Using a Position History-Based Protocol for Distributed Object Visualization

Sandeep K. Singhal David R. Cheriton

Department of Computer Science
Stanford University

Abstract

Users of distributed virtual reality applications interact with users located across the network. Similarly, distributed object visualization systems store dynamic data at one host and render it in real-time at other hosts. Because data in both systems is animated and exhibits unpredictable behavior, providing up-to-date information about remote objects is expensive. Remote hosts must instead apply extrapolation between successive update packets to render the object's true animated behavior. This paper describes and analyzes a "position history-based" protocol in which hosts apply several recent position updates to track the position of remote objects. The history-based approach offers smooth, accurate visualizations of remote objects while providing a scalable solution.

1 Introduction

In distributed object visualization systems, a host must accurately display dynamic data located at other hosts on a network. For example, each participant in a virtual reality environment moves about the virtual world while continually interacting with other participants physically located at other hosts (Figure 1); each machine must display the position and orientation of the local participant along with the position and orientation of all visible participants.

Because virtual reality systems are highly interactive, users expect a positionally accurate, behaviorally accurate, and smoothly rendered view of remote participants. Each user expects the visual representations of other participants to exhibit *positional accuracy*: the average position error of each remote representation should be small. For example, in an auto-racing simulation, the user requires an accurate representation of competitors' cars in order to steer his own car and avoid collisions. Any perceived inaccuracy leads to confusion, incorrect actions, and inconsistent responses. The visual representations of remote objects should also exhibit *behavioral accuracy*: the velocity and acceleration of each remote representation should mirror the behavior of the true object (though the actual values may not be accurate). For example, the user might want to see that a distant car is swerving repeatedly even if he tolerates seeing inaccuracies in the car's actual position. Finally, the remote object should be animated smoothly at the local frame rate. Distributed simulations therefore appear "seamless," meaning that users should be unable to distinguish between local and remote objects on the display.

Transmission at the sender's frame rate of state updates to remote hosts is infeasible because of network latency and bandwidth. Networks do not guarantee timely, in-order delivery of packets.

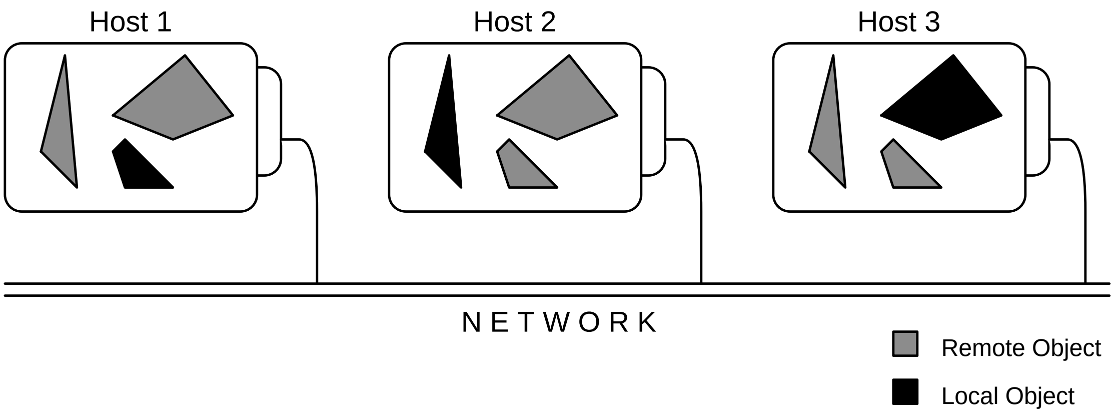


Figure 1: Distributed Virtual Reality Applications Display Local and Remote Participants

Remote hosts must repeat the frame if a packet is delayed because they cannot wait for the data to arrive. Moreover, most existing networks cannot support the necessary bandwidth for real-time data. Transmitting 128-byte update packets at 60 frames per second, only 52 objects are needed to impose a 40% load on a 10Mbps ethernet. Finally, a receiver's frame rate may be faster than that of the sender, in which case the remote object rendered on the receiver's display does not exhibit the same smoothness as local objects. The SGI Flight Simulator, for example, transmits frame-by-frame position updates. The display at remote sites therefore depends on the transmitter's update rate as well as the underlying network behavior. As a result, users often see a jerky view of non-local objects.

A few existing simulations have utilized "dead reckoning" algorithms to overcome the update rate problem, but almost no work has been done to study the accuracy of these techniques. In a dead reckoning protocol, each host broadcasts state information about local objects at lower than frame rate frequencies. All other hosts generate and display an approximate animation of the remote participant by applying predictive techniques based on the available information. For example, the Amaze multiplayer game [3] uses occasional position and velocity updates to track the location of remote players. However, the slow speed of objects relative to update rate in this game simplifies the remote tracking problem considerably because the potential error in the remote model is always small. Large virtual environments targeted by the DIS protocol [7] rely on position, velocity, and acceleration updates to produce remote animations. However, no study has measured the overall accuracy of the DIS dead reckoning algorithm for a general set of object motions. The Amaze and DIS protocols only use information from the most recent packet to predict future behavior of an object.

In this paper, we study a communication protocol in which objects transmit only their absolute position to remote hosts. Using this approach, remote sites track the future object position based on several previous updates and use curve fitting to characterize the overall behavior of the object. After describing the basic protocol and the supporting "position history-based" algorithms for remote object modeling, we analyze the accuracy of these techniques.[1] By studying various classes

[1] A separate paper [13] analyzes the network and data rate performance of the protocol.

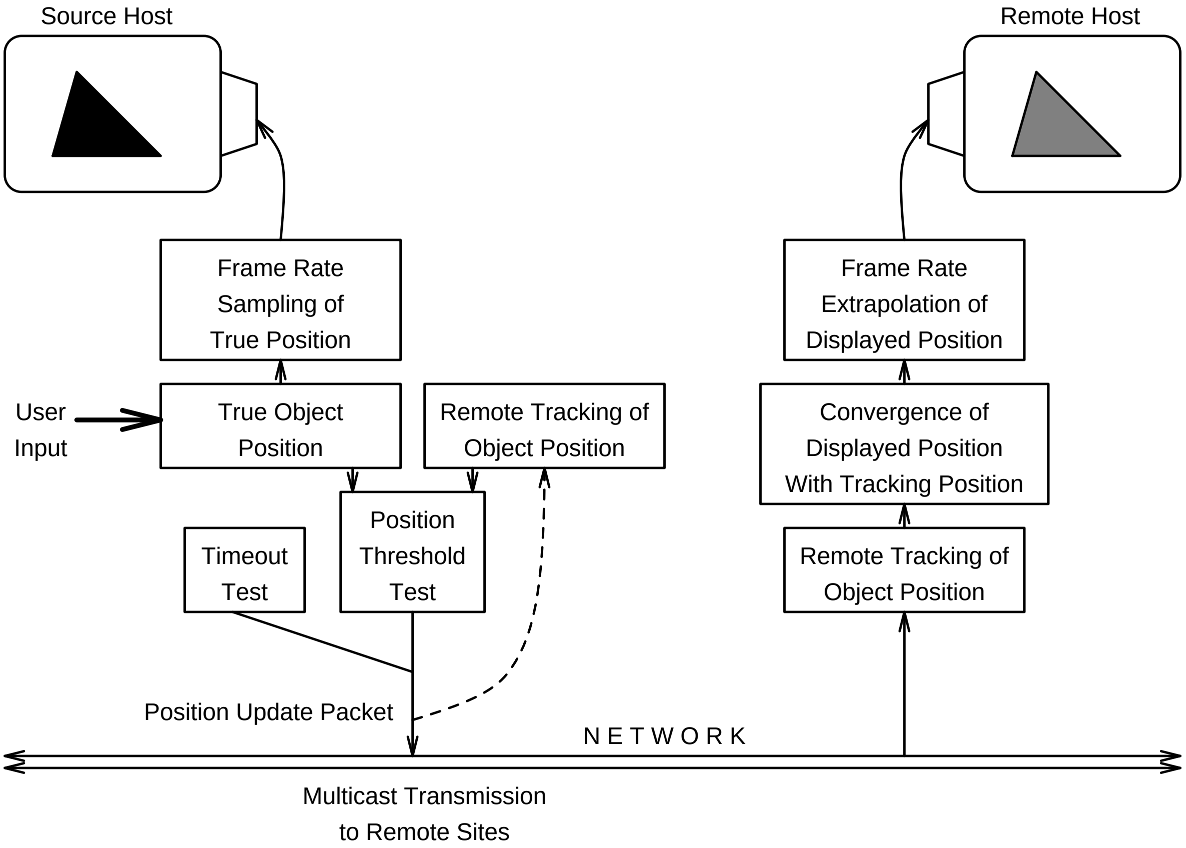


Figure 2: Sites Process Position Updates from Remote Objects

of object behavior, we show that the history-based technique yields visually accurate and smooth representations of object position and behavior. We conclude by discussing how this technique applies to general problems of rendering dynamic objects in distributed simulation and virtual reality applications.

2 Position History-Based Protocol and Algorithms

The position history-based protocol allows hosts to generate a visually accurate animation of remote objects in spite of typical network delays. As shown in Figure 2, a source host periodically multicasts the current object position to remote sites across the network. During the *tracking step*, each remote host uses a short history of these updates to estimate the object's position, velocity, and acceleration until the next update arrives. The *convergence step* calculates a velocity and acceleration that correct the object's current displayed position to smoothly converge with the tracked position. Finally, the *extrapolation step* samples the convergence path at the local frame rate to produce an animation on the display.

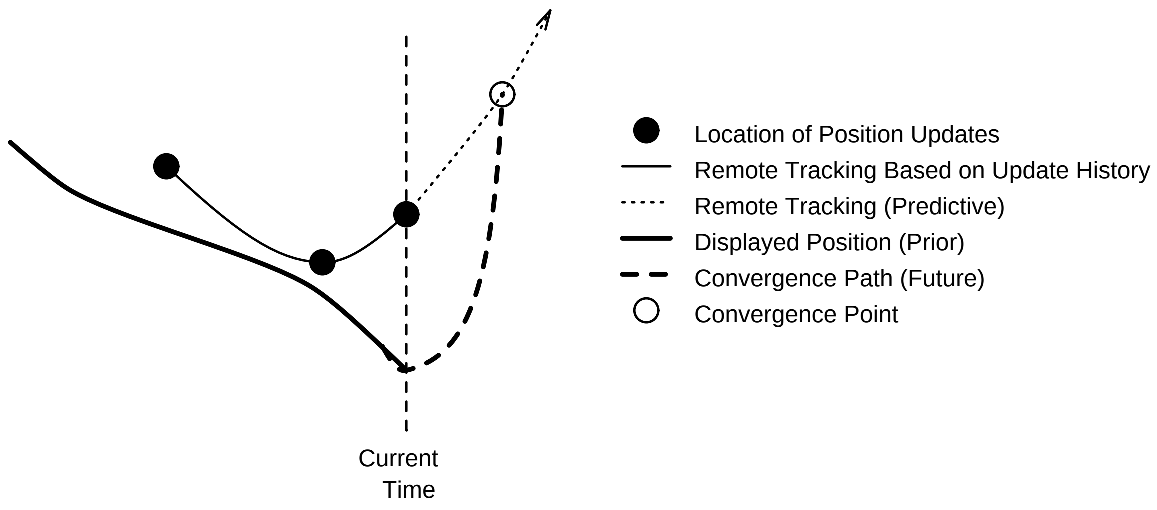


Figure 3: Tracking and Convergence Steps at Remote Sites

2.1 Source Generation of Updates

The source host transmits an update packet whenever it determines that the remote tracking of an object differs significantly from the object's true position. The host concurrently maintains two models of each object: the true position is determined indirectly from user input; the remote tracking position (described in the next section) estimates the position based on previous network updates. By calculating the difference between the true position and the remote tracking position, the source host therefore approximates the dead reckoning error at remote hosts. A maximum error threshold is associated with each object. The host transmits an update packet for an object whenever the distance between the true position and the remote tracking position exceeds this threshold. To prevent remote hosts from relying on old information when a packet is lost, the source host transmits an update if none is generated within a timeout period

An update packet only reports the object's current position (along each axis) and orientation (using Euler angles or quaternions). Remote sites use this "absolute state" information to track the object's position, velocity, acceleration, orientation, angular velocity, and angular acceleration. Each packet includes a timestamp which is used by receivers to account for transmission latency.

2.2 Receiver Processing of Updates

Upon receiving an update packet, a host updates its tracking of the remote object and converges the displayed position to the tracked position (Figure 3):

- **Tracking Step:** The host predicts the object location until the next update arrives, compensating for position updates not arriving at frame-rate frequencies. The dotted line in Figure 3 indicates the predicted object path.
- **Convergence Step:** The animated object smoothly converges with the tracked position at the *Convergence Point*, as shown by the dashed line in Figure 3. Because the host estimates

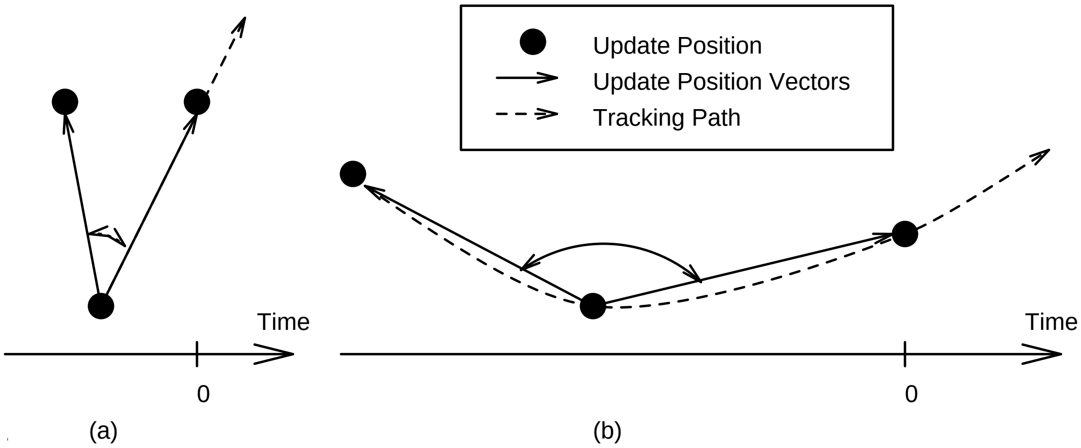


Figure 4: Local Angle of Embrace Determines Adaptive Tracking

the object's future location, the current displayed position is somewhat inaccurate relative to the actual position provided in the update packet. Rather than directly "jumping" to the correct position, smooth convergence provides users with a seamless view of remote objects.

2.2.1 Adaptive Tracking Algorithm

The tracking algorithm adapts to the object's behavior by using either a second-order (parabolic) estimation with the three most recent position updates or a first-order (linear) estimation with the two most recent position updates (Figure 4). To determine which of the two tracking techniques to use, the remote host calculates the angle between the three most recent update positions. This angle, defined in differential geometry as the *angle of embrace*, estimates the local Gaussian curvature of the object's path [5]. A small angle—implying a sharp turn—indicates first-order tracking (Figure 4a); a larger angle—implying smooth motion—indicates second-order tracking (Figure 4b).

The second-order tracking technique is used when the object is moving smoothly. If the three most recent positions are x_0 at time 0, x_{-1} at time $(-\delta_{-1})$, and x_{-2} at time $(-\delta_{-1} - \delta_{-2})$, then the remote tracking at time 0 has initial position $x(\tau)\Big|_{\tau=0} = x_0$, initial velocity

$$x'(\tau)\Big|_{\tau=0} = \frac{\delta_{-1}}{(\delta_{-1}+\delta_{-2})\delta_{-2}}x_{-2} - \left(\frac{1}{\delta_{-1}} + \frac{1}{\delta_{-2}}\right)x_{-1} + \left(\frac{1}{\delta_{-1}} + \frac{1}{\delta_{-1}+\delta_{-2}}\right)x_0$$

and acceleration of

$$x''(\tau) = \frac{2}{(\delta_{-1}+\delta_{-2})\delta_{-2}}x_{-2} - \frac{2}{\delta_{-1}\delta_{-2}}x_{-1} + \frac{2}{\delta_{-1}(\delta_{-1}+\delta_{-2})}x_0$$

By common subexpression elimination, these parameters are evaluated with ten multiplications and six additions. The displayed position converges with the tracked position in $\tau = \delta_{-1}$ seconds. The *convergence point* described in the next section is therefore

$$x(\tau)\Big|_{\tau=\delta_{-1}} = \frac{2\delta_{-1}^2}{\delta_{-2}(\delta_{-1}+\delta_{-2})}x_{-2} - \left(\frac{2\delta_{-1}}{\delta_{-2}} + 1\right)x_{-1} + \left(\frac{2\delta_{-1}}{\delta_{-1}+\delta_{-2}} + 1\right)x_0$$

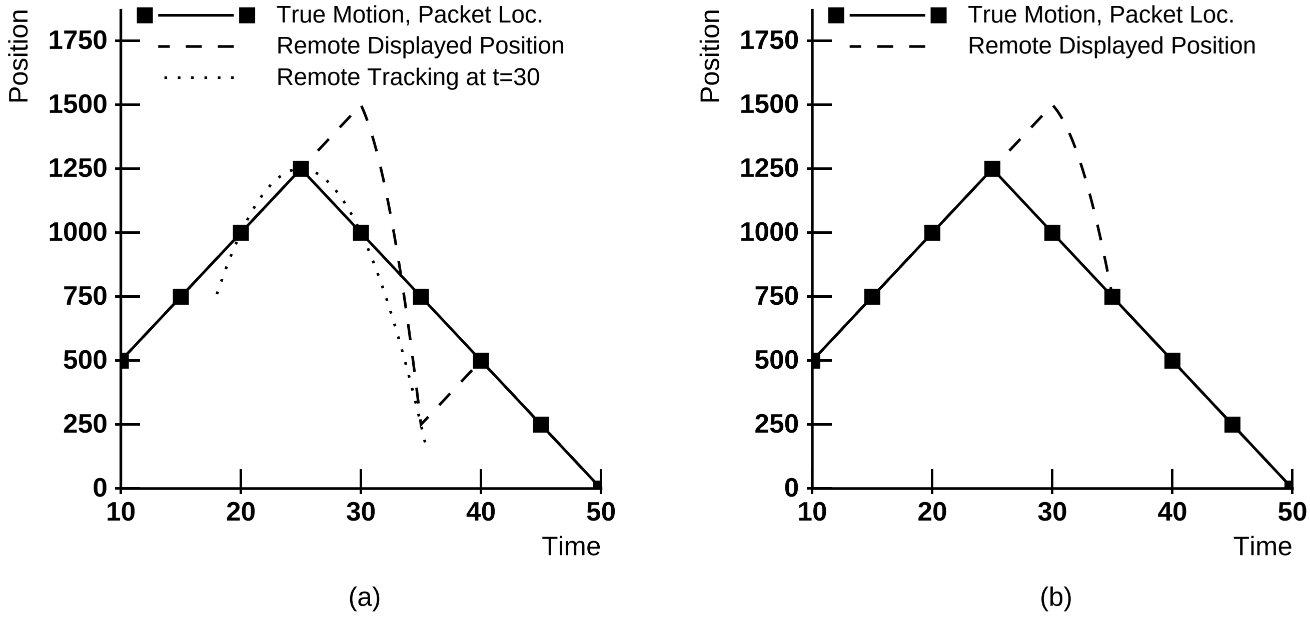


Figure 5: Using Adaptive Tracking to Model Sudden Path Changes

The chosen convergence period is long enough to eliminate jerky path corrections, yet it is short enough so that path correction is complete when the next update packet is expected to arrive.

The first-order tracking technique is used when the object makes a sharp change in direction, for example as a result of a collision. In this situation, a second-order technique as described above is inaccurate. The dotted line in Figure 5a shows that a second-order curve over-compensates for the turn and introduces new error in the remote tracking; the resulting displayed path, represented by the dashed line, does not reflect the object's true behavior. A first-order approach, on the other hand, offers better results by ignoring information from before the sharp turn (Figure 5b). The resulting displayed path converges to the true position more rapidly, as reflected in the dashed line. Using first-order tracking, the remote tracking at time 0 has initial position $x(\tau)\Big|_{\tau=0} = x_0$, and velocity $x'(\tau) = \frac{1}{\delta_{-1}}x_{-1} + \frac{1}{\delta_{-2}}x_0$. By common subexpression elimination, these parameters are evaluated with one multiplication and two additions. The displayed position converges with the tracked position in $\tau = \min(\delta_{-1}, 0.25)$ seconds, meaning that the convergence point is

$$x(\tau)\Bigg|_{\tau=\delta_{-1}} = -x_{-1} + 2x_0$$

or

$$x(\tau)\Bigg|_{\tau=0.25} = -\frac{0.25}{\delta_{-1}}x_{-1} + \left(1 + \frac{0.25}{\delta_{-1}}\right)x_0$$

Each receiver accounts for packet delay by "backdating" the position update to its transmission time provided by the packet timestamp. Although hosts receive the packet with different latencies, all remote sites consequently track the object in nearly the same manner. The source host therefore can accurately measure error in the remote tracking position.

Angle Between Three Recent Updates	Curve Characterization	Tracking Model	Convergence Model
Small	Sharp Turn	First-Order	Second-Order
Medium	Smooth Curve	Second-Order	Second-Order
Large	Straight Line	Second-Order	First-Order

Table 1: Adaptive Modeling Technqiues for Extrapolation and Convergence

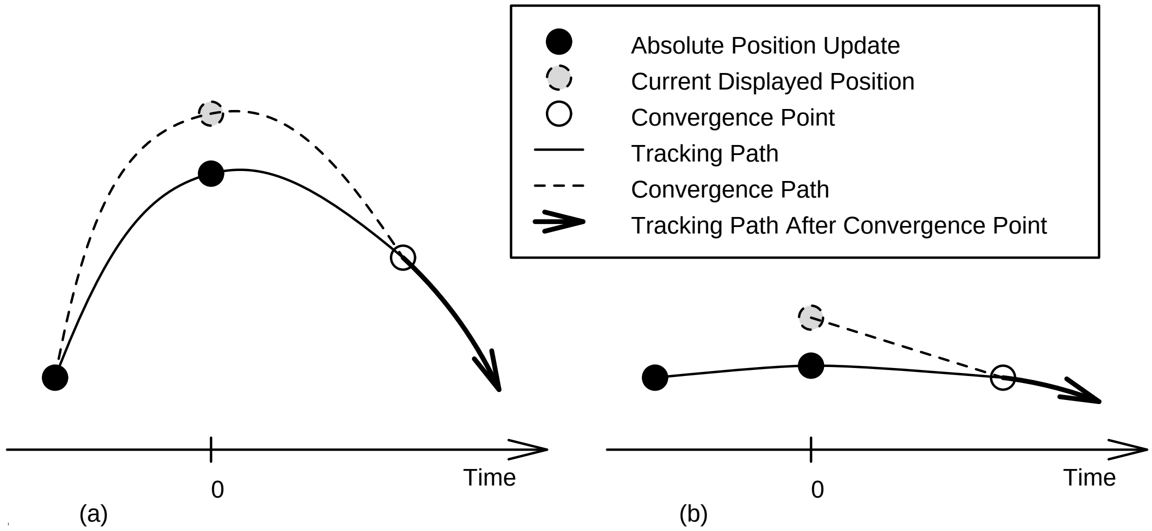


Figure 6: Convergence Smoothly Corrects the Current Displayed Position

2.2.2 Adaptive Convergence Algorithm

The convergence algorithm adapts to the object's behavior by producing either a second-order (parabolic) path of constant acceleration or a first-order (linear) path of constant velocity. The angle of embrace calculated during the tracking algorithm determines which path is used. A large angle, indicating almost linear motion, requires first-order convergence, and a smaller angle, indicating curved motion, requires second-order convergence. Table 1 summarizes how angle of embrace determines the adaptive tracking and convergence algorithms.

Second-order convergence generates a smooth curve between the object's previous absolute position, current displayed position, and the *convergence point* on the tracked path (Figure 6a). If the previous absolute position is x_{-1} at time $-\delta_{-1}$, the current displayed position is $\overline{x_0}$ at time 0, and the convergence point (determined in the previous section) is $\overline{x_1}$ at time δ_1, then convergence has initial position $\left.\overline{x(\tau)}\right|_{\tau=0} = \overline{x_0}$, initial velocity

$$\left.\overline{x'(\tau)}\right|_{\tau=0} = -\frac{\delta_1 x_{-1}}{\delta_{-1}(\delta_{-1}+\delta_1)} + \frac{(\delta_1 - \delta_{-1})\overline{x_0}}{\delta_{-1}\delta_1} + \frac{\delta_{-1}\overline{x_1}}{(\delta_{-1}+\delta_1)\delta_1}$$

Jerk Magnitude	Jerk Smoothness	Sample Motion	Algorithm(s) Providing Accurate Remote Display
Low	No Spikes	Line, Parabola	Tracking, Convergence
High	No Spikes	Oscillation	Tracking
Low	Spikes	Bouncing	Convergence
High	Spikes	Random	Neither

Table 2: Summary of Four Curve Classes

and acceleration

$$\overline{x''(\tau)} = \frac{2x_{-1}}{\delta_{-1}(\delta_{-1}+\delta_1)} - \frac{2\overline{x_0}}{\delta_{-1}\delta_1} + \frac{2\overline{x_1}}{(\delta_{-1}+\delta_1)\delta_1}$$

By common subexpression elimination, the parameters are computed in ten multiplications and six additions.

When the object is almost moving in a straight line, first-order convergence linearly joins the current displayed position to the convergence point as shown in Figure 6b. Using first-order convergence, the initial position is $\overline{x(\tau)}\Big|_{\tau=0} = \overline{x_0}$ and the velocity is

$$\overline{x'(\tau)} = \frac{\overline{x_1} - \overline{x_0}}{\delta_1}$$

These parameters are calculated in one multiplication and one addition.

The convergence process is complete once the object's displayed position reaches the convergence point. Until the next update arrives, the displayed object follows the position, velocity, and acceleration predicted by the remote tracking algorithm.

3 Position History-Based Protocol Accuracy

By using second-order curves to locally approximate smooth object motion, the protocol perfectly tracks any object exhibiting constant acceleration. Furthermore, the algorithm is highly accurate for smooth curves whose acceleration changes slowly because those paths locally exhibit near-parabolic behavior. For these curves, the remote error at each position update is relatively small, and the remote tracking easily corrects the error by perturbing the acceleration applied between position updates.

To evaluate the algorithm for rapidly changing acceleration—that is, non-zero jerk—we characterize object motion into three classes (summarized in Table 2):

- **High jerk with no spikes:** For example, a bus traveling along a winding road exhibits smooth jerk.
- **Low jerk with occasional spikes:** Jerk may spike when an external force is temporarily applied to the object. For example, a bouncing object exhibits near-zero jerk except at the ground, where the jerk spikes, changing the velocity. In this case, the convergence algorithm must quickly recover from errors resulting from the unpredicted collision with the ground.[2]

[2] We assume here that the object is tracked in isolation, without advance knowledge of the impending collision.

- **High jerk with frequent spikes:** This case includes all remaining types of motion, including random motion. For example, a person in a crowd moves around smoothly at varying velocity but is occasionally pushed by someone else. Tracking and convergence algorithms are of little benefit in predicting or correcting the remote object display.

Though an object's motion is generally complex, its behavior can at least locally be described in one of these cases.

We make the following observations:

Level-of-Detail Tradeoff Observation A lower protocol threshold provides greater level of detail but requires higher network bandwidth. The balance of this tradeoff depends on the particular curve class being modeled.

Behavioral Accuracy Observation A higher protocol threshold retains behavioral accuracy despite reducing the positional level-of-detail in remote tracking.

Threshold Estimation Observation Setting the protocol threshold to double the average tolerable visualization error provides a reasonable balance of visualization error and network bandwidth for most curves.

3.1 High Jerk With No Spikes

If the jerk is smooth, the object's acceleration changes uniformly. When a remote site uses the second-order tracking algorithm between position updates, jerk in the actual object motion introduces a minimum error of $\left(\frac{j}{6}t^3 + \frac{j'}{24}t^4\right)$ where j is the initial jerk and j' is the change in jerk per unit time. Jerk is mostly smooth, so j' is small and the first term dominates the expression. Because time between update packets is typically on the order of one second, the error is roughly proportional to jerk. The position history used by remote tracking offers a fair estimate of the future position because a smooth jerk cannot generate significant position error within a short time interval.

3.1.1 Oscillatory Motion

Oscillation is a common example of this class of behavior. At tight protocol thresholds, the position history-based protocol accurately tracks high-speed oscillatory motion (Figure 7a) by transmitting update packets soon after the object's behavior changes. A tight threshold requires higher packet rates of over five per second. By increasing the protocol threshold, the sender allows remote sites to "overshoot" the oscillation amplitude (Figures 7b-c) and consequently reduces the packet rate to less than one per second. Despite the lower positional level-of-detail, we observe that the remote tracking still exhibits oscillatory motion, maintaining behavioral accuracy.

3.1.2 Circular Motion

Circular or spherical motion results when jerk smoothly changes in direction rather than in magnitude. Analysis of circular motion has broader applicability, however, for a large family of curves can be locally approximated as circles [15]. Figure 8a shows the average remote tracking error as a function of the position threshold. The graph shows that average error is substantially lower

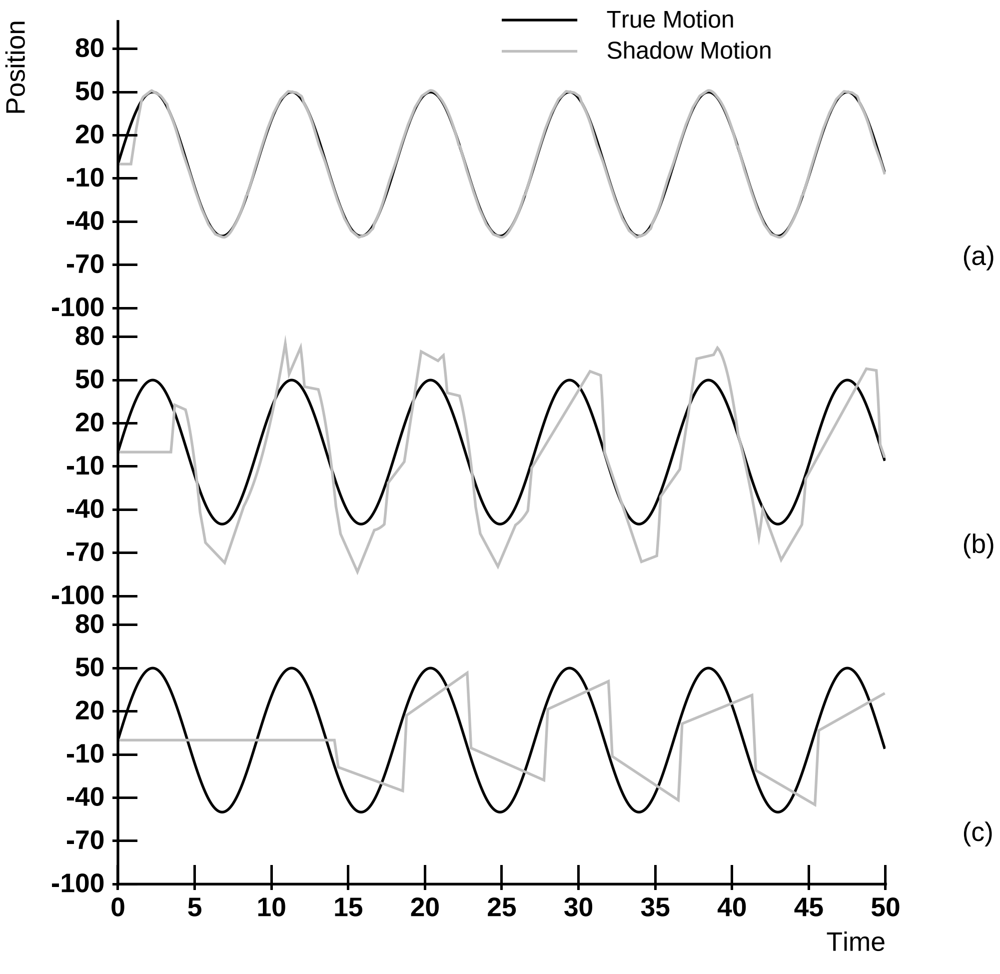


Figure 7: History-Based Modeling of Sinusoidal Oscillation (amplitude 50 meters; period 9 seconds; 5 second timeouts; threshold (a) 1, (b) 25, and (c) 50 meters)

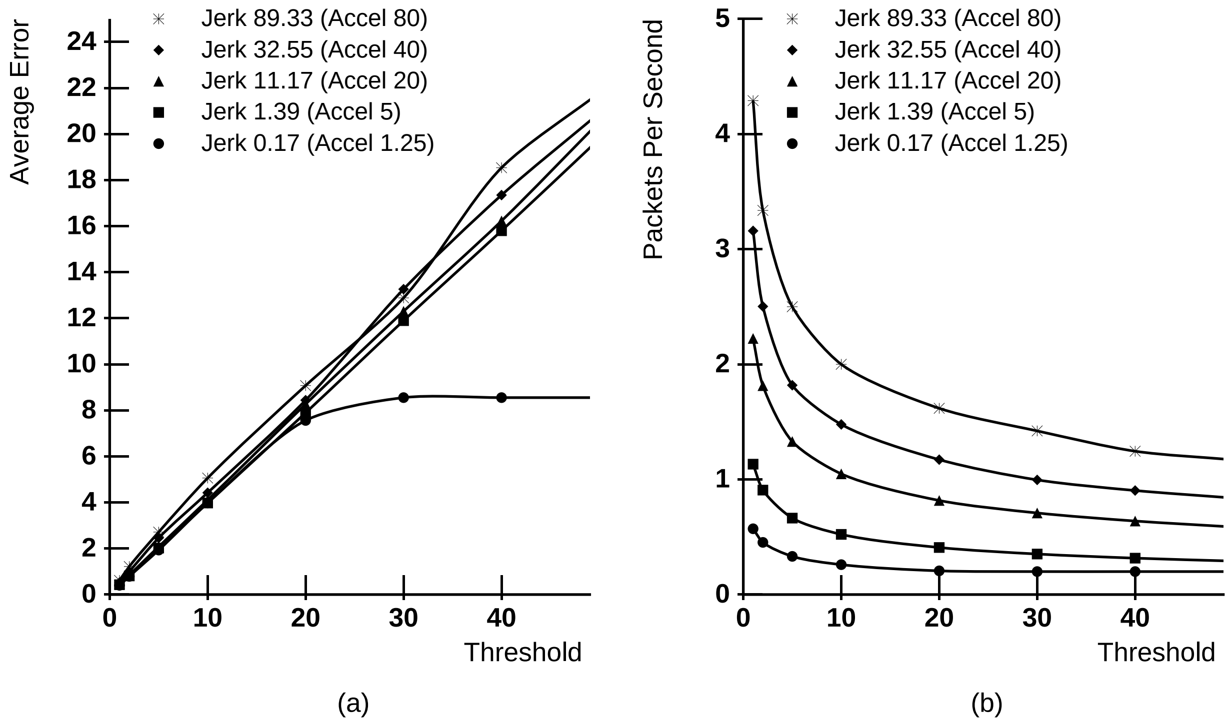


Figure 8: Average History-Based Protocol (a) Error and (b) Packet Rate for Circular Motion

than the protocol tolerance. Furthermore, the average error rises linearly with increasing protocol threshold, and the packet rate rises exponentially with tighter threshold (Figure 8b). When considering the Level-of-Detail Tradeoff, we observe that protocol threshold affects bandwidth usage more severely than average error. If all other factors are equal, the protocol threshold should therefore be as high as possible to minimize bandwidth requirements, as long as the minimum required level of visualization detail is provided. The Threshold Estimation Observation, which sets the threshold to twice the average visual tolerance, balances visual error requirements with the need to manage data traffic.

3.2 Low Jerk With Occasional Spikes

We now consider the situation in which the overall jerk is near zero but exhibits occasional "spikes" of high magnitude. In such motion, the acceleration remains fairly constant for some time but sometimes may change suddenly. For example, when two objects collide, they both instantaneously exhibit high acceleration as momentum is reversed. After the resulting almost instantaneous velocity change, the acceleration returns to a stable state.

A bouncing object provides an example of this class of behavior (Figure 9). The jerk is zero as the object is moving, but it exhibits a positive spike as the object changes direction and a negative spike as the object returns to the original acceleration. The corresponding acceleration remains constant as the object is moving, but it spikes as the object bounces and velocity reverses direction.

Remote sites cannot use prior position information to predict a sudden spike in jerk such as occurs during collisions; the tracking and convergence steps must instead quickly recover from the error after the change is reported. Because the acceleration changes instantaneously at the collision

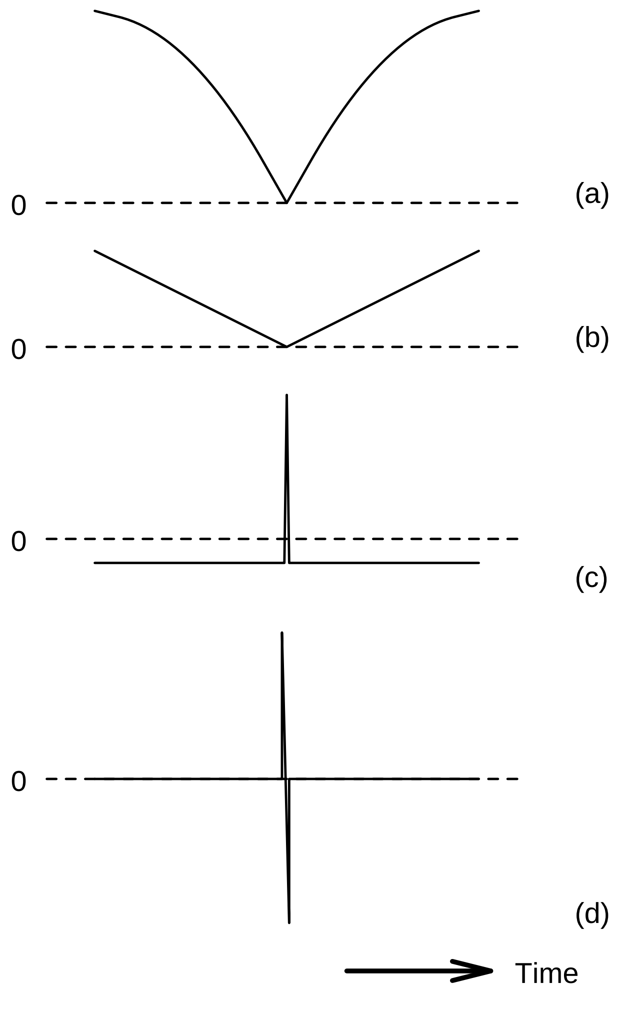


Figure 9: Object Collision (a) Position, (b) Velocity, (c) Acceleration, and (d) Jerk Over Time

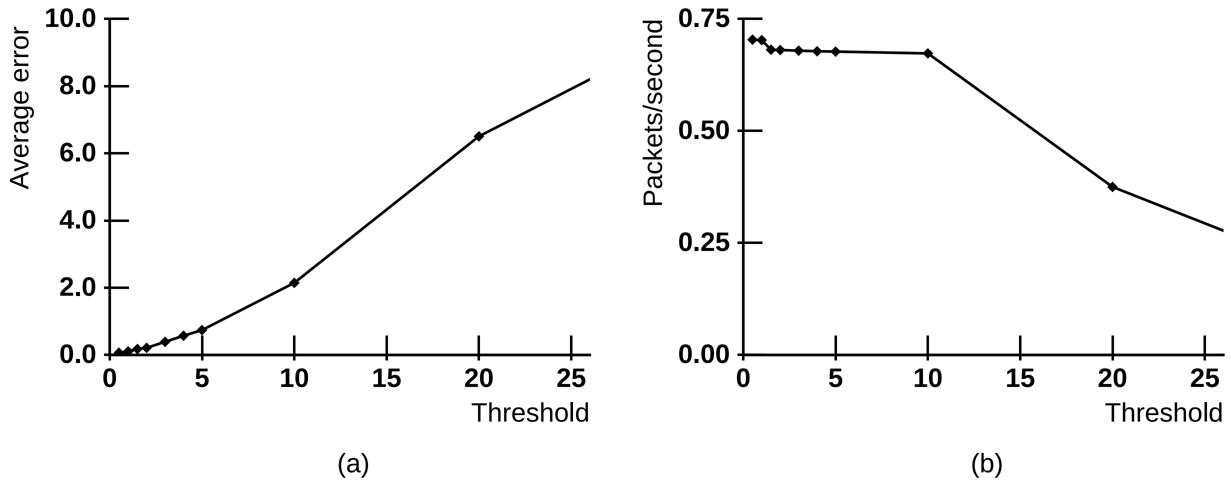


Figure 10: Average History-Based Protocol (a) Error and (b) Packet Rate for Bounce Motion

point, the error in the remote tracking increases quadratically after the collision until it exceeds the protocol threshold. In particular, suppose that the object acceleration and velocity respectively change by a_δ and v_δ during the collision (here, a_δ and v_δ are calculated from the instantaneous acceleration and velocity before and after the collision). The remote tracking error at time τ after the collision is $\left|\frac{1}{2}a_\delta\tau^2 + v_\delta\tau\right|$. Setting this expression equal to T, the protocol error threshold, yields the expected delay before remote tracking learns of the acceleration change. For large v_δ, the delay is less than $\frac{T}{v_\delta}$. For large a_δ, the delay is less than $\sqrt{\frac{2T}{a_\delta}}$. For roughly equal a_δ and v_δ, the delay is on the order of $\left(\frac{T}{v_\delta} - \frac{T^2}{2a_\delta^2}\right)$.

While network utilization is the sensitive resource in Level-of-Detail Tradeoff for smooth motion, average error is more critical when considering bouncing motion. Figure 10a shows that the average tracking error rises almost quadratically with increased protocol threshold. On the other hand, the number of packets transmitted over time is less dependent on the protocol threshold (Figure 10b). Because error after a sudden acceleration change increases quadratically over time, the remote tracking always eventually exceeds the protocol tolerance and requires an update. Figure 10 therefore indicates that the protocol threshold affects average error more severely than network bandwidth. If all other factors are equal, therefore, we desire to keep the threshold as tight as possible (to provide the most accurate remote visualization), without exceeding the available network bandwidth. Based on the data in Figure 10, the Threshold Estimation Observation appears to offer a reasonable starting point for achieving this balance.

Figure 11 shows the protocol's visual performance on bounce motion. As long as the acceleration is constant, the remote tracking is exact. When the acceleration suddenly changes, the remote tracking continues along the old path until it violates the protocol threshold. After the host receives an update packet, convergence spreads the acceleration change over a time interval to smoothly correct the displayed position.

Remote tracking of bounce motion demonstrates the Behavioral Accuracy Observation. Figure 11 shows that the tracking behavior remains fairly stable as the protocol threshold increases. For tight thresholds, the remote behavior follows the actual path accurately (Figure 11a). Although the object is bouncing at a high frequency, models with wide thresholds may display a lower fre-

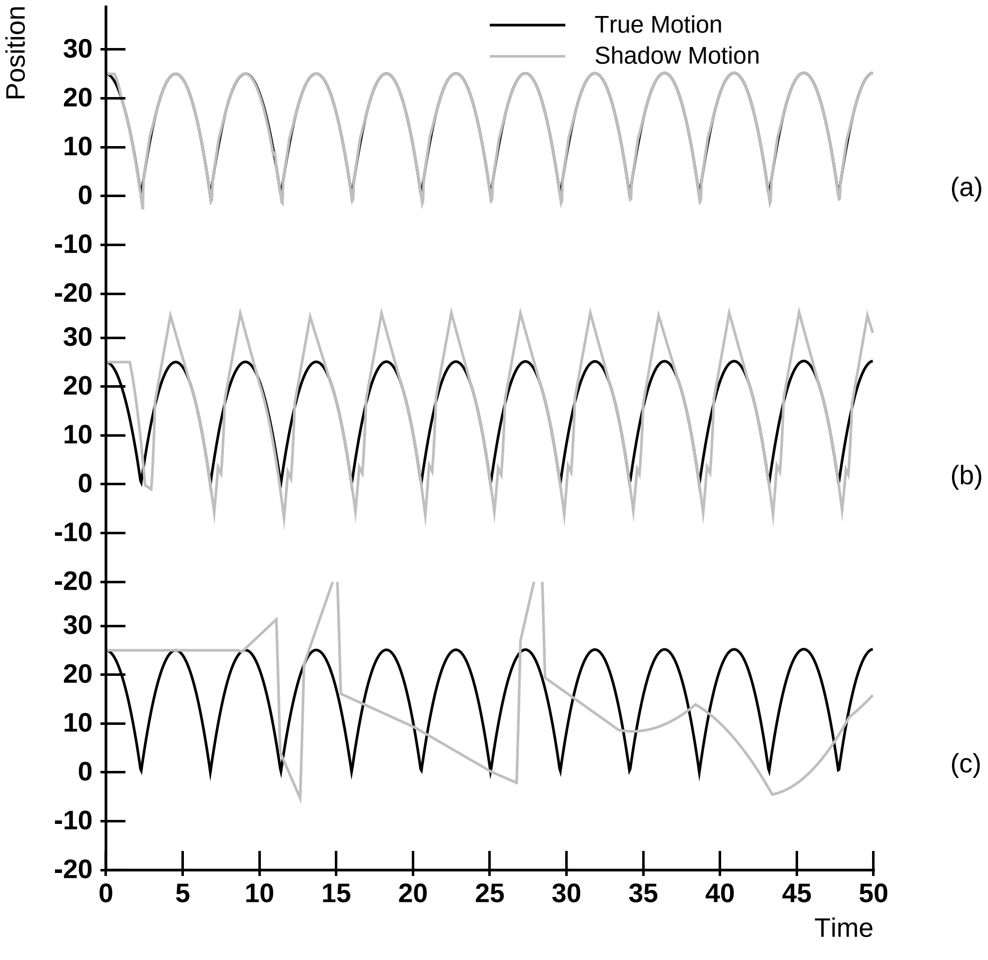


Figure 11: History-Based Modeling of Bounce Motion (height 25 meters; timeout 5 seconds; threshold (a) 1, (b) 10, and (c) 25 meters)

quency bounce of varying height, as shown in Figure 11c. Despite providing a lower positional level-of-detail because of the higher protocol threshold, the remote model retains considerable behavioral accuracy. This type of tracking is acceptable to a low-fidelity observer who is less interested in the local perturbations of the object but does need to observe the object's overall behavior.

3.3 High Jerk With Frequent Spikes

This type of curve exhibits both smooth, rapid changes and sharp, unpredictable changes. Such behavior includes most random motion, such as a person weaving through a crowd or a particle travelling through a wind tunnel. Because the curve behavior changes so quickly, remote modeling of these curves is error prone. The tracking step cannot accurately predict the future position, and the convergence step does not recover from a display error before the object behavior changes again.

Despite the complexity of this class of curves, the position history-based protocol provides good support for such erratic behavior. Figure 12 demonstrates the algorithm's behavior on a sample object path of this curve class. The curve trace was generated by randomly perturbing the acceleration smoothly over time, with large acceleration jumps occurring 5% of the time on average. For this and all other curves that we have studied from this class, the Threshold Estimation Observation provides a reasonable tradeoff between visual error and network utilization.

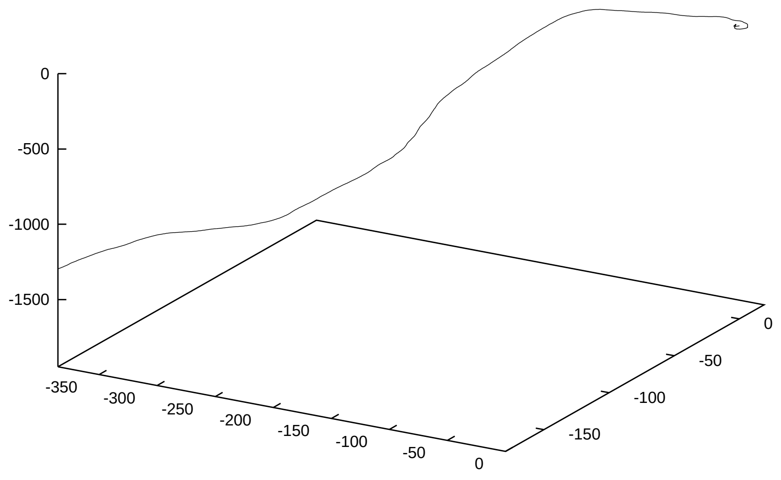
In summary, no remote modeling algorithm can expect to accurately support random motion. However, by sampling position periodically and smoothing between these samples, the position history-based protocol provides good behavioral accuracy with an acceptable positional accuracy.

4 Comparison to the DIS Protocol

Dead reckoning techniques are central to the large virtual environments targeted by the Distributed Interactive Simulation (DIS) protocols (IEEE Standard 1278). The current DIS protocol [7], [8] transmits position, velocity, and acceleration information whenever the remote object model exceeds a threshold or a five second timeout elapses. Using the most recent position, velocity, and acceleration information, DIS dead reckoning algorithms generate a second-order model to predict the future object location. We observe that the position history-based protocol performs at least as well as DIS for smooth object motion, and it potentially performs better than DIS for non-smooth object motion while requiring less network bandwidth.

At first, the DIS technique appears more effective than the history-based protocol for modeling a simple class of curves having smooth circular motion. Consider the path of an F-16 performing "Air Combat Maneuvering" (ACM) turning maneuvers as shown in Figures 13 and 14. The DIS protocol generates 25% lower error than the history-based protocol. Transmitting almost 25% fewer packets over the network, DIS would therefore appear to outperform the history-based technique for these types of curves.

However, the history-based protocol actually performs comparably to the DIS protocol in networked applications containing smoothly moving objects. In real applications, data updates are not delivered instantaneously by the network, and virtual reality systems must expect packet latencies of up to 250ms for cross-country traffic. The effectiveness of the DIS protocol is limited because the tracking relies on acceleration information. An object's acceleration can change more rapidly



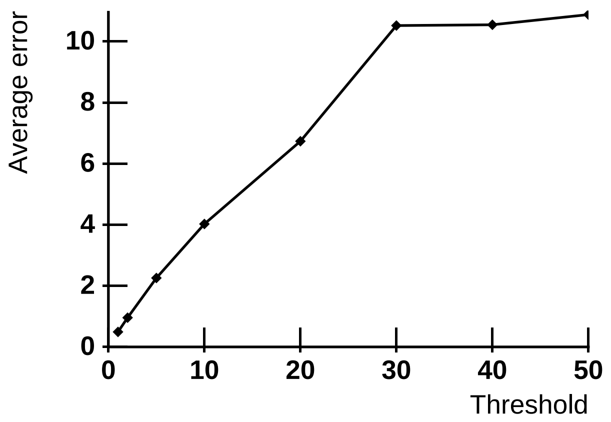


Figure 12: Average History-Based Protocol Error for Modeling Acceleration Changing Both Smoothly and Sharply

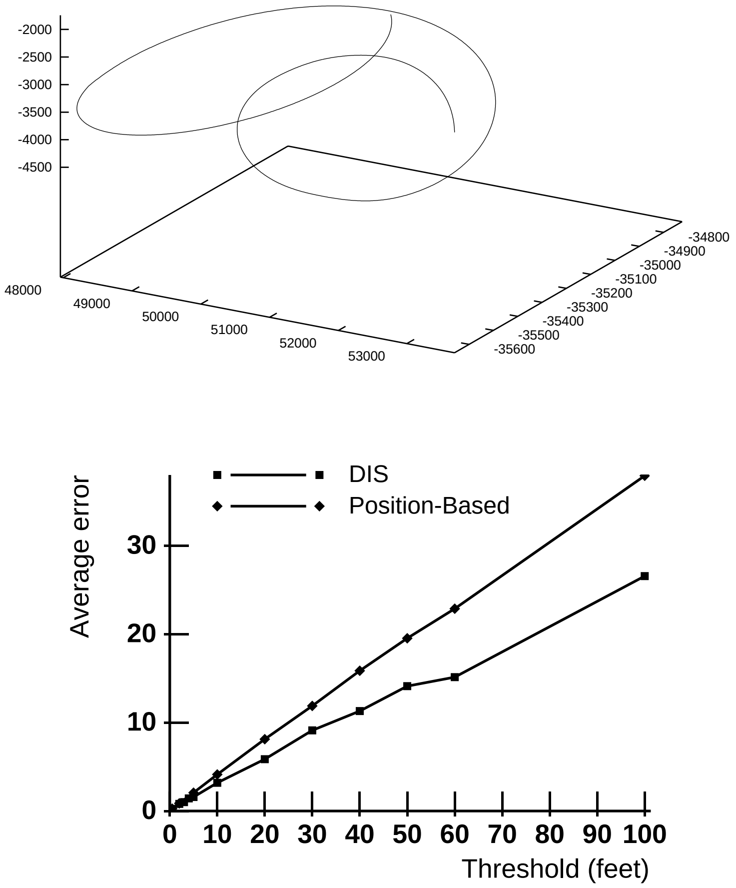
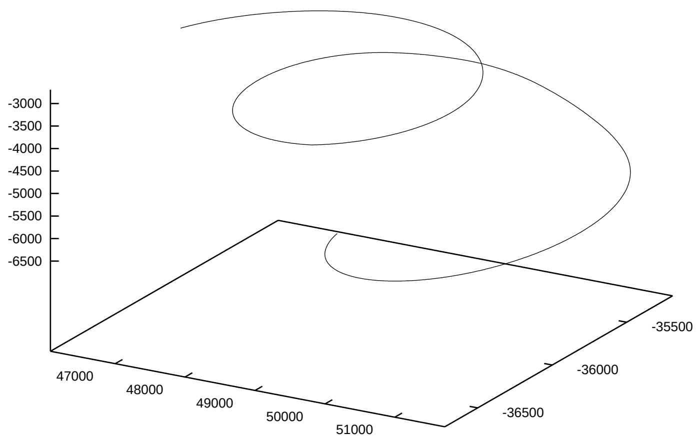


Figure 13: Modeling Error of F-16 Turning Maneuvers Using the DIS and History-Based Protocols



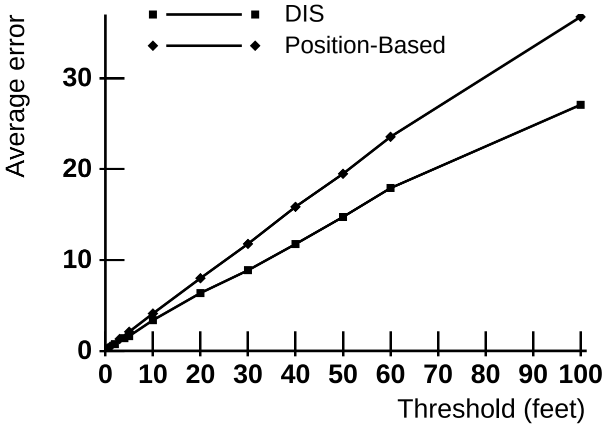
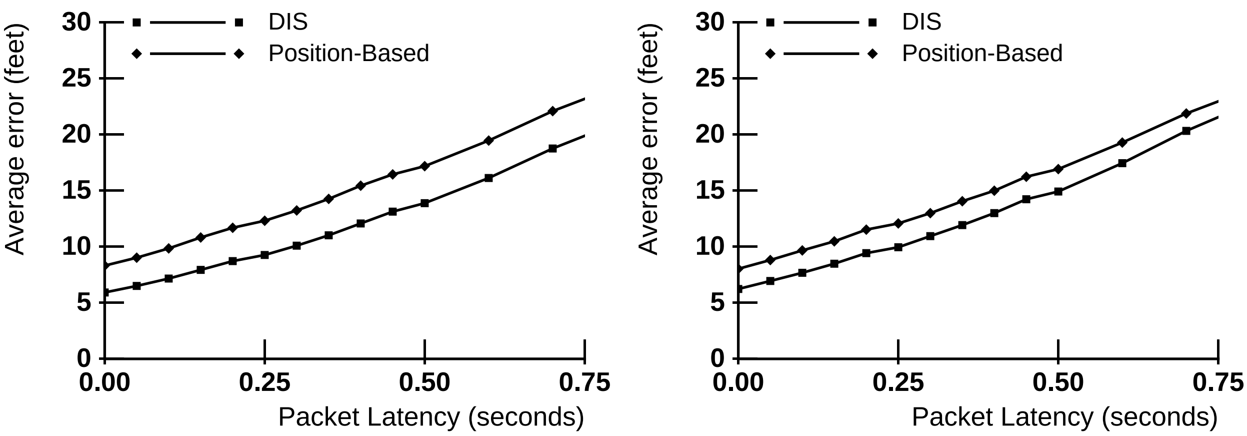


Figure 14: Second Example of Error for F-16 Turning Maneuvers Using the DIS and History-Based Protocols



(a) (b)

Figure 15: Effect of Packet Latency on Curves of (a) Figure 13 and (b) Figure 14

than its position, so if packets are delayed, then the DIS algorithm is likely to use out-of-date information to predict object behavior. Figure 15 demonstrates the effects of latency using the F-16 paths studied in Figures 13 and 14. In these tests, the sender's threshold level is set at 20 feet and the DIS algorithm is implemented using timestamp information to account for packet latency.[3] We observe that for realistic packet latencies, the average error in both algorithms rise comparably at their respective packet rates. This result indicates that both algorithms are generating very similar tracking models for circular motion in response to each update packet.

The position history-based protocol is furthermore potentially superior to the DIS protocol for tracking non-smooth object motion. The DIS protocol is highly sensitive to sudden acceleration changes because the algorithm utilizes only the most recent update information. Velocity and acceleration may change instantaneously by an order of magnitude or more. If an update is transmitted at the moment that the acceleration spikes, receivers use inaccurate information to track the object. This error becomes apparent, for example when modeling oscillatory motion (compare Figure 16 with Figure 7) and bouncing motion (compare Figure 17 with Figure 11). We observe that better results are obtained through a longer-term curve analysis as provided by the history-based protocol. This protocol relies on information—namely object position—that changes least rapidly and least randomly. In particular, a physical object's position must be continuous. Furthermore, position changes are generally an *indirect* response to physical or mechanical forces applied to the object. For example, position cannot change significantly until the velocity and acceleration have changed and acted for some time; position changes are thus delayed reactions to forces. By exploiting the relatively stable behavior of position, the history-based protocol offers better modeling than the DIS protocol relying on more transient attributes.

Superior performance on these non-smooth paths makes the history-based protocol more useful than DIS protocols in virtual reality applications and visual simulations. Accurate visualization is

[3] The current DIS dead reckoning protocol specification does not use timestamps to control the remote tracking step. Timestamps could easily be added to the DIS protocol, however, and this extension appears warranted. Without timestamp information, DIS dead reckoning error is higher by an order of magnitude.

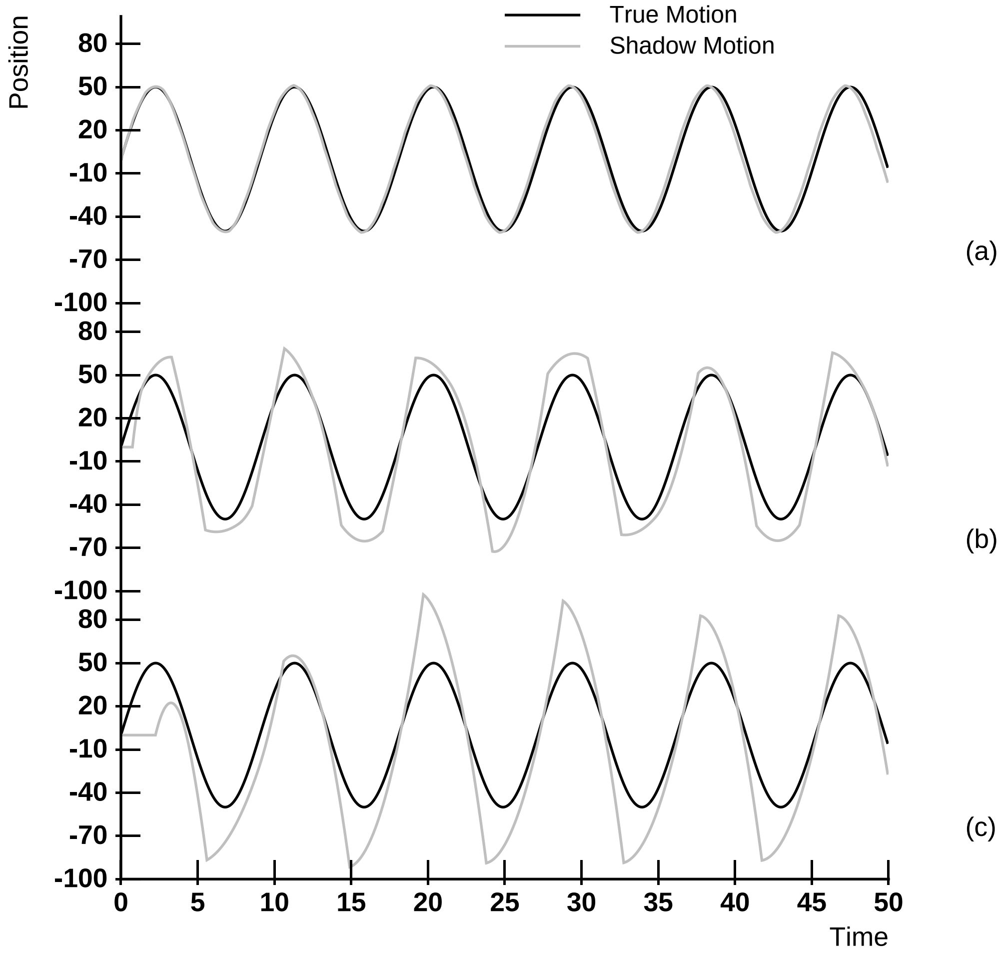


Figure 16: DIS modeling of Sinusoidal Oscillation (Amplitude 50 Meters; Period 9 Seconds; Timeout 5 Seconds; Threshold (a) 1, (b) 25, and (c) 50 Meters)

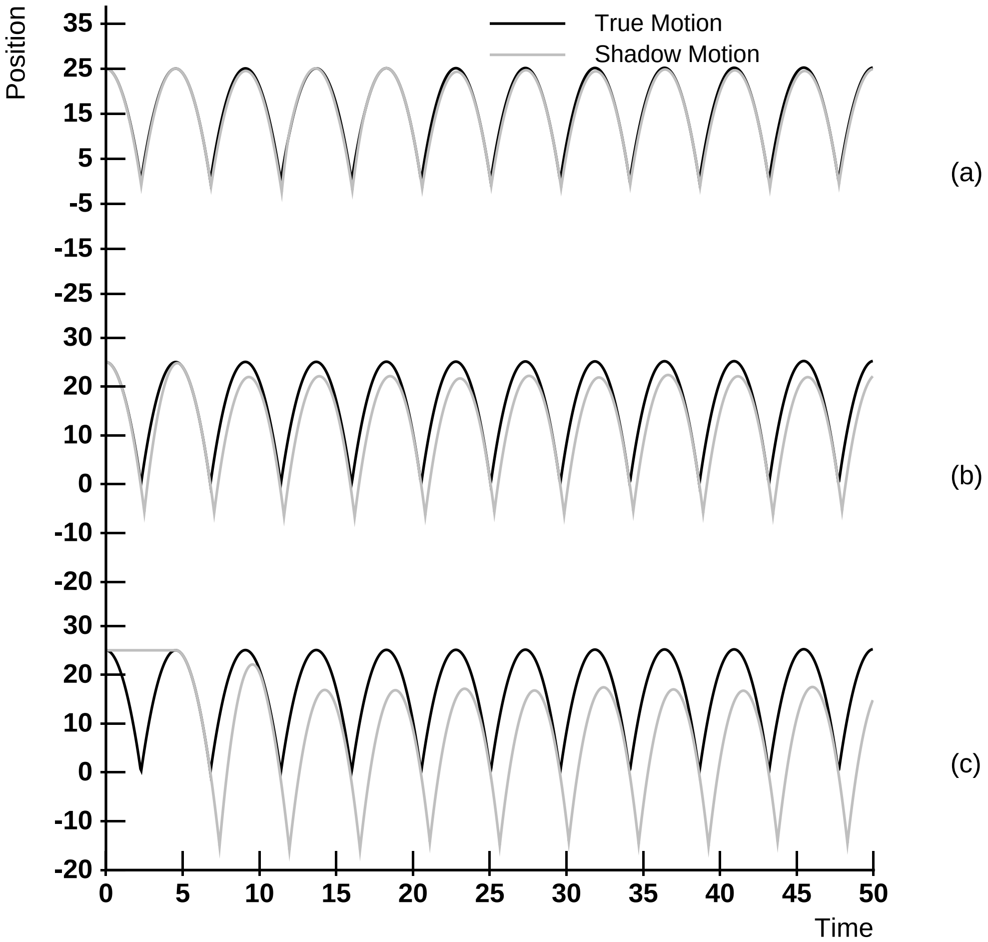


Figure 17: DIS Modeling of Bounce Motion (height 25 meters; timeout 5 seconds; threshold (a) 1, (b) 10, and (c) 25 meters)

most critical when the object is nearby in the virtual environment, but most objects do not exhibit smooth behavior when viewed at close range. For example, a car bounces slightly even as it travels over smooth roads. Similarly, a nearby adversary in a multi-player game most likely will move unpredictably to avoid being hit. Error is tolerable when the remote object is distant enough so that it appears to move smoothly, but when it comes closer, inaccuracy is unacceptable.

To transmit velocity and acceleration information, the DIS protocol also requires larger packets than the history-based protocol. By transmitting only position data in update packets, the history-based protocol can transmit over 1.75 times as often as DIS and still reduce bandwidth requirements. Taking this information into account when reconsidering the smooth F-16 paths, we observe that by transmitting 1.75 times as many update packets, the history-based protocol outperforms DIS because of the lower average tracking error.

5 Other Related Work

The generation of smooth curves and surfaces from discrete data has been an important issue within the computer graphics research community, but most of the prior research in this field cannot be applied to real-time distributed applications. A large body of graphics research concentrates on using quadratic and cubic Bezier splines for interpolating a set of points (consider [4], [6]). Emphasizing high-order continuity between piecewise splines, these techniques require that all sample points be provided in advance. These algorithms also attempt to generate smooth curvature, not allowing for sharp turns or random motion. An adaptive spline calculation technique [16] attempts to recursively subdivide the time period and locally apply linear, quadratic, or cubic splines within each interval. This technique requires that considerably more data samples are provided in advance. Non-linear optimization techniques, unsuitable for real-time use, have become popular for approximating curves, orientations, and surfaces (consider [1], [2], [12]). The history-based protocol obviates the need for complex algorithms of this sort by relying on short-term position history. This minimzation of parameters allows the protocol to derive extrapolation coefficients analytically and therefore quickly.

The idea of dead reckoning is similar to predictive encoding in control theory and signal processing, but most of this theory is also inapplicable to the remote object visualization problem. The most common signal processing techniques, Gaussian least-squares estimation [14] and Kalman filtering [10], [9], both assume Gaussian normal error in the sensor input. Distributed visualizations, however, receive accurate information from remote hosts. On the other hand, many control systems [11] utilize gradient descent methods which are unsuitable for use at frame-rate speeds. More importantly, both signal processing and control systems operate over a long-term series of samples, implicity assuming fairly stable data behavior. This assumption is unreasonable in simulation systems where object behavior may change quickly.

6 Conclusions and Future Work

Modeling of geometric position using periodic position updates and dead reckoning at receivers allows accurate, smooth visualizations of remote objects. The technique offers greater accuracy and better performance than both frame-rate protocols and competing algorithms; remote objects are visually indistinguishable from local objects. The protocol offers several notable features:

- Remote sites apply local computation on position history from multiple updates to accurately track position.

- Adaptive tracking and convergence characterize the object's overall behavior by using curve-fitting. The algorithm smoothly visualizes a broad range of object paths including straight lines, smooth curves, and sharp turns.

- Remote tracking uses timestamps to maintain a common view of the remote object at all sites despite network latency.

The history-based protocol, using several position updates instead of extra derivatives, more accurately tracks remote objects because it is less sensitive to sporadic changes in an object's state. This robustness is evident from the accurate remote modeling of oscillatory and bouncing motion. Protocols using instantaneous derivative data are more likely to see local velocity or acceleration spikes in response to external forces; such short-term fluctuations make these protocols unreliable. In addition, the history-based protocol requires less bandwidth than alternative algorithms because velocity and acceleration information are not transmitted.

Adaptive tracking techniques effectively model a variety of object behaviors. The protocol generates perfect representations for simple second-order (quadratic) curves, yet it also performs acceptably for complex motion characterized by unpredictable acceleration. Furthermore, adaptive extrapolation and convergence assure optimal performance for straight lines and sharp turns, two cases often neglected by other proposed algorithms.

Clock-time synchronization is more effective than frame-rate synchronization for addressing latency issues in real-time visualization systems. By using timestamp information to track remote objects, the position history-based protocol reduces real-time dependencies between hosts. The protocol can therefore withstand substantial network latency and different frame-rates between the source and receivers. This decoupling approach departs significantly from traditional multimedia and virtual reality systems which attempt to synchronize the senders and receivers by transmitting frame-by-frame data. Such systems rely on network bandwidth and latency guarantees to achieve acceptable performance.

The dead reckoning protocol appears applicable to a wider class of visualization problems. We are interested in applying this technique in several areas:

- Visualization of data from information services and large remote databases: For example, a local application program might visualize the performance of world-wide stock prices obtained from remote servers in real-time.

- Distributed CAD/CAM and architectural design: Sites transmit the location, size, color, etc. of components while remote sites depict the complete design.

- Factory automation and monitoring: Sites throughout the factory floor transmit performance information, sensor readings, and robot diagnostics while remote sites dynamically project factory output and maintenance requirements.

With increasing video resolution, network bandwidth, and processor speed, distributed virtual reality and visualization systems are becoming increasingly common in the entertainment industry and scientific community. However, these applications face numerous fundamental challenges. Position history-based dead reckoning stands to address many of these difficulties by providing a general framework for the real-time modeling of remote objects in distributed applications. Though considerable work remains to be done in this emerging area, we feel the technique offers a promising solution.

Acknowledgements

We thank Dan Schab of the Naval Air Warfare Center, Training Systems Division (NTSC) for providing the F-16 trace data studied in this paper. Hugh Holbrook assisted with the graphics rendering. The authors were respectively supported by a Fannie and John Hertz Foundation fellowship and by ARPA under contract DABT63-91-K-8001.

References

[1] Barr, Alan H., et al. "Smooth Interpolation of Orientation with Angular Velocity Constraints and Quaternions." SIGGRAPH '92 (Chicago, IL: 26–31 July 1992). In *Computer Graphics* 26.2 (July 1992): 313–320.

[2] Baart, M. L. and R. J. Y. McLeod. "Parabolic Curve Approximation in Design and Finite Element Applications." *Computer-Aided Design* 18.1 (Jan-Feb 1986): 29–32.

[3] Berglund, Eric J. and David R. Cheriton. "Amaze: A Multiplayer Computer Game." *IEEE Software* 2.1 (May 1985): 30–39.

[4] Cendes, Zoltan J. and Steven H. Wong. "C^1 Quadratic Interpolation Over Arbitrary Point Sets." *IEEE Computer Graphics and Applications* 7.11 (November 1987): 8–16.

[5] Colladine, C. R. "Gaussian Curvature and Shell Structures." In *The Mathematics of Surfaces.* Ed. J. A. Gregory. Oxford: Clarendon Press, 1986. 179–196.

[6] Hoschek, J. "Circular Splines." *Computer-Aided Design* 24.11 (November 1992): 611–618.

[7] Institute for Simulation and Training. *Standard for Information Technology—Protocols for Distributed Interactive Simulation Applications* [Version 2.0 Draft IEEE Standard]. University of Central Florida. Document IST–CR–93–15.

[8] Institute for Simulation and Training. "Dead Reckoning Definitions and Algorithms." *Enumeration and Bit Encoded Values for Use with Protocols for Distributed Interactive Simulation Applications.* University of Central Florida. Document IST–CR-93-02.

[9] Kailath, T. *Lectures on Wiener and Kalman Filtering.* New York: Springer-Verlag, 1981.

[10] Kalman, R. E. "A New Approach in Linear Filtering and Prediction Problems." *Transactions of the American Society of Mechanical Engineers [ASME], Journal of Basic Engineering* 82D (March 1960): 35–45.

[11] Mendel, Jerry M. *Discrete Techniques of Parameter Estimation: The Equation Error Formulation.* New York: Marcel Dekker, Inc., 1973. Chapters 4–6.

[12] Moreton, Henry P. and Carlo H. Sequin. "Functional Optimization for Fair Surface Design." SIGGRAPH '92 (Chicago, IL: 26–31 July 1992). In *Computer Graphics* 26.2 (July 1992): 167–176.

[13] Singhal, Sandeep K. and David R. Cheriton. "A Position History-Based Protocol for Remote Object Modeling in Distributed Applications." Unpublished paper.

[14] Sorenson, H. W. "Least-Squares Estimation: From Gauss to Kalman." *IEEE Spectrum* 7 (July 1970): 63–68.

[15] Van Wechel, R. "Analysis of Dead Reckoning Update Time." Eighth Workshop on Standards for the Interoperability of Defense Simulations (Orlando, FL: 22–26 March 1993).

[16] Waggenspack, Warren N. "Piecewise Parametric Approximations for Algebraic Curves." *Computer-Aided Geometric Design* 6.1 (January 1989): 33–53.