noise. In this case one is modeling the position as a random walk. Similarly if the user's velocity is relatively constant for periods of time, and the acceleration is somewhat random, a PV-motion model might be more appropriate. In this case one is modeling the *velocity* as a random walk. Similarly one can model the *acceleration* as a random walk.

For an $n \times 1$ state vector $\bar{x}$, the corresponding continuous-time change in state can be modeled by the linear *process* equation

$$\frac{d\bar{x}}{dt} = A_c \bar{x} + \bar{q}_c, \tag{2}$$

where A_c is an $n \times n$ continuous-time *state transition matrix*, and $\bar{q}_c = [0, \ldots, 0, \mathrm{N}(0,q)]^{\mathsf{T}}$ is an $n \times 1$ continuous-time *process noise vector* with corresponding $n \times n$ *noise covariance matrix* $Q_c = \mathrm{E}\{\bar{q}_c \bar{q}_c^{\mathsf{T}}\}$, where $\mathrm{E}\{\}$ indicates expected value. In Equation (2), $A_c \bar{x}$ reflects the deterministic component of the model and $\bar{q}_c$ the random component. See Table 1 for the process parameters corresponding to the P, PV, and PVA models in Figure 3.

Model	$\bar{x}$	A_c	Q_c
P	$\begin{bmatrix} x \end{bmatrix}$	$\begin{bmatrix} 0 \end{bmatrix}$	$\begin{bmatrix} q \end{bmatrix}$
PV	$\begin{bmatrix} x \\ \dot{x} \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & q \end{bmatrix}$
PVA	$\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & q \end{bmatrix}$

Table 1: Parameters for P, PV, and PVA motion models.

To evaluate the impact of the (expected) user motion over some time interval, one needs to examine the impact of the injected noise at each stage of the process (each element of the state). As time passes, the noise variance at and between each state grows, both deterministically as a result of the integration, and randomly as a result of the injected noise $\mathrm{N}(0,q)$. One can integrate the continuous-time process Equation (2) over a time interval δt to obtain an $n \times n$ *discrete*-time (sampled) Q matrix. The solution to this integration is given by

$$Q = \int_0^{\delta t} e^{A_c t} Q_c e^{A_c^{\mathsf{T}} t} dt \tag{3}$$

as described in [13]. With appropriately formed A_c and Q_c matrices one can use Equation (3) to compute the discrete time Q matrices for each of the motion-models in Figure 3. The three corresponding discrete-time (sampled) Q matrices are

$$Q_P = \begin{bmatrix} q\,\delta t \end{bmatrix}, \tag{4}$$

$$Q_{PV} = \begin{bmatrix} q\frac{\delta t^3}{3} & q\frac{\delta t^2}{2} \\ q\frac{\delta t^2}{2} & q\,\delta t \end{bmatrix}, \text{ and} \tag{5}$$

[1] We use the notation $x \sim \mathrm{N}(\mu, \sigma^2)$ to indicate that x is normally distributed with mean μ and variance σ^2.

$$Q_{PVA} = \begin{bmatrix} q\frac{\delta t^5}{20} & q\frac{\delta t^4}{8} & q\frac{\delta t^3}{6} \\ q\frac{\delta t^4}{8} & q\frac{\delta t^3}{3} & q\frac{\delta t^2}{2} \\ q\frac{\delta t^3}{6} & q\frac{\delta t^2}{2} & q\,\delta t \end{bmatrix}. \quad (6)$$

This stochastic formulation for the motion model is very general, and is often already available (modeled) for tracking systems that (for example) already use a Kalman filter [14, 19]. For example, systems by Intersense and 3rdTech (HiBall-3100™) use Kalman filters, as do many research systems [16, 20, 18, 4]. However it is possible to consider the optimal sample time using only deterministic motion integration, i.e. simple classical mechanics. For example one might simply integrate some expected velocity over the sample interval to obtain a first-order estimate of the impact of the user motion on the measurement.

2.2 Measurement Models

Similar to the motion noise model, the $m \times 1$ measurement vector $\bar{z}$ obtained from any device can be described by a linear combination of deterministic and random components, as

$$\bar{z} = H\bar{x} + \bar{r}, \quad (7)$$

where H is an $m \times n$ *measurement matrix* that defines the relationship between the $n \times 1$ state space and the $m \times 1$ measurement space, and $\bar{r}$ is an $m \times 1$ *measurement noise vector* with corresponding $m \times m$ *noise covariance matrix* R, i.e. $\bar{r} \sim \mathrm{N}(0, R)$. If the relationship between the state and the measurements is nonlinear, then H can be approximated by the Jacobian of the nonlinear function.

Typically both H and R can be determined by some sort of "bench top" calibration. For example, given a calibrated camera, the 2D measurement of a 3D point would be modeled as the transformation and projection of a point from the world into the camera and onto the image plane. The measurement noise covariance R can be determined experimentally, by computing the covariance of a set of fixed points (in the working volume) over time. If the variance in the measurements changes with distance (for example) then one can fit a parametric curve to the experimental results, and compute R as a function of distance.

Note that the measurement matrix H can also be used to transform state-space uncertainly, e.g., the Q matrix, into the measurement space where it can be compared with R. Because Q is a variance, Q is mapped into measurement space as HQH^{T}. All of the simulations in the following sections are one-dimensional, with $H = 1$ for P models, $H = [1,0]$ for PV, and $H = [1,0,0]$ for PVA.

Finally note that it is the measurement device(s) that determine the system update period δt. Each device may operate at a different δt, or every device may operate at the same rate. In any case, independent of the *rate* of updates we are now going to examine what happens *during* the measurement (sampling) associated with any one update.

3 Measurement-Level Optimization

Without loss of generality, let us assume a regular update cycle for a tracking system. Each cycle for a particular device can be divided into sampling, processing, and idle time as shown in Figure 4. As implied in the diagram, $\delta t = \tau_s + \tau_p + \tau_i$.

Typically one wants to minimize the idle time τ_i in an attempt to generate estimates as frequently as possible. Similarly one generally wants to minimize the processing time τ_p, to minimize the likely change in target state between system-level updates (δt), thus allowing for simplification of the estimation algorithm, which further reduces τ_p, minimizing state change, etc. This circular relationship, related to time-motion optimization, was originally described in [5]. Indeed when one reduces δt for a particular device one will likely see improved system-level performance [1].

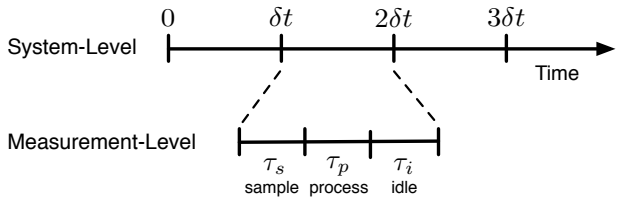

Figure 4: At the system level, a tracking system attempts to estimate the target pose at some regular interval δt (seconds). At the measurement level this can be further divided into sampling, processing, and idle time.

3.1 Sampling Time (τ_s)

In keeping with the apparent "less is more" circumstances described above, it would seem that one would also want to minimize the device sampling time τ_s. Indeed if the device measurements are obtained using instantaneous analog-to-digital conversion circuitry, then $\tau_s = 0$. In this case each measurement will have a constant amount of noise or uncertainty R associated with it.

However there are many situations where device measurements are not instantaneous, either because they cannot be in theory, or because it is impractical to do so. In such cases, $\tau_s > 0$. For example, designers sometimes choose to *average* some number of instantaneous device samples over τ_s in an attempt to reduce the effects of device noise. In addition, image-forming cameras inherently *integrate* photons over τ_s (the shutter time) and then deliver that integrated energy in the form of a single image. In both of these situations the noise or uncertainty associated with each measurement is no longer constant, but a function $R(\tau_s)$ that typically decreases as τ_s (and/or the number of averaged samples) increases.

3.2 Optimal Sampling Time (τ_s)

If increasing τ_s (and/or using more samples) reduces the effects of the device noise, then why not increase τ_s as much as possible? The problem is that for human tracking or motion capture systems, the target (the user) is likely to be *moving* throughout the sample time τ_s. This movement will, in effect, introduce some additional "noise" (uncertainty) into the measurement, and that noise will actually *increase* as τ_s increases. In effect, any measurements based on samples collected over $\tau_s > 0$ have an uncertainty that is a combination of the expected device noise and the target motion uncertainty. Typically when τ_s is small the device noise dominates, and as τ_s increases the target motion uncertainty dominates. In between there is an optimal τ_s where neither dominates—where the sum of the two is minimal. Using the stochastic motion and measurement models described in Section 2 one can determine this optimal τ_s.

Figure 5 depicts three common measurement situations: instantaneous, averaged, and integrated. For each example we model the uncertainty in the process (user motion) as a noise-driven linear PV model as described earlier in Section 2.1. Figure 5 (a) represents a situation where the target pose is measured instantaneously as indicated by the switch. This corresponds to a typical analog-digital conversion with sample & hold circuitry. In (b), s instantaneous samples are collected over τ_s and then *averaged*. This is the approach used in the HiBall™ system for example. In (c) the signals are continuously *integrated* over the interval τ_s and the integral is then sampled. This corresponds to a camera.

In Figure 6 we show the results of three one-dimensional simulations corresponding to the measurement situations in Figure 5. Each plot shows simulated target motion uncertainty Q and simulated device noise R for the corresponding system shown in Figure 5. In each case there is a point where the sum is minimal, i.e. the optimal τ_s. We now discuss each situation in turn.

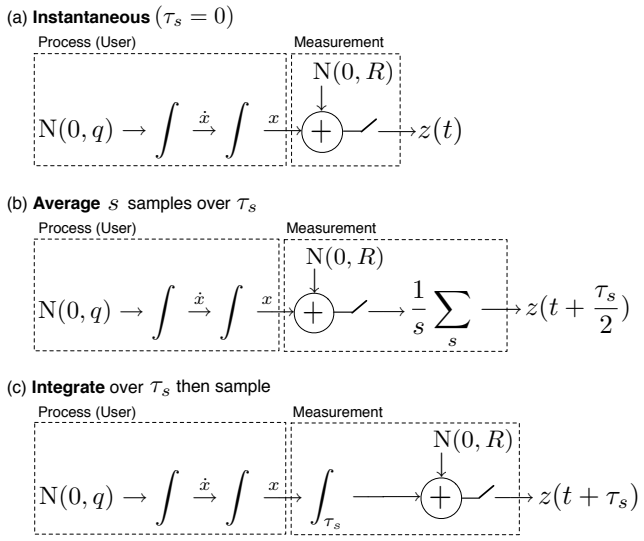

Figure 5: Three possible measurement scenarios for system with a position-velocity (PV) process model as described in Section 2.1: (a) instantaneous samples; (b) s instantaneous samples are *averaged* over τ_s; and (c) signals are *integrated* over τ_s and then sampled.

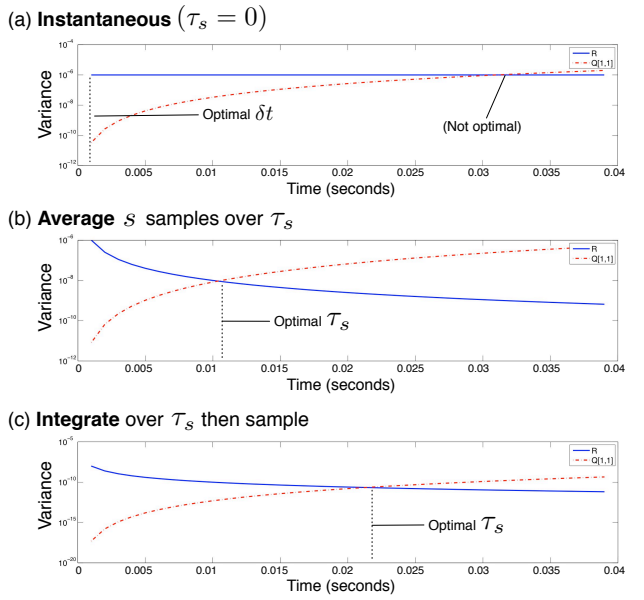

Figure 6: Three Q-R plots corresponding to the example measurement scenarios shown in Figure 5: (a) instantaneous samples; (b) s instantaneous samples are *averaged* over τ_s; and (c) signals are *integrated* over τ_s and then sampled. In each the solid blue curve represents the uncertainty due to random measurement noise, while the dashed red line represents the uncertainty due to (possible) user motion. (For these examples the relative positions of the optimal τ_s values are more important than the absolute numbers.)

(a) Single Instantaneous Samples ($\tau_s = 0$)

Case (a), where a single instantaneous device sample is collected at each system update time $t = k\delta t$ ($k = 0, 1, 2, \ldots$) is a unique situation. In this case, because $\tau_s = 0$, one is typically only concerned with determining the optimal system-level update period δt. When trying to determine the best δt in this special case, the device noise is essentially irrelevant because it is constant for every sample. And because the uncertainty in target state between system-level updates grows with δt, one essentially wants to keep δt as small as possible, minimizing τ_p and τ_i (see Figure 4). This situation is shown in Figure 6 (a), where R is constant, and $Q[1,1]$ is of the form of the upper-left element of the Q_{PV} in Section 2.1. In this special case, the optimal δt is simply as small as possible; the longer one waits between system-level measurement updates, the more uncertain the user pose, etc. per the circular relationship described in [5].

(b) Average s Samples Over τ_s

Case (b) represents the situation where at every system-level update cycle $t = k\delta t$ one collects s ($s > 1$) instantaneous samples (over period τ_s) and averages them to produce a single system-level measurement for time $t + \tau_s/2$. This is the approach used in the HiBall$^{\text{TM}}$ system for example. Because the device noise R and process (user motion) uncertainty are both being averaged, they are both now a function of time, and they both affect the variance (noise) in the averaged measurement $z(t + \tau_s/2)$. The variance in the average measurement due to R decreases with increasing time as the device noise is averaged. And while the measurement variance due to Q (target motion uncertainty) increases with time, it increases more *slowly* than in case (a). The optimal τ_s (over which one would average the s samples) is at $\min(Q(\tau_s) + R(\tau_s))$, about where the curves cross in Figure 6 (b).

That Q increases more slowly than in case (a) should make sense intuitively, as averaging is a form of low-pass filtering. However we can also model the effect stochastically. Averaging the target motion can be modeled by adding another element to the state. In case (a) the one-dimensional PV-model state was simply $\bar{x} = [x\ \dot{x}]^{\mathsf{T}}$. In case (b) it becomes $\bar{x} = [\mathrm{E}\{x\}\ x\ \dot{x}]^{\mathsf{T}}$. Noting that for a constant-velocity model, $d\mathrm{E}\{x\}/dt = \dot{x}/2$, and referring back to Section 2.1, we can augment the continuous time state transition matrix A_c with an additional averaging term, and increase the dimension of Q_c,

$$A_c = \begin{bmatrix} 0 & 0 & 1/2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad Q_c = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & q \end{bmatrix},$$

and use Equation (3) again to obtain the new discrete time Q

$$Q = \begin{bmatrix} q\frac{(\delta t)^3}{12} & q\frac{(\delta t)^3}{6} & q\frac{(\delta t)^2}{4} \\ q\frac{(\delta t)^3}{6} & & \\ q\frac{(\delta t)^2}{4} & & Q_{PV} \end{bmatrix}. \tag{8}$$

where Q_{PV} is as in Equation (5). The target motion uncertainty in Figure 6 (b) is modeled as $Q[1,1] = q(\delta t)^3/12$ (dashed red line).

(c) Integrate Over τ_s then Sample

Case (c) represents the situation where once every system-level update cycle $t = k\delta t$ the device signal is *integrated* over the interval τ_s, and the integral is then sampled to produce a single system-level measurement for time $t + \tau_s$. This corresponds to the operation of a camera, where the shutter opens to collect photons (the shutter time is τ_s) and the result is read back as an image. While it does not appear that the device noise R is being integrated, the fact that the *signal* is integrated means that the signal/noise ratio goes up with time, i.e. the noise/signal ratio goes down. So again the device noise R and process (user motion) uncertainty are both functions of time, and they both affect the variance (noise) in the integral measurement $z(t + \tau_s)$.

As in case (b), integrating the target position x can be modeled by adding another element to the state: $\bar{x} = [\int x\ \ x\ \ \dot{x}]^{\mathsf{T}}$. Referring back to Section 2.1, augmenting the process with an additional integrator effectively changes it from a PV to PVA model, with A_c and Q_c as shown in the PVA row of Table 1. We then use Equation (3) to arrive at the new discrete time Q as in Equation (9),

$$Q = \begin{bmatrix} q\frac{(\delta t)^5}{20} & q\frac{(\delta t)^4}{8} & q\frac{(\delta t)^3}{6} \\ q\frac{(\delta t)^4}{8} & & \\ q\frac{(\delta t)^3}{6} & & Q_{PV} \end{bmatrix} \quad (9)$$

where Q_{PV} is as in Equation (5). The target motion uncertainty in Figure 6 (c) is modeled as $Q[1,1] = q(\delta t)^5/20$ (dashed red line). The optimal τ_s (over which one would integrate the device signal) is at $\min(Q(\tau_s)+R(\tau_s))$, about where the curves cross.

3.3 Dynamic Optimal Sampling Time (τ_s)

As discussed in Section 2.2, one can typically characterize R as a function of time (and any other relevant parameters) using a bench-top setup consisting of the actual measurement devices. And while one cannot easily characterize Q experimentally, one can use stochastic process models as described above.

If the expected device noise and user motion statistics are unimodal and stationary, an optimal τ_s can be determined for each device during the hardware design process. However if the expected user motion and device noise statistics are *not* stationary during normal operation, or if the variance q of the driving process noise is expected to vary, or the form of the model might change, one can compute the optimal sampling time τ_s dynamically at run time. This could be done numerically if necessary, but if one has or can approximate analytical expressions for $R(\tau_s)$ and $Q(\tau_s)$, one can dynamically solve for the τ_s where $R(\tau_s)+Q(\tau_s)$ is minimized. This will allow the system to continually adjust the optimal sampling time based on the current circumstances. For example, if the target motion starts slowing down, τ_s could probably be increased, and when the target motion starts speeding up again, τ_s should probably be decreased.

4 Experiments

Beyond our simulations we have carried out a simple controlled experiment to illustrate the effect. In summary, we used two calibrated cameras to image a rotating LED, triangulated the LED position over time, fitted those points to a circle, and computed the statistical deviation from the circle for different shutter times (τ_s).

Our test setup is shown in Figure 7. We mounted a battery operated LED on a 15 cm (approximate) arm attached to a computer-controlled rotation stage (Directed Perception PTU-D46-70). About 50 cm away we mounted two PtGrey Dragonfly digital cameras pointed towards the rotating LED. The cameras had 6 mm lenses, and using the PtGrey software we fixed all of the imaging parameters (including gain at 1.0) except the shutter time (τ_s), which we varied under computer control.

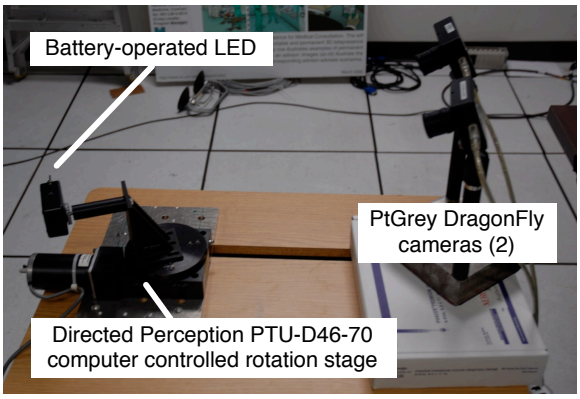

Figure 7: Experimental setup with rotating LED and two cameras.

Initially we performed two *static* experiments to model the measurement noise. First we positioned the LED at the near side of the circle (about 35 cm away from the cameras), held it still, and measured the standard deviation of the 2D centroids (on the image plane) as we varied the shutter time. We fit a curve to the deviations to obtain $R_{35cm}(\tau_s)$. The results are shown in Figure 8 (a), where the blue asterisks are the individual deviations, and the solid red line is the fitted curve. We repeated this experiment with the LED at the far side of the circle (about 65 cm away from the cameras) to obtain $R_{65cm}(\tau_s)$. The results are shown in Figure 8 (b).

Because there was essentially no uncertainty in the motion of the LED, we used simple classical mechanics to predict the likely blur of the LED in the images given the varying τ_s. Specifically, we predicted the blur by first computing the arc length of the LED path over the shutter interval, and then projecting that onto the camera image plane. The result is shown as a black dashed line in Figure 8. The sum of this predicted LED motion (blur) and the actual measurement noise (solid red) curves is shown in dashed green in Figure 8, and the predicted minimums (optimal τ_s) are noted.

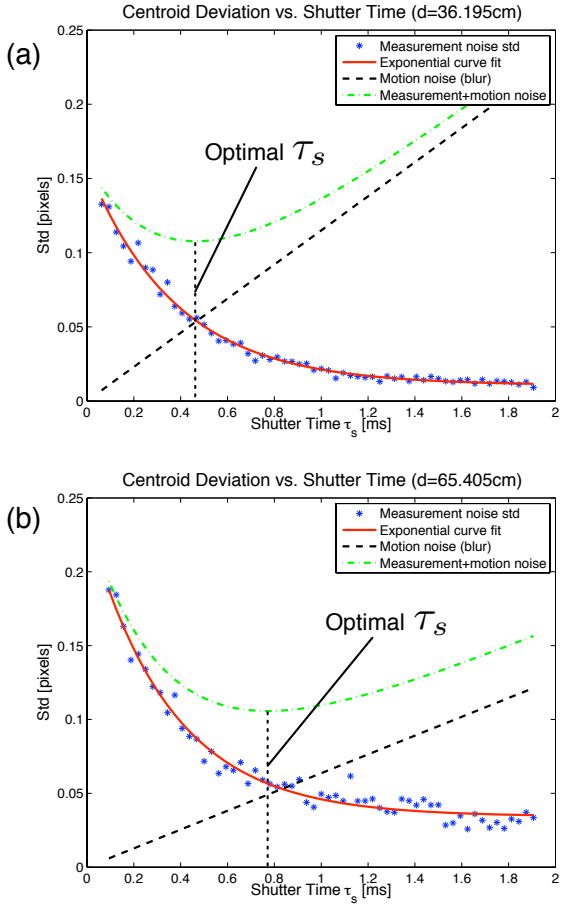

Figure 8: The actual measurement noise and predicted LED motion noise (blur) curves for a rotating LED and varying τ_s.

Finally we performed several *dynamic* experiments (rotating LED) to see if we could observe the "saddle" effect predicted in Figure 8 as we varied τ_s. We rotated the LED about one revolution every 10 seconds, capturing 30 frame-per-second imagery with the (synchronized) cameras, with around two complete revolutions per shutter setting (τ_s). We then undistorted the imagery, found the centroids of the LED in all images, triangulated to find the 3D posi-

tions, and fit the points to a 3D circle.[2] We computed the statistics on the fit in terms of the Euclidian distance to the circle, the distance to the plane of the circle, and the distance to the cylinder formed by the circle. The results are shown in Figure 9. The minimums appear to be at around $\tau_s = 0.6$ ms, which is about the average of the minimums predicted in Figure 8: around 0.4 ms in (a) and about 0.8 ms in (b). The minimums are statistically significant.

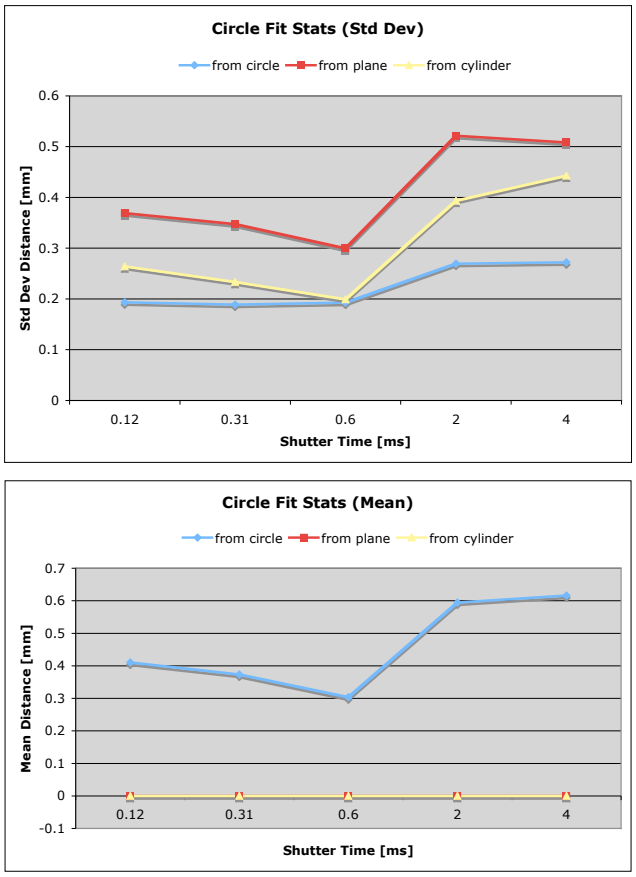

Figure 9: The statistics of the actual distances of the triangulated LED points (in 3D) to the fitted circle, for the rotating LED, and with varying τ_s. (The plane and cylinder differences are too small to see.)

5 Conclusions

In the future we plan on performing more controlled experiments, in particular for a camera-based motion capture system, where the human dynamics might be a factor in the precision one can achieve given the need for non-zero sample time. Mocap systems typically use LED-based "ring lights" to inject signal, thus improving the measurement noise curves, however it could be that non-trivial human motion can occur during the time it takes to turn on (pulse) the LEDs, take a picture, and turn off the LEDs. Controlled experiments on a large scale are hard, but we have some ideas.

References

[1] B. D. Allen and G. Welch. A general method for comparing the expected performance of tracking and motion capture systems. In *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 201–210, New York, NY, USA, 2005. ACM Press.

[2] R. Ashby. *Design for an Intelligence-Amplifier*, volume 34 of *Annals of Mathematics Studies, Automata Studies*. Princeton University Press, 1956.

[3] R. Ashby. *Design For a Brain*. John Wiley & Sons, New York City, N. Y., 2nd ed. edition, 1960.

[4] R. Azuma, H. N. III, M. Daily, and J. Leonard. Performance analysis of an outdoor augmented reality tracking system that relies upon a few mobile beacons. *Proceedings of 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2006)*, pages 101–104, October 2006.

[5] G. Bishop. *The Self-Tracker: A Smart Optical Sensor on Silicon*. Ph.d. dissertation, University of North Carlina at Chapel Hill, 1984. by Gary Bishop. ill. ; 29 cm. Thesis (Ph.D.)–University of North Carolina at Chapel Hill, 1984.

[6] J.-Y. Bouguet. Camera calibration toolbox for matlab.

[7] C. B. Chang and J. A. Tabaczyinski. Application of state estimation to target tracking. *IEEE Transactions on Automatic Control*, ac-29(2):98–109, February 1984.

[8] X. Chen. *Design of Many-Camera Tracking Systems For Scalability and Efficient Resource Allocation*. PhD thesis, Stanford University, 2002.

[9] L. Davis, E. Clarkson, and J. P. Rowland. Predicting accuracy in pose estimation for marker-based tracking. In *Proceedings of Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04)*, pages 28–35. Institute of Electrical and Electronics Engineers, IEEE Computer Society Press, October 2003.

[10] L. Davis, F. Hamza-Lup, and J. P. Rowland. A method for designing marker-based tracking probes. In *Proceedings of Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'04)*. Institute of Electrical and Electronics Engineers, IEEE Computer Society Press, November 2003.

[11] U. Erdem and S. Sclaroff. Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints. In *OMNIVIS*, page workshop, 2004.

[12] E. Foxlin, M. Harrington, and G. Pfeifer. Constellation: A wide-range wireless motion-tracking system for augmented reality and virtual set applications. In M. F. Cohen, editor, *Computer Graphics*, Annual Conference on Computer Graphics & Interactive Techniques, pages 371–378. ACM Press, Addison-Wesley, Orlando, FL USA, SIGGRAPH 98 conference proceedings edition, 1998.

[13] M. S. Grewal and P. A. Angus. *Kalman Filtering Theory and Practice*. Information and System Sciences Series. Prentice Hall, Upper Saddle River, NJ USA, 1993.

[14] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[15] G. Olague and R. Mohr. Optimal camera placement for accurate reconstruction. *Pattern Recognition*, 35(4):927–944, 2002.

[16] G. Reitmayr and T. Drummond. Going out: Robust model-based tracking for outdoor augmented reality. *Proceedings of 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2006)*, October 2006.

[17] A. State, G. Welch, and A. Ilie. An interactive camera placement and visibility simulator for image-based vr applications. In *Engineering Reality of Virtual Reality 2006 (3D Imaging, Interaction, and Measurement; IS&T-SPIE 18th Annual Symposium on Electronic Imaging Science and Technology)*, San Jose, CA, January 2006.

[18] A. Vorozcovs, A. Hogue, and W. Stuerzlinger. The hedgehog: a novel optical tracking method for spatially immersive displays. In *Virtual Reality, 2005. Proceedings. VR 2005. IEEE*, pages 83–89, 12–16 March 2005 2005.

[19] G. Welch and G. Bishop. An introduction to the Kalman filter: SIGGRAPH 2001 course 8. In *Computer Graphics*, Annual Conference on Computer Graphics & Interactive Techniques. ACM Press, Addison-Wesley Publishing Company, Los Angeles, CA, USA, SIGGRAPH 2001 course pack edition, August 12-17 2001.

[20] S. You, U. Neumann, and R. T. Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *IEEE Virtual Reality*, pages 260–267, Houston, TX USA, 1999.

[2] The circle fit optimization had six degrees of freedom: the 3D origin of the circle, two orientation angles, and the radius.