The Predictive Tracking Algorithm Testing Suite: A Tool for Developing and Analyzing Predictive Tracking Algorithms

Joseph J. LaViola Jr.

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-02-07
July 2002

The Predictive Tracking Algorithm Testing Suite: A Tool for Developing and Analyzing Predictive Tracking Algorithms

Joseph J. LaViola Jr.
Brown University
Department of Computer Science, Box 1910
Providence, RI 02912
jjl@cs.brown.edu

Abstract

We present The Predictive Tracking Algorithm Testing Suite, a software platform for developing and examining different predictive tracking algorithms across a variety of system and algorithmic parameters. The suite allows for the evaluation of predictors across a multidimensional parameter space which includes sampling rate, prediction time, noise variance, specific algorithmic parameters, and different types of user motion. By providing a general testing suite, VR and AR system designers can ask a variety of questions about how to maximize predictor performance in tracking systems and applications. We describe the features of the testing suite and it's underlying software architecture as well as our data preparation methodology. We also present some simple experiments that show the testing suite's utility by examining four Kalman filter-based predictors (two for position and two for orientation) on head and hand motion datasets.

Keywords: Kalman filter, Extended Kalman filter, prediction algorithms, sampling rate, prediction time, noise variance

1 Introduction

People have many problems when interacting with virtual environments. One of these problems is that the displayed image is often presented incorrectly to the user because it does not reflect the user's current position and orientation (i.e., pose). To remedy this problem, the image displayed to the user should be based on the user's pose at *display time*, the time right after scanout when the image is displayed on the screen.[1] However, it is generally difficult to precisely know the user's display time pose at *image computation start time*, the time at which the computation of the display image begins and the user's pose is required.

One naive but rather common technique for determining the user's pose at display time is to simply assume that the user's image computation start time pose is the display time pose. If the user does not move during *image computation time*, the time between display time and image computation start time, the technique works well.[2] However, if the user does move during image computation time the displayed image will not be synchronized with the user's current pose.

A more sophisticated approach is to estimate or predict the user's display time pose from prior observations at image computation start time. Ideally, this approach would allow the image computation process to produce an image that is synchronized with the user's motion at display time. However, it is difficult to accurately determine where the user will be at display time even with the most sophisticated prediction algorithms. With this approach there are a number of parameters, such as device sampling rate and noise as well as the type of user motion, the length of the prediction time, and specific algorithmic parameters that contribute to the overall prediction algorithm error. Therefore, having a predictive tracking algorithm testing suite for understanding how these different variables affect the prediction accuracy of the user's pose would provide a valuable tool to tracking system and virtual and augmented reality application developers.

In theory, such a tool would aid in answering a variety of questions about how to minimize user pose prediction error and maximize display-to-user-motion synchronization. In addition, these questions could be asked in a number of different ways. For example, the testing suite could help determine what prediction algorithm works best, given specific values for the parameters shown above. In another case, given a particular prediction algorithm, determining what kind of error one could expect and what the values are for the system parameters that influence it would be useful for VR and AR system designers. Finally, given an error metric, the tool could determine the prediction algorithm(s) that fit under a particular error threshold and what the values for the system parameters need

[1] For the purposes of this paper, we will assume that *display time* is an instant in time instead of a time interval.

[2] Note that even if the user does not move there still would be some error associated with the displayed image due to noise from the input device.

to be to maintain that level of performance. More specifically, the following list represents some example questions which typify how a testing suite could be used.

1. With hand motion data, a tracker running at 60Hz with a noise variance of 0.0025 inches in position and 0.5 degrees in orientation, and prediction times between 50 and 100ms, what prediction algorithm should I use which has an average prediction error under 1 inch in position and 3 degrees in orientation?

2. I want my prediction algorithm output error to be below 0.2 inches in position and 2 degrees in orientation. What prediction algorithms will fit my needs if I am estimating the user's head pose? hand pose?

3. How far can I predict into the future with a given prediction algorithm, a sampling rate of 20, 40, or 80 Hz, and noise variance of 0.003 inches in position and 1 degree in orientation?

4. Does prediction algorithm A perform better than B, given I am predicting with hand tracking data vs. head tracking data?

5. How robust is prediction algorithm A across different sampling rates? prediction times? noise variances?

This paper presents The Predictive Tracking Algorithm Testing Suite, a software platform for developing and testing prediction algorithms in the context of virtual and augmented reality. In addition to describing the testing suite, we also demonstrate the features of the suite by showing the results of some simple experiments on how four Kalman filter-based predictors are affected by different sampling rates, prediction times, and noise variances for both head and hand motion data. We also discuss how the empirical data could be used to provide insight into three hypothetical questions which might be asked when developing a tracking system or VR application.

1.1 Paper Organization

In the next section we present some related work on predictive tracking algorithm performance analysis and briefly discuss their similarities and differences to our work. Section 3 describes The Predictive Tracking Algorithm Testing Suite by examining its features and software architecture as well as our motion data preparation methodology. Sections 4 and 5 present some simple experiments which show the utility of the testing suite. Finally, we discuss some future work for extending the testing suite and using it to develop general predictive tracking algorithm models.

2 Related Work

There has been a significant amount of work done on predictive tracking algorithms in both augmented and virtual reality. However, most of this work examined these algorithms with specific systems in mind and did not study their performance more generally. For example, Azuma and Bishop developed Kalman filter-based predictors for a see-through HMD utilizing velocity and acceleration sensors[1]. Three separate head motion sequences were studied along with two different predictors. However, differing sampling rates, prediction times, and sensor noise variances were not taken into account. Friedmann et.al. used Kalman filter-based predictors as well, but their studies were limited to simple hand motion (i.e., drumming) and were not thoroughly tested across different system parameters[3]. This is one of the few cases where predictive tracking was used with hand motion data. Our work presents one of the first significant experiments on predictive hand tracking. Other Kalman filter-based predictors have been developed and studied with similar restrictions on system parameters and user motion[5, 7, 9].

Besides Azuma and Bishop[1], there has been little work on comparing different predictive tracking algorithms. Wu and Ouhyoung[12] presented experiments comparing a Kalman filter-based predictor, simple extrapolation, and Grey System prediction[13] but once again did little with varying system parameters such as sampling rate and prediction time.

Finally, Azuma and Bishop's frequency-domain analysis of predictive tracking algorithms is similar to our work in that they developed theoretical formulae for characterizing predictor behavior based, in part, on prediction time and the motion signal's power spectrum[2]. With these formulae, extensive comparative studies are not as important since these "closed form solutions" provide a powerful measure of predictor accuracy. However, these theoretical formulae do have limitations in that they do not take into account sampling rate or static sensor error, and they were only developed for two specific predictors. In addition, they were not developed with hand motion in mind and do not take nonlinear or adaptive predictors into account. Extending the concepts behind these theoretical formulae to include other existing and novel prediction algorithms and more system input parameters represents a fundamental and significant extension to previously documented results. Although our work is mostly empirical, we believe that our testing suite provides the ability for more general and robust predictor analysis which we demonstrate with the example experiments presented in Section 4 and 5.

3 Predictive Tracking Algorithm Testing Suite

To develop predictive tracking algorithms and robustly test them across the multidimensional parameter space, we require a tool that makes it easy to run a variety of different test scenarios. In addition, we require useful error metrics which help us to gauge predictor performance. The Predictive Tracking Algorithm Testing Suite provides such a tool and makes it easy to perform interactive and automatic testing procedures. Automatic testing procedures allow for tests across different system parameters while the interactive tests allow us to look at specific parameter configurations in more detail. In the following subsections, we describe the testing suite functionality, our motion data preparation process and the suite's software architecture.

3.1 Testing Suite Functionality

The testing suite provides a number of useful features. It currently has over 20 different predictive tracking algorithms including simple polynomial extrapolation methods, exponential smoothing algorithms, Grey System prediction[13], and a variety of Kalman, extended Kalman, and unscented Kalman filter-based predictors[10].

By setting specific parameters (i.e., sampling rate, prediction time, noise variance, and algorithmic parameters), users can run a single test which provides global and component wise error results for a number of different error metrics. The suite also provides testing procedures for examining predictors across different sampling rates, prediction times, error variances, and algorithmic parameters (depending on the particular predictor). These tests are single variable tests in that one parameter varies while the rest stay constant according to user specifications. In addition, the suite provides a full system parameter test where sampling rate, prediction time, and error variance are all variables while algorithmic parameters stay constant. This test can be used for collecting data points across the whole multidimensional space.

There are currently three different error metrics, root mean square (RMSE), max norm, and percent better, implemented in the testing suite. All of these error metrics are computed for global position and orientation as well as the individual components of the position vector and quaternion. For the purposes of this paper, we only describe the global error metrics. Given a sequence of predicted poses (denoted by subscript p) and truth poses (denoted by subscript t), the root mean square position error, $RMSE_{pos}$, is

$$\sqrt{\frac{1}{n}\sum_{i=0}^{n-1} l_i^2} \tag{1}$$

where $RMSE_{pos}$ is in inches and

$$l_i = \sqrt{(x_{p_i} - x_{t_i})^2 + (y_{p_i} - y_{t_i})^2 + (z_{p_i} - z_{t_i})^2}. \tag{2}$$

For root mean square orientation error with truth quaternion q_{t_i} and predicted qauternion q_{p_i},

$$RMSE_{rot} = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} e_i^2} \tag{3}$$

where $RMSE_{rot}$ is in degrees and

$$e_i = \frac{2(180)}{\pi} \arccos\left((q_{t_i}(q_{p_i})^{-1})[w]\right). \tag{4}$$

In addition to RMSE, which shows how predictors are performing on average, we also utilize the max norm error metric which simply finds the maximum distance between the predicted pose and the truth pose. In other words,

$$MAX_{pos} = max(l_i), i = 0...n-1 \tag{5}$$

$$MAX_{rot} = max(\mid e_i \mid), i = 0...n-1 \tag{6}$$

help to determine how predictors are performing in the worst case.

Finally, the percent better error metric determines how predictors are performing relative to the random walk predictor. The random walk predictor is just the naive solution to the prediction problem where the last pose from the tracking system is used as the predicted pose. We compute this metric by counting the number of times the predicted pose is closer to the true pose than the random walk pose and divide by the total number of predictions made.

3.2 Data Preparation

Our test data has to be in a format general enough to handle the variety of different predictive tracking algorithms as well as the different testing procedures. For example, we wish to make it easy to add noise of different variances to the motion signals and to easily make computations between predicted and truth signals. Therefore, we require a set of "ground truth" datasets which are assumed to be the user's actual motion. Although these "ground truth" datasets may not be the exact motion of a given user, we can consider them to be representative of what the user might do and this is adequate for our purposes.

To develop the "ground truth" datasets, we collected a set of head and hand motion sequences with the user performing various tasks that are typical in a virtual environment. The data was collected using the InterSense IS900 tracking system in a Cave. Each motion sequence is about 20 seconds in length and was sampled between 210 and 217Hz. Each pose record consists of a timestamp, a position vector and a unit length quaternion. Each motion sequence was then resampled five times higher than its original sample rate using lowpass interpolation. This makes each motion sequence's sampling rate a little over 1kHz. The resampling makes testing over different sample rates much easier to perform.

The resampled motion sequences have noise associated with them from both the lowpass interpolation and the sensors themselves. To create clean signals[3], we examined the power spectrum of each to determine the lowpass and highpass parameters to a zero-phase shift filter used to remove high frequency noise. Note that once the signals are filtered, the quaternions are renormalized to ensure they are of unit length.

Many successful predictors utilize velocity and/or acceleration information from other sensors like gyroscopes and accelerometers[1]. Therefore, it is important to have the flexibility to examine these predictors. Since we do not have access to these types of sensors, we do the next best thing by simulating them. Using the clean motion sequences, first and second derivatives (i.e., velocity and acceleration) were calculated using 4th order finite central differencing schemes. Because the derivatives were calculated on clean signals with high sampling rates, the 4th order differencing schemes provide a nice approximation to what one would get with today's gyroscopes and accelerometers. With the derivative information for both position and orientation, each pose record contains a timestamp, a position, velocity, and acceleration vector and an orientation quaternion, an angular velocity pure vector quaternion, and an angular acceleration pure vector quaternion. These motion sequences are then used as input to the testing suite.

3.3 Software Architecture

The Predictive Tracking Algorithm Testing Suite's software architecture, shown in Figure 1, has four main components. The first component, the user interface component, was developed with the Qt interface toolkit and provides the user with control panels for loading datasets, setting system parameters, setting prediction algorithm parameters, and running different experiments. In addition, it allows the user to start a MATLAB engine used to produce graphical error plots from the various test types.

The UI component has a two way connection to the second component, the Dispenser. The Dispenser acts as a link between the UI component and the prediction algorithm library. It takes all the required system, algorithmic, and test parameter information plus the active dataset from the UI component and simulates a tracking system by providing pose records to the prediction algorithm library's active predictor. The Dispenser is also in charge of adding noise to the clean motion sequences according to the noise variance parameter (we assume zero mean Gaussian white noise).

The Dispenser's other job is to send the predicted as well as the clean and corrupted motion sequences to the Results and Error Analysis component. This component performs the error computations as well as prepares the results for both tabular and graphical display according to the particular test run. Finally, the fourth component is the Prediction Algorithm Library. The library contains all the developed predictors and allows for others

[3] We need a clean signal so we can add noise to them later.

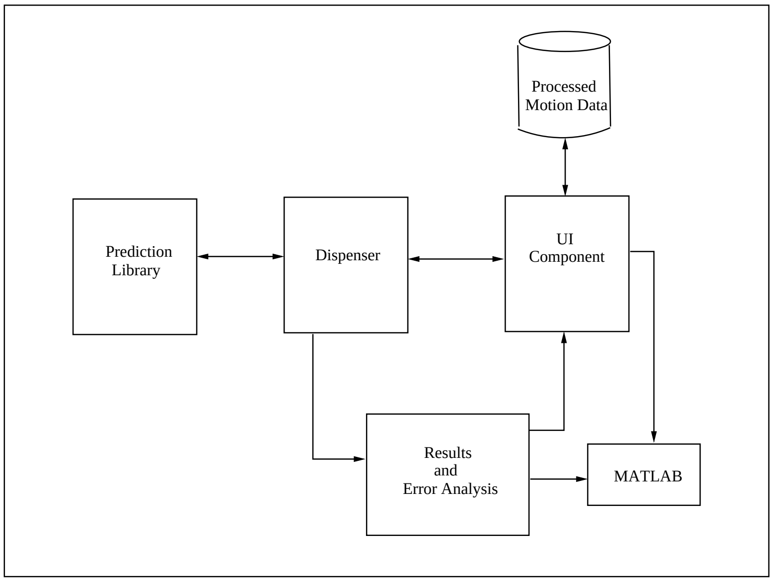

Figure 1: A software architecture diagram of The Predictive Tracking Algorithm Testing Suite.

to be easily added through a simple API and a small addition to the UI component. It has a two way connection to the Dispenser so it can receive pose information as well as algorithmic parameters and send predicted poses back to the Dispenser. The library's only connection to the rest of the testing suite is through the Dispenser since we want it to be as independent as possible. This independence will make porting the predictors to real-time VR and AR applications much easier.

4 Example Experiments using the Testing Suite

Utilizing The Predictive Tracking Algorithm Testing Suite, we present some simple experiments on the performance of 4 different Kalman filter-based predictors across different sampling rates, prediction times, and noise variance for both head and hand motion datasets. The two position predictors utilize a continuous-discrete Kalman filter while the two orientation predictors utilize the extended Kalman filter, an extension to standard Kalman filtering used with nonlinear process models. For more details on Kalman and extended Kalman filtering see [1, 8, 11, 14].

4.1 Datasets

Two head motion and two hand motion datasets are used to test the various predictors. The two head motion sequences, denoted by HEAD1 and HEAD2, are typical of those found when using a Cave. The HEAD1 dataset samples a user standing in the center of the Cave rotating the head and body as she virtually navigates through the VE. The HEAD2 dataset samples a user physically walking around in a Cave environment. This type of motion is common when a user interacts with the StepWIM[6], an interaction widget placed on the floor of the Cave used in navigation. The two hand motion datasets, HAND1 and HAND2, represent two different hand motion styles. The HAND1 dataset samples a user grabbing, manipulating, and placing virtual objects in the VE. The HAND2 dataset samples a user working in CavePainting[4], a tool for creating 3D artistic scenes. The HAND2 dataset is essentially free form hand motion and will be the most difficult to predict accurately.

4.2 Predictors

We examine four Kalman filter-based predictors using the testing suite, two KF predictors for position and two EKF predictors for orientation. The predictors are designated by their process model and the types of measurement sensors used. For position we have the following predictors:

- PVSM – position/velocity process model with single position measurement
- PVASM – position/velocity/acceleration process model with single position measurement.

For orientation we use the following predictors:

- OVSM – orientation/angular velocity process model with single position measurement
- OVASM – orientation/angular velocity/angular acceleration process model with single position measurement.

4.3 Experimental Parameters

Three separate tests are run on each predictor for every dataset giving a total of 48 different tests scenarios. For the test scenarios we utilize RMSE metric. The predictors are run across different sampling rates, prediction times, and error variances. For the sampling rate test, predictors are tested where the datasets are sampled using 40 different sample rates between 25 and approximately 1000Hz. For the prediction time test, the predictors

RMS Error Ranges from Lowest to Highest Sampling Rate (25 to 1000Hz) 100ms Prediction Time, 5e-5 Position Noise and 5e-6 Orientation Noise Variance				
	HEAD1	HEAD2	HAND1	HAND2
PVSM	0.46 - 0.34	0.58 - 0.47	0.67 - 0.51	2.23 - 2.05
PVASM	≈ 0.54	0.66 - 0.7	0.83 - 0.63	3.18 - 2.36
OVSM	1.9 - 1.4	2.15 - 1.71	2.7 - 2.3	12.6 - 11.4
OVASM	2.62 - 3.13	3.03 - 3.72	3.56 - 4.37	18.6 - 27.1

Table 1: A summary of the RMS error ranges from low to high sampling rates. The entries for the first two predictors are in inches, and the entries for the second two predictors are in degrees.

are tested 40 times between 15 and 240ms. Finally, for the error variance tests, predictors are tested 20 times where noise is added to the motion signals with variances between 0.000001 and 0.0001 for position and between 0.0000001 and 0.00001 for orientation. For the noise added to the position signals, this is equivalent to a Gaussian distributed range of ± 0.036 inches in the smallest case and ± 0.36 inches in the largest case, and for orientation, ± 0.28 degrees in the smallest case and ± 2.66 degrees in the largest. For each of the three tests, the system parameters that are not varied are constant with sampling rate of approximately 120Hz, prediction time of 100ms and error variance at 0.00005 for position signals and 0.000005 for orientation signals.

5 Results and Discussion

Tables 1-3 shows the basic results of the different predictor tests. For the sampling rate and noise variance tests, RMS error ranges are shown from the lowest to highest sampling rate and the lowest to highest noise variance. For the prediction time tests, the highest prediction times are shown for RMS errors under one inch and 2 degrees. Note that these results are rounded for clarity.

One of the important goals of the testing suite is to help VR and AR system designers in making decisions on how to use predictive tracking algorithms. With the results of these experiments, we can begin to examine some of the questions posed in Section 1 given the experimental parameters discussed in Section 4.3. For example, suppose a tracking system designer is interested in knowing if increasing sampling rate will vastly improve prediction performance using the PVSM predictor. Therefore, how is the PVSM predictor's performance affected by varying sampling rate with a prediction time of 100ms and a noise variance of 5e-5 across the HEAD1, HEAD2, and HAND1 datasets? Figure 2 provides insight into this question, and we can see that performance does improve although

RMS Error Ranges from Lowest to Highest Error Variance Position: 5e-6 to 1e-4, Orientation: 5e-7 to 1e-5 100ms Prediction Time, 120Hz Sampling Rate				
	HEAD1	HEAD2	HAND1	HAND2
PVSM	0.33 - 0.46	0.44 - 0.58	0.47 - 0.67	1.9 - 2.4
PVASM	0.52 - 0.56	0.67 - 0.69	0.59 - 0.82	2.2 - 3.2
OVSM	1.3 - 1.9	1.6 - 2.2	2.1 - 2.7	10.8 - 12.4
OVASM	2.93 - 3.07	3.46 - 3.56	3.99 - 4.15	24.43 - 23.98

Table 2: A summary of the RMS error ranges from low to high noise variance. The entries for the first two predictors are in inches, and the entries for the second two predictors are in degrees.

Prediction Times Up to 1 Inch and 2 in Degrees RMS Error Noise Variance: Position: 5e-5 , Orientation: 5e-6 Test Runs between 15 and 240ms, 120Hz Sampling Rate				
	HEAD1	HEAD2	HAND1	HAND2
PVSM	217	166	146	60
PVASM	166	144	123	49
OVSM	110	100	84	32
OVASM	76	71	63	16

Table 3: All entries in the table show the highest achieved prediction time for which position predictors attained an RMS error under 1 inch and orientation predictors attained an RMS error under 2 degrees. Times are in milliseconds.

not in a dramatic fashion. A possible explanation is that the noise variances are constant across different sampling rate tests. As the sampling rate increases, a constant noise variance will make the motion sequence harder to predict. To determine the true benefit or hindrance of different sampling rates requires testing against different noise variances simultaneously. Our hypothesis here is that higher sampling rates will make significant improvements in predictor performance as noise variances get smaller.

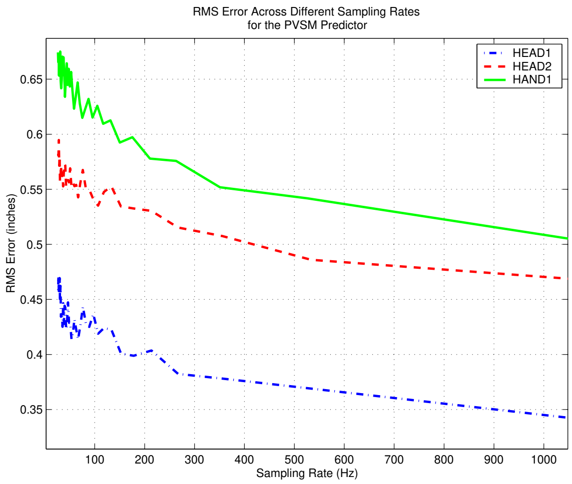

Figure 2: PVSM predictor performance over different sampling rates.

In another example, the same tracking system designer is interested in how increasing noise variance will affect the OVSM predictor's performance given a prediction time of 100ms and a sampling rate of 120Hz across the HEAD1, HEAD2, and HAND1 datasets? Figure 3 shows the results of this experiment which indicate that RMS errors do get worse as noise variance increases. However, at the given sampling rate and prediction time, the RMS errors stay within one degree from lowest to highest noise variance indicating that the OVSM predictor handles the noise fairly well.

Finally, in our last example, a VR systems programmer knows that predicting hand motion will be an important requirement in his clients VR applications. How do all four predictors compare across prediction times for the HAND2 dataset with a sampling rate of 120 Hz, a position noise variance of 5e-5, and an orientation noise variance of 5e-6? From Figure 4, we see that the PVSM and OVSM predictors perform better then their PVASM and OVASM counterparts. However, even the PVSM and OVSM predictors do a poor job for any reasonable prediction times. This result indicates that we need a higher sampling rate and lower noise variance or more sophisticated prediction algorithms to improve performance for the motion data exemplified by HAND2. The tracking system designer also examines predictor performance across prediction times for the HAND1

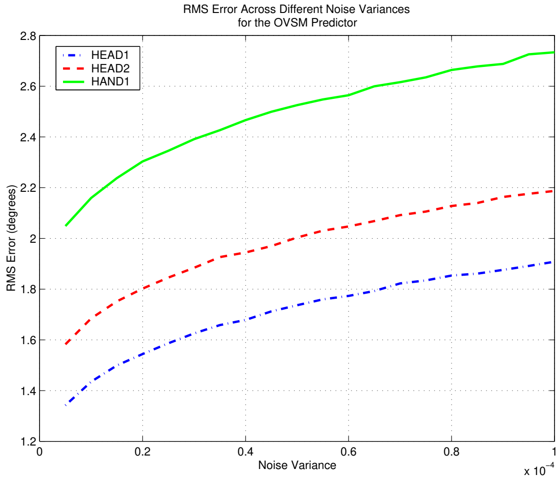

Figure 3: OVSM predictor performance over increasing noise variance.

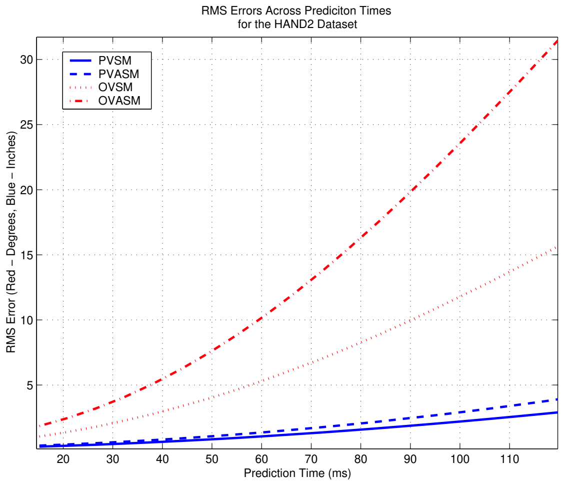

Figure 4: Predictor performance over prediction time for the HAND2 dataset.

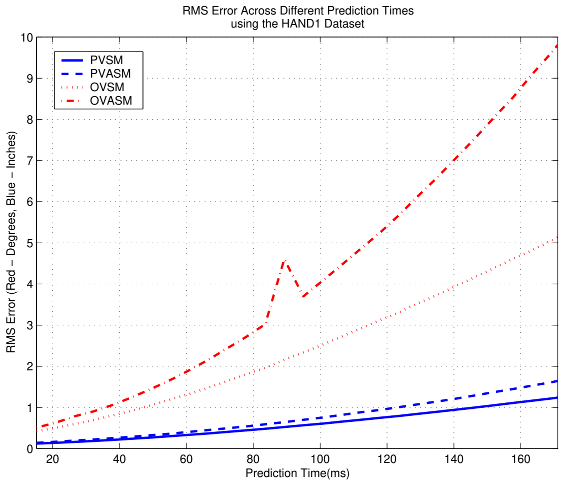

Figure 5: Predictor performance over prediction time for the HAND1 dataset.

dataset (shown in Figure 5) and sees, once again, that the PVSM and OVSM predictors perform better than their PVASM and OVASM counterparts. For the HAND1 dataset, these predictors show adequate performance under 100ms for position and under 80ms for orientation indicating the PVSM and OVSM predictors could be used if the HAND1 dataset is a typical example of the type of hand motion used in the client's VR applications.

Note that these examples are just a small set of the many possible questions that a VR system designer or programmer might ask to determine how predictors can be used in their systems. Providing a testing suite with enough flexibility to handle a variety of system and algorithmic parameters enables them to ask such questions and have the empirical evidence to make informed decisions to improve VR and AR applications.

6 Future Work

There are a number of significant areas for future work with The Prediction Algorithm Testing Suite. We wish to continue to extend the feature set of our testing suite by including more error metrics, more sophisticated testing tools, and additional prediction algorithms such as Multiple Model Adaptive Estimation and Covariance Intersection.

In addition, we wish to use the testing suite to develop general predictive tracking algorithm models that characterize prediction algorithm performance. The example experiments presented in this paper examined representative slices through the multidimensional parameter space. Although these slices provide a large amount of information about predictor performance, they are not sufficient for developing general prediction algorithm models. Therefore, we wish to perform more rigorous studies to get enough data points for model development.

This multidimensional parameter space not only contains sampling rate, prediction time and noise variance, but also the type of user motion and algorithmic parameters. Developing quantitative measures of user motion and models for determining optimal algorithmic parameters settings will be important for fully understanding predictive tracking algorithm performance. With this information, prediction algorithm models could be developed on a prediction by prediction basis leading to adaptive predictive tracking systems. Once we have developed these predictive tracking models we wish to incorporate them into real tracking systems to improve interaction in VR and AR applications.

7 Conclusion

We have presented The Predictive Tracking Algorithm Testing Suite, a software tool for developing and examining different predictive tracking algorithms. The suite is a general framework for examining prediction algorithms across different sampling rates, prediction times, noise variance, specific algorithmic parameters, and different types of user motion. The suite is general enough to allow VR system designers, prediction algorithm developers, and tracking system designers to ask a variety of questions about how predictors will perform under given conditions.

We have also presented some example experiments using the testing suite and some possible questions that might be asked to demonstrate the testing suites flexibility. Although the testing suite is still in its early stages of development, it has the potential to be a valuable tool to anyone interesting in understanding how to use predictive tracking algorithms in VR and AR systems.

Acknowledgments

Fill in later.

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] Azuma, Ronald and Gary Bishop. A Frequency Domain Analysis of Head Motion Prediction. In *Proceedings of SIGGRAPH'95*, 401-408, 1995.

[3] 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.

[4] Keefe, D., Acevedo, D., Moscovich, T., Laidlaw, D., and LaViola, J. CavePainting: A Fully Immersive 3D Artistic Medium and Interactive Experience. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, 85-93, 2001.

[5] 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.

[6] LaViola, Joseph, Daniel Acevedo, Daniel Keefe, and Robert Zeleznik. Hands-Free Multi-Scale Navigation in Virtual Environments. In the *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, ACM Press, 9-15, 2001.

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

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

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

[10] , Wan, E. A., and R. van der Merwe. The Unscented Kalman Filter, In *Kalman Filtering and Neural Networks*, S. Haykin (ed.), Wiley Publishing, 2001.

[11] 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.

[12] 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.

[13] Wu, Jiann-Rong and Ming Ouhyoung. A 3D Tracking Experiment on Latency and its Compensation Methods in Virtual Environments. In *Proceedings of UIST'95*, 41-49, 1995.

[14] 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.