7. International Immersive Projection Technologies Workshop
9. Eurographics Workshop on Virtual Environments (2003)
J. Deisinger, A. Kunz (Editors)

Double Exponential Smoothing: An Alternative to Kalman Filter-Based Predictive Tracking

Joseph J. LaViola Jr.

Brown University Technology Center
for Advanced Scientific Computing and Visualization
PO Box 1910, Providence, RI, 02912, USA
jjl@cs.brown.edu

Abstract

We present novel algorithms for predictive tracking of user position and orientation based on double exponential smoothing. These algorithms, when compared against Kalman and extended Kalman filter-based predictors with derivative free measurement models, run approximately 135 times faster with equivalent prediction performance and simpler implementations. This paper describes these algorithms in detail along with the Kalman and extended Kalman Filter predictors tested against. In addition, we describe the details of a predictor experiment and present empirical results supporting the validity of our claims that these predictors are faster, easier to implement, and perform equivalently to the Kalman and extended Kalman filtering predictors.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Virtual Reality

1. Introduction

Predictive tracking algorithms represent an important component of any virtual or augmented reality system. Without these algorithms, virtual reality(VR) and augmented reality(AR) systems must use the current user pose to compute and display new images for each frame. This naive approach can cause problems since the user might be moving during the computation and display process, resulting in stale images and a display-to-user-motion synchronization mismatch. This mismatch degrades the user experience because dynamic tracking error produces perceived latency[1] and possible cybersickness[9].

In general, we require predictive tracking algorithms to be accurate, fast, robust (to different user motions), and simple to understand and implement. Accurate prediction is important since we want to mask latency and keep images fresh. Unfortunately, any prediction algorithm will introduce latency into the rendering pipeline because it takes some amount of time to make a prediction. Fast prediction algorithms are, therefore, an important requirement since we want to minimize any additional latency introduced into the virtual environment(VE). In addition, fast prediction algorithms do not have to predict as far into the future as slower ones to compensate for any computational overhead. Robustness is important for a predictive tracking algorithm to be useful in a VE, as it needs to handle a variety of different motion dynamics and styles in different applications. Finally, predictive tracking algorithms should be simple to understand and implement because we want them to be easily used in VE systems and applications.

Kalman and extended Kalman filter-based (KF/EKF) predictors have received considerable attention in the literature[1, 5, 8, 10, 12, 17] and appear to be the prediction method of choice by many researchers. These predictors can be derived in a number of ways depending on the underlying process and measurement models[18]. Most of the work with these predictors has been with derivative free motion models (i.e., the tracking system does not provide derivative information from rate gyros and/or acceleration sensors). Although Azuma and Bishop have shown superior performance with the use of derivatives in KF and EKF measurement models[1], our work focuses on KF/EKF predictors with derivative free motion models since most commercial tracking systems do not have sensors that measure velocity and acceleration. Previous work using these algorithms have shown that they, in general, are relatively fast and accurate for a variety of user motion dynamics and are fairly straightforward to understand and implement.

In this paper, we present a faster alternative to KF/EKF predictors with derivative free measurement models, using double exponential smoothing, a common technique in business and economic forecasting[3, 6, 14]. Double exponential smoothing, which has similarities with the α-β-γ filter[15] used in aircraft tracking, relies on the idea that user motion can be adequately modeled by a simple linear trend equation with slope and y-intercept parameters that vary slowly over time[4]. This idea is the basis for our assumptions that double exponential smoothing is an appropriate choice for predicting user motion. Additionally, these predictors are simpler to understand and implement than Kalman filter-based predictors. To support the validity of the double exponential smoothing predictors, we describe the results of a study that shows these new predictors are as accurate as KF/EKF predictors for a variety of user motion sequences.

In the next section, we describe the double exponential smoothing algorithms for predicting user position and orientation followed by a description of the KF/EKF predictors we are testing against. Then, we present our predictor experiment and the results which indicate the validity of our new algorithms. Finally, we discuss some areas for future work and present conclusions.

2. Double Exponential Smoothing-Based Prediction

Double exponential smoothing-based prediction (DESP) models a given time series using a simple linear regression equation where the y-intercept β_0 and slope β_1 are varying slowly over time[2]. An unequal weighting is placed on these parameters that decays exponentially through time so newer observations get a higher weighting than older ones. The degree of exponential decay is determined by the parameter $\alpha \in [0,1)$. We can use such an evolving regression equation to make user pose predictions.

To predict user position, we assume that at time $t-1$, we have the estimates $\vec{b}_0(t-1)$ and $\vec{b}_1(t-1)$ for $\vec{\beta}_0(t-1)$ and $\vec{\beta}_1(t-1)$ respectively. Note that each estimate is a vector representing the x, y, and z components of position. We also assume we have a new user position $\vec{p}_t$ at time t. To update the estimates of $\vec{\beta}_0(t-1)$ and $\vec{\beta}_1(t-1)$, we require two smoothing statistics defined by

$$\vec{Sp}_t = \alpha\vec{p}_t + (1-\alpha)\vec{Sp}_{t-1} \tag{1}$$

$$\vec{Sp}_t^{[2]} = \alpha\vec{Sp}_t + (1-\alpha)\vec{Sp}_{t-1}^{[2]}, \tag{2}$$

where the first equation smoothes the original position sequence and the second equation smoothes the $\vec{Sp}_t$ values.

Using $\vec{Sp}_t$ and $\vec{Sp}_t^{[2]}$, we can calculate $\vec{b}_0(t)$ and $\vec{b}_1(t)$ with the following:

$$\vec{b}_1(t) = \frac{\alpha}{1-\alpha}(\vec{Sp}_t - \vec{Sp}_t^{[2]}) \tag{3}$$

$$\vec{b}_0(t) = 2\vec{Sp}_t - \vec{Sp}_t^{[2]} - t\vec{b}_1(t). \tag{4}$$

Given these estimates, the user's position is predicted time τ into the future with

$$\vec{p}_{t+\tau} = \vec{b}_0(t) + \vec{b}_1(t+\tau). \tag{5}$$

With some algebraic manipulation (see [2] for details), our position prediction equation is

$$\vec{p}_{t+\tau} = \left(2 + \frac{\alpha\tau}{(1-\alpha)}\right)\vec{Sp}_t - \left(1 + \frac{\alpha\tau}{(1-\alpha)}\right)\vec{Sp}_t^{[2]}. \tag{6}$$

We predict the user's orientation using the same formulations for position prediction except that quaternions are used instead of 3D vectors. Therefore, substituting q for $\vec{p}$, equations 1, 2, and 6 are applied to each of the four quaternion components and an explicit renormalization is done to make sure the predicted quaternion is on the unit sphere.

The DESP algorithm predicts a user's pose an integral multiple (i.e., τ) of Δt (i.e., 1.0 divided by the sampling rate) into the future. For example, if the sampling rate is 20 Hz, then Δt is 50 milliseconds, and the prediction scheme can only predict user poses at $50, 100, 150, \ldots, n$ milliseconds into the future with $\tau = 1, 2, 3, \ldots, i$. If τ is not an integer the algorithm provides no answer.

To overcome this limitation and predict user poses any time into the future, we have extended the basic DESP algorithm. If τ is not an integer then we make a low and high prediction using $\lfloor\tau\rfloor$ and $\lceil\tau\rceil$ respectively. Then, we interpolate between these two predicted values to find the prediction at the correct future time. For position, using linear interpolation,

$$\vec{p}_{t+\tau} = (\vec{phi}_{t+\lceil\tau\rceil} - \vec{plo}_{t+\lfloor\tau\rfloor})(\tau - \lfloor\tau\rfloor)) + \vec{plo}_{t+\lfloor\tau\rfloor}. \tag{7}$$

Since we are representing orientation with quaternions, we use spherical linear interpolation[13] (i.e., SLERP) to find the predicted orientation at the correct time. This predicted orientation is given by

$$q_{t+\tau} = \frac{qlo_{t+\lfloor\tau\rfloor}\sin(1-\rho)\Omega + qhi_{t+\lceil\tau\rceil}\sin\rho\Omega}{\sin\Omega}, \tag{8}$$

where $\rho = \tau - \lfloor\tau\rfloor$ and $\Omega = \arccos(qlo_{t+\lfloor\tau\rfloor} \odot qhi_{t+\lceil\tau\rceil}))$.

The $\odot$ symbol stands for a quaternion dot product in this case. Note that we make sure to normalize $qlo_{t+\lfloor\tau\rfloor}$ and $qhi_{t+\lceil\tau\rceil}$ before applying the SLERP function to ensure they represent rotations on the unit sphere. Finally, to initialize both the position and orientation predictors we simply set the smoothing statistics at time 0 to the initial observations in a motion sequence.

3. Kalman Filter-Based Prediction

We develop the KF and EKF-based predictors so as to make comparisons with DESP. To describe all the details of the KF and EKF predictors is beyond the scope of this paper. Therefore, we present the basics using a more algorithmic description. See [1, 11, 16, 18] for more detail on Kalman and extended Kalman filtering.

3.1. Kalman Filter-Based Predictor for Position

The Kalman filter is a set of mathematical equations that fuse information from multiple sources; it uses a predictor-corrector mechanism to find an optimal estimate in the sense that it minimizes the estimated error covariance[16]. In other words, the filter uses an underlying process model to make an estimate of the current system state and then corrects the estimate using any available sensor measurements. Then, after the correction is made, we use the process model to make a prediction.

For position, we use a simple position/velocity (PV) model given by

$$\frac{d\vec{p}}{dt} = \vec{v}. \tag{9}$$

Note that we use three separate Kalman filter-based predictors, one for each translational axis, which simplifies the computations. We can trivially express each component of these process models in state space form and determine the system dynamics matrix $\mathbf{F}$. Since we are using polynomial Kalman filters (see [18] for more details), we can find the fundamental matrix $\phi(\mathbf{t})$ that propagates the state vector through time using the Laplace transform,

$$\phi(\mathbf{t}) = \mathcal{L}^{-1}[(\mathbf{sI} - \mathbf{F})^{-1}]. \tag{10}$$

Substituting Δt for $\mathbf{t}$ we get the discrete form of $\phi(\mathbf{t})$, ϕ_k. For a single axis of the PV process model

$$\phi_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}. \tag{11}$$

The advantage of using the fundamental matrix is that explicit numerical integration techniques are not needed.

Given the state vector at step $k-1$, we propagate the process model through time using ϕ_k.

$$\hat{x}_k^- = \phi_k \hat{x}_{k-1}, \tag{12}$$

where $\hat{x}_k^-$ is the *a priori* state estimate. The correction step, which fuses any sensor measurements with $\hat{x}_k^-$ determines the *a posteriori* state estimate,

$$\hat{x}_k = \hat{x}_k^- + \mathbf{K}_k(z_k - \mathbf{H}\hat{x}_k^-), \tag{13}$$

where $\mathbf{K}_k$ is the Kalman gain or blending factor and $\mathbf{H}$ is the constant measurement matrix used to combine the measurement vector z_k with $\hat{x}_k^-$. The Kalman gain is computed using the matrix Ricatti equations,

$$\begin{aligned} \mathbf{P}_k^- &= \phi_k \mathbf{P}_{k-1} \phi_k^T + \mathbf{Q} && (14) \\ \mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}^T (\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R})^{-1} && (15) \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}) \mathbf{P}_k^-, && (16) \end{aligned}$$

where $\mathbf{P}_k^-$ is the *a priori* estimate of the error covariance matrix, $\mathbf{Q}$ is the constant process noise covariance, $\mathbf{R}$ is the constant measurement noise covariance, and $\mathbf{P}_k$ is the *a posteriori* estimate of the error covariance matrix.

Once the *a posteriori* state vectors are computed, we can use the fundamental matrices to compute the predicted position by setting $\Delta t = t_{pred}$, the value which determines how far to predict into the future.

3.2. Extended Kalman Filter-Based Predictor for Orientation

We are using quaternions to represent rotations. Therefore, the process models used to propagate the state vector through time are nonlinear. Since the standard Kalman filter is a linear estimator, we need to linearize about the current mean and covariance, which is referred to as the extended Kalman filtering[16].

The process model we use is an orientation/angular velocity (OV) model defined by

$$f = \frac{dq}{dt} = \frac{1}{2} q\omega, \tag{17}$$

where ω is a pure vector quaternion. In this case, a single EKF is used, where the state vector contains the current quaternion as well as angular velocity. Since these process models are nonlinear, we cannot use ϕ_k to propagate the state vector through time, so we use 4th order Runge-Kutta integration. However, we can approximate ϕ_k which we use in the correction step by taking the Taylor expansion of $\phi(\mathbf{t})$ around the system dynamics matrix

$$\mathbf{F}_{[i,j]} = \frac{\partial f_{(i)}}{\partial x_{(j)}}(x_k^-), \tag{18}$$

a Jacobian matrix which linearizes the process function f, and substituting Δt for $\mathbf{t}$. For the correction step we can reuse equations (14)-(16) with some minor modifications. First,

$$\mathbf{H}_{[i,j]} = \frac{\partial h_{(i)}}{\partial x_{(j)}}(x_k^-), \tag{19}$$

a Jacobian matrix that linearizes around the nonlinear measurement function h. In our case this function is just quaternion normalization. Second, $\mathbf{F}, \mathbf{H}, \mathbf{Q}$ are now denoted by $\mathbf{F}_k, \mathbf{H}_k, \mathbf{Q}_k$ since they are no longer constant and must be updated at each filtering step. Note that after we compute the *a posteriori* state vector, the quaternion is renormalized.

Once the *a posteriori* state vector is computed, predictions are made with the OV model,

$$q_{pred} = q + (t_{pred} - t)\frac{1}{2}q\omega. \tag{20}$$

3.3. KF/EKF Parameters and Initialization

Each predictor requires a $\mathbf{Q}$ ($\mathbf{Q}_k$ in the EKF case) and $\mathbf{R}$ matrix which represent the process noise covariance and the measurement noise covariance. $\mathbf{R}$ is determined empirically and accounts for the uncertainty in the tracking data. Setting these matrices properly go a long way toward making the predictors robust. We determine $\mathbf{Q}$ and $\mathbf{Q}_k$ using the continuous process noise matrix $\tilde{\mathbf{Q}}$ which assumes that the process noise always enters the process model on the highest derivative[18]. For example, with the PV model

$$\tilde{\mathbf{Q}} = \phi_s \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \tag{21}$$

where ϕ_s is a scaling parameter which acts as a confidence value for how sure we are that the process model is an accurate description of the real world. Therefore, we compute $\mathbf{Q}$ and $\mathbf{Q}_k$ using

$$\int_0^{\Delta t} \phi(\tau)\tilde{\mathbf{Q}}\phi(\tau)^T \, dt. \tag{22}$$

The KF and EKF predictors also need to be initialized on startup. The position values and quaternion in the state vectors at time 0 are simply set to the first observations in the motion sequence and the velocity and angular velocity components are set to zero. The *a priori* estimate of the error covariance and the elements in the these matrices are set to 0 for the off-diagonal entries and to relatively large numbers in the diagonal entries. In the KF case, the diagonals are set to 100, and in the EKF case, the quaternion variance diagonals are set to 1 and the angular velocity variances are set to 100.

4. Prediction Algorithm Experiment

Sections 2 and 3 have shown the inherent complexity of the KF and EKF predictors compared to the double exponential smoothing prediction approach. The KF and EKF predictors require the calculation of partial derivatives, matrix multiplication and inverses, and a more complex infrastructure while the double exponential smoothing predictors only need a small set of simple equations. In this section, we describe an experiment comparing both running time and numerical accuracy between double exponential smoothing-based prediction, the KF and EKF predictors, and no prediction at all.

4.1. Experimental Setup

Six datasets (three head and three hand) were used in our study representing a variety of different motion dynamics collected from applications and interaction techniques developed in our Cave facility. Each dataset is about 20 seconds in length captured from an Intersense IS900 tracking system. The names and descriptions of the experimental datasets are as follows:

- HEAD1 – simple head movement where the user stands roughly in place and rotates to view the display screens
- HEAD2 – head movement from the user both walking and looking around in the Cave
- HEAD3 – head motion from the user examining a fixed object in order to gain perspective about it's structure
- HAND1 – hand motion used to navigate the user through the virtual world
- HAND2 – hand motion used in object selection, manipulation, and placement
- HAND3 – complex, free-form hand motion used in 3D painting.

Each dataset was tested with sampling rates of 70 and 180Hz for prediction times of 50 and 100ms giving us four different test scenarios. We use a small Monte Carlo simulation on each test scenario (5 runs) since random Gaussian noise is added to the motion signals, which is used to simulate jittery tracking data. Constant values were set for the random noise variances, 5e-5 for position and 5e-6 for orientation providing noise added to the motion signals with a Gaussian distributed range of ± 0.021 inches for position and ± 1.19 degrees for orientation. All tests were run on a AMD Athelon XP 1800+ with 512Mb of main memory.

ϕ_s Parameter Values for the KF/EKF Predictors				
	(70Hz,50ms)	(70Hz,100ms)	(180Hz,50ms)	(180Hz,100ms)
HEAD1	1	1	1	1
HEAD2	1	1	1	1
HEAD3	1	1	1	1
HAND1	3	3	2	2
HAND2	2	2	1	1
HAND3	40	103	35	50

Table 1: *The ϕ_s parameter values used across the different datasets and test scenarios for both the KF and EKF predictors. In this case, the same parameter value is used for both position and orientation prediction.*

α Parameter Values for the Double Exponential Smoothing Predictors				
	(70Hz,50ms)	(70Hz,100ms)	(180Hz,50ms)	(180Hz,100ms)
	Pos : Rot	Pos : Rot	Pos : Rot	Pos : Rot
HEAD1	0.34 : 0.46	0.34 : 0.46	0.18 : 0.25	0.18 : 0.25
HEAD2	0.41 : 0.48	0.41 : 0.48	0.26 : 0.27	0.26 : 0.27
HEAD3	0.44 : 0.39	0.44 : 0.39	0.24 : 0.21	0.24 : 0.21
HAND1	0.53 : 0.57	0.58 : 0.57	0.30 : 0.62	0.31 : 0.62
HAND2	0.47 : 0.53	0.45 : 0.55	0.26 : 0.31	0.26 : 0.31
HAND3	0.81 : 0.95	0.87 : 0.97	0.49 : 0.62	0.51 : 0.66

Table 2: *The α parameter values used across the different datasets and test scenarios for the double exponential smoothing position and orientation predictors. The numbers to the left of the colon are position α values while the numbers to the right of the colon are the orientation α values.*

4.2. Evaluation Method

Comparing predicted output with reported user poses is problematic since these records have noise and small distortions associated with them. Thus, any comparison with the recorded data would count tracking error with the prediction error. We obtain "ground truth" datasets by passing them through a zero phase shift filter to remove high frequency noise. We determine the lowpass and highpass filter parameters by examining each signal's power spectrum. Depending on the particular dataset, the lowpass/highpass pairs were anywhere between 1/3 and 6/8 Hz. This cleaning step (see Figure 1) gives us the truth datasets we need to test against and makes it easy to add noise of known characteristics for simulating jittery tracking data. With the truth datasets, we can calculate the root mean square error for each test (RMSE) and take the average over the Monte Carlo simulation runs.

Since a predicted pose consists of a position and orientation, we calculated the running times for the prediction algorithms by grouping the KF/EKF predictors together and the position and orientation DESPs together. Each test makes a number of predictions depending on sampling rate, so we took an average over all the predictions for a given test. We then took a random sampling of 100 tests from the KF/EKF predictor, the DESP, and no prediction at all and computed the overall average running times.

4.3. Prediction Parameter Determination

Algorithm parameter tuning plays an important role in determining predictor accuracy. For each dataset and test scenario, we first perform a parameter search routine to find the best parameter settings for that particular configuration. This approach is not ideal from a practical standpoint since it is a difficult problem to know exactly what the user's mo-

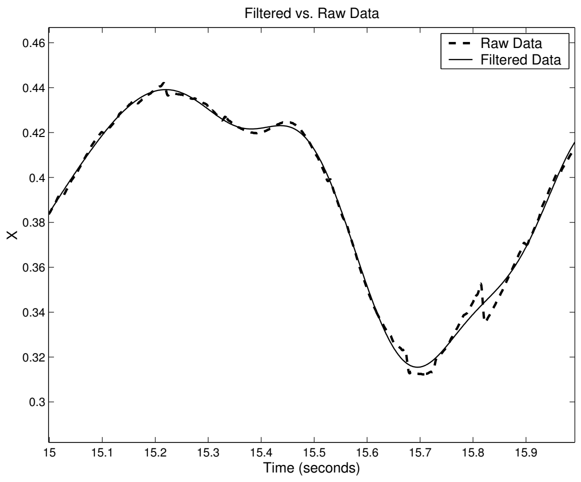

Figure 1: *A segment of the HEAD1 dataset's X component showing the raw and clean signals.*

tion will be in a real application. However, we chose this approach for two reasons. First, we want to examine how the different motion datasets affect the optimal parameter values. Second, for purposes of this experiment, we want to examine the predictors on a level playing field so we can judge there performance under optimal settings. Note that we are currently exploring ways of determining parameter settings adaptively in order to achieve the best predictor performance possible for any user motion.

For the KF/EKF predictors, we need to determine the $\mathbf{R}$ and $\mathbf{Q}$ ($\mathbf{Q_k}$ for the EKF) covariance matrices. Since we know the variances of the Gaussian white noise we are injecting into the motion signals, we set the off-diagonal entries of the $\mathbf{R}$ to zero and set the diagonal entries to be the value of the noise variance parameters (5e-5 position and 5e-6 for orientation). Thus we are making the assumption that our measurement noise is based on the variability of a stationary tracker. As shown in Section 3.3, we calculate the $\mathbf{Q}$ and $\mathbf{Q_k}$ matrices using equation 22 leaving ϕ_s as our free parameter. The search routine ran over different integer values for ϕ_s and Table 1 shows the values we used for ϕ_s in our experiment.

For DESP, we also did a parameter search for the best values of α. For each dataset and testing scenario, values for α (between 0 and 1) were determined for both the position and orientation predictors. These values are shown in Table 2.

5. Results and Discussion

Table 3 shows the results of the running time experiments. From the results, we can see that DESP runs approximately 135 times faster than the KF/EKF predictors. For a frame of reference, we also timed how long it would take to simply take the previous user pose and use it as the predicted pose. These timings show that DESP does not take much longer to predict a pose than simply using no prediction at all.

	KF/EKF	DESP	No Prediction
Average:	458.7803	3.3360	1.1912
Variance:	24.2354	0.0285	0.0152

Table 3: *Average running times(μs) and variances for the different predictors.*

To examine the relationship between the accuracy results for both predictors, we looked at how many times better the given predictor performs than no prediction at all. This is easily calculated by taking the RMSE results from each predictor for all test scenarios and dividing them by the corresponding RMSE for no prediction.

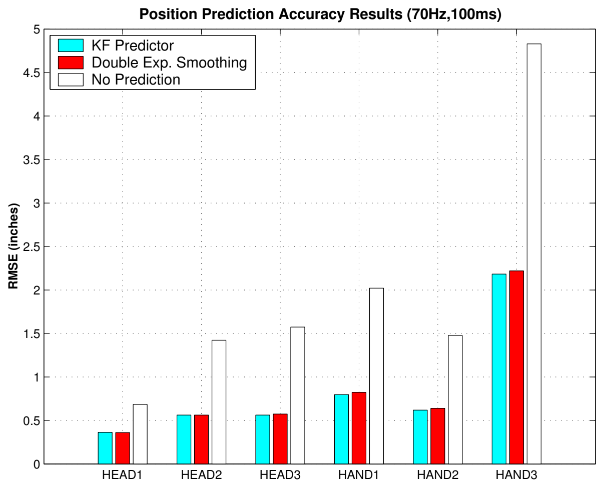

Figure 2: *Prediction accuracy results from the 70Hz, 100ms test scenario comparing DESP with KF prediction and no prediction at all across head and hand motion datasets.*

Table 4 shows these "times better" metrics for the KF/EKF and DESP in relation to no prediction at all. The table shows that on average both predictor types perform between two to three times better than no prediction at all for all cases which first confirms previous results about KF/EKF predictors[1] and second indicates DESP has roughly the same performance as the KF/EKF algorithms. Additionally, the differences between KF/EKF and DESP "times better" metrics are no larger than 0.1 further indicating their similar performance. On average, DESP gets two to three times better prediction accuracy than no prediction at all with a cost of approximately 2 additional μs whereas the KF/EKF predictors also get to two to three times better prediction accuracy but with an additional cost of approximately 456 μs.

	KF	DESP
Head Position	2.53	2.50
Hand Position	2.69	2.59
	EKF	DESP
Head Orientation	2.69	2.60
Hand Orientation	2.06	2.00

Table 4: *Performance of the KF/EKF predictors and DESP in relation to no prediction at all.*

Figures 2 and 3 show some specific results from the experiment that are representative of all the scenarios run. The graphs show the relatively minute differences between the prediction accuracies for KF/EKF and the double exponential smoothing predictors. Although for the majority of the test runs, the KF/EKF predictors performed slightly better than double exponential smoothing, the average differences between the RMSE numbers was only 0.0163 of an inch for position and 0.0709 of a degree for orientation. These differences show any additional accuracy improvements obtained with the KF/EKF predictors are negligible.

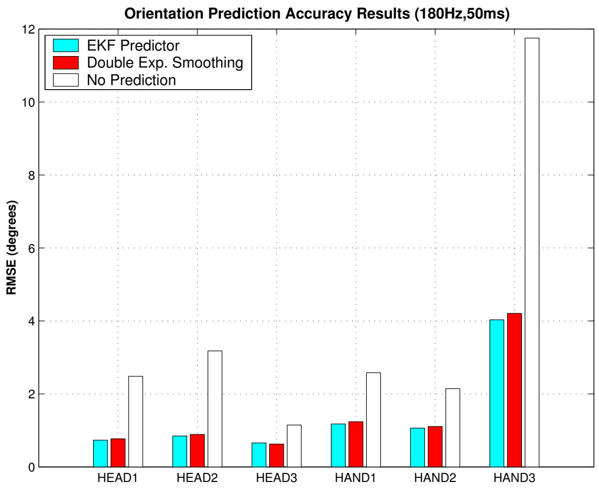

Figure 3: *Accuracy results from the 180Hz, 50ms test scenario comparing DESP with EKF prediction and no prediction at all across head and hand motion datasets.*

The results from this experiment provide empirical evidence showing that the DESP algorithms are significantly faster than the KF/EKF predictors with roughly the same accuracy in RMSE. These results are consistent across the different datasets and testing scenarios. In addition to these results, the DESP algorithms are conceptually easier to understand and implement than the KF/EKF predictors and this can be seen from the descriptions presented in Sections 2 and 3.

One possible negative characteristic of DESP compared to the KF/EKF predictors is that they will be slightly more difficult to apply in a real time tracking system that produces a variety of signals with different motion dynamics. The algorithmic parameter searches (results are shown in Tables 1 and 2) show that the ϕ_s parameters for the KF/EKF predictors have good stability in that the majority of them are set to one. This indicates that the analytical determination of $\mathbf{Q}$ and $\mathbf{Q_k}$ represent good process noise covariance matrices. The stability makes the KF/EKF predictors easier to use in a real tracking system in the general case. Only the HAND3 data sets require inflating these matrices with ϕ_s which is probably due to the free form nature of these motion signals.

The α values for the DESP algorithms have more variation across the different datasets. If the application domain and the types of user motion are known, this variation is not a problem since similar datasets can be used to find α without much loss in performance. However, in the general case, these predictors will be slightly more difficult to apply. One possible approach to making DESP more robust is to build an adaptive parameter tuning model which examines a window of previous user poses then calculates the value for α on the fly based on the signal's characteristics. This approach should add relatively little computational overhead to the DESP algorithm assuming the window sizes are relatively small. Note that the KF/EKF predictors would also benefit from such an approach especially when dealing with free form hand motion (i.e., HAND3).

The significantly faster running times and simplicity of DESP gives them a number of important benefits. Because it takes the DESP algorithms only slightly more time to compute than doing no prediction at all, they introduce hardly any latency into the VE system providing more time for rendering. Another benefit is in dealing with a tracking system's sampling rate. Sampling rate plays an important role in prediction accuracy since a higher sampling rate will improve a predictor's performance. The short running times of DESP makes them better suited to handle tracking systems with very high sampling rates. DESP will not have to skip tracker samples due to the prediction computation. DESP's simplicity makes them easier to implement than the KF/EKF predictors and also make the algorithms suitable for implementation on a microcontroller that could be incorporated directly into the tracking hardware (assuming an accurate parameter setting model). These are just a few of the many possible benefits that the double exponential smoothing predictors can provide which we are continuing to explore.

6. Future Work

Although our empirical results present strong evidence to support our claims, we would like to obtain more evidence both empirically and analytically. Empirically, we wish to continue to run experiments testing different datasets from other tracking systems, applications, and interaction techniques. Further experimentation will aid in showing the robustness of the double exponential smoothing techniques. Analytically, there has been work done that has shown mathematical equivalence between the Kalman filter and exponential smoothing under certain conditions[7]. We would like to apply these analytical techniques to our problem domain and see if we can find such an equivalence between both the KF/EKF predictors and their DESP counterparts. This work would provide a much stronger theoretical basis for the validity and accuracy of the DESP algorithms in the context of user motion prediction. We also plan to develop the adaptive parameter tuning models discussed in Section 5. These models will not only will make DESP more applicable in the general case but any prediction algorithm that needs parameter tuning. Finally, we wish to incorporate DESPs into our tracking system for work in our Cave facility.

7. Conclusion

We have presented novel prediction algorithms using double exponential smoothing for predicting a user's pose. We have described the technique and contrasted it with Kalman and extended Kalman filter-based predictors that employ derivative free measurement models. Our experiments indicate these prediction algorithms run approximately 135 times faster than a KF/EKF predictor with roughly the same accuracy. With this paper, we have shown that double exponential smoothing predictors are a viable alternative for predictive tracking over their KF/EKF predictor counterparts due to their simplicity, speed, and accuracy and have shown they should to be considered when employing a predictive tracking algorithm in a virtual environment.

Acknowledgments

Special thanks to Gary Bishop, Greg Welch, John Hughes, and Andy van Dam for valuable guidance and discussion. This work is supported in part by the NSF Graphics and Visualization Center, IBM, the Department of Energy, Alias/Wavefront, Microsoft, Sun Microsystems, and TACO.

References

1. Azuma, Ronald and Gary Bishop. Improving Static and Dynamic Registration in a See-Through HMD. In *Proceedings of SIGGRAPH'94*, 197-204, 1994.

2. Bowerman, Bruce J. and Richard T. O'Connell. *Forecasting and Time Series: An Applied Approach*. Duxbury Thomson Learning, 1993.

3. Box, G. Understanding Exponential Smoothing: A Simple Way to Forecast Sales and Inventory, In *Quality Engineering*, 4(1):143-151, 1991.

4. Chatfield, Chris. *Time-Series Forecasting*. Chapman and Hall/CRC, 2001.

5. Friedmann, Martin, Thad Starner, and Alex Pentland. Device Synchronization Using and Optimal Linear Filter. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, 57-62, 1992.

6. Gardner Jr., Everette S. and David G. Dannenbring, Forecasting with Exponential Smoothing: Some Guidelines for Model Selection, In *Decision Sciences*, 11., 370-383, 1980.

7. Harvey, A. C. *Forecasting, Structural Time Series Models, and the Kalman Filter*, Cambridge University Press, 1991.

8. Kiruluta, Andrew, Moshe Eizenman, and Subbarayan Pasupathy. Predictive Head Movement Tracking Using a Kalman Filter. In *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, 27(2):326-331, April 1997.

9. LaViola, Joseph. A Discussion of Cybersickness in Virtual Environments, *SIGCHI Bulletin*, 32(1):47-56, January 2000.

10. Liang, Jiandong, Chris Shaw, and Mark Green. On Temporal-Spatial Realism in the Virtual Reality Environment. In *Proceedings of UIST'91*, 19-25, 1991.

11. Maybeck, Peter S. *Stochastic models, estimation, and control*. Volume 1, Academic Press, 1979.

12. Mazuryk, Tomasz and Michael Gervautz. Two-Step Prediction and Image Deflection for Exact Head Tracking in Virtual Environments. In *EUROGRAPHICS'95*, 29-41, 1995.

13. Shoemake, Ken. Animating Rotations with Quaternion Curves. In *Proceedings of SIGGRAPH 85*, ACM Press, 245-254, 1985.

14. Snyder, R. D., A. B. Koehler, and J. K. Ord. Forecasting for Inventory Control with Exponential Smoothing. In the *International Journal of Forecasting*, 18(1), 5-18, January-March 2002.

15. Tenne, D. and T. Singh. Characterizing Performance of α-β-γ Filters. In *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):1072-1087, July 2002.

16. Welch, Greg and Gary Bishop. An Introduction to the Kalman Filter. Technical Report TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, 1995.

17. Wu, Jiann-Rong and Ming Ouhyoung. On Latency Compensation and Its Effects on Head-Motion Trajectories in Virtual Environments. *The Visual Computer* 16(2): 79-90, 2000.

18. Zarachan, Paul and Howard Musoff. *Fundamentals of Kalman Filtering: A Practical Approach*. Progress in Astronautics and Aeronautics, Volume 190, American Institute of Aeronautics and Astronautics, Inc., 2000.