Hand Motion Prediction for Distributed Virtual Environments

Addison Chan, Rynson W. H. Lau, *Senior Member*, *IEEE*, and Lewis Li

Abstract—We use our hands to manipulate objects in our daily life. The hand is capable of accomplishing diverse tasks such as pointing, gripping, twisting, and tearing. However, there is not much work that considers using the hand as input in distributed virtual environments (DVEs), in particular, over the Internet. The main reasons are that the Internet suffers from high network latency, which affects interaction, and the hand has many degrees of freedom, which adds additional challenges to synchronizing the collaboration. In this paper, we propose a prediction method specifically designed for human hand motion to address the network latency problem in DVEs. Through a thorough analysis of finger motion, we have identified various finger motion constraints, and we propose a constraint-based motion prediction method for hand motion. To reduce the average prediction error under high network latency, for example, over the Internet, we further propose a revised dead-reckoning scheme here. Our performance results show that the proposed prediction method produces a lower prediction error than some popular methods, and the revised dead-reckoning scheme produces a lower average prediction error than the traditional dead-reckoning scheme, particularly at high network latency.

Index Terms—Motion prediction, hand motion prediction, hand interaction, network latency.

1 Introduction

As our hand is capable of accomplishing diverse tasks, there has been much work to make use of the hand for interaction [1], [2], [3]. In [4], we proposed techniques to allow multiple concurrent users to modify a design with their hands by wearing the CyberGlove, which is an electronic glove for capturing hand gestures. Due to the advance in computer technologies in the past decade, the performance bottleneck has shifted from local machine latency to network latency. A major challenge of the project is to overcome the high network latency of the Internet, which has significant impact on data synchronization and user interactivity. To address this limitation, we propose in this paper a prediction method specifically designed for modeling human hand motion in order to cope with the network latency problem.

We have conducted a thorough analysis of human finger motion and noticed that it has an elliptic motion behavior similar to our earlier observation on hand motion in manipulating a 2D mouse [5]. We have identified various finger motion constraints. By considering these motion characteristics and constraints, we propose a motion prediction method for agile hand motion. Note that our work differs from existing work on hand motion capture in that we focus on modeling the state transition density rather than the posterior density of states. In addition, the intuitive constraints applied do not require extensive training. Our results show that the new method is significantly more accurate than other popular methods in predicting hand motion. In addition, we also propose a revised dead-reckoning scheme, which sends more update messages only at times when we anticipate that the predictor may be less reliable. Our results show that the revised scheme has a lower average prediction error, particularly at high network latency.

The rest of the paper is organized as follows: Section 2 briefly reviews relevant work. Section 3 outlines our prediction method for hand motion. Section 4 presents our hand motion model. Section 5 discusses the static and the dynamic constraints of hand motion and how we integrate them into our hand motion model. Section 6 presents our revised dead-reckoning scheme. Section 7 demonstrates the performances of our prediction method and the revised dead-reckoning scheme through a number of experiments. Finally, Section 8 briefly concludes the work presented in this paper.

- *A. Chan and R.W.H. Lau are with the Department of Computer Science, University of Durham, South Road, Durham, UK. E-mail: {a.s.k.chan, rynson.lau}@dur.ac.uk.*
- *L. Li is with the Department of Computer Science, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon, Hong Kong. E-mail: kwfli@cs.cityu.edu.hk.*

Manuscript received 12 Sept. 2006; revised 1 Feb. 2007; accepted 20 Feb. 2007; published online 24 Apr. 2007.
Recommended for acceptance by C. Shaw.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0165-0906.
Digital Object Identifier no. 10.1109/TVCG.2007.1056.

2 Related Work

Although there has been a lot of work on motion prediction, work on human hand motion prediction is very limited. In this section, we first examine the network latency problem in distributed virtual environments (DVEs). We then survey prediction methods that are developed for various applications. Finally, we review existing work on human hand modeling and discuss the constraints of finger motion.

2.1 Network Latency in DVEs

Due to network latency, it is not possible to reflect changes immediately in all client machines of a DVE. Hence, different users may observe different states of the same object shared among them, resulting in inconsistency. This

affects system interactivity and, hence, user performance in performing a collaborative task.

Some solutions have been proposed to address the network latency problem. One method is to acknowledge the existence of network latency by explicitly showing the delay information to the users [6]. It is then up to the users to adjust their own actions to anticipate the delay. Such adjustment may be via a move-and-wait strategy [7]. The limitation of this approach is that user interaction becomes more difficult, particularly when the latency fluctuates. Another method is to defer updating an object change until the client with the highest latency has received the update [8]. This, however, further increases the response time. A more popular solution to network latency is dead reckoning [9], [10]. This method runs a motion predictor of the object in the host machine of the object and in all clients that access the object. The host only sends out update messages when the error between the predicted state and the actual state of the object is higher than a given threshold. In between updates, the clients approximate the object state with the local motion predictor. Thus, the performance of the dead-reckoning scheme depends on the accuracy of the predictor. When sharing objects—for example, avatars—in a DVE, dead reckoning not only alleviates the network latency problem, but also reduces the number of update messages needed to be sent through the network.

2.2 Motion Prediction

In dead reckoning, the polynomial predictor is often used to extrapolate/predict the future locations of an entity. Although polynomials of any orders may be used, second-order polynomials are the most popular. In [11], a hybrid approach is suggested. The first-order polynomial is used if the acceleration is either small or high; otherwise, the second-order polynomial is chosen. In [12], an EWMA scheme is used in a distributed virtual walkthrough. The model assigns different weights to past movement vectors, with higher weights to recent vectors.

There has been some work on predicting human body motion. Azuma and Bishop [13] propose a predictor for head motion based on the Kalman filter, which was originally used to filter measurement noise in linear systems by recursively minimizing the mean square estimation error. This method may work with the polynomial predictor or other prediction models to further reduce the prediction error. Results show that during rapid motion, 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 model to predict head motion [14] and human motion [15]. Wu and Ouhyoung [16] also attempt to predict head motion using a gray-system-theory-based predictor, whose accuracy is similar to that of the polynomial-based predictor with Kalman filtering.

Other than Kalman filtering, sequential Monte Carlo [17] is also used in motion tracking. Although this method can approximate arbitrary distribution functions without a unimodel Gaussian assumption, it is relatively expensive computationally. LaViola Jr. [18] suggests a double-exponential-smoothing-based predictor, which is efficient despite its slightly lower accuracy.

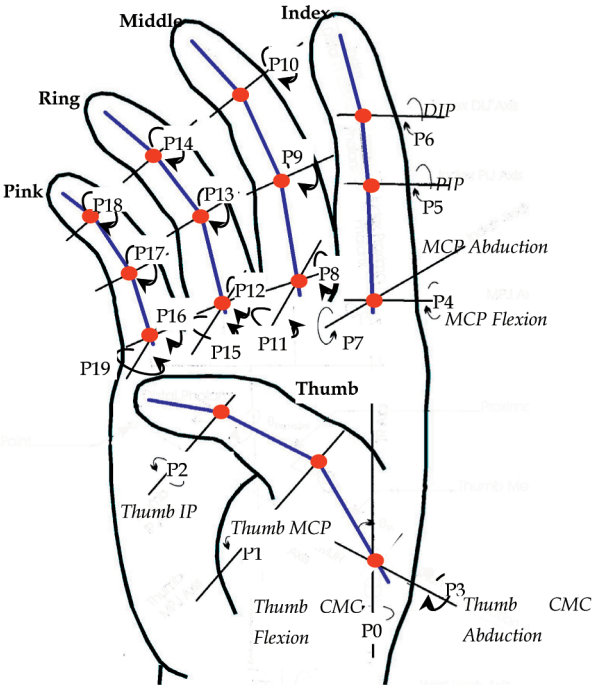

Fig. 1. The human hand kinematic model.

In general, specialized prediction methods produce more accurate results and are more capable of predicting further ahead in time even if the motion is vigorous. This is because they take into account the motion behavior of the target objects, which the polynomial predictor fails to do. The trade-off is the loss of generality. In this work, we consider the motion constraints of the human hand and propose a hand motion predictor based on an elliptical motion model to approximate the hand motion.

2.3 Human Hand Motion Modeling and Constraints

To model human hand motion, we need to define a kinematic model for the hand. Some hand models have been proposed for various applications, including hand animation and motion capture. We adopt a model similar to the one used in the CyberGlove [19], as shown in Fig. 1. This model resembles those used by the biomechanics specialists [20] but simplifies the complicated human hand biomechanical structure.

The adopted kinematic hand model focuses on the major components that dominate hand motion. In Fig. 1, each node represents a joint and the associated circular arrows indicate the rotation axes. We can see that although the distal interphalangeal (DIP) joints, the proximal interphalangeal (PIP) joints, the thumb interphalangeal (IP) joints, and the thumb metacarpophalangeal (MCP) joints have only 1 degree of freedom (DOF), others, such as the metacarpophalangeal joints (except the thumb), have two DOFs. The middle, ring, and pinky fingers have the same joint configuration as the index, although some of the information is omitted for clarity. The motion of these four fingers may affect each other. Although the thumb has similar motion characteristics, it moves independently from the other four fingers.

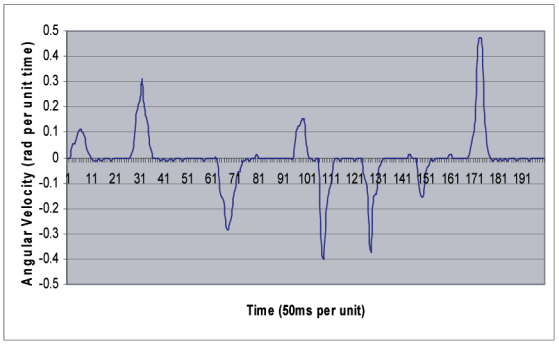

Fig. 2. Angular velocity of the index MCP flexion.

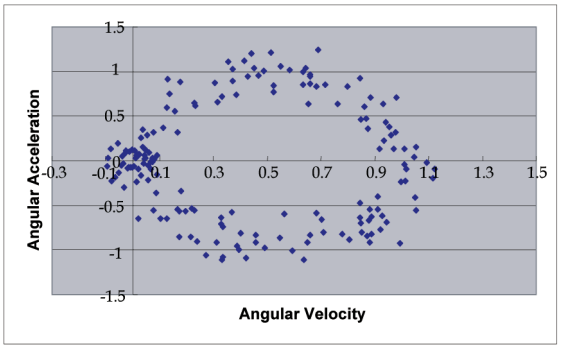

Fig. 3. Motion acceleration versus velocity.

The lines connecting joints or joints and finger tips represent bone segments. Finger motion is generated by contracting the muscles across bones. There are two main kinds of muscle driving the motion: *extrinsic* and *intrinsic muscles*. The extrinsic muscles spread over the forearm and the palm. Most of them are primarily responsible for finger flexions. A few of them are also responsible for thumb abduction. The intrinsic muscles only attach to the wrist and hand. They are responsible for finger flexions, abductions, and adductions. Note that flexion is to move the finger toward or away from the center of the palm, whereas abduction and adduction are to move two fingers away from and toward each other, respectively. Some of the hand motion characteristics are omitted from our hand model. For example, a finger rotates slightly about itself during abduction [21], and the shape of the palm may change slightly as the lower bones of the fingers (located at the palm region) are not fixed in position [22], for example, during grasping. Nevertheless, these characteristics do not play a significant role on hand motion, and it is impossible to include all physical constraints in the model as it will overcomplicate the model and increase the computational cost. Besides, the CyberGlove does not measure these kinds of movement.

In [23], physical constraints are used to model hand motion. The most obvious one is that any rotation of a joint has a range, which can be static (that is, fixed) or dynamic (that is, affected by neighboring joints). Dynamic constraints of finger motion may appear in many forms. For example, rotating PIP would cause the rotation of DIP [24]. Rotating a finger toward its palm through rotating the MCP reduces the range of abduction of the finger [22]. As pointed out by [23], the flexion of a finger depends on the flexion of its adjacent fingers (except the thumb). In [25], the principle component analysis (PCA) is used to identify the constraints for reducing the DOFs of the hand model. The DOFs can be further reduced by regarding the state space constructed by linear manifold spanning [26]. Although those in [25] and [26] are generic approaches for reducing DOFs in motion tracking, they involve extensive training, as each new user or motion style has a different configuration space. In addition, the correlation of some finger states is nonlinear, which is not considered by PCA. On the contrary, by using explicit physical constraints, we do not need a computationally expensive retraining process for switching users and motion styles. Only the limits of the motion parameters are needed, which can be computed efficiently.

3 Method Overview

The novelties of our work include the introduction of appropriate constraints suitable for efficient prediction without extensive training and a method to integrate these constraints into a motion prediction method for hand motion prediction. Our motion model is different from those used in motion tracking [25], [17], [26], which are primarily zero-mean Gaussian or polynomial-based. These methods work well in tracking motion, which does not need to handle a long prediction length, that is, predicting further ahead in time, or agile motion. Their focus is the posterior probability density of model parameters. In our situation, the latency of the Internet is typically high (0.3 second or more). Hence, we need to anticipate long prediction lengths and focus on state transition density or motion modeling [5], [11].

To develop our hand motion model, we have studied hand motion data. Fig. 2 shows a typical example of an index MCP flexion. As we flex the finger, a series of pulses are generated. (Similar motion patterns are also found in other fingers.) Each pulse represents a single movement step of the finger—the start of a pulse signals the start of the finger motion, whereas the end of it signals a motion pause. If we normalize the pulses and plot the acceleration against velocity, we may observe an elliptic pattern as shown in Fig. 3, which is similar to our earlier observation on manipulating a 2D mouse with the hand [5]. Hence, we apply the hybrid elliptic motion model [5] here for hand motion prediction. When a finger joint is at low-velocity motion, we use a linear model for prediction. When it picks up speed, we switch to an elliptic model for prediction. Details of our hand motion model are discussed in Section 4.

Our method reads the state of the glove regularly (100 Hz in our implementation), which is composed of 20 finger joint angles (the thumb: one for IP, one for MCP, and two for CMC; other fingers: one for DIP, one for PIP, and two for MCP) as shown in Fig. 1. Twenty Kalman filters are used to filter the measurement noise of the 20 input joint angles. When a new pulse is detected, the predictor will predict the joint angle at the end point of this pulse, which is then compared with the static and the dynamic

constraints of the joint. If the predicted joint angle exceeds the constraints (that is, the allowable range), we adjust the parameters (that is, the state vectors) of the relevant predictors in order to revise the predicted joint angle.

First, we consider the static constraints of each joint angle. If the predicted joint angle at the end of the pulse exceeds its static limits, the state vector of the corresponding predictor is adjusted so that the predicted joint angle will satisfy the static constraints. Details of this are discussed in Section 5.1. Second, we consider the dynamic constraints of the joint angles, which always involve the motion of two or more adjacent fingers. The MCP flexions of the index, middle, ring, and pinky are checked. (Since the motion of the thumb does not affect the motion of other fingers, it need not be checked here.) We compare the predicted angular differences of adjacent MCP flexion angles (index-middle, middle-ring, and ring-pinky) at the end of their pulses. If any pair of the fingers exceeds its corresponding dynamic limits, the state vectors of the two predictors are adjusted. If there is more than one pair of fingers violating the constraints, we prioritize them according to the time remaining from the current moment to the end of the pulse. The pair of fingers with the shortest remaining time is given the highest priority. The state vectors of the predictors for this pair will then be adjusted. The predicted angular differences of other MCP flexion angles are then checked again, and the state vectors of the predictors for any violated joint angles are adjusted in order to satisfy the dynamic constraints. Details of this are discussed in Section 5.2.

Third, we consider the dynamic constraints between MCP flexion angles and MCP abduction angles of adjacent fingers. Usually, flexion involves more energetic motion than abduction, and we adjust the state vectors of the predictors for abduction according to the predicted angles at the end of the MCP flexion pulses. However, if abduction is found to be more energetic, the state vectors of the predictors for the MCP flexion will be adjusted instead. Details of this are discussed in Section 5.3.

After the above adjustments, the predicted joint angle at the end of the pulse should satisfy the constraints. We may now use the predictors to predict the future angles of individual finger joints. The whole prediction method repeats when a new set of joint angles are read from the glove. Fig. 4 summarizes our method.

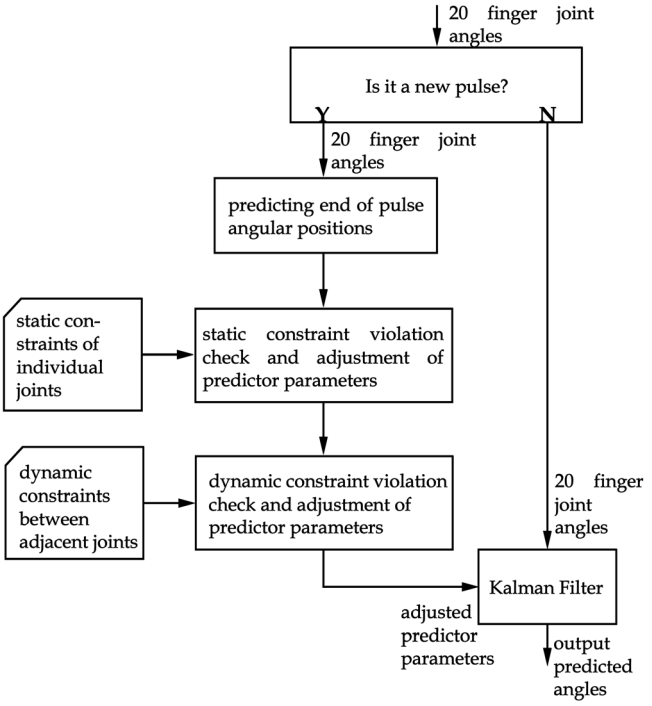

Fig. 4. Overview of prediction method.

4 The Hand Motion Model

In Figs. 2 and 3, the angular velocity $\omega(t, X)$ within a pulse can be approximated by an *elliptic model*, given the state vector $X = (K_1, K_2)$:

$$\omega(t, X) = K_1(\cos(K_2\, t) - 1), \tag{1}$$

where t is the time relative to the beginning of the pulse. X is a nonzero vector. To reduce measurement noise and the error due to the elliptic approximation, we apply an extended Kalman filter to the X of each joint, computed from (1). Hence, X is considered as the components of the state vector to the Kalman filter. The predicted angle of a joint can thus be found by integrating $\omega(t, X)$ in (1) over the prediction interval as follows:

$$\theta_{pred}(t_{pred}) = \theta_{cur} + [(K_1 \sin(K_2\, t)/K_2) - K_1 t]_{t_{cur}}^{t_{pred}}, \tag{2}$$

where t_{cur} is the current time. θ_{pred} and θ_{cur} are the predicted and the current joint angles, respectively. Let τ be the prediction length. The prediction time t_{pred} is then

$$t_{pred} = \min(t_{cur} + \tau, 2\pi/K_2).$$

In Fig. 2, we observe that the angular velocity is nearly zero during the period between any two pulses. It is not appropriate to model the velocity using the elliptic model due to the increased significance of noise in the sampled data. Hence, we model low-velocity motion with a generic linear model commonly used in dead reckoning:

$$\theta_{pred} = \theta_{cur} + \omega(t_{pred} - t_{cur}). \tag{3}$$

Note that although a joint angle is approximated by the linear model, the Kalman filter is not used. Once the absolute value of the angular velocity is higher than a predefined threshold, we compute the state vector X and then switch to use the elliptic model. In our implementation, we set the threshold for switching from the linear model to the elliptic model as 0.01 rad/s.

4.1 Configuring the Kalman Filter

The purpose of the Kalman filter is to filter noise and measurement error by iteratively minimizing the mean square estimation error using the following four steps:

1. Update the a priori estimate $\hat{X}_{k+1}^{-}$.
2. Compute the Kalman gain, which is used to relate the state estimation and the measurement.

3. Update the estimated state vector $\hat{X}_k$ using the measured velocity value Z_k.
4. Compute the error covariance, which represents the estimation error of the Kalman filter.

More details on Kalman filters can be found in [27].

The a priori estimate $\hat{X}_{k+1}^-$ can be set as follows:

$$\hat{X}_{k+1}^- = \Phi k \hat{X}_k + U_k, \tag{4}$$

where Φ_k is the transition matrix equal to I, as the state vector remains unchanged within a pulse. It relates the states at time t_k and t_{k+1}. U_k is the optional control input and should be set as a nonzero vector for the first iteration of the Kalman filter at each pulse and a zero vector for the rest of the pulse. At the beginning of each pulse, we need not evaluate U_k explicitly. We simply assign the most recently found state vector X to $\hat{X}_{k+1}^-$, which is $(K_1, K_2)^T$. X is determined by the starting velocity values of a pulse, which is discussed in Section 4.2.

Since the measurement update (Step 3) is in a nonlinear form, we need to apply the extended Kalman filter [27] here, which measurement update step is

$$\hat{X}_k = \hat{X}_k^- + G_k(Z_k - h_{t_k}(\hat{X}_k^-)), \tag{5}$$

where $h_{tk}(\hat{X}_k^-)$ is a function equivalent to (1). By applying linearization on $h_{tk}(\hat{X}_k^-)$, we obtain the matrix H_k, which corresponds to the noiseless connection between the state vector and the measurement as follows:

$$\begin{aligned} H_k &= ((\partial h_{t_k}(x)/\partial x)|_{x=\hat{x}_k^-}) \\ &= (\cos(K_2\, t_k) - 1, -K_1\, t_k \sin(K_2\, t_k))^T. \end{aligned} \tag{6}$$

4.2 Determining the State Vectors

The initial guess of the state vector X at the start of each pulse is crucial to the convergence of the filter and the accuracy of the prediction. An inappropriate initial guess may lead to instability or divergence. Hence, we use three consecutive nonzero velocity values to compute the vector and need to find an optimized X that matches the three velocity values. Let $W = (\omega_1, \omega_2, \omega_3)^T$, consisting of the first, second, and third angular velocity values, respectively, and taken since the start of a pulse. We define the optimization problem to minimize $f(X)$ as follows:

$$f(X) = \|g(X) - W\|^2, \tag{7}$$

where $g(X) = (\omega_1(X), \omega_2(X), \omega_3(X))^T$. To speed up the optimization process and to guarantee a global minimizer, an explicit solution is beneficial. The solution to the problem can be found by setting the gradient of $f(X)$ to zero. This can then be solved by Newton's method or other numerical methods. Note that if we use more velocity values in the initial guess of the state vector, the estimation will be more accurate, but the computational cost, as well as sampling delay, will be higher too.

Note also that we use only K_1 and K_2 in (1) to form the components of the state vector of the Kalman filter. This is different from the approach in [15] and [14], which applies the Gauss-Markov model and uses velocity and acceleration to form the state vector. Here, as K_1 and K_2 already depend on the values of velocity and acceleration, using K_1 and K_2, as well as velocity and acceleration, would increase the chance of having a singular or nearly singular covariance matrix, which will lead to numerical problems. Another consideration is the reduction in dimensions. Using only K_1 and K_2 helps reduce the computational cost significantly.

5 Incorporating Motion Constraints

Each joint angle should have lower and upper flexion limits, which are independent of other joints. They are referred to as *static constraints* [23], which are discussed in Section 5.1. When two or more adjacent fingers are moving, additional constraints may exist among these fingers. These constraints vary as the fingers move and are referred to as *dynamics constraints*. To avoid overcomplicating the hand model, we only focus on two important types of dynamic constraints in our method. One exists in MCP flexion and the other exists between MCP flexion and MCP abduction, which are discussed in Sections 5.2 and 5.3, respectively. Since the motion of the thumb is independent of other fingers, we only need to consider the dynamic constraints among the other four fingers. The limits defined by each constraint can be easily obtained by physically recording finger motion.

There are several ways of incorporating the constraints into the Kalman filter. One straightforward solution is to minimize the parameterization by substituting constraints into the measurement equations [28]. Unfortunately, it is not always possible to do the substitution as it may lead to an ill-formed parameterization [29]. Another approach is to treat the constraints as perfect measurements [30]. However, this may increase the chance of getting numerical problems and the dimension of the measurement vector as stated in [31]. Instead of regarding the constraints as perfect measurements, the approach in [32] ensures no violation of the constraints at each iteration of the Kalman filter by optimizing the estimated state vector after the measurement update subject to satisfying the constraints. Although this does not cause additional numerical problems nor does this increase the dimension, it has a higher computational cost due to additional optimization in each iteration. It will be even worse if the constraints involve high-dimensional vectors (20 in our case). Here, we propose to validate the constraints at the start of each pulse only. As the duration of a pulse is short, the chance of violating the constraints within the pulse is small. Even if it happens, the amount of violation should be small. In addition, as we optimize the state vectors subject to relevant constraints only, the dimension of the problem is greatly reduced.

5.1 Static Constraints of Finger Motion

If the state vector $(K_1, K_2)^{\mathrm{T}}$ estimated by (7) causes violation of static constraints, it needs to be evaluated again by another formula. The angle of a joint θ should satisfy the inequalities

$$s_{min} \le \theta \le s_{max}, \tag{8}$$

where s_{min} and s_{max} are the lower and the upper limits, respectively. They are different for each joint. Let ψ be the initial angular position at the start of the pulse. After the

TABLE 1
Change in Pulse Length Due to the Motion of Other Fingers

Average pulse length	491ms.
Average error in estimated pulse length	47 ms.
Average error / average pulse length	0.096

preliminary state vector $(K_1, K_2)^{\mathrm{T}}$ is found, we predict the value of θ at the end of the pulse by (2) as follows:

$$\theta = \psi + (-2\pi K_1/K_2). \tag{9}$$

If the constraints in (8) are violated, we may rewrite (8) as

$$\varphi = -2\pi K_1/K_2 \tag{10}$$

where $\varphi = s_{max} - \psi$ or $s_{min} - \psi$. If we substitute (10) into (7), a similar optimization problem but with only one variable is obtained. Formally, the solution to the problem can be obtained by minimizing $f(X)$ in (7) subject to the linear constraint in (10). The solution can be found by setting the gradient of $f(X)$ to zero.

5.2 Dynamic Constraints of MCP Flexion

The dynamic constraints of MCP flexions specify the maximum angular difference between any two adjacent fingers as follows:

$$F_{min} \leq D\, X_{flex} \leq F_{max}, \tag{11}$$

where

$$D = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix},$$

X_{flex} is a vector composed of the angular positions of the four MCP flexions $(\theta_{ind\_mcp\_flex}, \theta_{mid\_mcp\_flex}, \theta_{rin\_mcp\_flex}, \theta_{pin\_mcp\_flex})^{\mathrm{T}}$, and *ind*, *mid*, *rin*, and *pin* represent the index, middle, ring, and pinky, respectively. F_{min} and F_{max} are vectors containing the relative minimum and maximum differences among the four MCP flexions, that is, $\theta_{ind\_mcp\_flex} - \theta_{mid\_mcp\_flex}$, $\theta_{mid\_mcp\_flex} - \theta_{rin\_mcp\_flex}$, and $\theta_{rin\_mcp\_flex} - \theta_{pin\_mcp\_flex}$. This kind of constraint is primarily due to the tension of ligaments and skins between adjacent fingers.

In order to apply the constraints defined by (11), we need to study how the pulse length of an MCP flexion may be affected by other MCP flexions. We have conducted an experiment by moving the fingers arbitrarily and examining how much the pulse length of a finger is altered by the movement of other fingers. We collected thousands of samples from several users. The results, as shown in Table 1, indicate that the pulse length of the flexion motion is not really affected by the movement of other fingers. We can estimate the pulse length as $2\pi/K_2$. The absolute difference between the actual and the estimated pulse lengths indicates the error of the estimated pulse length. We can see that the ratio of the average error to the average pulse length is very small (less than 0.1).

When two adjacent fingers concurrently move in the opposite directions, their moving speeds may be lowered by each other as they are near their own dynamic limits. This speed reduction increases the pulse lengths. On the other hand, the room for moving is reduced, which shortens the pulses. These contradicting factors appear to cancel out each another. We have performed an experiment to move the MCP joints of two adjacent fingers until they stop at their own dynamic limits. All four fingers are examined as shown in Table 2. The average length of all motion pulses and the average absolute difference in stopping time between two correlating flexions are recorded. We have found that the ratio of the average difference to the average pulse length is small enough to conclude that the MCP flexions of two correlating fingers stop simultaneously when they reach their dynamic constraints.

TABLE 2
Change in Pulse Length Due to Concurrent Opposite Finger Motion

Average pulse length	449 ms.
Average stopping difference	12 ms.
Average difference / average pulse length	0.027

To adjust the MCP flexion state X, we can minimize $\sum_{X \subset S_{flex}} f(X)$ subject to the linear constraints in (11), where S_{flex} is the set of MCP flexion states. The dimension of the problem is eight. This is similar to the optimization problem in inverse kinematics. Due to the high computational cost, it is not suitable for real-time prediction. From the results shown in Tables 1 and 2, it is possible to reduce the cost by half. Since we have assumed that the pulse length remains the same, K_2 should not change. However, we note from Table 2 that there is a small change in the pulse length. Instead of adjusting K_2, we add a δ to K_2 and then minimize $\|\delta\|^2$ subject to

$$2\pi/\left(K_2^{(i)} + \delta^{(i)}\right) - t_{cur}^{(i)} = 2\pi/\left(K_2^{(j)} + \delta^{(j)}\right) - t_{cur}^{(j)}, \tag{12}$$

where $\delta = (\delta^{(i)}, \delta^{(j)})^T$. i and j are the MCP flexions of the two adjacent fingers that predicted that the angular positions violate the constraints defined by (11). t_{cur} is the time elapsed since the start of the pulses. The right-hand side and left-hand side of (12) represent the time left before both fingers are stopped by their MCP flexion dynamic constraints. They are expected to stop at the same time. K_2 is then incremented by δ. The complexity of the problem can be further reduced by rearranging and trimming the constraints stated in (11). After the state vector of each MCP flexion has been adjusted by considering the static constraints, the angular positions from (9) are checked against the inequality in (11). The columns and rows of D corresponding to the joints not violating the constraints are eliminated. The K_2 of all joints are removed from the problem because they are adjusted by the method described above. As a result, the size of D is decreased, and (11) becomes

$$D^* X^*_{flex} \leq F^*, \tag{13}$$

where the components of X^*_{flex} are picked from X_{flex}, and the size of X^*_{flex} is smaller than X_{flex}. Since only the minimum or maximum limit can be reached at any time,

F_{min} and F_{max} can be combined into F^* by negating appropriate rows in D.

Although the complexity of the problem is now much reduced, we do not apply (13) to minimize $\sum_{X \subset S_{flex}} f(X)$ because it ignores the finger motion characteristics. We note that when the fingers of a free hand are moving vigorously, more than one pair of fingers may violate their dynamic constraints. If a pair of fingers reaches the limits first, it may affect the motion limits of other fingers, and we need to handle this pair of joints first. We compute the remaining time of the pulse l_{rem} as $2\pi/K_2 - t_{cur}$. We then choose the MCP flexions of two adjacent fingers, i and j, violating the dynamic constraints and with the smallest l_{rem}. As the K_2 of both flexions have been adjusted, we only need to minimize the following functions:

$$f_{K1}^{((i,ti),(j,tj))}(V_{K1}) = \sum\nolimits_{n=i,j} \| g_{K1}^{(n,tn)}(K_1^{(n)}) - W^{(n,tn)} \|^2, \quad (14)$$

where

$$g_{K1}^{(n,tn)}\left(K_1^{(n)}\right) = \left(\omega_{tn+1}\left(X\left(K_1^{(n)}\right)\right), \omega_{tn+2}\left(X\left(K_1^{(n)}\right)\right), \omega_{tn+3}\left(X\left(K_1^{(n)}\right)\right)\right)^T,$$
$$W^{(n,tn)} = \left(\omega_{tn+1}, \omega_{tn+2}, \omega_{tn+3}\right)^T.$$

t_n is the time elapsed for joint n, where n can be i or j in (14), since the start of the motion pulse, and $V_{K_1} = (K_1^{(i)}, K_1^{(j)})^T$. $X(K_1^{(n)})$ is the state vector with a fixed $K_2^{(n)}$. The minimization of (14) is subject to

$$\theta_{pred}^{(i)}\left(2\pi/K_2^{(i)}\right) - \theta_{pred}^{(j)}\left(2\pi/K_2^{(j)}\right) = F_{i-j}^*,$$

where F_{i-j}^* is one of the components in F^* that correlates MCP flexions i and j. Functions $\theta_{pred}^{(i)}(2\pi/K_2^{(i)})$ and $\theta_{pred}^{(j)}(2\pi/K_2^{(j)})$ are defined by (2). The constraint is actually extracted from the one in (13). This can be easily transformed into a linear form as follows:

$$A^T V_{K1} = b, \quad (15)$$

where A is a constant vector, and b is a constant scalar.

After the first pair of state vectors has been adjusted, we check other joints violating the flexion constraints. We select and adjust the pair of joints with the lowest l_{rem}. As more state vectors are fixed through this iteration, the cost of the minimization problem is reduced. The iteration stops when there are no more flexion constraints violated.

5.3 Dynamic Constraints between MCP Flexion and Abduction

After adjusting the MCP flexions, the MCP abduction states may also need to be adjusted according to the dynamic constraints existing between flexion and abduction. This may in turn require modifying the adjusted MCP flexions again. To study the relationship between the abduction limits and MCP flexion limits of a finger, we have captured and plotted the index MCP abduction against the index MCP flexion by stretching the index MCP joint in all directions so as to draw a circle as large as possible. This helps find the dynamic abduction limit at every MCP flexion angle. As shown in Fig. 5, the range of the abduction angle reduces as the MCP flexion approaches its static limits and increases as the MCP flexion approaches the midpoint between the static limits. We may approximate the data points by two quadratic curves, one above and another below the x-axis. The widths of the two parabolas indicate the static limits of the MCP flexion, whereas the heights indicate the static limits of the MCP abduction. Similar graphs can be found by moving the MCP of other fingers.

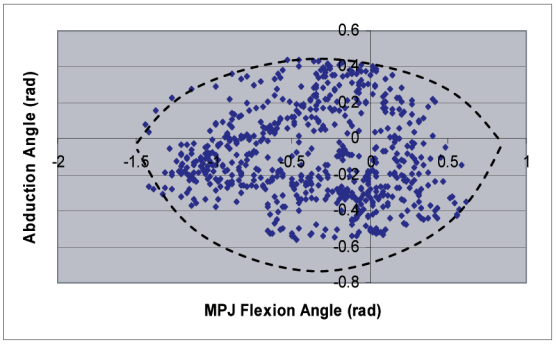

Fig. 5. Index MCP abduction versus index MCP flexion.

Here, we attempt to relate the dynamic abduction limits d_{min_abd} and d_{max_abd} with the MCP flexion angle θ_{flex} by a quadratic model. To model the dynamic abduction maximum (upper curve d_{max_abd} in Fig. 5), we need to obtain three points to define the curve. We observe that the curve cuts the x-axis at two points, the static minimum and the static maximum of the MCP flexion:

$$(s_{min\_flex}, 0),$$
$$(s_{max\_flex}, 0),$$

where $s_{min\_flex}/s_{max\_flex}$ are the static limits of the MCP flexion. If we also consider the midpoint between them,

$$((s_{min\_flex} + s_{max\_flex})/2, s_{max\_abd}),$$

the curve becomes

$$d_{max\_abd} = -4s_{max\_abd}/(s_{min\_flex} - s_{max\_flex})^2(\theta_{flex}^2 - (s_{min\_flex} + s_{max\_flex})\theta_{flex} + s_{min\_flex}s_{max\_flex}). \quad (16)$$

Similarly, the lower curve is defined by

$$(s_{min\_flex}, 0),$$
$$(s_{max\_flex}, 0),$$
$$((s_{min\_flex} + s_{max\_flex})/2, s_{min\_abd}),$$

where s_{min_abd} and s_{max_abd} are the static limits of the MCP abduction. Hence, d_{min_abd} can be computed as

$$d_{min\_abd} = -4s_{min\_abd}/(s_{min\_flex} - s_{max\_flex})2(\theta_{flex}^2 - (s_{min\_flex} + s_{max\_flex})\theta_{flex} + s_{min\_flex}s_{max\_flex}). \quad (17)$$

Since MCP flexion is usually more energetic and produces higher velocity motion than MCP abduction, we may

assume MCP flexion, rather than MCP abduction, to be the dominating factor as in (16) and (17).

After the state vector for the MCP flexion has been evaluated by the method shown in Section 5.2, the dynamic abduction limits of each finger are found by (16) and (17). We compute θ_{flex} at the end of the velocity pulse by using (9) to estimate the values of d_{min_abd} and d_{max_abd}. The predicted abduction angle θ_{abd} at the end of the pulse (again by (9)) is then checked against the following constraints:

$$\theta_{abd} \geq d_{min\_abd}, \tag{18}$$

$$\theta_{abd} \leq d_{max\_abd}. \tag{19}$$

Violation of any one of the above constraints requires the adjustment of the state vector $(K_1^{abd}, K_2^{abd})^{\mathrm{T}}$ for the MCP abduction. Since (18) and (19) are a complimentary pair, only one can be violated. If (18) is violated, the constraints can be satisfied simply by multiplying K_1 by a factor as follows:

$$K_1^{abd} = (\Omega/\Omega_{ori})K_1^{abd\_ori}. \tag{20}$$

Ω is the displacement from the current position to the dynamic abduction limit d_{min_abd}, and Ω_{ori} is the displacement from the current position to the expected end of the pulse. They are found as follows:

$$\Omega = d_{min\_abd} - \theta_{abd\_cur} \tag{21}$$

$$\begin{aligned}\Omega_{ori} = &-2\pi K_1^{abd\_ori}/K_2^{abd\_ori} - (K_1^{abd\_ori}/\\ &K_2^{abd\_ori}\sin(K_2^{abd\_ori}t_{cur}) - K_1^{abd\_ori}t_{cur})\end{aligned} \tag{22}$$

where $(K_1^{abd\_ori}, K_2^{abd\_ori})^{\mathrm{T}}$ is the state vector before the adjustment, and θ_{abd_cur} is the current abduction angle. Here, K_2^{abd} need not be adjusted as we assume that the pulse duration is not changed (Table 1). On the other hand, if (19) is violated, (21) is replaced by

$$\Omega = d_{max\_abd} - \theta_{abd\_cur}. \tag{23}$$

Sometimes, abduction may be energetic enough to alter flexion motion, when flexion is stationary or has a weak motion, that is, flexion is approximated by a linear model. In this case, if the predicted θ_{abd} at the end of the pulse violates either (18) or (19), a new θ_{flex} is evaluated by solving (16) or (17), where d_{min_abd} and d_{max_abd} are known values. Two solutions are usually found, and we choose the one nearer to the current flexion angle θ_{flex_cur}. The angular velocity of the flexion motion is set as

$$\omega_{flex} = (\theta_{flex} - \theta_{flex\_cur})/l_{rem}, \tag{24}$$

where l_{rem} is the remaining time of abduction pulse.

6 A Revised Dead-Reckoning Scheme

One major problem of dead reckoning is that it takes time for the update messages to reach the remote client, in particular, when the network latency is high. After the host has detected that the prediction error is too high and sent an update message to the client, it will take time for the message to reach the client, and the error may become very significant when the message actually reaches it. One obvious solution to this problem is to lower the error threshold, that is, increasing the update frequency. However, this also increases network loading.

In our framework, we have implemented a revised dead-reckoning scheme to address the problem. Our idea is to send more update messages during the period when we anticipate a higher prediction error and less in other times to optimize the network bandwidth consumption. This approach is considered more aggressive as we need to predict not only the object states but also when the prediction errors may become high. Fortunately, with our proposed hand motion model, we already know that the prediction error tends to fluctuate significantly at the beginning of a pulse when we are refining the motion parameters for the pulse. Hence, we reduce the error threshold at the beginning period of each pulse and then return it to normal for the rest of the pulse. (In our implementation, we set this beginning period as the initial 5 percent of the estimated pulse length.)

Although there are some adaptive schemes proposed [33], [11], [10], they focus on reducing network loading through either adjusting the thresholds or reducing the message size. In our framework, we focus on improving the prediction accuracy while minimizing the effect on network loading. The revised dead-reckoning scheme, although simple, has the advantage that it further reduces the error between the predicted and the actual states of the object, that is, improving the system consistency, as shown in Section 7.3.

7 Results and Discussion

To study the performance of the new prediction method, we have compared the performance of four predictors:

- The second-order polynomial predictor [13]: P_{sop}.
- The Gauss-Markov model [15]: P_{gmm}.
- The hybrid motion model without considering any physical constraints [5]: P_{hm}.
- The new constraint-based hybrid motion model: P_{ehm}.

All these methods apply Kalman filtering. The noise parameters P and Q are found by Powell's method [34] using arbitrary hand motion. The objective is to minimize the absolute error for one step prediction. Due to page limitation, we will not show the noise covariance matrices of all joint angles here. We only show the mean values and the variances of all the components in the noise covariance matrices as follows:

$$\begin{aligned}&Mean\ component\ values\ of\ Q \approx \begin{pmatrix}0.54 & 0.16\\ 0.16 & 0.06\end{pmatrix},\\ &Mean\ component\ values\ of\ P \approx (0.70),\\ &Component\ variance\ of\ Q \approx \begin{pmatrix}0.11 & 0.07\\ 0.07 & 0.04\end{pmatrix},\\ &Component\ variance\ of\ P \approx (0.12).\end{aligned}$$

We note that tuning P and Q for each user or for each motion style has negligible effect on prediction accuracy. Thus, we may fix their values after they are obtained.

TABLE 3
Computational Costs of the Four Predictors

P_{sop}	P_{gmm}	P_{hm}	P_{ehm}
5ms	4ms	1ms	3ms

TABLE 4
Network Latency of Various Network Connections

Network connection	Latency
LAN (within a department)	5 ms.
Internet (within a university)	40 ms.
Internet (between Hong Kong and US)	160 ms.
Internet (between Hong Kong and UK)	325 ms.

The experiments are presented as follows. Section 7.1 shows the experiment settings and some preliminary experiments conducted. Section 7.2 compares the prediction accuracy of the four predictors. Section 7.3 investigates the accuracy of the predictors and the revised dead-reckoning scheme with regard to network loading and network latency.

7.1 Settings and Preliminary Experiments

We have integrated the four prediction methods into a glove-based system for collaborative design [4] with 18-sensor CyberGloves. Our tests show that the system can capture up to about 100 samples per second in practice. The resolution of the gloves is about 0.5 degree, and the repeatability is about 1 degree. To compute the angular velocity, we first apply the eighth-order Butterworth filter [35] to the input samples to reduce the effect of noise and then divide the angular difference of two consecutive filtered samples by the sampling interval. The use of Kalman filter helps further reduce the noise. All the experiments shown in this section were conducted on a PC with a 2 GHz Pentium 4 CPU.

First, we have compared the size of the update messages of our method with those of the second-order polynomial and Gauss-Markov predictor. For each predictor, our method needs to send four 32-bit floating-point numbers in each update message, consisting of θ_{cur}, t_{cur}, K_1, and K_2 in (2). The formulas for the second-order polynomial P_{sop} and for the Gauss-Markov predictor P_{gmm} are as follows:

$$\boldsymbol{P_{sop}}: \quad \theta_{pred} = \theta_{cur} + \omega t + 0.5\alpha t^2, \tag{25}$$

$$\boldsymbol{P_{gmm}}: \quad \theta_{pred} = \theta_{cur} + \omega t + \alpha/\beta(t + e^{-\beta t}/\beta - 1/\beta), \tag{26}$$

where ω is the angular velocity, α is the angular acceleration, t is the time elapsed, and β is a predefined parameter of the Gauss-Markov model. Both methods require sending three 32-bit floating-point numbers, consisting of θ_{cur}, ω, and α.

Second, we have conducted an experiment to study the speed of the predictors. Table 3 shows the times taken for the predictors to compute 20 joint angles. We can see that the computational costs of these methods are small enough to be neglected.

Third, we have conducted an experiment to determine the network latency of various network connections as shown in Table 4. To conduct this experiment, we use the "ping" function available in Unix and record how long it takes for the sender to receive an echo from the remote machine. This measures the round-trip delay. The network latency can then be approximated as half of this value.

7.2 Prediction Accuracy

Four types of typical hand action are used in this experiment: *grabbing*, *counting*, *typing*, and *sculpting*. Grabbing involves clenching and releasing an object. All the fingers move back and forth at the same time, and most of the joints reach their static limits. Counting involves moving some MCP joints rapidly. Usually, only a few joints are changed significantly. Typing is common in some applications such as pressing the keyboard and playing the piano. It involves moving multiple MCP joints but with a smaller range than grabbing and counting. It also involves a greater amount of abduction. Sculpting involves a large amount of random finger flexion, as well as abduction. Since finger motion in sculpting is the most arbitrary among the four types of action and may involve the characteristics of the other three types of hand action, it is the most difficult to predict.

We have invited 20 users to perform this experiment. To capture the grabbing motion, the users are told to grab something handy enough to be held in the palm, such as a tennis ball or a glass, from the virtual scene and then place it on a virtual desk. To capture the counting motion, the users are asked to represent a random number (between 0 and 10) by their fingers. To capture the typing motion, the users are asked to type an article with the keyboard. In all the above captures, the users are told to stop after 3 minutes at each task. Capturing the sculpting motion is a bit tricky. Performing real sculpting may involve using materials such as clay, which may damage the CyberGloves. Therefore, we ask the users to perform virtual sculpting using our virtual sculpting prototype [4] to modify a model arbitrarily for 30 sec.

Fig. 6 shows the prediction errors of the four prediction methods on the four types of hand action. *Prediction length* refers to how far ahead in time (in terms of seconds) that we want to predict. We perform the experiments using five discrete values of prediction length, from 0.1 second to 0.5 second. The prediction error E (the vertical axis) is measured as the average difference between the actual and predicted joint angles in radians as

$$E = \sum_{t=1}^{N} \sum_{i=1}^{I} \frac{|\theta_{i,t} - \hat{\theta}_{i,t}|}{N * I} \tag{27}$$

where i is the index to a joint angle, t is the sample number, I is the total number of joints, N is the total number of samples, $\theta_{i,t}$ is the sampled angle, and $\hat{\theta}_{i,t}$ is the predicted angle.

In general, the prediction errors of the four prediction methods increase roughly linearly but at different rates as we increase the prediction length. At a low prediction length (0.1 second), the difference in performance of all four methods is not so obvious, although we may observe that P_{ehm} (the proposed method) does slightly better than other methods. However, as the prediction length increases, P_{ehm} clearly outperforms all other methods, and P_{sop} performs

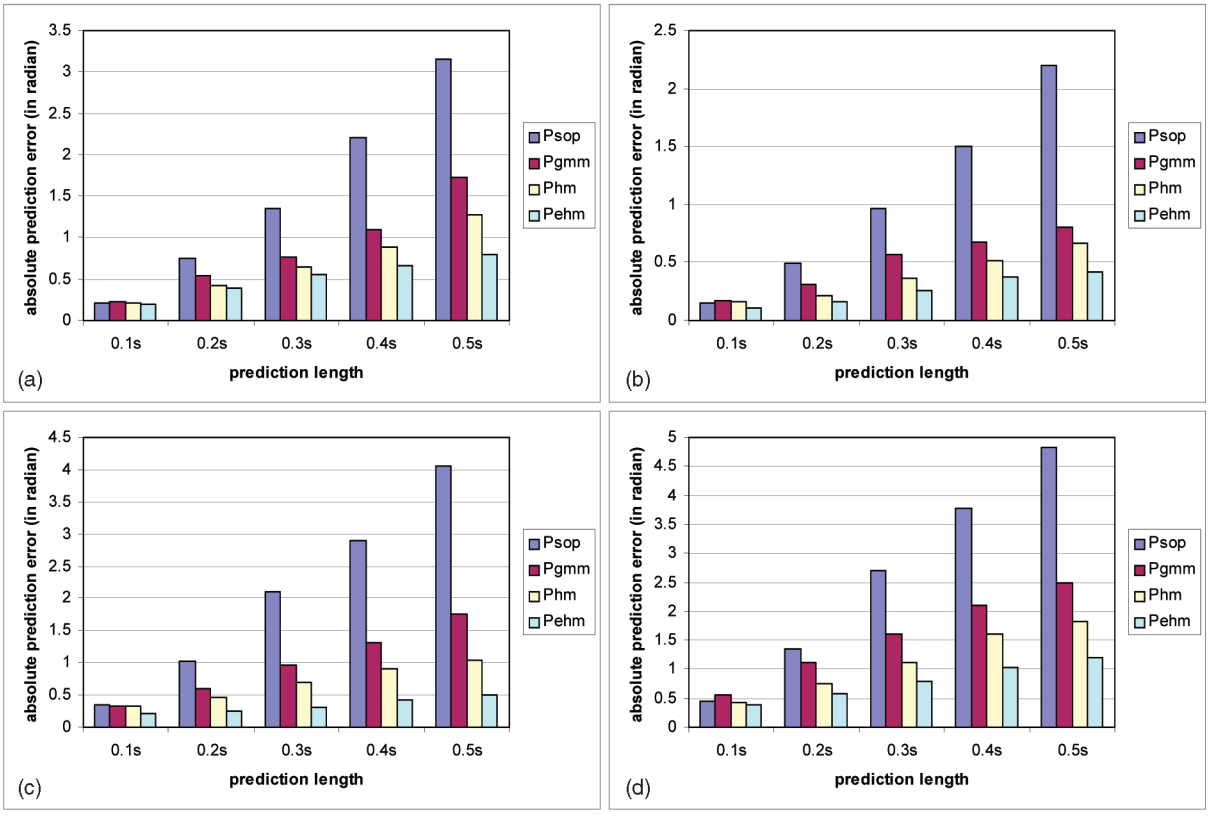

Fig. 6. Comparing the prediction errors of the four predictors with four types of hand action. (a) Grabbing. (b) Counting. (c) Typing. (d) Sculpting.

the worst, especially at a high prediction length of 0.5 second, as shown from the snapshots in Fig. 7.

Consider (25) for P_{sop} and (26) for P_{gmm}. If we differentiate the two equations, we get

$$\boldsymbol{P_{sop}}: \quad \omega_{pred} = \omega_{cur} + \alpha\, t \tag{28}$$

$$\boldsymbol{P_{gmm}}: \quad \omega_{pred} = \omega_{cur} + \alpha(1 - e^{-\beta t})/\beta. \tag{29}$$

If we compare (1), (28), and (29), we can see that all three equations are increasing functions when t is small, as shown in Fig. 8. However, as t increases, P_{sop} will continue to increase, P_{gmm} will gradually level off, and the elliptic model of P_{hm} and P_{ehm} will peak at some point and then gradually return to zero. This explains why the prediction errors of the four methods are very close at a low prediction length. As the prediction length increases, the velocity curve of P_{sop} deviates more and more from the shape of the pulses shown in Fig. 2, whereas the predicted velocity curves of P_{hm} and P_{ehm} resemble the shape of the pulses more closely. This explains why P_{hm} and P_{ehm} perform better than the other two methods. On the other hand, P_{ehm} considers the static and dynamic motion constraints and uses them to adjust the state vectors for prediction. This allows P_{ehm} to have a much lower prediction error than P_{hm} at high prediction lengths.

Comparing the four types of action, we notice that the prediction error for counting in Fig. 6b is generally lower than those for other actions. This is because counting involves the rapid movement of only one or a few joints. Most of the other joints are nearly stationary. Hence, only a few MCP flexions contribute to the prediction error. Grabbing is similar to counting but with more joints moving at the same time. Thus, the prediction error for grabbing is higher than that for counting. Although typing also involves the rapid movement of only a few joints at any time, the movement usually lasts for longer period of time. Sculpting involves more fingers for a longer period of movement. This is why the prediction errors for typing and for sculpting are relatively high compared with those for the other two actions.

As we analyze the results further, we note that a major source of error for P_{ehm} is due to the irregular shape of some pulses. In particular, some pulses may have a dip within the curve of a pulse. We may classify this kind of pulse based on the extent of the dip into small, medium, and large, as shown in Fig. 9. The closer it is to zero, the greater the extent of the dip is. We believe that this problem is mainly due to the mechanical noise of the CyberGloves. Judging from all the motion data that we have collected, these three types of dip have roughly the same percentage of occurrences, which is 8 percent each.

In general, our method can cope with the small dips (Fig. 9a) and the large ones (Fig. 9c) very well. For the small dips, our method simply treats the pulses as normal pulses. For the large ones, our method treats each pulse as two separate pulses. However, the medium ones induce a much higher error. To study the error contributions of these three types of dip, we remove the small ones and the large ones from the motion data, which resulted in a 3 percent reduction in prediction error. On the other hand, if we remove the medium ones from the motion data, we get a 12 percent reduction in prediction error.

In summary, our proposed predictor outperforms others methods, in particular, at high prediction lengths. When the

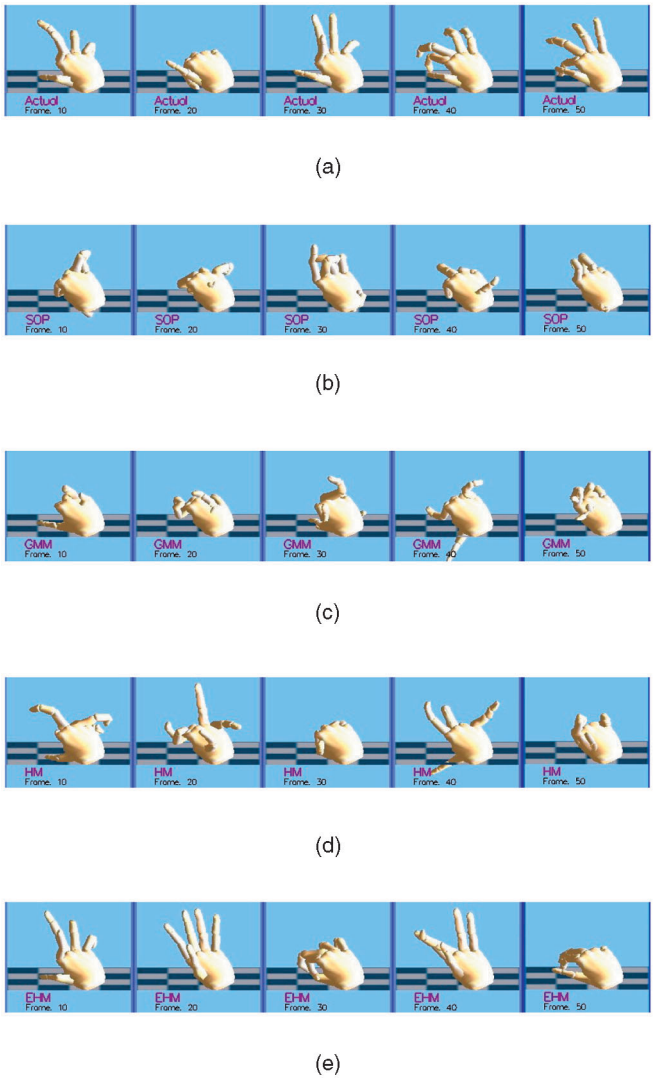

Fig. 7. Predicted gesture comparison at 0.5 second intervals. (a) Actual. (b) P_{sop}. (c) P_{gmm}. (d) P_{hm}. (e) P_{ehm}.

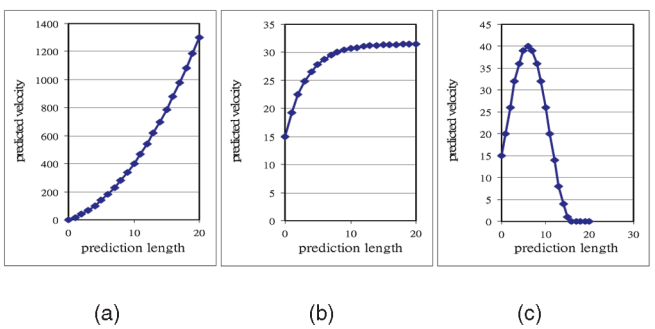

Fig. 8. Sketches of the three predictors. (a) P_{sop}. (b) P_{gmm}. (c) P_{hm} and

prediction length is as high as 0.5 second, the performances of the four methods deviate significantly. P_{sop} shows the worst performance. Both P_{gmm} and P_{hm} deteriorate in accuracy. P_{ehm} has a significantly higher accuracy, and its predicted gestures are still relatively close to the actual ones. We have conducted the experiment with 10, 15, and 20 users and found that the results are consistent despite the change in the number of users.

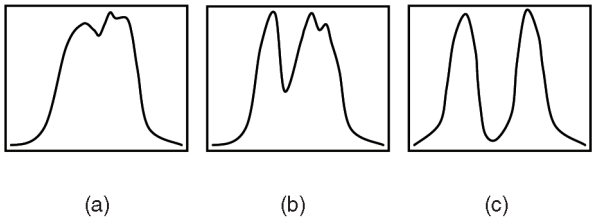

Fig. 9. Extent of dip in a pulse. (a) Small. (b) Medium. (c) Large.

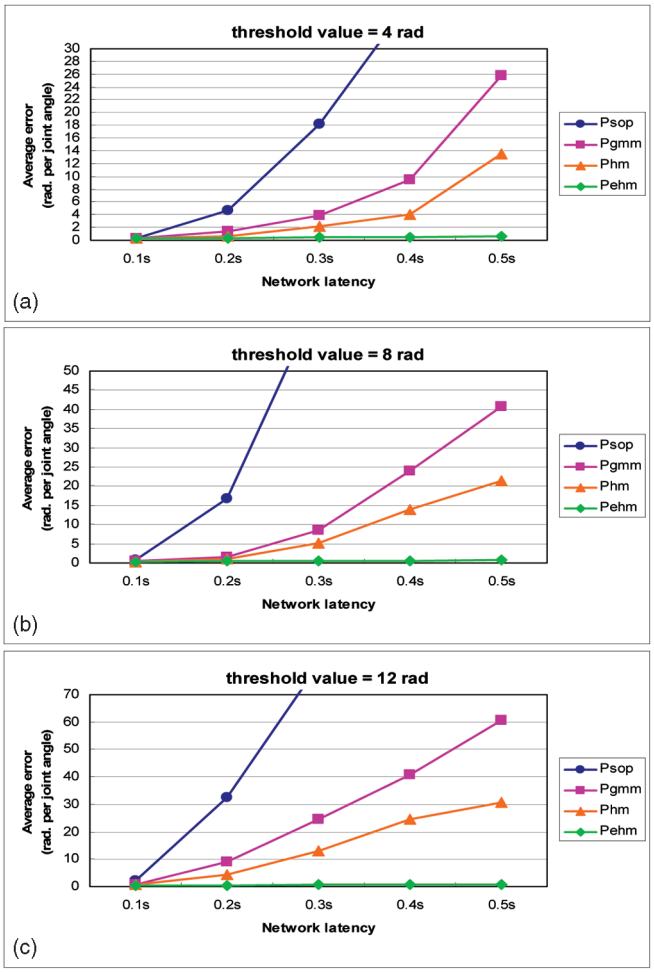

Fig. 10. Performance comparison of the four predictors in dead reckoning at different threshold values. (a) 4 rad. (b) 8 rad. (c) 12 rad.

7.3 Network Latency/Loading on Prediction Accuracy

In this experiment, we evaluate the proposed prediction method and the revised dead-reckoning scheme on how the prediction accuracy is affected by network latency and network loading. Through studying the prediction accuracy against network latency and the prediction accuracy against network loading, we attempt to achieve a comprehensive evaluation on the proposed methods. To study this under the worst scenario, we perform the experiment using the sculpting motions that we used in the last experiment, as these motions are the most arbitrary among the four types of actions.

To study the effect of network latency on prediction accuracy, we plot the average error (the average difference between the actual and the predicted joint angles computed by (27)) against network latency, as shown in Figs. 10

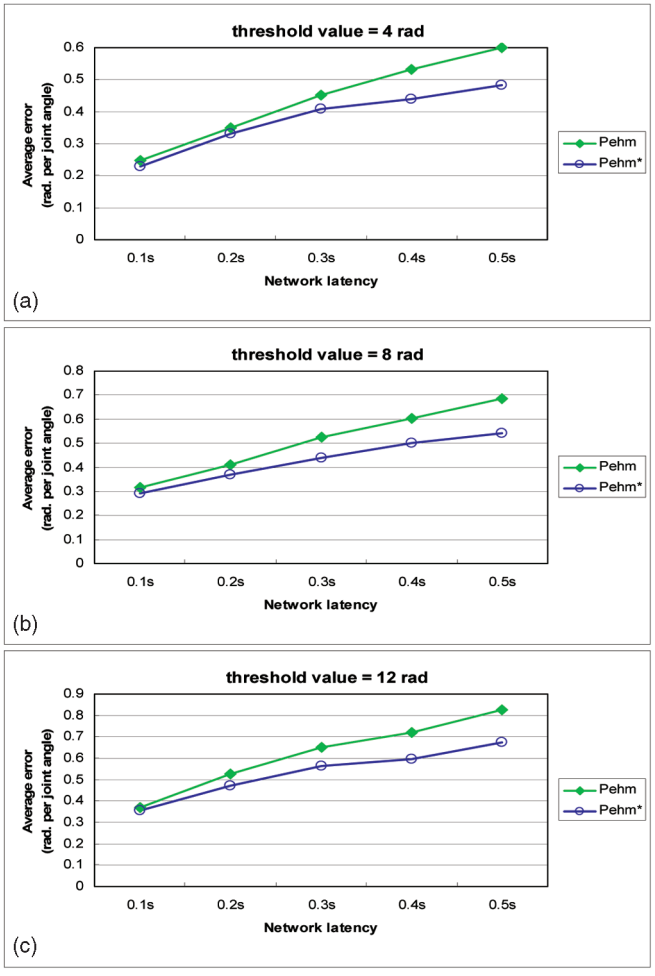

Fig. 11. Performance comparison of the two dead-reckoning schemes at different threshold values. (a) 4 rad. (b) 8 rad. (c) 12 rad.

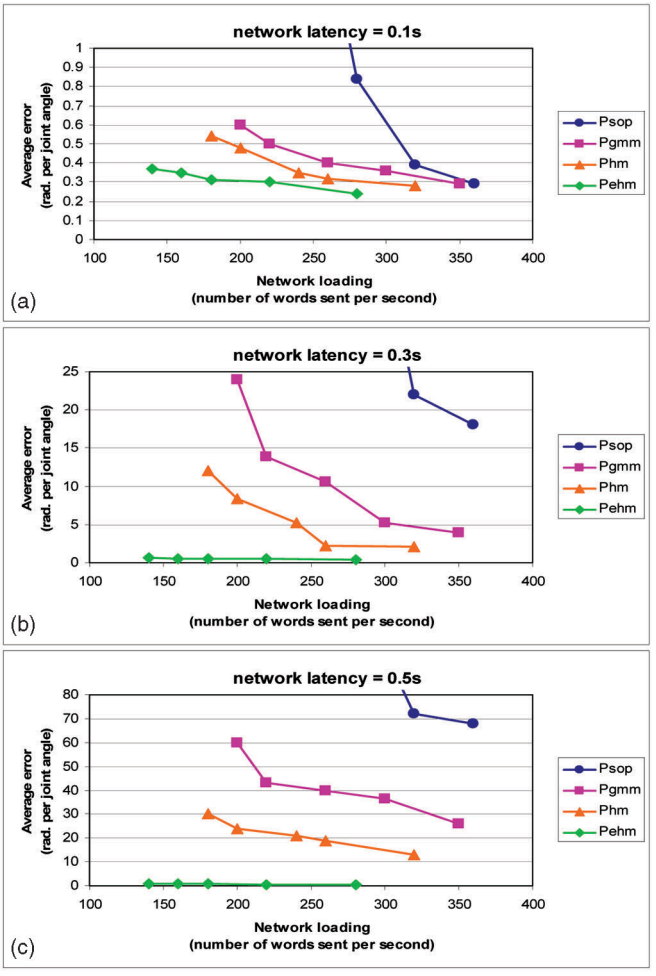

Fig. 12. Performance comparison of the four predictors in dead reckoning at different network latencies. (a) 0.1 second. (b) 0.3 second. (c) 0.5 second.

and 11. The three diagrams in Fig. 10 are produced by setting the threshold value of dead reckoning to 4, 8, and 12 radians. When the sum of the absolute differences between the actual and the predicted joint angles of a single sample of the glove is higher than the given threshold value, we send out an update message. In general, we observe that the prediction error increases with the network latency. This is because, by the time the update message has arrived at the remote client, the error may become much higher and the higher the network latency, the higher this error is. In Fig. 10, we can see that P_{ehm} clearly outperforms the other methods, in particular, at high prediction length. This agrees with our results presented in Section 7.2.

We have integrated our predictor into the revised dead-reckoning scheme presented in Section 6 P^*_{ehm}, as well as the original dead-reckoning scheme P_{ehm} used in Fig. 10. Fig. 11 shows the performance comparison. We can see that P^*_{ehm} performs even better than P_{ehm}. When the latency is at 0.1 second, its prediction error is about 4 percent lower than P_{ehm}. As the latency increases, the difference in performance also increases. When the latency is at 0.5 second, the prediction error of P^*_{ehm} is about 18 percent lower than P_{ehm}.

To study the effect of network loading on prediction accuracy, we have plotted the same set of data in a different way. We look at the induced network loading as a result of changing the threshold value of dead reckoning (from 4 to 12 radians), and we plot the average error against network loading as shown in Figs. 12 and 13. Network loading is measured in terms of the number of 32-bit floating-point numbers per second sent through the network to update the remote client.

In general, the three diagrams in Fig. 12 show that the average error decreases as the network loading increases. This means that as we send out more update messages by lowering the threshold value, the prediction accuracy improves at the expense of higher network loading. From all diagrams in Fig. 12, we can see that P_{ehm} performs better than those of the other methods under the same network loading. This performance improvement is even more significant at higher network latency. In other words, given an average error, P_{ehm} incurs a lower network loading. This implies that a more accurate predictor helps reduce the network loading, as less update messages are needed to be sent. Note that in Fig. 12c, the average error of P_{sop} is so high that its curve is not shown.

Fig. 13 compares the performance of the revised dead-reckoning scheme P^*_{ehm} with that of the original scheme P_{ehm} used in Fig. 12. Fig. 13a shows that P^*_{ehm} performs only slightly better than P_{ehm} at low network latency, but it incurs a slightly higher network loading. However, as the

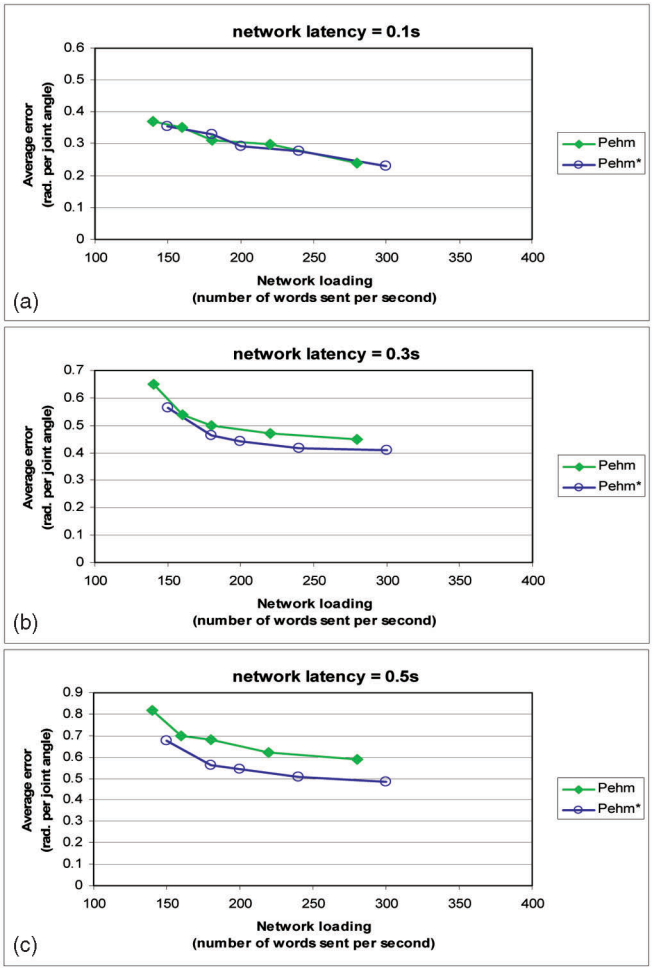

Fig. 13. Performance comparison of the two dead-reckoning schemes at different network latencies. (a) 0.1 second. (b) 0.3 second. (c) 0.5 second.

network latency increases, P^*_{ehm} outperforms P_{ehm} more and more, as shown in Figs. 12b and 12c. If we compare Fig. 13 with Fig. 11, we can see that when the latency is at 0.1 second, although P^*_{ehm} is more accurate than P_{ehm}, it needs to send more update messages, causing a higher network loading. This is due to the extra messages sent at the beginning of each pulse. Hence, if we look at the two curves in Fig. 13a, at the same loading, P^*_{ehm} and P_{ehm} actually produce similar average errors. However, when the latency is at 0.3 second and 0.5 second, the improvement in accuracy due to the extra messages sent becomes much more significant. If we look at the two curves in Figs. 13b and 13c, at the same loading, P^*_{ehm} now produces a lower average error.

8 Conclusion

In this paper, we have presented a motion prediction method for modeling human hand motion by considering the physical constraints of finger motion. We classify the constraints into static and dynamic. By predicting if a moving joint will violate the constraints at the end of each movement step, we may adjust the state vector of its predictor to refine the model. This provides a feedback to the predictor to improve the prediction accuracy. We have demonstrated with a number of experiments that the proposed predictor performs much better than other popular prediction methods, in particular, at long prediction lengths.

We have also proposed in this paper a revised dead-reckoning scheme, which would send update messages at times when we expect the prediction error to be high. Our results show that the new scheme produces a lower average prediction error than the original dead-reckoning scheme, in particular, at high network latency.

The new method proposed in this paper is expected to benefit applications that require the remote sharing of hand gestures, such as collaborative design.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful and constructive comments on this paper.

References

[1] L. Cutler, B. Frohlich, and P. Hanrahan, "Two-Handed Direct Manipulation on the Responsive Workbench," *Proc. ACM Symp. Interactive 3D Graphics,* pp. 107-114, 1997.
[2] C. Shaw and M. Green, "THRED: A Two-Handed Design System," *Multimedia Systems J.,* vol. 5, pp. 126-139, 1997.
[3] J. Wong, R.W.H. Lau, and L. Ma, "Virtual 3D Sculpting," *J. Visualization and Computer Animation,* vol. 11, no. 3, pp. 155-166, July 2000.
[4] F. Li, R.W.H. Lau, and F. Ng, "VSculpt: A Distributed Virtual Sculpting Environment for Collaborative Design," *IEEE Trans. Multimedia,* vol. 5, no. 7, pp. 570-580, 2003.
[5] A. Chan, R.W.H. Lau, and B. Ng, "Motion Prediction for Caching and Prefetching in Mouse-Driven DVE Navigation," *ACM Trans. Internet Technology,* vol. 5, no. 1, pp. 1-22, 2005.
[6] I. Vaghi, C. Greenhalgh, and S. Benford, "Coping with Inconsistency due to Network Delays in Collaborative Virtual Environments," *Proc. ACM Symp. Virtual Reality Software and Technology (VRST '99),* pp. 42-49, 1999.
[7] T. Sheridan and W. Ferrell, "Remote Manipulative Control with Transmission Delay," *IEEE Trans. Human Factors in Electronics,* vol. 4, no. 1, pp. 25-29, 1963.
[8] M. Mauve, J. Vogel, V. Hilt, and W. Effelsberg, "Local-Lag and Timewarp: Providing Consistency for Replicated Continuous Applications," *IEEE Trans. Multimedia,* vol. 6, no. 1, pp. 47-57, 2004.
[9] *IEEE Std. 1278, Distributed Interactive Simulation—Application Protocols,* IEEE, 1998.
[10] S. Singhal and M. Zyda, *Networked Virtual Environments: Design and Implementation.* ACM Press, 1999.
[11] S. Singhal and D. Cheriton, "Exploiting Position History for Efficient Remote Rendering in Networked Virtual Reality," *Presence,* vol. 4, no. 2, pp. 169-193, 1995.
[12] J. Chim, M. Green, R.W.H. Lau, A. Si, and H. Leong, "On Caching and Prefetching of Virtual Objects in Distributed Virtual Environments," *Proc. ACM Multimedia,* pp. 171-180, Sept. 1998.
[13] R. Azuma and G. Bishop, "A Frequency-Domain Analysis of Head-Motion Prediction," *Proc. ACM 22nd Ann. Conf. Computer Graphics and Interactive Techniques (Siggraph '95),* pp. 401-408, 1995.
[14] J. Liang, C. Shaw, and M. Green, "On Temporal-Spatial Realism in the Virtual Reality Environment," *Proc. ACM Symp. User Interface Software and Technology (UIST '91),* pp. 19-25, 1991.
[15] T. Capin, I. Pandzic, N. Magnenat-Thalmann, and D. Thalmann, "A Dead-Reckoning Algorithm for Virtual Human Figures," *Proc. IEEE Virtual Reality Ann. Int'l Symp. (VRAIS '97),* pp. 161-169, 1997.
[16] J. Wu and M. Ouhyoung, "On Latency Compensation and Its Effects on Head-Motion Trajectories in Virtual Environments," *The Visual Computer,* vol. 16, no. 2, pp. 79-90, 2000.
[17] M. Isard and A. Blake, "CONDENSATION—Conditional Density Propagation for Visual Tracking," *Int'l J. Computer Vision,* vol. 29, no. 1, pp. 5-28, 1998.

[18] J. LaViola, Jr., "An Experimental Comparing Double Exponential Smoothing and Kalman Filter-Based Predictive Tracking Algorithms," *Proc. IEEE Virtual Reality Conf. (VR '03)*, pp. 283-284, 2003.
[19] *VirtualHand Software Library Reference Manual.* Virtual Technologies, 1998.
[20] E. Biryukova and V. Yourovskaya, "A Model of Human Hand Dynamics," *Advances in the Biomechanics of the Hand and Wrist.* Plenum Press, pp. 107-122, 1994.
[21] R. Cailliet, *Hand Pain and Impairment,* third ed. F.A. Davis, 1982.
[22] I. Kapandji, *The Physiology of the Joints,* vol. 1. Churchill Livingstone, 1982.
[23] J. Lee and T. Kunii, "Model-Based Analysis of Hand Posture," *IEEE Computer Graphics and Applications,* vol. 15, no. 8, pp. 77-86, 1995.
[24] H. Rijpkema and M. Girard, "Computer Animation of Knowledge-Based Human Grasping," *Proc. ACM 18th Ann. Conf. Computer Graphics and Interactive Techniques (Siggraph '91),* pp. 339-348, 1991.
[25] T. Heap and D. Hogg, "Towards 3D Hand Tracking Using a Deformable Model," *Proc. IEEE Int'l Conf. Automatic Face and Gesture Recognition (FG '96),* pp. 140-145, 1996.
[26] H. Zhou and T. Huang, "Tracking Articulated Hand Motion with Eigen Dynamics Analysis," *Proc. Eighth IEEE Int'l Conf. Computer Vision (ICCV '01),* pp. 1102-1109, 2003.
[27] G. Welch and G. Bishop, *An Introduction to the Kalman Filter,* http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html.
[28] W. Wen and H. Durrant-Whyte, "Model Based Active Object Localization Using Multiple Sensors," *Proc. IEEE/RSJ Int'l Workshop Intelligent Robots and Systems (IROS '91),* pp. 1448-1452, 1991.
[29] H. Durrant-Whyte, *Integration, Coordination and Control of Multi-Sensor Robot Systems.* Kluwer Academic, 1988.
[30] J. Geeter, H. Brussel, and J. Schutter, "A Smoothly Constrained Kalman Filter," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 11, pp. 1171-1177, 1997.
[31] D. Simon and T. Chia, "Kalman Filtering with State Equality Constraints," *IEEE Trans. Aerospace and Electronic Systems,* vol. 39, no. 1, pp. 128-136, 2002.
[32] D. Simon and D. Simon, "Aircraft Turbofan Engine Health Estimation Using Constrained Kalman Filtering," *Proc. ASME Turbo Expo,* 2003.
[33] J. Ohlenburg, "Improving Collision Detection in Distributed Virtual Environments by Adaptive Collision Prediction Tracking," *Proc. IEEE Virtual Reality Conf. (VR '04),* pp. 83-90, 2004.
[34] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C++,* second ed. Cambridge Univ. Press, 2002.
[35] A. Oppenheim and A. Willsky, *Signal and Systems,* second ed. Prentice Hall, 1997.

Addison Chan received the BEng degree in computer engineering from the Hong Kong University of Science and Technology and the MPhil degree in computer science from the City University of Hong Kong. From 2004 to 2006, he was a senior research assistant at the City University of Hong Kong. He is now a PhD candidate at the University of Durham, United Kingdom. His research interests include motion prediction for distributed virtual environments.

Rynson W. H. Lau received the BSc degree (top, first-class honors) in computer systems engineering from the University of Kent in 1988 and the PhD degree in computer graphics from the University of Cambridge in 1992. He is currently a professor at the University of Durham. Prior to his current appointment, he taught at the City University of Hong Kong from 1998 to 2006 and at the Hong Kong Polytechnic University from 1993 to 1998. He serves on the editorial board of *Computer Animation and Virtual Worlds*. He has served as the guest editor of a number of journal special issues, including the *ACM Transactions on Internet Technology*, *IEEE Transactions on Multimedia*, *IEEE Transactions on Visualization and Computer Graphics*, *Presence*, and *IEEE Computer Graphics and Applications*. He has also served in the conference committee of a number of conferences, including as a program cochair of the ACM Symposium on Virtual Reality Software and Technology (VRST) 2004, the 16th International Conference on Artificial Reality and Telexistence (ICAT '06), and as a conference cochair of ACM VRST 2005 and the Computer Animation and Social Agents Conference (CASA) 2005. He is a senior member of the IEEE.

Lewis Li received the BSc degree in computer studies from the City University of Hong Kong. From 2003 to 2005, he was a research assistant at the same university. Since 2005, he has been a research student there. His research interests include computer graphics and distributed virtual environments.