An Adaptive Approach to Exponential Smoothing for CVE State Prediction

Fred Stakem
Georgia Institute of Technology
School of Electrical and Computer Engineering
fstakem@gatech.edu

Ghassan AlRegib
Georgia Institute of Technology
School of Electrical and Computer Engineering
gregib@ece.gatech.edu

ABSTRACT

Creating an immersive experience in a collaborative virtual environment, or CVE, involves more than just high definition graphics and expensive hardware. An immersive experience requires participant's input to be translated in a timely fashion to the local environment as well as to others connected across the internet. A well designed prediction algorithm will reduce the lag caused by virtual environment hardware and communications networks. The purpose of this experiment was to test the quality of an adaptive exponential smoothing algorithm at predicting human arm movement in a CVE. The results show that adaptive exponential smoothing performs as well as or better than dead reckoning at position estimation. When applied to a CVE adaptive Holt's exponential smoothing can help to reduce the overall lag of the system without being as computationally complex as many other techniques.

Categories and Subject Descriptors

H.5.1 [**Information Systems**]: Multimedia Information Systems—*artificial, augmented, and virtual realities*; C.2.2 [**Computer Systems Organization**]: Network Protocols—*applications*; I.5.4 [**Computing Methodologies**]: Applications—*signal processing, waveform analysis*

General Terms

Algorithms, Design, Human Factors

Keywords

Virtual reality, dead reckoning, exponential smoothing, distributed systems, state prediction, signal processing

1. INTRODUCTION

For a CVE to be truly immersive a user's avatar must respond to changes in state in a timely and accurate manner. Any delay or jitter in conveying a user's new state can cause a reduced sense of immersion and responsiveness [12, 13, 11, 4, 9]. In extreme cases this disconnected feeling can even lead to participants becoming disoriented or sick [14]. To reduce the local delay high quality hardware components can be used, but some degree of latency will always exist. In addition, the communication lag from interconnecting CVEs across the internet will cause a delay for remote users. No matter what hardware or technology is used some amount of delay is inevitable and must be dealt with in the design of the virtual environment. One key technique used to mitigate the effects of latency in a CVE is state prediction. Prediction algorithms are used to predict the current state of an avatar, either local or networked, given past states so the virtual environment can respond without waiting for the current state to be communicated to it. When combined with other visual techniques prediction algorithms can greatly increase the responsiveness of a CVE allowing for a more immersive and enjoyable experience.

The local delay found in a CVE is typically small and mainly caused by the hardware sensors and their communication back to the software. Additional delay is often added to the system when low pass filtering techniques are used to smooth out the noisy signals obtained from the sensors. Since system designers have no control over the internals of the hardware or their drivers it is up to prediction algorithms in the software to mask the system's delay. No technique has been adopted by industry as a standard so each hardware vendor tends to develop their own techniques. In the research literature particle filters, neural networks, and various forms of kalman filters have been studied and found to be adequate prediction techniques [16, 1, 7, 18, 19]. One of the drawbacks with virtual environment research vividly displayed on this topic is the inherent complexity of virtual environments has caused no standard research method. Researchers vary in the type of hardware utilized, the type of movement studied, and even the complexity of the movement studied. Such a diverse set of literature helps to give a broad understanding but the results are not conclusive for every virtual environment developed.

In contrast, dead reckoning, the prediction technique used in internetworking virtual environments across the internet, is well studied and fairly uniformly adopted in industry and academia [15, 2, 3, 5]. Dead reckoning has its roots in large scale military simulations, but was later adapted for use in the online game industry where it has since become popular. Dead reckoning is a prediction technique that uses an entities position, velocity, and potentially its acceleration and jerk to calculate the entities future position. In essence

IMMERSCOM 2009, May 27-29, Berkeley, USA
Copyright © 2009 978-963-9799-39-4
DOI 10.4108/immerscom.2009.17

dead reckoning is an extrapolation technique used to reduce network delay and bandwidth. Precious bandwidth is saved because an entity only needs to transmit new information when its current position strays further than an error bound away from its predicted location. Although dead reckoning is well understood in online gaming its application to human movement signals has not been studied extensively.

In short, many complicated prediction techniques have been shown to work for local signal prediction, but dead reckoning is preferred in networked environments where the delay can be much larger. Dead reckoning is a relatively simple technique that requires no special mathematical libraries or large SDKs. Another prediction technique that is similar to dead reckoning in its simplicity and ease of implementation is exponential smoothing [6, 10]. Exponential smoothing is a technique that has many different variations but in general it uses a sum of weighted past samples to predict future samples. The term exponential comes from the fact that the weight, or value applied to the past sample, exponentially decreases the further in the past the sample is. This gives recent samples, which are more likely to be close in value to the predicted sample, a much higher weight than older samples. Dead reckoning can be thought of as a case of exponential smoothing that only uses a single previous sample instead of multiple samples. The use of multiple samples makes the algorithm behave as a low pass filter smoothing out high frequency components along with predicting the future value.

The degree of filtering, or smoothing, is controlled by the value of the smoothing parameter which also controls the prediction error. When picking the smoothing parameter a difficult tradeoff must be made between the amount of filtering and the precision of the predicted value. An example of a signal and its prediction by an exponential smoothing predictor can be seen in Figure 1. In this case the predictor has a low smoothing parameter, $\alpha = 0.4$, causing the filter to act more as a low pass filter than an accurate predictor. The high frequency component of the original signal is noticeably diminished in the predicted signal, but the accuracy of the prediction is also decreased.

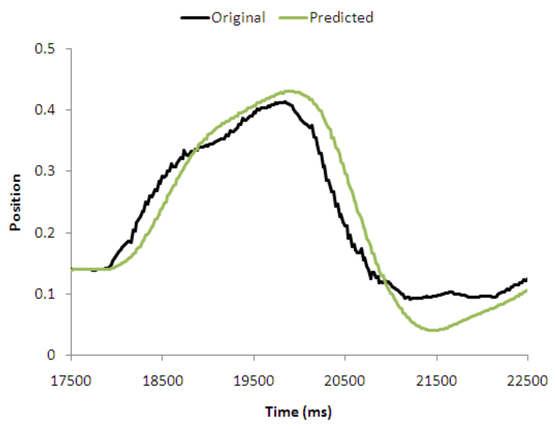

Figure 1: Exponential smoothing predictor with excessive filtering.

The application of exponential smoothing for state prediction in virtual environments has been rather sparse. Although exponential smoothing is a well known and highly regarded technique it is mainly used in the management research community. In the research paper by LaViola double exponential smoothing, a form of exponential smoothing, was compared to kalman filtering for state prediction in a virtual environment [8]. The research concluded that double exponential smoothing was an acceptable means of position estimation in virtual environments, and double exponential smoothing performed comparably to kalman filtering with a much faster execution time. The main drawback stated in the paper was the optimal estimation of the smoothing parameter. The optimal smoothing parameter was found in the experiment using a searching algorithm which can potentially be computationally expensive and is dependent on the searched data set. In addition once the optimal smoothing parameter was found it was constant and did not adapt to the signal. In Section 2 of this paper a form of exponential smoothing called Holt's exponential smoothing is discussed along with an adaptive approach for calculating the smoothing parameters. Section 3 discusses the virtual environment used along with the experiments performed to collect human movement data. Finally Section 4 analyzes exponential smoothing versus dead reckoning at predicting the human movements captured.

2. PREDICTION ALGORITHMS

One of the key weaknesses in using double exponential smoothing is that a fixed value must be chosen for the smoothing parameter. Using a search algorithm to find the optimal smoothing parameter can help to narrow down the parameters to choose from, but an expert is still required for analysis of the data. Often times small changes in the smoothing parameter can have a large effect on the error of the predicted signal. In addition to the computational time spent finding the optimal parameter, researchers will often need to spend time manually analyzing the predicted signal to find the truly optimal parameter. An alternative form of exponential smoothing similar to double exponential smoothing but requiring multiple smoothing parameters, that can potentially be made adaptive, is Holt's exponential smoothing. Holt's exponential smoothing uses a different smoothing parameter for each degree of differentiation used for prediction. For example if only the previous position is used for prediction, a variation commonly called simple exponential smoothing and shown below in Equation 1, then only one smoothing parameter needs to be defined (α). If both the previous position and velocity are used for prediction then two smoothing parameters need to be defined; one parameter for the position prediction term and one for the velocity prediction term. Tying each smoothing parameter to a different degree of differentiation allows fine tuning of the algorithm and for some individuals this makes the algorithm more intuitive to tweak.

$$\hat{s}_{t+k} = \alpha s_t + (1 - \alpha)\hat{s}_{t-k} \tag{1}$$

Holt's exponential smoothing is a simple technique that can be described by in Equations 2-4. The equations describe the estimated position, $\hat{p}_{t+k}$, and the estimated velocity, $\hat{v}_{t+k}$, for one axis of an entities state given the known position, p_t. For a virtual environment in 3D space each of the three different axes would have its own separate set of equations. The k step ahead predicted position, $\hat{s}_{t+k}$, is simply the addition of the estimated position and velocity terms. Since this form of exponential smoothing only uses velocity in addition to position there is only one additional smoothing parameter compared to simple exponential smoothing (α, γ). For each additional derivative term added a new equation and smoothing parameter need to be added to the set of equations. For example if the acceleration, $\hat{a}_t$, was used in addition to the position and velocity the equations would change and one new equation and smoothing parameter would need to be added (α, γ, ϕ). This can be seen in Equations 5-8.

$$\hat{p}_{t+k} = \alpha p_t + (1-\alpha)(\hat{p}_{t-k} + \hat{v}_{t-k}) \tag{2}$$

$$\hat{v}_{t+k} = \gamma(\hat{p}_t - \hat{p}_{t-k}) + (1-\gamma)\hat{v}_{t-k} \tag{3}$$

$$\hat{s}_{t+k} = \hat{p}_{t+k} + \hat{v}_{t+k} \tag{4}$$

Even if Holt's exponential smoothing is used instead of double exponential smoothing the optimal smoothing parameters must be found. Since movement patterns will differ according to how the virtual environment is used, finding parameters that will be optimal for all movements is unlikely. For example, the optimal parameters found for a precise hand movement will unlikely be the same as those found for a fast twitch hand movement. To remedy these problems the best solution is to make the smoothing parameters adaptive. By making the smoothing parameters adaptive the data will dictate changes to the smoothing parameter so that they change with every new sample. This was accomplished using the same Holt's exponential smoothing equations defined in equations 2-4 but by having the two smoothing parameters change with time.

$$\hat{p}_{t+k} = \alpha p_t + (1-\alpha)(\hat{p}_{t-k} + \hat{v}_{t-k} + \hat{a}_{t-k}) \tag{5}$$

$$\hat{v}_{t+k} = \gamma(\hat{p}_t - \hat{p}_{t-k}) + (1-\gamma)(\hat{v}_{t-k} + \hat{a}_{t-k}) \tag{6}$$

$$\hat{a}_{t+k} = \phi(\hat{v}_t - \hat{v}_{t-k}) + (1-\phi)\hat{a}_{t-k} \tag{7}$$

$$\hat{s}_{t+k} = \hat{p}_{t+k} + \hat{v}_{t+k} + \hat{a}_{t+k} \tag{8}$$

The key difference in adaptive Holt's exponential smoothing is how the smoothing parameters are recalculated at every new sample. The equations for the recalculation of the position smoothing parameter can be seen in Equations 9-12. First the prediction error, e_t, must be calculated as the difference between the real position, p_t, and the estimated position, $\hat{p}_t$. From the error term two new terms are calculated that are used to calculate the new smoothing parameter. The numerator for the new smoothing parameter, E_t, is calculated from current and past error terms. Like the numerator the denominator, D_t, is calculated from the current and past errors terms except the absolute value of the error terms is taken. The new smoothing parameter is simply the absolute value of the ratio of the two smoothed error terms. The only value that must be specified is β, the rate of change of the smoothing parameter. Although this seems to substitute one parameter for another, the effect of the rate of change parameter is much less volatile than the smoothing parameter. For each level of differentiation the smoothing parameter is calculated in this fashion with the only difference being the initial error term. If velocity, acceleration, or jerk were used then a new set of calculations would be needed for each smoothing parameter. In addition to being individually calculated for each differentiation term, the new smoothing parameters are also calculated individually for each axis making the algorithm very adaptable.

$$e_t = p_t - \hat{p}_t \tag{9}$$

$$E_t = \beta e_t + (1-\beta)E_{t-k} \tag{10}$$

$$D_t = \beta\,|e_t| + (1-\beta)D_{t-k} \tag{11}$$

$$\alpha_{t+k} = \left|\frac{E_t}{D_t}\right| \tag{12}$$

3. TESTING THE ALGORITHMS

In order to compare the different prediction algorithms data needed to be collected from a CVE. An existing CVE designed for close multiuser interaction was chosen as the test bed. The CVE was designed for individuals to collaborate around a central virtual table top mainly using their hands and arms for interaction. For dexterous hand manipulations 5DT data gloves were used to sense a user's fingers extension and flexion and relay this information back the CVE. For larger arm and body movements the CVE used Nest of Birds magnetic trackers. Two magnetic tracking sensors were attached to the back of the 5DT data gloves and another one was placed on a pouch the user carried around their midsection. This allowed the sensors to record the user's hand and central body location relatively accurately. Two large displays were used to project the 3D virtual environment. Although HMDs would seem to produce a more immersive experience, they were not used in order to reduce the overall all delay the user experienced.

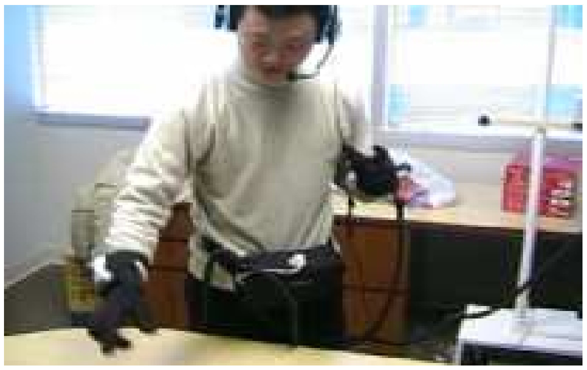
Figure 2: User donning CVE hardware sensors.

The software component of the virtual environment consisted of a typical 3D virtual space including virtual objects that users could interact with. Although the virtual environment was designed to allow all different types of objects to be used, the research focused on the use of small interconnecting blocks resembling legos. The choice of blocks allowed the users to build many different types of structures for interaction, and the blocks were general enough to be used for many different types of tests. In addition, the use of blocks allowed the users to perform different types of movements from large reaching movements, to fast twitch movements, to very small manipulations. The only drawback to using a multipurpose object such as a block was that object specific movements were excluded from the tests. For example, movement patterns exhibited while using a virtual hammer to strike a nail are hard to replicate with virtual blocks. Overall it was determined that the general nature of the virtual blocks was a better choice for studying algorithms that can potentially be applied to many different types of virtual environments.

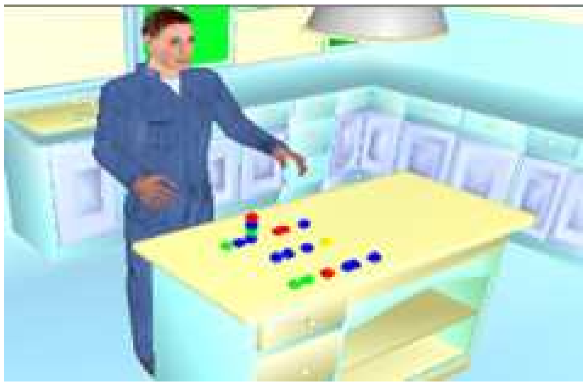

Figure 3: Virtual environment software including central table top and virtual blocks.

While collecting the data it was important that the subjects of the experiment performed a wide variety of different arm movements typically used in virtual environments so that the algorithms could be thoroughly tested. Since the vast majority of tasks done in a CVE involve reaching for objects and manipulating them, the focus of the experiment was around a simple task that involved both reaching and manipulative movements for its completion. The experiment started with the subject placing their right hand on a target in the center of the virtual table top. Once the subject's hand was placed on the target a random number of blocks were set at random locations on the table top. The subject was instructed to pick three blocks and to stack them on top of each other on the target where their hand started the experiment. Once the last block was placed on the stack the experiment log was saved and the subject was instructed to repeat the experiment. The experiment was conducted with three test subjects who repeated the task more than 100 times over the course of three days of testing.

In addition to the main stacking task previously described the subjects were also instructed to perform a variety of other tasks during the testing period. The alternative tasks were not meant to replicate the movements found in the stacking task, but were instead used to test more extreme movements than those typically found in a virtual environment. By testing movements rarely seen in a CVE and comparing the results to those of the stacking task a better indication of the generality of the predictive algorithms could be found. Various different movements were recorded by the three test subjects, but each of the different movements was only recorded from two to six times.

Two types of movements that illustrate the more extreme well are very quick movements and very intricate complex movements. To collect a twitch type of movement, more typical of movement found in a gaming system such as the wii, a task was designed where the subject was asked to start with their hand on a target in the center of the table as with the stacking task. This time the subject was instructed to attempt to catch or hit a falling virtual block, that fell from a random location, before it hit the table. Once the subject placed their hand on the target the experiment started and the block was dropped. Since the block fell quickly, the subject had to perform a quick twitch movement to catch the block. In addition, the random placement of the block's initial position made the task more difficult so that the subject could not anticipate the blocks location ahead of time.

The second type of movement captured focused on complicated small manipulations. The subject was instructed to tie their own shoes while donning the virtual reality equipment. Instead of performing this movement in the virtual world, which would currently be impossible with most virtual environments, the subject performed the task in the real world with the virtual environment equipment only used for logging purposes. Even though tying ones shoes seems like a simple task an actual analysis of all of the different movements performed during the task reveals its complexity. Once the subject tied their shoes the information was logged and the task was repeated. Although the movements necessary to tie one's shoes are not currently possible in a virtual environment the intricate movements will someday be possible and represent movement just beyond the bounds of current technology.

4. RESULTS

Once all of the data was collected it was tested against Holt's exponential smoothing, adaptive Holt's exponential smoothing, and dead reckoning to judge the effectiveness of the different prediction algorithms. Before Holt's exponential smoothing could be tested its optimal smoothing parameters needed to be found. First Holt's exponential smoothing was tested against all of the data with different prediction intervals and different smoothing parameters to obtain an estimate of the best smoothing parameters for each prediction interval. Each of the original signals was judged against its predicted signal using Euclidean distance as the error metric. After a rough estimate was found several predicted signals were plotted for various prediction intervals and smoothing parameters in order to better visualize the changes different smoothing parameters made. Both the rough estimate and an analysis of individual predicted signals were used to find the optimal parameters. Finally the mean distance error of the stacking data was found and plotted for all three prediction algorithms (Figure 4).

The algorithms were tested for their prediction capabilities under 100 ms which is typically where they would be utilized for both local and network prediction. Figure 4 shows that both Holt's exponential smoothing, HES, and adaptive Holt's exponential smoothing, AHES, can predict as well as dead reckoning, DR, or better. Although Holt's exponential smoothing outperforms its adaptive version in prediction under 40 ms it must be remembered that this is only the case when optimal parameters are used for Holt's exponential smoothing. When the prediction interval is large, 200 ms or greater, dead reckoning outperforms the other two algorithms. The likely cause of the reduction in accuracy by exponential smoothing is the fact that large prediction intervals use samples that are further in the past for prediction. The further in the past the samples used are the greater the likelihood of the samples being uncorrelated with the predicted sample and therefore unreliable for prediction.

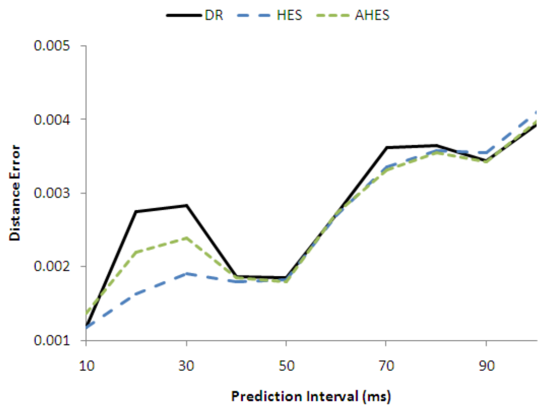

Figure 4: Prediction error for the three state prediction algorithms

Even though all of the algorithms can predict further ahead than 100 ms, that level of prediction is seldom needed for most situations. In case the algorithms need to predict even further ahead the data was tested with prediction intervals up to 500 ms and showed a steady increase in the error. How this effects the original signal can be seen in Figure 5. The figure shows a portion of the signal from the original stacking task and the adaptive Holt's exponential smoothing algorithms prediction of the signal 200 ms in the future. Clearly shown in the predicted signal is a small amount of high frequency noise and some distortion of the signals envelope, but even at 200 ms the prediction algorithms works fairly well. The most noticeable errors occur when the original signal abruptly changes its direction such as at 17,700 and 19,800 ms. When the original signal is constant or has a roughly linear slope the prediction algorithm works well except for the addition of the high frequency noise. The high quality of the algorithm at predicting a linear slope can be attributed to the fact that velocity, or slope, is an integral part of the algorithm. Given this it can be theorized that also including acceleration into the algorithm would increase the prediction accuracy even more. Unfortunately this was not found to be the case as the added acceleration terms tended to add more high frequency noise to the predicted signal.

Comparing the quality of prediction for the different movements using adaptive Holt's exponential smoothing did illustrate some interesting results. The stacking tasks had many natural long reaching movements followed by short manipulative movements. On the other hand the catching task had one quick twitch movement where the subject's

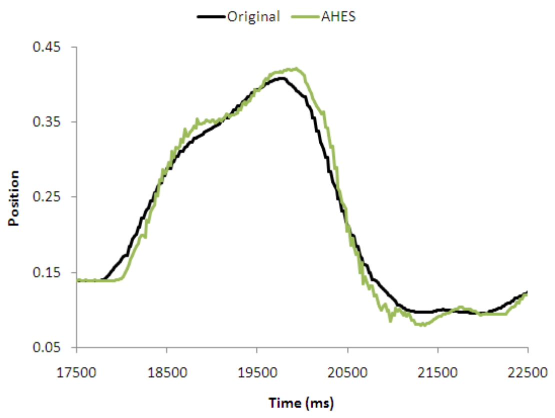

Figure 5: Adaptive Holt's exponential smoothing predicting 200 ms ahead.

hand shot out trying to catch a block. Although these tasks seem quite different the average distance error was similar for both. This result tends to reinforce the concept that reaching movements tend to have similar Gaussian speed envelopes with the peak being the defining difference [17]. Although the peak speed of the movements were different it is likely the low frequency nature of the two signals makes them easy to predict with exponential smoothing.

The same result was not true for the shoe tying task. In this task the average error was much higher than the stacking task. As expected the average error increased with the further in the future the algorithm predicted, but Figure 6 clearly shows a much steeper curve for the shoe tying movement. Similar results occurred for dead reckoning illustrating both algorithms difficulty at predicting sharp intricate movements. The likely cause for the greatly increased error is the higher frequency of the signals that make prediction further ahead more difficult. Only very short term prediction can be done because of the lack of correlation between samples in longer intervals. The threshold for quality prediction is therefore highly dependent on the type of movement exhibited as much as the algorithm used for prediction. The results suggest that adaptive Holt's exponential smoothing and dead reckoning are good algorithms for the majority of movements, but very intricate movements may require a different prediction algorithm.

5. CONCLUSIONS

A CVE like any virtual environment requires state prediction algorithms to combat the delays inherent in the system. Dead reckoning and other complex techniques have been proposed and implemented to accomplish this with various degrees of success. One simple technique that has been experimented with and shown to work well is exponential smoothing. The only problem with exponential smoothing is that the smoothing parameters which dictate the precision of the prediction are static. By using an adaptive exponential smoothing algorithm state prediction was accomplished without the need for a static smoothing parameter.

Adaptive exponential smoothing was shown to be able to predict as well as dead reckoning and even somewhat better

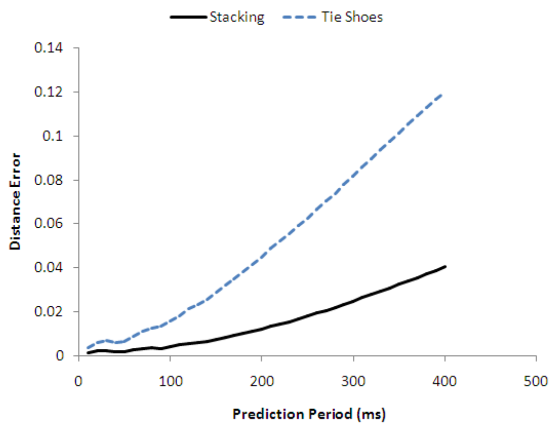

Figure 6: Adaptive Holt's exponential smoothing quality of prediction for different movements.

for some prediction intervals. In addition adaptive exponential smoothing was shown to be able to predict typical CVE signals up to 200 ms in the future relatively accurately. Although adaptive exponential smoothing was able to predict more extreme twitch movements both dead reckoning and adaptive exponential smoothing failed to predict intricate movements very far into the future. For short prediction intervals adaptive exponential smoothing can predict intricate movements reasonably well, but longer prediction intervals have too little correlation between samples. Further research needs to be done to find better prediction algorithms that will work on intricate movements as was as normal movements in a CVE.

6. REFERENCES

[1] F. Ababsa, M. Mallem, and D. Roussel. Comparison between particle filter approach and kalman filter-based technique for head tracking in augmented reality systems. *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, Vol.1:1021 – 6, 2004.

[2] S. Aggarwal, H. Banavar, A. Khandelwal, S. Mukherjee, and S. Rangarajan. Accuracy in dead-reckoning based distributed multi-player games. *Proceedings of the ACM SIGCOMM 2004 Workshops*, pages 161 – 165, 2004.

[3] W. Cai, F. Lee, and L. Chen. An auto-adaptive dead reckoning algorithm for distributed interactive simulation. *Proceedings Thirteenth Workshop on Parallel and Distributed Simulation. PADS 99. (Cat. No.PR00155)*, pages 82 – 9, 1999.

[4] L. Chen, G.-C. Chen, H. Chen, X.-L. Xu, and C. Chen. Study on effects of network characteristics on cooperation performance in a desktop cve system. *CSCWD 2004 - 8th International Conference on Computer Supported Cooperative Work in Design - Proceedings*, 1:121 – 126, 2004.

[5] T. Duncan and D. Gracanin. Pre-reckoning algorithm for distributed virtual environments. *Proceedings of the 2003 Winter Simulation Conference (IEEE Cat. No.03CH7499)*, vol.2:1086 – 93, 2003.

[6] J. Gardner, E.S. Exponential smoothing: the state of the art - part ii. *International Journal of Forecasting*, 22(4):637 – 66, 2006.

[7] A. Garrett, M. Aguilar, and Y. Barniv. A recurrent neural network approach to virtual environment latency reduction. volume 3, pages 2288–92, Honolulu, HI, USA, 2002. Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02.

[8] J. LaViola, J.J. Double exponential smoothing: an alternative to kalman filter-based predictive tracking. *IPT/EGVE 2003. Seventh Immersive Projection Technology Workshop. Ninth Eurographics Workshop on Virtual Environments*, pages 199 – 206, 2003.

[9] C. Ling, C. Gen-Cai, C. Hong, X. Xiao-Lei, and C. Chun. Study on effects of network characteristics on cooperation performance in a desktop cve system. *8th International Conference on Computer Supported Cooperative Work in Design (IEEE Cat. No.04EX709)*, Vol.1:121 – 6, 2004.

[10] A. Maia and F. de A.T. de Carvalho. Neural networks and exponential smoothing models for symbolic interval time series processing - applications in stock market. *2008 8th International Conference on Hybrid Intelligent Systems (HIS)*, pages 326 – 31, 2008.

[11] S. B. M. C. Nathan Sheldon, Eric Girard and E. Agu. The effect of latency on user performance in warcraft iii. *NetGames*, May 2003.

[12] K. S. Park and R. Kenyon. Effects of network characteristics on human performance in a collaborative virtual environment. pages 104–11, Houston, TX, USA, 1999. Proceedings IEEE Virtual Reality.

[13] P. Quax, P. Monsieurs, W. Lamotte, D. De Vleeschauwer, and N. Degrande. Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game. *Proceedings of the ACM SIGCOMM 2004 Workshops*, pages 152 – 156, 2004.

[14] R. Ruddle. The effect of environment characteristics and user interaction on levels of virtual environment sickness. *IEEE Virtual Reality 2004 (IEEE Cat. No.04CH37557)*, pages 141 – 285, 2004.

[15] S. Singhal and M. Zyda. *Networked Virtual Environments: Design and Implementation*. ACM Press, York, New York, 1999.

[16] F. Stakem and G. AlRegib. Neural networks for human arm movement prediction in cves. Atlanta, GA USA, June 2008. Proceedings of 3DPVT'08 - the Fourth International Symposium on 3D Data Processing, Visualization and Transmission.

[17] F. Stakem, G. Alregib, and B. Juang. Towards modeling human arm movement in a cve. Verona, Italy, 2007. IMMERSCOM, 2007.

[18] A. van Rhijn, R. van Liere, and J. Mulder. An analysis of of orientation prediction and filtering methods for vr/ar. pages 67–74, Bonn, Germany, 2005. IEEE Virtual Reality 2005.

[19] J. Wu and M. Ouhyoung. On latency compensation and its effects on head-motion trajectories in virtual environments. *Visual Computer*, vol. 16(2):79–90, 2000.