Motion Prediction for Caching and Prefetching in Mouse-Driven DVE Navigation

ADDISON CHAN, RYNSON W. H. LAU, and BEATRICE NG
City University of Hong Kong, Hong Kong

A distributed virtual environment (DVE) allows geographically separated users to participate in a shared virtual environment via connected networks. However, when the users are connected by the Internet, bandwidth limitation and network latency may seriously affect the performance and the interactivity of the system. This explains why there are very few DVE applications for the Internet. To address these shortcomings, caching and prefetching techniques are usually employed. Unfortunately, the effectiveness of these techniques depends largely on the accuracy of the prediction method used. Although there are a few methods proposed for predicting 3D motion, most of them are primarily designed for predicting the motion of specific objects by assuming certain object motion behaviors. We notice that in desktop DVE applications, such as virtual walkthrough and network gaming, the 2D mouse is still the most popular device used for navigation input. Through studying the motion behavior of a mouse during 3D navigation, we have developed a hybrid motion model for predicting the mouse motion during such navigation—a linear model for prediction at low-velocity motion and an elliptic model for prediction at high-velocity motion. The predicted mouse motion velocity is then mapped to the 3D environment for predicting the user's 3D motion. We describe how this prediction method can be integrated into the caching and prefetching mechanisms of our DVE prototype. We also demonstrate the effectiveness of the method and the resulting caching and prefetching mechanisms through extensive experiments.

Categories and Subject Descriptors: G.3 [**Probability and Statistics**]: *Probabilistic algorithms*; I.3.2 [**Computer Graphics**]: Graphics Systems—*Distributed / network graphics*; I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction techniques*

General Terms: Algorithms, Design, Human Factors

Additional Key Words and Phrases: Mouse motion prediction, caching and prefetching, distributed virtual environments, virtual navigation

1. INTRODUCTION

Distributed virtual environments (DVEs) allow remote users to participate in the same environment through networks. There are many applications

This research was partially supported by three CERG grants from the Research Grants Council of Hong Kong (RGC Reference: CityU 1080/00E, CityU 1308/03E, and CityU 1133/04E).
Authors' addresses: A. Chan, Department of CEIT, City University of Hong Kong, Hong Kong; email: addi@cs.cityu.edu.hk; R. W. H. Lau and B. Ng, Department of Computer Science, City University of Hong Kong, Hong Kong; email: {rynson,beatrice}@cs.cityu.edu.hk.

of DVE including remote training, virtual touring, collaborative gaming and collaborative design [Normand et al. 1999; Li et al. 2003, 2004]. However, most of these applications, in particular those requiring speedy response from the participants such as 3D multi-user online games, are mainly developed for use in a LAN environment. The Internet generally suffers from low-network bandwidth and high-network latency, which will certainly affect the interactivity of those applications. This is why we do not see many interactive DVE applications for the Internet. To address this, we need to optimize the usage of network bandwidth and to minimize the effect of network latency.

In DVE, there are two approaches to distribute the geometry data from the server to the clients: full replication and on-demand transmission. Most systems, such as DIVE [Carlsson and Hagsand 1993], SIMNET [Calvin et al. 1993], and VLNET [Pandzic et al. 1997], employ the full replication approach to distribute all geometry data to the clients before the start of the application. Since the geometry database is usually large in size, this approach assumes offline downloading or the use of a high-speed network in order to reduce the preloading time. The on-demand transmission approach is to distribute the geometry data to the clients on demand at run-time [Capps 2000; Falby et al. 1993; Singh et al. 1994]. The idea of it is that when the virtual environment (VE) is large, a viewer will likely visit only a small part of it. Hence, it is only necessary to transmit the visible region of the VE to the client to reduce startup time and optimize the usage of network bandwidth. In addition, a client may also prefetch objects from the server in advance so that the objects will be available at the client when they are needed. This may help minimize the effect of network latency on system interactivity. However, the effectiveness of prefetching depends very much on the accuracy of the prediction method used to predict the future movement of the viewer.

On the other hand, a client machine is likely to have a limited local cache. Hence, a caching mechanism is often used to identify objects in the cache which are unlikely needed in the near future and replace them with the incoming objects. This can be regarded as the classical caching issue. It is in contrast with the native "VRML" approach [The VRML97 Specification] that retains all the data downloaded in the client machine. Here, if the future usage of the data can be predicted more accurately, a better caching performance in terms of cache hit rate can be achieved. Hence, the prediction method used in prefetching can also be applied to caching. Ellsworth [2001] and Funkhouser et al. [1992] propose implicit predictors by cluttering interrelated or neighboring objects contiguously. Whenever they load an object from the database or secondary storage, objects with high affinity will be loaded as well. However, no precise prediction method is proposed, which can reduce unnecessary prefetching.

We note that in desktop DVE applications, the 2D mouse is still the primary device used for navigation input. In order to have an efficient caching and prefetching scheme in such a setting, we have developed a hybrid motion model for predicting the mouse motion during 3D navigation by studying the motion behavior of a mouse during those applications [Chan et al. 2001a].

At low-velocity mouse motion, we use a linear model for prediction. At high velocity motion, we use an elliptic model for prediction. To improve the accuracy of the prediction, we also consider the user's motion characteristics [Chan et al. 2001b]. Having predicted the mouse motion, we can estimate the viewer's future position and viewing angle.

We show in this article how this prediction method can be integrated into the DVE system that we have developed [Chim et al. 1998, 2003] and how it can improve the performance of the caching and prefetching mechanisms, which in turn reduces the effect of network latency and minimizes network bandwidth consumption. We demonstrate the effectiveness of the prediction method and the resulting caching and prefetching mechanisms through extensive experiments on the system.

The rest of the article is organized as follows: Section 2 summarizes existing motion prediction methods. Section 3 presents our hybrid motion prediction method. Section 4 briefly describes the caching and prefetching mechanisms used in our DVE system and how the prediction method can be integrated into these mechanisms. Section 5 demonstrates the effectiveness of our prediction method and the resulting caching and prefetching mechanisms through various experiments. Finally, Section 6 briefly concludes the article and discusses possible future work.

2. RELATED WORK

Although there are many prediction methods proposed, most of them are developed for specific purposes. Methods that are developed for navigation prediction are very limited. In this section, we survey different prediction methods that either are developed primarily for navigation prediction or can be adopted for navigation prediction in a straightforward manner.

In mobile computing, Liu and Maguire [1996] noted that most people tend to take the same path when travelling to attend a regular activity, such as going to work or to school. However, a person may occasionally choose a different path from the regular one. To model these two cases, two separate models are proposed. One is responsible for recognizing the regular path. If a user does not follow the regular path, the predictor will estimate the next movement using the Markov model, which is based on Markov process analysis. Unfortunately, the analysis requires collecting a large amount of data and may involve expensive computations. In Liu and Bahl [1998], user movement is again predicted using two models, *Global Mobility Model* (GMM) and *Local Mobility Model* (LMM). GMM exploits the fact that most of the mobiles exhibit some regularity everyday, which is similar to Liu and Maguire [1996]. LMM is used to predict the user's short-term trajectory with dynamic system modeling by examining the current position and velocity of the mobile.

In DVEs, dead reckoning [DIS Steering Committee 1998] is often used to extrapolate or predict the movement of an entity by assuming that the motion of the entity is deterministic, in order to reduce network traffics. The most popular predictor that the DIS standard supports is the polynomial predictor,

where polynomials of any orders may be used. Some examples are:

$$\begin{aligned} \textit{zero order} : p_{new} &= p \\ \textit{first order} : p_{new} &= p + t * v \\ \textit{second order} : p_{new} &= p + t * v + 0.5 * t^2 * a, \end{aligned}$$

where p is the initial position, p_{new} is the predicted position, v is the velocity, a is the acceleration, and t is the time at which p_{new} is to be predicted. In Singhal and Cheriton [1995], a hybrid approach is suggested. The first-order polynomial is used if the acceleration is either minimal or substantial; otherwise, the second-order polynomial is chosen. Cai et al. [1999] suggest the adaptive dead reckoning by adjusting the threshold according to the distance between the viewer and the entity.

Another prediction model is the EWMA scheme as suggested in our previous work [Chim et al. 1998, 2003]. The model assigns different weights to past movement vectors, with more recent vectors given higher weights. Although the EWMA scheme is simple to implement, it is not so effective for predicting rapid movements because it takes time for the predicted movement vectors to adjust to a new direction.

There has been some work on predicting human body motion. In Azuma and Bishop [1995], the Kalman-based predictor is proposed, which was originally used to filter measurement noise from linear systems by recursively minimizing the mean square estimation error. It may work with the polynomial predictor or other prediction models to further reduce the prediction error. Similar to the polynomial predictor, this method assumes the predicted motion to follow a fixed motion model. Experimental results from Azuma and Bishop [1995] show that during rapid movement, the predictor becomes less effective and its performance is similar to one without using the Kalman filter. Another method is to use the Gauss–Markov process modeling to predict head motion [Liang et al. 1991] and human animation [Capin et al. 1997]. Wu and Ouhyoung [2000] also attempt to predict head motion using a grey system theory-based predictor, which accuracy is similar to the polynomial-based predictor with Kalman filtering. In general, this kind of specialized prediction method usually produces more accurate results because it takes into account the motion behavior of the target objects. The tradeoff is the loss of generality.

3. A HYBRID MOTION PREDICTION METHOD

In our earlier work, we use the EWMA method to directly predict the viewer's 3D movement inside the VE [Chim et al. 1998, 2003]. Although this method performs very well in our simulation experiments, it performs less well in the prototype experiments. This is because in our simulation experiments, the viewer's movement is directly specified as 3D coordinates and vectors. Hence, our implementation simply uses the EWMA method to predict the viewer's 3D motion vectors based on his/her past 3D motion vectors. However, in the prototype experiments, the viewer's movement is specified by a 2D mouse through a mapping process. The performance of the EWMA method is affected by both the noise from the 2D mouse and the mapping process. In order to improve the

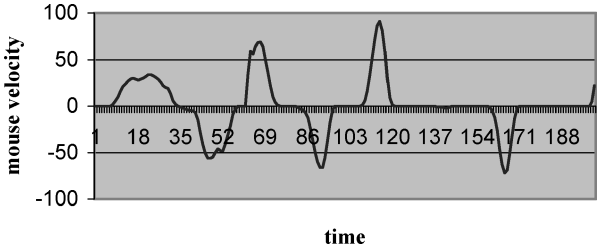


Fig. 1. Mouse motion velocity obtained from a sample navigation step.

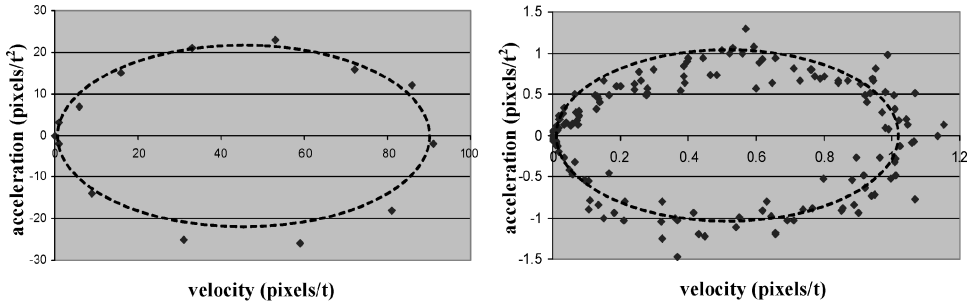


Fig. 2. Mouse motion acceleration against velocity of: (a) a positive pulse and (b) ten positive pulses.

prediction accuracy in such a setting, we propose here a hybrid model for predicting the motion of a 2D mouse during 3D navigation. The predicted mouse motion is then mapped to 3D for predicting the viewer's 3D movement.

To develop a reliable motion model for a moving mouse, we first analyze the movement pattern of the mouse while the user is navigating in a 3D environment. Figure 1 plots the vertical component of the mouse motion velocity against time during a sample navigation step. Here, a navigation step refers to the period when the mouse button is pressed until it is released. A typical navigation step contains a number of pulses. A positive pulse represents forward movement and a negative pulse represents backward movement. A similar graph can be obtained from the horizontal component of the mouse motion velocity.

3.1 An Elliptic Model for High-Velocity Motion

Figure 2 shows a plot of the mouse motion acceleration against velocity. Figure 2(a) shows a plot of one of the positive pulses from Figure 1. We can see that the trajectory of the scattered points can be approximated by an ellipse, which starts and ends at the origin, in a clockwise direction. The principle axis of the ellipse lies on the x-axis of the graph. The ellipse can be formulated as:

$$v^2B^2 - 2vAB^2 + A^2a^2 = 0, \tag{1}$$

where v is the velocity and a is the acceleration (i.e., $\mathrm{d}v/\mathrm{d}t$). $2A$ and $2B$ are the lengths of the major and the minor axes of the ellipse, respectively. Figure 2(b) shows a plot of ten positive pulses from Figure 1 by normalizing the width and height of each pulse. We can see that the elliptic shape of the trajectory of the data points is even more obvious here.

For a negative pulse, the ellipse can be formulated as:

$$v^2B^2 + 2vAB^2 + A^2a^2 = 0. \tag{2}$$

We may solve the differential equation (1) by defining the initial condition $v = 0$ at $\mathrm{t} = 0$, and we have:

$$v = K_1(\cos(K_2t) - 1), \tag{3}$$

where $K_1 = -A$ and $K_2 = {}^{B}/_{A}$. Both of them are arbitrary nonzero constants. Similarly, by solving Eq. (2), we have the same equation (Eq. (3)), but with $K_1 = A$ and $K_2 = {}^{B}/_{A}$. Hence, K_1 is a negative value for a positive pulse and a positive value for a negative pulse. The height and width of a pulse determine the absolute values of K_1 and K_2, respectively. A higher pulse causes a higher absolute value of K_1 and a wider pulse causes a smaller absolute value of K_2. Since K_1 and K_2 are constants within the period of a pulse and need to be recomputed for every new pulse, we refer to them as the *pulse constants*. Note that t needs to be reset to zero once a new pulse is detected.

3.2 A Linear Model for Low-Velocity Motion

We observe from Figures 1 and 2 that most of the sample points are located around the origin, representing low-velocity motion. It is inappropriate to interpret the trajectory as an ellipse due to the increased significance of noise and distortion on the sampled data. Hence, we model low-velocity motion with a generic formula commonly used in dead reckoning:

$$v = v_0 + a * \Delta t, \tag{4}$$

where v_0 is the initial velocity and v is the velocity after time Δt. Once the velocity is high enough, we may compute the pulse constants and then switch to the elliptic model as shown in Eq. (3). Hence, when a new pulse is detected, the first two velocity values, which are higher than the threshold value, are used to determine the pulse constants.

3.3 Determining the Pulse Constants

To reduce the effects of noise in the sampled data and the elliptic approximation of the mouse motion, we use a Kalman filter in our predictor to determine the values of the two pulse constants K_1 and K_2 for each pulse. To apply the Kalman filter on Eq. (3), K_1 and K_2 can be regarded as components of the state vector in the filter. This is different from Azuma and Bishop [1995], since our method does not take the velocity and acceleration values as the components of the state vector. Here, we first determine the preliminary values of the pulse constants and then use them as the state vector of the Kalman filter. Let v_1 and v_2 be

the first and the second velocity values, respectively, taken since the start of a pulse. We have:

$$v_1 = K_1(\cos(K_2(1)) - 1) \tag{5}$$

$$v_2 = K_1(\cos(K_2(2)) - 1). \tag{6}$$

By solving these two equations, we obtain

$$K_1 = 2v_1^2 \,/\, (v_2 - 4v_1) \tag{7}$$

$$K_2 = \cos^{-1}(v_2 \,/\, (2v_1) - 1). \tag{8}$$

If v_1 and v_2 taken are not valid, for example, when one of them is zero, the system will continue to use the linear model for prediction and v_1 and v_2 taken previously are ignored. Sometimes, t may reach $2\pi/K_2$. If this happens, the predictor will regard the pulse as terminated and wait for the start of a new pulse. In the case if v does not reach zero when t exceeds $2\pi/K_2$, we set $v_1 = v$ and assume it to be the beginning of a new pulse without a "zero start". Another point to note is that v_2 should be greater than v_1 in terms of their absolute values. Otherwise, they are considered as invalid.

Since most of the sample points are clustered around the origin during low-velocity motion, if we interpret the data points with our elliptic model in order to estimate the preliminary values of the pulse constants, the increased significance of noise and distortion in the sampled data may increase the error of the estimation. This error will affect the prediction accuracy for the complete pulse period. To reduce this estimation error, we further adjust the values of the pulse constants obtained from Eq. (7) and (8) by the characteristics of the user's past motion behavior as follows:

$$K_1^* = K_1(\delta_1 + \beta_1 \bar{v}) \tag{9}$$

$$K_2^* = K_2(\delta_2 + \beta_2/\bar{l}), \tag{10}$$

where K_1^* and K_2^* are the *adjusted pulse constants*. δ_1, δ_2, β_1 and β_2 are some constants. $\bar{v}$ is the EWMA absolute velocity determined in the last prediction step and $\bar{l}$ is the width of the EWMA pulse obtained from the last motion pulse. $\bar{v}$ and $\bar{l}$ can be calculated in a way similar to our earlier EWMA scheme Chim et al. 1998]:

$$\bar{v} = \begin{cases} a_1\hat{v} + (1-\alpha)v & \text{if } v \neq o \\ \text{unchange} & \text{otherwise} \end{cases} \tag{11}$$

$$\bar{l} = a_2\hat{l} + (1-\alpha_2)l, \tag{12}$$

where α_1 and α_2 are the weight constants, v is the distance moved by the mouse during the last step, l is the width of the last motion pulse, $\hat{v}$ and $\hat{l}$ are in fact $\bar{v}$ and $\bar{l}$ computed in the last step. After the sampling of the mouse velocity at each step, we update the value of $\bar{v}$ with Eq. (11). At the end of each pulse, we update the value of $\bar{l}$ with Eq. (12). From Eq. (9) and (10), K_1^* increases with increasing $\bar{v}$ while K_2^* decreases with increasing $\bar{l}$, to capture the user's past motion behavior. In addition, we use the EWMA values of v and l instead of

their mean values to adjust the values of the pulse constants as EWMA may adapt quicker to the change in the user's motion characteristics.

3.4 Determining Other Parameters

Typically, the process noise covariance matrix Q used in a predictive Kalman filter is fixed. However, we use two process noise covariance matrices here, Q_H and Q_L. For the first iteration of each pulse, we use Q_H, which has high absolute component values. Then, we switch to Q_L, which has relatively lower absolute component values, for the rest of the pulse. This design is based on the fact that the preliminary values of the pulse constants derived from Eq. (7) and (8) may introduce a higher error than those generated by the Kalman filter in its steady state. So, we use Q_H to compensate for the higher uncertainty. On the other hand, we fix the measurement error covariance matrix R.

If we assign the state vector as $\hat{x}_k = (K_1^*, K_2^*)^T$, the update of the state vector in a linear form is:

$$\hat{x}_{k+1}^- = \hat{x}_k + u_k, \tag{13}$$

where u_k is some vector for the first iteration of the Kalman filter at each pulse and a zero vector for the rest of the pulse. For the evaluation of u_k at the beginning of each pulse, we need not determine it explicitly. We may just evaluate $(K_1^*, K_2^*)^T$ for the new pulse as described above and assign it to $\hat{x}_{k+1}^-$. With measurement update, the Kalman filter will generate $\hat{x}_k$ as the a posteriori estimate for the current step and $\hat{x}_{k+1}^-$ as the a priori estimate for the next step. The state transition matrix ϕ_k is fixed to I. On the other hand, the measurement process as shown in Eq. (3) is in a nonlinear form and hence, by applying the extended Kalman filter, the measurement update step can be written as:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - h_t(\hat{x}_k^-)), \tag{14}$$

where $h_t(\hat{x}_k^-)$ is a function equivalent to the right-hand side of Eq. (3), t is the number of time units since the start of the current pulse, z_k is the measured velocity value and K_k is the Kalman gain. The matrix corresponding to the noiseless connection between the state vector and the measurement is denoted as H_k. It is determined by:

$$H_k = ((\partial h(x)/\partial x)|_{x=\hat{x}_k^-}), \tag{15}$$

which can be rewritten as:

$$H_k = (\cos(K_2 t) - 1, -K_1 t \sin(K_2 t)), \tag{16}$$

by substituting Eq. (3) into Eq. (15). (Please refer to Grewal and Andrews [1993].) The predicted mouse position can be calculated by:

$$p = p_0 + K_1(\cos(K_2(\tau + t_0)) - 1), \tag{17}$$

where t_0 is the current t value, τ is the prediction length and p_0 is the current position of the mouse.

To find the values of parameters Q_H, Q_L, R, β_1, β_2, δ_1, δ_2, α_1 and α_2, we apply the Powell's method [Press et al. 1992] by minimizing the absolute error

for one step prediction as the objective function:

$$Q_H \approx \begin{pmatrix} 37.852 & 0.991 \\ 0.991 & 47.852 \end{pmatrix} Q_L \approx \begin{pmatrix} 30.021 & 0.000 \\ 0.000 & 40.011 \end{pmatrix}$$

$$R \approx (2.000) \quad \beta_1 \approx 0.001 \quad \beta_2 \approx 3.448 \quad \delta_1 \approx 0.03$$

$$\delta_2 \approx 0.165 \quad \alpha_1 \approx 0.650 \quad \alpha_2 \approx 0.800$$

For simplicity, the predictor for the horizontal component of the mouse motion and the predictor for the vertical component of the mouse motion share this same set of parameters. Moreover, the threshold for switching from the linear model to the elliptic model Δ_V is set to 2 pixels.

3.5 Summary of Our Prediction Method

The prediction method that we propose here can be summarized in a pseudocode segment as shown in Figure 3. Basically, the prediction starts as soon as the user presses the mouse button to indicate the beginning of a navigation step. Whenever the mouse velocity goes below the threshold value, we assume the end of a pulse and a possible beginning of a new one. We use the linear model for prediction. When two consecutive samples are found to be higher than the threshold value and show an increasing trend, we assume the start of a new pulse. We determine the pulse constants and switch to the elliptic model for prediction for the rest of the pulse.

4. CACHING AND PREFETCHING MODEL

To guarantee object availability in DVE, Falby et al. [1993] maintains a region of the VE around the viewer at the client. This region is larger than the *area of interest* (or AOI) of the viewer. As the viewer moves towards one direction, new objects entering the region may begin to be transmitted to the client and hence, there is no need to do any prefetching. In Aliaga et al. [1999], a simple method is used to prefetch object models from the secondary memory to the main memory based on the velocity and line of sight of the viewer. In both of these methods, the amount of data prefetched is often far higher than necessary. A motion prediction method specifically designed for 3D navigation will certainly help reduce the amount of data needed to be prefetched. In Funkhouser et al. [1992], objects nearer to the viewer are assumed to have a higher chance to be retrieved. Although this assumption can be applied to cache replacement, it is not an accurate predictor.

In our earlier work [Chim et al. 1998, 2003], we proposed the MRM (most required movement) scheme in defining the access score of each object. It is based on the observation that the farther an object is from the viewer, the longer it will take for the viewer to move directly in front. Hence, the value of caching this object locally or prefetching it from the server is lower. Similarly, the larger the angular distance of an object is from the viewer's line of sight, the value of caching or prefetching the object is also lower. Figure 4 shows this relationship, with D_o representing the distance of object o from viewer v and

```
count = 0;
while (navigating) {
 mouse_v = GetMouseVelocity();
 if (|mouse_v| < threshold Δ_V ) {
      /* a new pulse is detected */
      predicted_v = LinearModel(mouse_v);
      count = 0;
 } else if (count == 0) {
      v_1 = mouse_v;
      predicted_v = LinearModel(mouse_v);
      count ++;
 } else if (count == 1) {
      if ((|mouse_v| > |v_1|) && (Sign(mouse_v) == Sign(v_1)) {
           v_2 = mouse_v;
           count ++;
           (K_1, K_2) = FindPreliminaryPulseConst(v_1, v_2);
           ( K_1^* , K_2^* ) = FindAdjustedPulseConst(K_1, K_2);
           state vector x̂_k = ( K_1^* , K_2^* );
           process noise Q = Q_H ;
      } else
           v_1 = mouse_v;
      predicted_v = LinearModel(mouse_v);
 } else {
      ( x̂_k , x̂⁻_{k+1} ) = KalmanFilter(Q);
      if (t == 3)
           process noise Q = Q_L ;
      /* detect irregular end of pulse */
      if ((count+1) > 2π / K_2^* ) {
           /* end of the current pulse and start of a new one */
           if (|mouse_v| < threshold Δ_V ) {
                v_1 = mouse_v;
                count = 0;
           }
           predicted_v = 0;
      } else
           predicted_v = EllipticModel( K_1^* , K_2^* );
      count ++;
 }
 output(predict_v);
}
```

Fig. 3. The pseudocode of the prediction algorithm.

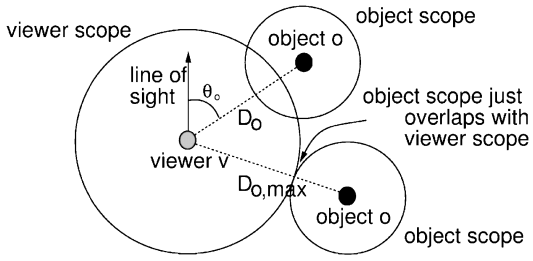


Fig. 4. Determining the access score of an object.

θ_o representing the angular distance of o from v's line of sight. Here, a *viewer scope* defines how far the viewer can see. An *object scope* defines how far the object can be seen and is set proportional to the size of the object.

The access score $S_{\mathrm{o,v}}$ of object o to viewer v is computed as:

$$S_{\mathrm{o,v}} = \alpha\left(1 - \frac{D_{\mathrm{o}}}{D_{\mathrm{o,max}}}\right) + (1-\alpha)\left(1 - \frac{|\theta_{\mathrm{o}}|}{\pi}\right), \tag{18}$$

where r_o is the radius of the object scope, r_v is the radius of the viewer scope, and α is a parameter between 0 and 1 to determine the contributions of D_{o} and θ_{o} to $S_{\mathrm{o,v}}$.

In our DVE prototype, we employ a navigation mapping technique similar to that of a typical VRML browser. We map the vertical component of the mouse motion to the viewer's forward or backward motion velocity and the horizontal component to the viewer's rotational velocity, that is, change of line of sight, in the 3D environment.

Previously, we use the EWMA scheme to predict where the viewer will be in the next step. Based on this predicted location, the client will request the server for objects that the viewer will see if the viewer does indeed move to the location. However, due to the dramatic increase in error as we attempt to predict more steps ahead using the EWMA scheme, we could only prefetch objects one step ahead. Due to the typical long latency of the Internet, unfortunately, the objects requested may not have reached the client in time for rendering the next frame. In addition, due to the limitation in accuracy of the EWMA scheme, we did not use the predicted location for object caching. Instead, we simply cached objects based on the current location of the viewer. This caching strategy has the problem that when the cache is full, the caching algorithm will remove objects based on the current location of the viewer. However, objects that are considered as visually less important at the current location and removed from the cache may possibly become visually important in the next few steps.

To apply the new prediction method in the DVE prototype, we need to perform two prediction computations at each frame time based on the current mouse velocity, one for predicting the horizontal velocity and the other for the vertical velocity. Each of the prediction computations requires its own K_1 and K_2. The prediction results are mapped to the 3D environment as the viewer's predicted velocity vectors, which can then be used to determine the viewer's future positions in the 3D environment. As will be shown in the next section, the new prediction method is very accurate even when we predict a few steps ahead. Hence, we have revised both the caching and the prefetching algorithms of our DVE prototype to use the proposed prediction method. We have experimented the accuracy of the two algorithms with different numbers of prediction steps. As will be shown later, increasing the number of prediction steps can improve the cache hit ratio significantly.

5. RESULTS AND DISCUSSIONS

We have implemented three predictors, P_{GMM}, P_{SOP}, and P_{HM} for our experiments. All of them are operated with a Kalman filter. P_{GMM} and P_{SOP} are two

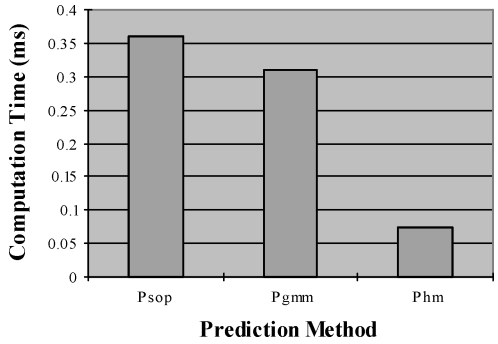


Fig. 5. Comparison of computational costs.

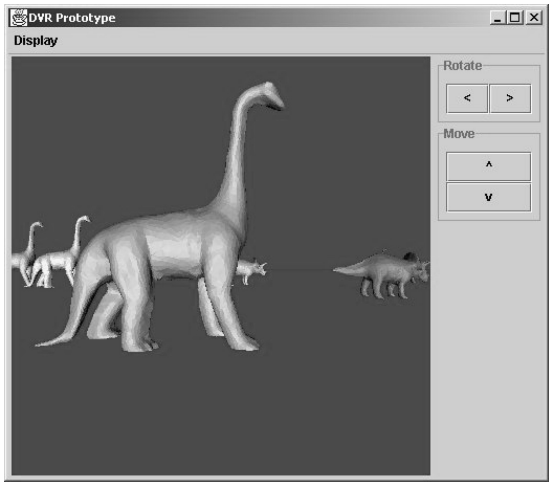
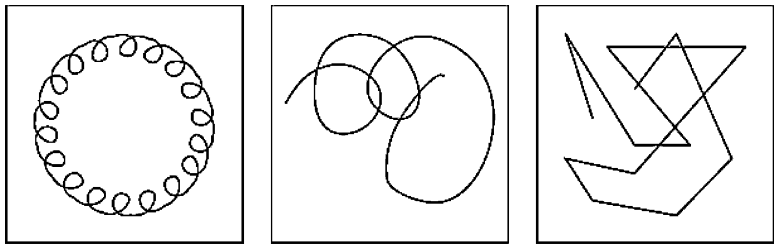


Fig. 6. Snapshot of our DVE prototype system.

popular predictors. P_{GMM} is a predictor using the Guass–Markov process modeling and P_{SOP} is a second-order polynomial predictor. P_{HM} is our hybrid model prediction method proposed here. We have integrated all these three predictors into the caching and prefetching algorithms in our DVE system for comparison. During all our experiments, the client module with the three predictors ran on a PC with a Pentium III 800 MHz CPU, while the server module ran on a Sun workstation with an UltraSPARC-IIi 270 MHz CPU.

Figure 5 compares the computational costs of the three predictors. We observe that P_{SOP} has the highest computational cost. This is because its state vector and the measurement vector have sizes of 3 and 2, respectively. They are larger than those in P_{GMM}, which have sizes of 2 and 2. P_{HM} has the smallest state vector and measurement vector sizes of 2 and 1, respectively. Since the computational cost increases with the sizes of the two vectors, our method is the fastest of all three methods.

We have conducted some navigation experiments on the three prediction methods using our DVE prototype. Figure 6 shows a snapshot of the prototype



(a) CP (b) CCP (c) RW

Fig. 7. Navigation patterns: (a) CP, (b) CCP, and (c) RW.

system. The VE that we use for our experiments has a size of $10^4 \times 10^4$ square units containing a total of 5,000 objects. These objects are similar in size and are evenly distributed over the VE. The average size of each object is about 300 KB and the total database size is approximately 1.5 GB.

We have experimented four types of navigation patterns as shown in Figure 7. Figure 7(a) shows the *CP* pattern, which models a constant circular translation pattern. Figure 7(b) shows the *CCP* pattern, which models a changing circular pattern. Figure 7(c) shows the *RW* pattern, which models a random walk around the environment. Here, we further divide the RW pattern into two types. One has a fast-navigation speed, called *RW-F*, and the other has a slow-navigation speed, called *RW-S*. Note that the high-speed random motion of *RW-F* does not model real situation; we include it in our experiments just to test the performance of our prediction method under extreme condition.

To demonstrate the effectiveness of the proposed prediction method, we first compare the prediction errors of it with other methods. We then investigate the cache hit ratios as a result of applying various prediction methods. Finally, we discuss the limitation of the proposed method. Note that in all our experiments, we set the rendering rate as 10 frames per second, just to simply our discussion. Since the prediction rate is the same as the rendering rate, each prediction step is therefore 0.1 s.

5.1 Prediction Errors

We have studied the prediction error of the proposed method in two steps. First, we investigate the prediction error of the method when a typical user uses the prototype system. Second, we look at how the prediction error of the method changes with different users.

5.1.1 *Typical Prediction Errors.* Before studying the prediction error, we need to measure the network latency with respect to the change in the geographical distance between the client and the server. To do this, we send "ping" messages to some public sites and measure the time for us to receive echoes. This is equivalent to measuring the round trip time (RTT) between two sites. Table I shows that when the client and the server are located within the same LAN, the network latency is very small and can be neglected in our application. When the two machines are linked with a local Internet connection (within

Table I. Network Latencies of Different Network Connections

Network Connection	No. of Hops	RTT
LAN (two hosts within the university network)	4	0.64 ms
Internet (local connection: two hosts within HK)	10	0.01 s
Internet (overseas connection: one host in HK, another in US)	13	0.2 s

Hong Kong), the latency is still relatively small. However, when the two machines are linked with an overseas Internet connection, the latency becomes significant and is equivalent two prediction steps in our experiment (not including the data processing time). Note that all the figures shown in Table I assume a small packet size. In a LAN environment, the effect of large packet size may be less significant as the network bandwidth is usually large. However, in the Internet environment (whether it is a local or an overseas connection), the network bandwidth is usually much lower. It may take extra time to download all the objects requested within a single prediction step. Hence, it is important to study the prediction error of our method at different prediction lengths.

We have compared the accuracy of the three predictors P_{GMM}, P_{SOP}, and P_{HM} at one to five prediction steps with each of the four navigation patterns: CP, CCP, RW-F and RW-S. The parameters used in P_{GMM} and P_{SOP} are also determined by the Powell's method, in a way similar to those determined for P_{HM}. Figure 8 shows the results of the experiment. The navigation system computes the predicted location every 0.1 s. From the figure, we observe that P_{SOP} may be the most inaccurate method among the three. As pointed out by Azuma and Bishop [1995], the accuracy of a polynomial-based predictor is highly dependent on the prediction length and the contribution of high frequencies in its signal spectrum [Oppenheim et al. 1997]. The larger they are, the greater the error is expected. So, as the adverse performance of P_{SOP} shows, the prediction length or the high-frequencies contribution may be too high for this method. It is believed that the performance may improve significantly by reducing the sample size. This can be regarded as a reduction on the prediction length or the high-frequencies contribution since the velocity value will be lowered if the time unit is decreased. However, neither of them can be manipulated by us in DVE applications.

Figure 9 shows the responses of the three predictors. Figure 9(a) sketches the second order polynomial predictor (P_{SOP}), where $p = 0$, $v = 5$ and $a = 15$. At high prediction length, the slope of the curve becomes steeper and steeper. This does not match with the shape of the pulses shown in Figure 1. Hence, the generic polynomial predictor is not suitable for predicting mouse motion. By comparing Figure 9(b)—P_{GMM} and Figure 9(c)—P_{HM}, we observe that they are similar in accuracy at low prediction length. However, as we increase the prediction length, P_{HM} becomes more and more outstanding. This can be easily explained by comparing the outputs of the equations. According to the transition matrix in P_{GMM}, the predicted velocity is:

$$v = v_o + a * (1 - e^{-\beta t})/\beta \tag{19}$$

where β is some constant. The curve is plotted in Figure 9(b), with v = 5 and a = 15. At the beginning of the pulse, the output is similar to those in Figure 1. However, as we increase the prediction length, the output approaches

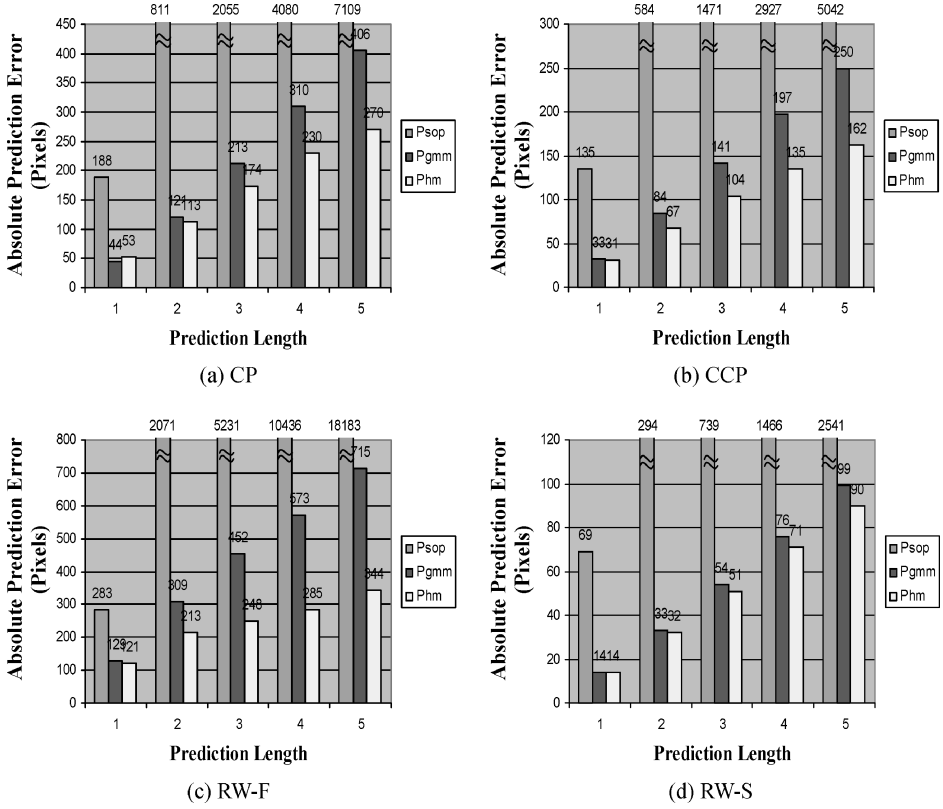


Fig. 8. Comparison of prediction errors among different prediction methods.

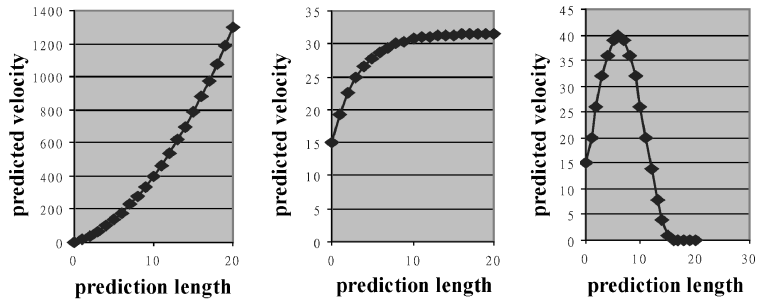


Fig. 9. A sketch of the responses of the three predictors.

to some constant value. On the other hand, PHM drops to zero eventually, which matches nicely with those in Figure 1. This accounts for the outstanding performance of P_{HM} over P_{GMM}.

5.1.2 *Change in Prediction Errors with Different Users.* To study how the prediction error of our predictor varies with different users, we have conducted

Table II. The Statistics of Prediction Errors from 10 Users

User	x Mean	x Variance	y Mean	y Variance	Total Samples
#1	−0.0432	60.054	−0.0227	41.052	240
#2	0.0276	55.879	0.0198	29.939	187
#3	0.0299	39.994	0.0235	33.581	335
#4	0.0232	57.64	0.0210	40.087	200
#5	−0.0420	60.504	−0.0278	35.768	566
#6	0.0290	54.564	−0.0261	34.523	323
#7	0.0157	52.435	0.0105	23.359	182
#8	−0.0169	49.954	−0.0256	35.929	591
#9	0.0123	53.255	−0.0383	36.069	243
#10	0.0123	53.345	−0.0356	35.120	137

a simple experiment to examine the error variances, S2, of 10 users at one prediction step. S2 is defined as:

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}(e_i - \bar{e})^2 \tag{20}$$

where e_i is the observed error, $\bar{e}$ is the error mean, and N is the number of samples. It is reasonable to assume a Gaussian distribution of error with a zero mean. As the prediction error increases, the corresponding S^2 is expected to increase also. Table II shows the error mean, the variance and the sample size of each of the 10 users in x and y directions.

We can tell whether the prediction results of all the users are similar by testing the null hypothesis that all variances are the same, H_0: $\sigma_1 = \sigma_2 = \cdots = \sigma_k$, against the alternative hypothesis that there is at least one different variance, H_1: not all are the same, where $k = 10$ is the number of experimental groups. We adopt the Bartlett's test [Montgomery 2001] here, which test statistics are:

$$\chi_0^2 = \frac{(N-k)\ln S_p^2 - \sum_{i=1}^{k}(N-1)\ln S_i^2}{1 + \frac{1}{3(k-1)}\left(\left(\sum_{i=1}^{k}\frac{1}{N_i-1}\right) - \frac{1}{N-k}\right)}, \tag{21}$$

where S_i^2 is the variance of ith group, N_i is sample size of the ith group, and N is the total sample size. The test statistics χ_0^2 is approximated by the chi-square distribution with $k-1$ degrees of freedom. The pooled variance, S_p^2, is given by:

$$S_p^2 = \frac{1}{N-k}\sum_{i=1}^{k}(N_i - 1)S_i^2. \tag{22}$$

Now, H_0 can be rejected if $\chi_0^2 > \chi_{\alpha,k-1}^2$ at significant level α, which is set to be 0.01. We have found that χ_0^2 of the x and the y prediction errors are 21.18 and 21.03, respectively. Both figures are less than $\chi_{\alpha,k-1}^2$, which is equal to 21.67. Thus, we may conclude that all the users have similar error variances.

Table III. Settings of Experiment #1

Test	Prefetching	Determining the Access Score at
# 1	No	Current Viewing Location
# 2	No	*EWMA* Predicted Location
# 3	No	P_{SOP} Predicted Location
# 4	No	P_{GMM} Predicted Location
# 5	No	P_{HM} Predicted Location
# 6	Yes	*EWMA* Predicted Location
# 7	Yes	P_{SOP} Predicted Location
# 8	Yes	P_{GMM} Predicted Location
# 9	Yes	P_{HM} Predicted Location

5.2 Hit Ratios

To study the performance of the three predictors under a real situation, we have performed some experiments on the prototype system [Chim et al. 2003]. The system employs a multiresolution modeling technique [Lau et al. 1998] to organize each object model for progressive transmission. The access score of each relevant object in the VE is computed at each step to determine the objects for caching and for prefetching and their appropriate resolutions, so as to increase the availability of objects while minimizing the transmission and rendering costs. The cache size is fixed at 4 MB.

To compare the performance of the three motion predictors, we describe here three experiments conducted with the prototype system to measure the cache hit ratios of the three predictors. The cache hit ratio refers to the percentage of bytes of renderable objects, that is, objects which are inside the viewing region of the viewer, available from client's local cache at the time of rendering. A high cache hit ratio implicitly indicates that the output images have high visual quality.

5.2.1 *Experiment* #1. In the first experiment, we investigate the performance of the caching and prefetching mechanisms with different prediction methods, including P_{GMM}, P_{SOP}, P_{HM} and the EWMA method [Chim et al. 1998, 2003]. (Although the EWMA prediction method is not suitable for predicting mouse motion, we have included here for reference.) The weight of the EWMA method is set at 0.5. Table III shows the settings of the experiment. Although the experiment was conducted based on the RW-S walking pattern, similar results are found with other walking patterns. Test #1 is a base case for comparison—no prefetching is preformed and the access score is determined by the user's current location. In tests #2 to #5, we use the four prediction methods to predict the viewing location and line of sight of the viewer in the next step, which are in turn used to determine the access scores of objects for caching. However, no prefetching is performed. Tests #6 to #9 are similar to tests #2 to #5 except that prefetching is performed this time based on the predicted viewing location and line of sight.

From Figure 10, we may observe that even if there is no prefetching performed in test #1, the hit ratio is still reasonably high, due to the small incremental change in object visibility between consecutive frames and the

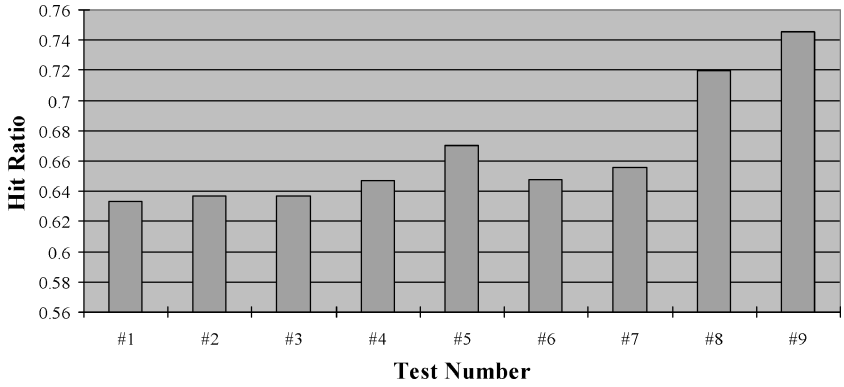


Fig. 10. Performance of different methods under different settings.

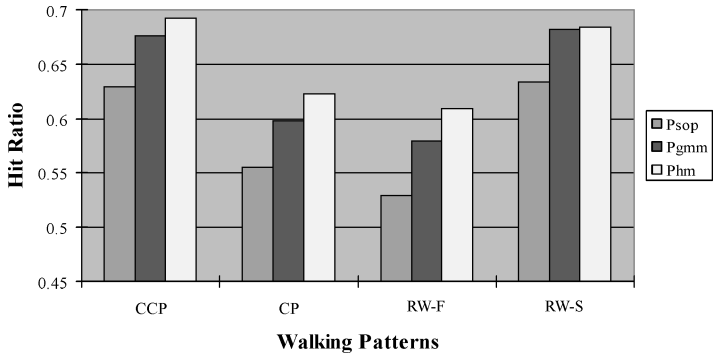


Fig. 11. Performance of different methods under different navigation patterns.

satisfactory performance of the caching mechanism. From tests #2 and #3, we observe that EWMA and P_{SOP} perform only marginally better than test #1, which simply uses the current location to determine caching. From test #4, we observe that P_{GMM} performs slightly better than the other methods. However, from test #5, we can see that our P_{HM} method performs significantly better than all the other methods shown in tests #1 to #4. This indicates that when a prediction method is accurate enough, using the predicted location to compute the access score helps determine future object visibility more accurately. From tests #6 to #9, we observe that all the methods perform relatively better when prefetching is used. In particular, we note that both P_{GMM} and P_{HM} have significant improvement on hit ratios when used with prefetching. This again indicates that both methods are more accurate in predicting object visibility in the next step.

5.2.2 *Experiment* #2. In the second experiment, the performance of the three predictors under different navigation patterns is evaluated. We have experimented the same four navigation patterns: CP, CCP, RW-F and RW-S. Figure 11 shows the hit ratios obtained under the four navigation patterns when all the predictors attempt to predict the viewer's location and line of sight

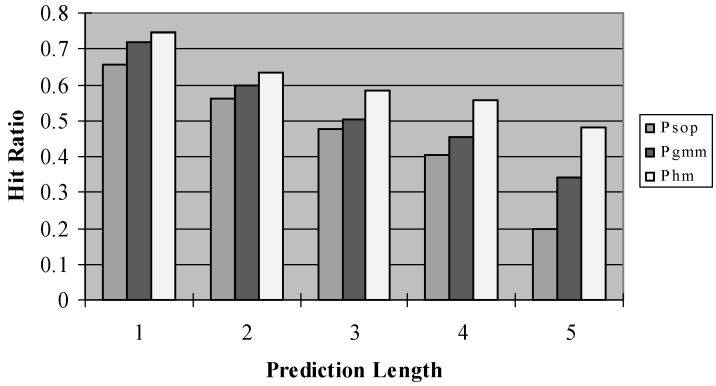


Fig. 12. Performance of different methods with different prediction lengths.

in the next step. We may observe that the hit ratios obtained from the RW-F pattern are generally lower than those from other navigation patterns. This is because under the RW-F pattern, the viewer moves more rapidly in a random manner, reducing the accuracy of the prediction. We may also observe that our method does better than the other two predictors under all the navigation patterns, due to its higher accuracy in predicting the mouse motion velocity.

5.2.3 *Experiment* #3. In the third experiment, we investigate the performance of the predictors at different prediction lengths and the performance of the resulting prefetching mechanism. Again, the experiment was conducted based on the RW-S walking pattern.

We test the three predictors P_{GMM}, P_{SOP}, and P_{HM} in predicting from 1 to 5 steps ahead, that is, the system will try to prefetch objects which may be accessed in the subsequent 1 to 5 steps, respectively. In Figure 12, we observe that the hit ratios of all three methods decrease with increasing prediction and prefetching steps. This is because at higher prediction length, the predictor needs to predict the viewer's location and line of sight further ahead of time, that is, few more steps later. As shown in Figure 8, this will generally result in a high prediction error. Here, we may observe that P_{HM} performs better than all the other methods in particular in higher prediction length. This again agrees with Figure 8 that the prediction error of P_{HM} is comparatively lower than the other methods in particular in high prediction length.

5.3 Experimental Evaluation

In this section, we have demonstrated the performance of the proposed prediction method for predicting the mouse motion through various experiments. In particular, we have shown that in the Internet environment, where the network latency is high and the network bandwidth is low, it is important for the proposed method to be able to predict more steps ahead with low prediction error. Our results show that the proposed method has lower prediction error compared with some popular prediction methods, in particular at high prediction length.

A major limitation of our method, however, is that its error rate can be affected by the input noise caused by the nonlinear mouse motion. When a user moves the mouse forward, for example, due to mechanical or fictional problems, the mouse may sometimes slow down in the middle of its motion and then continue its forward motion again. This motion jagginess may cause a dip in the otherwise elliptic pulse and affect the prediction error. However, with the popularity of the optical mouse, this problem should be much reduced.

6. CONCLUSIONS

The Internet is known to suffer from high-network latency and low-network bandwidth. These limitations have particularly high impact on interactive applications, such as DVE applications. In this article, we have proposed to address these limitations through prediction. We note that most desktop DVE applications still use the 2D mouse for navigation input and present a hybrid method for predicting the mouse motion velocity during 3D navigation. At low-velocity mouse motion, we use a linear model for prediction, and at high-velocity motion, we use an elliptic model for prediction. In addition, we also consider the user's motion characteristics to further improve the accuracy of the prediction.

There are many DVE applications that the proposed prediction method will be found useful, in particular, Internet gaming and collaborative design. The game industry is becoming increasingly more important in recent years. Its revenue is now comparable with that of the movie industry in some countries. Among all types of games, network games are receiving a lot of attention as most players enjoy the opportunity to play games with other people via the networks. However, Internet games are not yet popular due to the two limitations mentioned above. Another type of applications is collaborative design. Due to globalization, most companies would now have offices around the world. The design of a new product would often involve people of different expertises located at various offices around the world. Flying all these people for numerous meetings during the design stage would not only be expensive but also time consuming. A system that allows the team to interactively visualize and manipulate the design would be extremely useful.

One possible future research of this work is to consider applying the prediction method to other types of hand manipulated input devices, such as joysticks and driving wheels. We feel that these types of devices have motion characteristics similar to those of the 2D mouse. As an example, we are currently adapting the prediction method for use with the CyberGlove in collaborative design [Li et al. 2003]. (The CyberGlove is an electronic glove for sensing the user's hand gesture. It has 22 sensors to sense the relative joint positions of individual fingers.) In a multi-user design environment, timely delivery of the user's hand/finger positions is crucial in order to generate the real-time visual feedback. However, since the human hand has high degrees of freedom, if we predict each of the finger joints individually using our method, the accumulated error could be significant. To handle this issue, we are currently attempting to reduce the degrees of freedom by considering the kinematic constraints of the

finger motions in the prediction. Hopefully, this will help improve the accuracy of predicting the hand gesture.

REFERENCES

ALIAGA, D., COHEN, J., WILSON, A., BAKER, E., ZHANG, H., ERIKSON, C., HOFF, K., HUDSON, T., STURZLINGER, W., BASTOS, R., WHITTON, M., BROOKS, F., AND MANOCHA, D. 1999. MMR: An interactive massive model rendering system using geometric and image-based acceleration. In *Proceedings of ACM Symposium on Interactive 3D Graphics*. ACM, New York, 199–237.

AZUMA, R. AND BISHOP, G. 1995. A frequency-domain analysis of head-motion prediction. In *Proceedings of ACM SIGGRAPH'95*. ACM, New York, 401–408.

CAI, W., LEE, F., AND CHEN, L. 1999. An auto-adaptive dead reckoning algorithm for distributed interactive simulation. In *Proceedings of Workshop on Parallel and Distributed Simulation*, 82–89.

CALVIN, J., DICKEN, A., GAINES, B., METZGER, P., MILLER, D., AND OWEN, D. 1993. The SIMNET Virtual World Architecture. In *Proceedings of IEEE VRAIS'93*. IEEE Computer Society Press, Los Alamitos, Calif., 450–455.

CAPIN, T., PANDZIC, I., MAGNENAT-THALMANN, N., AND THALMANN, D. 1997. A dead-reckoning algorithm for virtual human figures. In *Proceedings of IEEE VRAIS'97*. IEEE Computer Society Press, Los Alamitos, Calif., 161–169.

CAPPS, M. 2000. QUICK Framework for task-specific asset prioritization in distributed virtual environments. In *Proceedings of IEEE VR'00*. IEEE Computer Society Press, Los Alamitos, Calif., 143–150.

CARLSSON, C. AND HAGSAND, O. 1993. DIVE—A multi-user virtual reality system. In *Proceedings of IEEE VRAIS'93*. IEEE Computer Society Press, Los Alamitos, Calif., 394–400.

CHAN, A., LAU, R. W. H., AND SI, A. 2001a. A motion prediction method for mouse-based navigation. In *Proceedings of CGI'01*. 139–146.

CHAN, A., LAU, R. W. H., AND NG, B. 2001b. A hybrid motion prediction method for caching and prefetching in distributed virtual environments. In *Proceedings of ACM VRST'01*. ACM, New York, 135–142.

CHIM, J., GREEN, M., LAU, R. W. H., LEONG, H., AND SI, A. 1998. On caching and prefetching of virtual objects in distributed virtual environments. In *Proceeding of ACM Multimedia'98*. ACM, New York, 171–180.

CHIM, J., LAU, R. W. H., LEONG, H., AND SI, A. 2003. CyberWalk: A web-based distributed virtual walkthrough environment. *IEEE Trans. Multimed. 5*, 4 (Dec.), 503–515.

DIS STEERING COMMITTEE. 1998. *IEEE Standard for Distributed Interactive Simulation—Application Protocols*. IEEE Standard 1278.

ELLSWORTH, D. 2001. Accelerating demand paging for local and remote out-of-core visualization. Available from *NAS Technical Report NAS-01-004*.

FALBY, J., ZYDA, M., PRATT, D., AND MACKEY, R. 1993. NPSNET: Hierarchical data structures for real-time three-dimensional visual simulation. *Comput. Graph. 17*, 1, 65–69.

FUNKHOUSER, T., SEQUIN, C., AND TELLER, S. 1992. Management of large amount of data in interactive building walkthroughs. In *Proceedings of ACM Symposium on Interactive 3D Graphics*. ACM, New York, 11–20.

GREWAL, M. AND ANDREWS, A. 1993. *Kalman Filtering: Theory and Practice*, Prentice-Hall, Englewood Cliff, N.J.

LAU, R. W. H., GREEN, M., TO, D., AND WONG, J. 1998. Real-time continuous multi-resolution method for models of arbitrary topology. *Presence 7*, 1 (Feb.), 22–35.

LI, F., LAU, R. W. H., AND NG, F. 2003. VSculpt: A Distributed virtual sculpting environment for collaborative design. *IEEE Trans. Multimed. 5*, 4 (Dec.), 570–580.

LI, F., LAU, R. W. H., AND KILIS, D. 2004. GameOD: An internet based game-on-demand framework. In *Proceedings of ACM VRST'04* (Nov.), ACM, New York, 129–136.

LIANG, J., SHAW, C., AND GREEN, M. 1991. On temporal-spatial realism in the virtual reality environment. In *Proceedings of ACM UIST'91*. ACM, New York, 19–25.

LIU, G. AND MAGUIRE, G. 1996. A class of mobile motion prediction algorithms for wireless mobile computing and communications. *Mobile Netw. Applic. 1*, 2, 113–121.

LIU, T. AND BAHL, P. 1998. Mobility modeling, location tracking, and trajactory prediction in wireless ATM networks. *IEEE J. Sele. Areas Commun. 16*, 6 (Aug.) 922–936.

MONTGOMERY, D. 2001. *Design and Analysis of Experiment* (5th Ed.). Wiley, New York.

NORMAND, V., BABSKI, C., BENFORD, S., BULLOCK, A., CARION, S., FARCET, N., FRECON, E., KUIJPERS, N., MAGNENAT-THALMANN, N., RAUPP-MUSSE, S., RODDEN, T., SLATER, M., SMITH, G., STEED, A., THALMANN, D., TROMP, J., USOH, M., VAN LIEMPD, G., HARVEY, J., AND KLADIAS, N. 1999. The COVEN project: Exploring applicative, technical, and usage dimensions of collaborative virtual environments. *Presence*, *8*, 2, 218–236.

OPPENHEIM, A., WILLSKY, A., AND NAWAB, S. 1997. *Signals and Systems* (2nd Ed.), Prentice Hall, Englewood Cliff, N.J.

PANDZIC, I., CAPIN, T., LEE, E., MAGNENAT-THALMANN, N., AND THALMANN, D. 1997. Flexible architecture for virtual humans in networked collaborative virtual environments. In *Proceedings of Eurographics'97*, 177–188.

PRESS, W., TEUKOLSKY, S., VETTERLING, W., AND FLANNERY, B. 1992. *Numerical Recipes in C* (2nd Ed.), Cambridge University Press.

SINGH, G., SERRA, L., PNG, W., AND NG, H. 1994. BrickNet: A software toolkit for network-based virtual world. *Presence 3*, 1, 19–34.

SINGHAL, S. AND CHERITON, D. 1995. Exploiting position history for efficient remote rendering in networked virtual reality. *Presence 4*, 2, 169–193.

The VRML97 Specification. Available at http://www.web3d.org/.

WU, J. AND OUHYOUNG, M. 2000. On latency compensation and its effects on head-motion trajectories in virtual environments. *The Visual Comput. 16*, 2, 79–90.

Received February 2003; revised August 2003; accepted October 2003