Performance Animation from Low-dimensional Control Signals

Jinxiang Chai[*]
Carnegie Mellon University

Jessica K. Hodgins[†]
Carnegie Mellon University

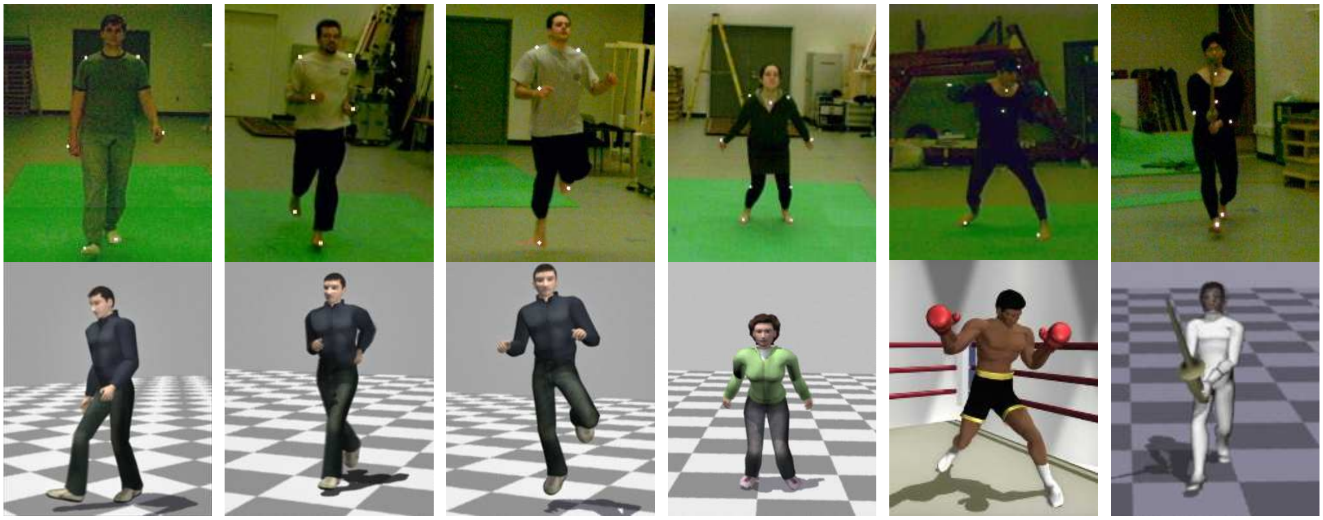


Figure 1: Users wearing a few retro-reflective markers control the full-body motion of avatars by acting out the motion in front of two synchronized cameras. From left to right: walking, running, hopping, jumping, boxing, and Kendo (Japanese sword art).

Abstract

This paper introduces an approach to performance animation that employs video cameras and a small set of retro-reflective markers to create a low-cost, easy-to-use system that might someday be practical for home use. The low-dimensional control signals from the user's performance are supplemented by a database of pre-recorded human motion. At run time, the system automatically learns a series of local models from a set of motion capture examples that are a close match to the marker locations captured by the cameras. These local models are then used to reconstruct the motion of the user as a full-body animation. We demonstrate the power of this approach with real-time control of six different behaviors using two video cameras and a small set of retro-reflective markers. We compare the resulting animation to animation from commercial motion capture equipment with a full set of markers.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—animation; I.3.6 [Computer Graphics]: Methodology and Techniques—interaction techniques

Keywords: performance animation, vision-based interface, motion capture data, dimensionality reduction, online control of human motion, local modeling, lazy learning

1 Introduction

The ability to accurately reconstruct a user's motion in real time would allow the intuitive control of characters in computer games, the control of avatars for virtual reality or electronically mediated communication, and the rapid prototyping of character animations. This problem has been solved by commercially available motion capture equipment, but this solution is far too expensive for common use. It is also cumbersome, requiring the user to wear 40–50 carefully positioned retro-reflective markers and skin-tight clothing, 15 magnetic motion sensors, or a full exoskeleton. In this paper, we present a different approach to solving this problem: reconstructing the user's motion from the positions of a small set of markers captured with two video cameras. This necessarily incomplete information about the user's motion is supplemented by a database of pre-recorded human motion. The results are visually comparable in quality to those obtained from a commercial motion capture system with a full set of markers provided that similar behaviors are found in the pre-recorded database. The cost is low because only two synchronized video cameras are required. The system is easy to set up and relatively non-intrusive because the user is required to wear only a small set of markers (6–9 for the experiments reported here) and can wear street clothes (figure 1).

Providing accurate control of full-body motion based on a small set of markers is difficult because the information about the user's motion is incomplete. The control signals, or input, are the locations of the markers. This information is quite low-dimensional (less than twenty degrees of freedom) compared to a typical human model (approximately sixty degrees of freedom). The control signals cannot be used directly to create a full-body animation because they will be consistent with many disparate solutions for the character's pose. We eliminate this ambiguity by building a local, low-dimensional model of the user's motion on the fly from a motion database of pre-recorded, high-quality motion. The key insight

[*] e-mail: jchai@cs.cmu.edu
[†] e-mail: jkh@cs.cmu.edu

behind this approach is that natural human motion is highly coordinated and the movement of the degrees of freedom are not independent. As a result, the local models can be quite low-dimensional while accurately representing the motion.

We demonstrate the power and flexibility of this approach by having users control six behaviors in real time without significant latency: walking, running, hopping, jumping, boxing, and Kendo (Japanese sword art) (figure 1). The reconstructed motion is based on a single large human motion database. Our experiments indicate that this approach scales well with the size and heterogeneity of the database and is robust to variations in kinematics between users. The resulting animation also captures the individual style of the user's motion through spatial-temporal interpolation of the data. The database, however, must contain the basic actions required for the application domain.

We assess the quality of the reconstructed motion by comparing against ground truth data simultaneously captured with a full marker set in a commercial motion capture system. We also compare alternative techniques for the constituent elements of the system: dimensionality reduction for the human motions and the local models used for synthesis.

2 Background

In the next two sections, we discuss related work in control interfaces for human motion. Because we use a motion capture database in our system, we also briefly review research utilizing motion capture data for animation.

2.1 Control Interfaces for Human Motion

Computer and video games offer a variety of interfaces for controlling human motion such as mice, joysticks, and button or key presses. These interfaces can provide direct control over only a limited number of degrees of freedom and the details of the motion must be computed automatically.

Control at a more detailed level can be provided if the user acts out the desired motion using his or her body in a performance animation system. Such systems have proved quite successful for television characters who respond in real time to the actions of human actors [Shin et al. 2001]. Active optical, passive optical, magnetic, and exoskeleton-based motion capture systems all now have the ability to perform real-time capture of a typical human model. While very effective for television, trade shows, and other performance venues, the time required to suit up the user (commonly referred to as the time to don and doff) prevents their use in location-based entertainment and other applications where the system will be used by many people. These systems are also not appropriate for home use because of cost.

Systems that can extract meaningful information about the user's motion from only a few sensors are appealing because they dramatically reduce don and doff time. The infrared sensor-based games (Mocap Boxing and Police 911 by Konami [2001], for example) are a successful commercial example of this class of interface. These systems track the motion of the hands and render the hands, a first person view of part of the upper body, and the effect of the boxing gloves or gun on the environment. Sony's EyeToy [2003] is a vision-based system that requires no markers but is capable of extracting the 2D locations of simple gestures such as a punch or a wave. An earlier system that implemented a number of different interfaces for computer games was presented in the research community [Freeman et al. 1998]. None of these systems attempted to fully capture or animate the user's motion but instead focused on recognizing or locating a limited set of simple actions and showing their effect on the scene.

Researchers have also explored techniques for using a few sensors to reconstruct full-body motion. Badler and colleagues [1993] used four magnetic sensors and real-time inverse kinematics algorithms to control a standing figure in a virtual environment. Their system adopted a heuristic approach to handling the kinematic redundancy while we use a data-driven approach. Semwal and colleagues [1998] provided an analytic solution to the inverse kinematics algorithm based on eight magnetic sensors. Yin and Pai [2003] used a foot pressure sensor to develop an interface that extracts full-body motion from a database. Their system was successful at reproducing full body motion for a limited range of behaviors with a latency of one second. However, foot pressure patterns may be insufficient to accurately reconstruct a motion with detailed upper body motions.

The interface problem becomes more tractable if the motion is performed in several layers so that not all degrees of freedom need to be animated simultaneously. Oore and colleagues [2002] used a pair of six degree-of-freedom tracking devices to provide interactive control over the stepping and walking motions of a character. Dontcheva and colleagues [2003] also used layering in their puppeteering system. Recently, Grochow and colleagues [2004] applied a global nonlinear dimensionality reduction technique, Gaussian Process Latent Variable Model (GPLVM) [Lawrence 2004], to human motion data. They combined the learned probabilistic model with kinematic constraints to create a character that could be interactively posed with a mouse. Their global, nonlinear dimensionality reduction technique works well with a small homogenous data set, but might not be suitable for a large heterogeneous motion data set.

Lee and colleagues [2002] built a vision-based interface to transform noisy silhouette data obtained from a single video camera to full-body movements with a latency of about two seconds. Their approach searched a motion graph using Hu moments computed from the input silhouettes. Ren and colleagues [2004] used silhouettes from a three-camera system and a motion capture database to select among a large set of simple features for those most suited to identifying the yaw orientation and pose of the user from three silhouettes. Their application was domain specific in that the features were selected using training data of a specific behavior, swing dancing. Their approach produced high-quality motion that approximated that of the user with 0.8 second latency. Neither of these systems gives the user precise control over the character's motion because a motion graph-based approach cannot modify the existing motions in the database. In addition to eliminating the synthesis latency in these systems, our approach uses a series of local models to interpolate the motions in the database for more accurate control.

Another alternative is to employ vision-based tracking to capture the movement of the user. However, that technique has not yet been successfully used to accurately reconstruct complex full-body human motion in real time [Howe et al. 1999; Brand 1999; Sidenbladh et al. 2002; Cheung et al. 2003]. The work of Howe [1999] and Sidenbladh [2002] and their colleagues is perhaps most closely related to that presented in here in that they also use motion capture data. Howe and colleagues [1999] published one of the earliest papers on using global PCA to reduce the dimensionality of human motion. They incorporated the reduced model into a probabilistic Bayesian framework to constrain the search of human motion. Sidenbladh and colleagues [2002] reduced the dimensionality of the database using global PCA and then constrained the set of allow-able trajectories within a high-dimensional state space. Our goals are different, however, because we focus on high-quality animation and real-time control.

2.2 Animation with Motion Capture Data

A number of researchers have developed techniques for synthesizing animated sequences from motion capture data. Three distinct approaches have been used: constructing models of human motion [Li et al. 2002; Brand and Hertzmann 2000], reordering motion clips employing a motion graph [Arikan and Forsyth 2002; Kovar et al. 2002; Lee et al. 2002; Pullen and Bregler 2002; Arikan et al. 2003] and interpolating motion to create new sequences [Guo and Roberge 1996; Wiley and Hahn 1997; Rose et al. 1998; Kovar and Gleicher 2004]. In our work, we construct a graph of nearest neighbors for fast search of the motion examples that are close to the current control signals and use it to build a local linear model of the motion for interpolation. We therefore discuss motion graphs and motion interpolation in more detail.

Motion graphs create an animation by cutting pieces from a motion database and reassembling them to form a new motion. Because the motion that is selected is not modified, it retains the subtle details of the original motion data but the synthesized motions are restricted to those in the motion capture database. For example, a motion graph cannot be used to synthesize a walking motion for a slope of a particular angle unless the database included data for that slope.

Interpolation addresses this problem by allowing synthesis of motion variations that are not in the database. Both Guo and Roberge [1996] and Wiley and Hahn [1997] produced modified motions using linear interpolation. Rose and colleagues [1998] used radial basis functions to interpolate motions located irregularly in the parameter space. Generally, this approach requires segmenting the motion data into structurally similar sequences, building a temporal correspondence among them, and annotating each with a small set of meaningful, high-level control knobs. Given new values of the control parameters, the sequences can be interpolated to compute a motion that matches the specified parameters. Recently, Kovar and Gleicher [2004] introduced a method for automatically locating logically similar motion segments in a data set and using them to construct parameterized motions. These algorithms produce high-quality motion for new parameter values that are within the space of the interpolated examples.

Like interpolation, our approach can generate spatial/temporal variations that are not in the database. Because interpolation occurs only in the local region of the current control signals with our approach, it does not require that the motions be structurally similar at a high level.

3 Overview

Our system transforms low-dimensional control signals obtained from only a few markers into full-body animation by constructing a series of local models from a database of human motion at run time and using those models to fill in probable values for the information about the user's motion not captured by the markers.

We first perform a series of off-line captures to create a large and heterogeneous human motion database (about 1 hour) using a Vicon optical motion capture system with twelve 120 Hz Mx-40 cameras [Vicon Systems 2004]. The database contains ten full-body behaviors: boxing (71597 frames), walking (105963 frames), running (18523 frames), jumping (40303 frames), hopping (18952 frames), locomotion transitions (36251 frames), dancing (18002 frames), basketball (12484 frames), climbing on playground equipment playground (51947 frames), and Kendo (59600 frames). We used a marker set with 41 markers, an adaptation of the Helen Hayes marker set. We added four extra markers on the bamboo sword for Kendo. For Kendo, boxing, and locomotion, the subjects were instructed to repeat each action at least five times in order to capture variations in performance and to ensure that the local model was constructed from sufficiently similar data.

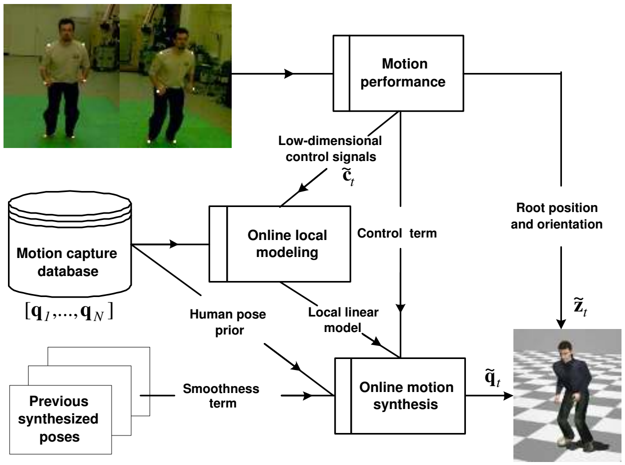


Figure 2: System overview.

Each motion in the database has a skeleton that includes the subject's limb lengths and joint range of motion computed automatically from calibration captures. Each motion sequence contains trajectories for the absolute position and orientation of the root node (pelvis) as well as relative joint angles of 18 joints. These joints are head, thorax, upper neck, lower neck, upper back, lower back, and left and right humerus, radius, wrist, femur tibia, and metatarsal.

We denote the set of motion capture data in the database as $\{\mathbf{q}_n | n = 1,...,N\}$, where $\mathbf{q}_n$ is the joint angle representation of a specific pose in the database. The control signals obtained from the interface at time t are represented by the locations of a small set of retro-reflective markers worn by the user, denoted as $\tilde{\mathbf{c}}_t$. We always place markers on the torso of the user so that the absolute position and orientation of the user, denoted as $\tilde{\mathbf{z}}_t$, can be directly obtained from the control signals, $\tilde{\mathbf{c}}_t$. The online motion control problem is to synthesize the current human body pose, $\tilde{\mathbf{q}}_t$, in real time based on the current low-dimensional control signals, $\tilde{\mathbf{c}}_t$, obtained from the vision-based interface, motion capture data in the database, $\{\mathbf{q}_1,...,\mathbf{q}_N\}$, and the synthesized poses in the previous frames, $[\tilde{\mathbf{q}}_1,...,\tilde{\mathbf{q}}_{t-1}]$.

The system contains three major components (figure 2):

Motion performance. The user wears a small set of retro-reflective markers to perform a motion in front of two synchronized video cameras. The system automatically extracts the locations of the markers, $[\tilde{\mathbf{c}}_1,...,\tilde{\mathbf{c}}_t]$, and the absolute position and orientation of the motion, $[\tilde{\mathbf{z}}_1,...,\tilde{\mathbf{z}}_t]$, from the video streams in real time. The trajectories of the markers specify the desired trajectories of certain points on the animated character.

Online local modeling. To synthesize the current pose $\tilde{\mathbf{q}}_t$, we first search the motion capture database for examples that are close to the current control signals $\tilde{\mathbf{c}}_t$ and the synthesized poses in the previous frames $[\tilde{\mathbf{q}}_1,...,\tilde{\mathbf{q}}_{t-1}]$. Because the runtime computational cost depends on the efficiency of the nearest neighbor search process,

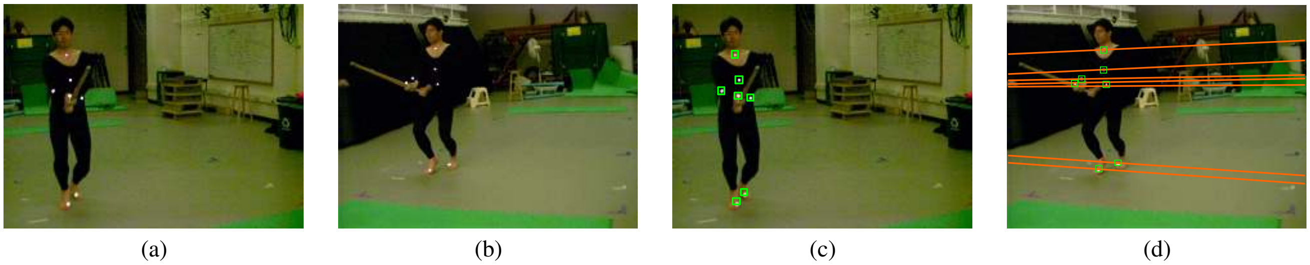


Figure 3: Marker detection and correspondence: a user acts out the motion in front of two synchronized video cameras. (a) and (b) The images from the left and right cameras respectively. (c) The detected marker positions in the left image. (d) The detected marker locations in the right image and the epipolar lines of the markers that were detected in the left image. For each marker in the left image, the matching marker in the right image should be located on its corresponding epipolar line.

we introduce a data structure, a *neighbor graph*, and an algorithm that accelerates the nearest neighbor search by utilizing the temporal coherence of the control signals. The nearest neighbors, denoted as $\{\mathbf{q}_{t_k}|k = 1,...,K\}$, are then used to learn a local linear model of human pose near the current control signal.

Online motion synthesis. The local linear model is used to reconstruct the user's pose, $\tilde{\mathbf{q}}_t$, based on the control signals obtained from the vision-based interface, $\tilde{\mathbf{c}}_t$, a human pose prior term that ensures that the synthesized motion satisfies the probabilistic distribution of human motions in the database, and a smoothness term that minimizes velocity changes in the synthesized motion.

We describe these components in more detail in the next three sections.

4 Motion Performance

In this section, we describe a simple but robust vision algorithm to extract the locations of the retro-reflective markers from two synchronized video cameras. We then describe the subject calibration process that makes the vision-based interface robust to users of different sizes and to variations in marker placement.

4.1 Motion Analysis

Our system employs two Pulnix video cameras (TMC-6700-CL), which have 640×480 image resolution and a frame rate of 60 fps, as input devices (figure 3). We use the method described by Zhang [1999] to determine the intrinsic parameters of the cameras and the relative transformation between them. To illuminate the retro-reflective markers, we placed a photography light near each camera. To make detection more robust, we apply background subtraction techniques based on the statistics of the images and search just the foreground pixels. The system computes the mean and standard deviation of each background pixel in each color channel for a sequence of frames where the user is not present. During online tracking, pixels that differ in at least one color channel by more than a user-defined threshold from the background distribution are labelled as foreground pixels. We then perform a morphological filter (dilation) on the foreground pixels to enlarge the region and to ensure that markers on the boundary are included.

After the system locates the markers in each image, we establish a correspondence between the markers using epipolar geometry and color similarity constraints. The epipolar geometry constraint reduces the search space for the corresponding marker to only those markers that lie on a single epipolar line [Xu and Zhang 1996]. Figure 3 (c) and (d) shows the marker locations and the corresponding epipolar lines. Because the number of markers is small, the epipolar line constraint is generally sufficient to find the correct point correspondences. Occasionally, more than one point in the second camera might satisfy the epipolar geometry constraint from the first camera, but temporal coherence can be used to reduce this ambiguity. Given a point correspondence between the two calibrated cameras, we compute the marker's 3D location by finding the intersection of rays cast from the 2D markers in both cameras. Occlusion might prevent a marker from being seen by both cameras. To address this problem, we also include the 2D locations of the markers seen by only one camera in the control signals.

Once the system labels the markers in the first frame, it can label the markers in each subsequent frame by matching the current marker locations to the marker set found for the most recent synthesized motion (described in section 6). Because the synthesized motion is reconstructed from the motion capture database, it includes any occluded markers. Therefore, the system can automatically handle missing markers by labelling a marker that was occluded once it can be seen again. The marker tracking and labelling system runs in real time and did not require manual intervention for the experiments reported here. The marker labelling appears to be robust to variations in body type and occlusion.

4.2 Subject Calibration

Subject calibration ensures that the vision-based interface is robust to users of different sizes and to variations in marker placement. Subject calibration consists of two steps: skeleton calibration and marker calibration.

Skeleton calibration. Skeleton calibration estimates the user's skeleton model from the 3D locations of a few markers obtained from the vision-based interface. In our experiments, we place markers on the left hand, left elbow, left foot, left knee, and each shoulder. Two markers are placed on the front of the waist. We instruct the user to assume a "T" Pose and capture the 3D locations of the markers. The locations of this small set of markers are not sufficient to compute a detailed skeleton model; therefore we use these measured 3D marker locations to interpolate a database of detailed skeleton models from a variety of subjects. We then place markers on the right limb and model the right side of the skeleton model in a similar fashion. Each user need perform the skeleton calibration step only once.

Marker calibration. The goal of marker calibration is to determine the location of the control markers used in the interface relative to the inboard joint. For example, the location of the hand marker relative to the coordinate system of the wrist. We first measure the location of the markers for the "T" pose in the world coordinate frame. Given this information and user's skeleton model, we compute the 3D positions of the inboard joints relative to the world coordinate frame in the "T" pose via forward kinematics. The location of markers in the coordinate system of the inboard joint can then be found by computing the difference between the location of the inboard joint and the marker relative to the world coordinate frame. This calibration step must be repeated if the marker placement changes. The system can handle extra markers if the marker calibration step is repeated for the new marker set.

We preprocess the motion capture database by computing the 3D location of the control markers $\mathbf{c}_n$ corresponding to the motion capture data for each frame in the database $\mathbf{q}_n$:

$$\mathbf{c}_n = \mathbf{f}(\mathbf{q}_n; \tilde{\mathbf{s}}, \tilde{\mathbf{v}}_l, \mathbf{z}_0) \tag{1}$$

where the function $\mathbf{f}$ is the forward kinematics function that computes the marker positions from the joint angles of the current frames, $\mathbf{q}_n$, given the user's skeleton model, $\tilde{\mathbf{s}}$, and the locations of the control markers, $\tilde{\mathbf{v}}_l$, relative to the inboard joint. We choose the default root position and orientation, $\mathbf{z}_0$, as $\mathbf{0}_{6\times 1}$.

5 Online Local Modeling

The motion synthesis problem is difficult because the positions of the small set of markers worn by the user do not adequately constrain the joint angles of a full-body human model. The key idea of our approach is to use a *lazy learning* algorithm to automatically construct a series of simple local models that sufficiently constrain the solution space. The lazy learning algorithm postpones all computation until an explicit request for information (e.g. prediction or local modeling) is received [Aha 1997].

The motions to be synthesized, $[\tilde{\mathbf{q}}_1, ..., \tilde{\mathbf{q}}_t]$, form a nonlinear manifold embedded in a high-dimensional space. At run time, the system automatically learns a series of low-dimensional linear models to approximate this high-dimensional manifold. To build a local model, we search the motion capture database for examples that are close to the current marker locations and recently synthesized poses. These examples are then used as training data to learn a simple linear model via Principal Component Analysis (PCA) [Bishop 1996]. A new local model is created to synthesize each pose.

The system relies on the current control signals from the interface $\tilde{\mathbf{c}}_t$ and the synthesized poses in the previous two frames $[\tilde{\mathbf{q}}_{t-1}, \tilde{\mathbf{q}}_{t-2}]$ to find the K closest examples $\{\mathbf{q}_{t_k} | k = 1, ..., K\}$ for the current pose $\tilde{\mathbf{q}}_t$. The query metric, for each example $\mathbf{q}_n$ in the database, is

$$\alpha \|\mathbf{c}_n - \mathtt{T}(\tilde{\mathbf{z}}_t, \mathbf{z}_0)\tilde{\mathbf{c}}_t\|^2 + (1-\alpha)\|\mathbf{q}_n - 2\tilde{\mathbf{q}}_{t-1} + \tilde{\mathbf{q}}_{t-2}\|^2 \tag{2}$$

where $\|\cdot\|$ denotes a Euclidean norm. $\mathtt{T}(\tilde{\mathbf{z}}_t, \mathbf{z}_0)$ is the transformation matrix that aligns the current root position and orientation of the control markers, $\tilde{\mathbf{z}}_t$, with the default root position and orientation of the motion capture data, $\mathbf{z}_0$. The first term evaluates how well the control parameters associated with the example pose match the control signals from the interface. The second term evaluates the continuity of the motion that would result if $\mathbf{q}_n$ were to be placed after $\tilde{\mathbf{q}}_{t-1}$ and $\tilde{\mathbf{q}}_{t-2}$. In our experiments, α is set to 0.8.

After subtracting the mean, $\mathbf{p}_t$, of the K closest examples in the local region we apply PCA to the covariance matrix of these K examples, $\{\mathbf{q}_{t_k} | k = 1, ..., K\}$, in the joint angle space. We obtain a linear model for the current pose $\tilde{\mathbf{q}}_t$:

$$\tilde{\mathbf{q}}_t = \mathbf{p}_t + U_t \cdot \mathbf{w}_t \tag{3}$$

where U_t is constructed from the eigenvectors corresponding to the largest eigenvalues of the covariance matrix of the local examples. $\mathbf{w}_t$ is a B-dimensional vector, which is a low-dimensional representation of the current pose $\tilde{\mathbf{q}}_t$. The number of nearest neighbors, K, and the dimension of the space, B, are selected locally and adjusted for the current query point by a leave-one-out cross-validation [Stone 1974; Atkeson et al. 1997a]. More specifically, each time a prediction is required, a set of local models are identified, each with a different dimension, B, and each including a different number of neighbors, K. The generalization ability of each model is then assessed through a local leave-one-out cross-validation procedure and the best model is selected for reconstruction. The dimension of the new space is usually less than seven in the experiments reported here and is therefore much lower than the dimension of the original space.

The local model avoids the problem of finding an appropriate structure for a global model, which would necessarily be high-dimensional and nonlinear. Instead, we assume that a series of low-dimensional, local models are sufficient to approximate the global high-dimensional manifold. Our local models do not require any parameter tuning because all parameters are automatically selected by cross-validation.

5.1 Fast Online K-nearest Neighbor Search

The main drawback of the local model is the time required to find the nearest neighbors. Because the number of queries will be large, the computational cost can be significantly reduced by preprocessing the motion capture database to create a data structure that allows fast nearest-neighbor search. This section introduces a *neighbor graph* and an algorithm that accelerates the runtime query by utilizing temporal coherence.

We first build a neighbor graph, each node of which represents a pose in the human motion database $\mathbf{q}_n$. We connect the i-th node and j-th node if and only if they satisfy:

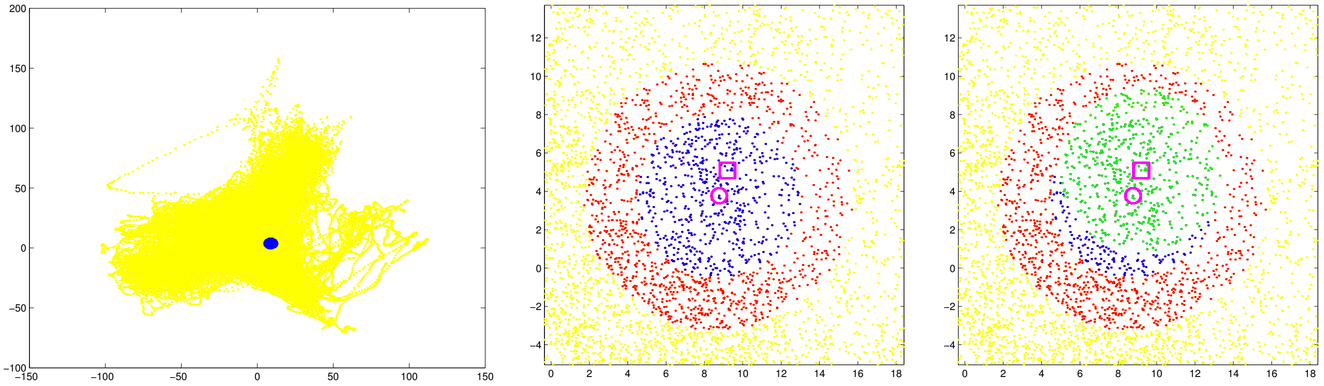
$$\|\mathbf{q}_i - \mathbf{q}_j\|_{L_1} < max\{\frac{f_m \Delta d}{f_c}, \varepsilon\} \tag{4}$$

where $\|\cdot\|_{L_1}$ denotes the L_1 distance. Δd represents the largest L_1 distance between two consecutive poses in the database. f_m and f_c are the camera frame rates used for the motion capture and the control interface respectively. ε is a specified search radius for nearest neighbors. In our experiments, Δd is 1.75 degrees per joint angle and ε is 3 degrees per joint angle.

Let $\{\mathbf{q}_{t-1_k} | k = 1, ..., K\}$ be the nearest neighbors of the previous frame. The nearest neighbors of the query point $\{\mathbf{q}_{t_k} | k = 1, ..., K\}$ can be approximately found by searching $\{\mathbf{q}_{t-1_k} | k = 1, ..., K\}$ and their neighbors in the neighbor graph. A 2D example of our nearest

Database	size	mean	min.	max.	error
Boxing	71597	752.31	36	8968	0.025
Hetergeneous	433622	710.1	31	9101	0.014

Table 1: Performance of the nearest neighbor search algorithm, where mean, min. and max. are the mean number of nodes searched, the minimal number of nodes searched, and the maximum number of nodes searched. All three numbers are significantly smaller than the size of the database. The error (degree per joint) is measured by computing the L_2 distance between the pose synthesized by the examples found using exhaustive search and the pose synthesized by the examples found using our nearest neighbor search.



(a) (b) (c)

Figure 4: A 2D example of the fast nearest neighbor search using two dimensions of the neighbor graph for the boxing database: (a) the data points in the database after we project them into the 2D eigen-space; (b) the magenta circle represents the previous pose, $\tilde{\mathbf{q}}_{t-1}$, and the magenta square represents the current pose, $\tilde{\mathbf{q}}_t$. At run time, we use the neighbors of the previous frame (blue points) $\{\mathbf{q}_{t-1_k}|k=1,...,K\}$ and a precomputed neighbor graph to find the neighbors of $\{\mathbf{q}_{t-1_k}|k=1,...,K\}$ in the neighbor graph (red points). The algorithm then searches only the red and blue points to find the nearest neighbors of the current query point. (c) the green points are the nearest neighbors found using this algorithm.

neighbor search algorithm is shown in figure 4. The neighbor graph significantly reduces the computational time of the search by just examining the data points in the neighboring nodes of the last query point. Table 1 shows that the performance scales well with the size and heterogeneity of the database.

6 Online Motion Synthesis

The system automatically extracts the absolute root position and orientation, $[\tilde{\mathbf{z}}_1,...,\tilde{\mathbf{z}}_t]$, of the reconstructed motions directly from the vision-based interface. This section focuses on how to reconstruct the joint angle values, $[\tilde{\mathbf{q}}_1,...,\tilde{\mathbf{q}}_t]$, from the low-dimensional control signals, $[\tilde{\mathbf{c}}_1,...,\tilde{\mathbf{c}}_t]$. At run time, the system automatically transforms the low-dimensional control signals from the vision-based interface to full-body human motions frame by frame using the learned local linear model (Equation 3), the training examples in the local region, $\{\mathbf{q}_{t_k}|k=1,...,K\}$, the previous synthesized poses, $[\tilde{\mathbf{q}}_1,...,\tilde{\mathbf{q}}_{t-1}]$, and the current low-dimensional control signals, $\tilde{\mathbf{c}}_t$. We use the local linear model (Equation 3) as a hard constraint and optimize the current pose $\tilde{\mathbf{q}}_t$ in the low-dimensional space $\mathbf{w}_t$ using a set of three energy terms: prior, control, and smoothness:

The prior term, E_{prior}, measures the a-priori likelihood of the current pose using the knowledge embedded in the motion capture database. The prior term is used to constrain the reconstructed pose to satisfy the probabilistic distribution determined by the training examples $\{\mathbf{q}_{t_k}|k=1,...,K\}$ in the local region. We assume the poses in the local region are a multivariate normal distribution, and the pose prior term maximizes

$$P(\tilde{\mathbf{q}}_t|\mathbf{q}_{t_1},...,\mathbf{q}_{t_K}) = \frac{\mathtt{exp}(-\frac{1}{2}(\tilde{\mathbf{q}}_t-\mathbf{p}_t)^T{\Lambda_t}^{-1}(\tilde{\mathbf{q}}_t-\mathbf{p}_t))}{(2\pi)^{\frac{d}{2}}|\Lambda_t|^{\frac{1}{2}}} \qquad (5)$$

where d is the dimension of $\tilde{\mathbf{q}}_t$. The vector $\mathbf{p}_t$ and the matrix Λ_t are the mean vector and covariance matrix of the data samples $\{\mathbf{q}_{t_k}|k=1,...,K\}$ in the local region. $|\Lambda_t|$ is the determinant of the covariance matrix Λ_t. We minimize the negative log of $P(\tilde{\mathbf{q}}_t|\mathbf{q}_{t_1},...,\mathbf{q}_{t_K})$, yielding the energy formulation

$$E_{prior} = (\tilde{\mathbf{q}}_t-\mathbf{p}_t)^T{\Lambda_t}^{-1}(\tilde{\mathbf{q}}_t-\mathbf{p}_t) \qquad (6)$$

The control term, $E_{control}$, measures the deviation of the marker locations in the reconstructed motion from the control inputs obtained from the vision-based interface:

$$E_{control} = \|\mathbf{f}(\tilde{\mathbf{q}}_t;\tilde{\mathbf{s}},\tilde{\mathbf{v}}_l,\tilde{\mathbf{z}}_t)-\tilde{\mathbf{c}}_t\|^2 \qquad (7)$$

where the function $\mathbf{f}$ is the forward kinematics function (Equation 1). Generating the animated sequence from only this constraint in the original joint angle space is similar to performing per-frame inverse kinematics as was done by Badler and Yamane and their colleagues [Badler et al. 1993; Yamane and Nakamura 2003]. If markers are visible to only one camera, we use the intrinsic and extrinsic parameters of that camera to project the 3D locations to 2D locations in the camera's image plane and then minimize the difference between the projected 2D locations and the 2D marker locations from the single camera.

The smoothness term, $E_{smoothness}$, measures the smoothness of the synthesized motion if $\tilde{\mathbf{q}}_t$ were placed after $[\tilde{\mathbf{q}}_1,...,\tilde{\mathbf{q}}_{t-1}]$. We assume that the pose at time t depends only the poses at time t-1 and t-2, and the smoothness term is

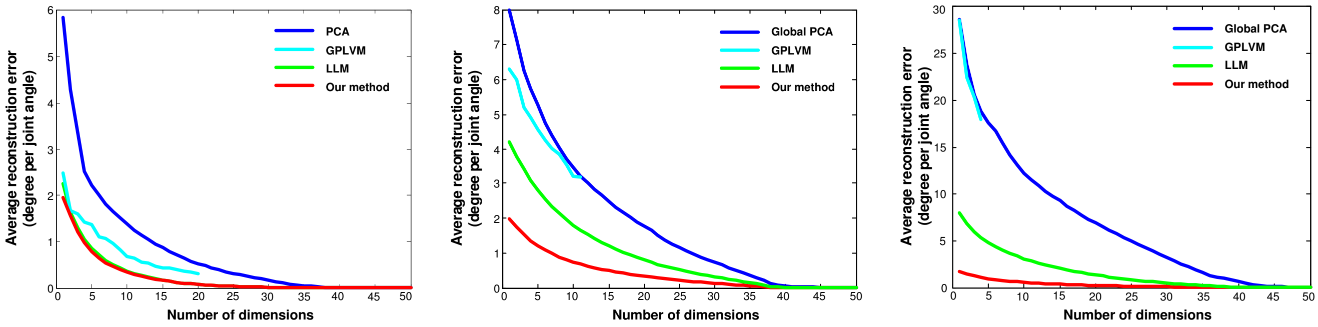
$$E_{smoothness} = \|\tilde{\mathbf{q}}_t-2\tilde{\mathbf{q}}_{t-1}+\tilde{\mathbf{q}}_{t-2}\|^2 \qquad (8)$$

where $\tilde{\mathbf{q}}_{t-1}$ and $\tilde{\mathbf{q}}_{t-2}$ are the synthesized poses in the previous two frames.

Combining Equations 6, 7 and 8 and substituting $\tilde{\mathbf{q}}_t$ using the local model (Equation 3), the complete energy function for motion synthesis is

$$\begin{aligned} arg\,min_{\mathbf{w}_t}\ \mathbf{w}_t^T {U_t}^T{\Lambda_t}^{-1}U_t\mathbf{w}_t+\alpha\|\mathbf{f}(\mathbf{w}_t;U_t,\mathbf{p}_t,\tilde{\mathbf{s}},\tilde{\mathbf{v}}_l,\tilde{\mathbf{z}}_t)-\tilde{\mathbf{c}}_t\|^2 \\ +\beta\|U_t\mathbf{w}_t+\bar{\mathbf{p}}_t-2\tilde{\mathbf{q}}_{t-1}+\tilde{\mathbf{q}}_{t-2}\|^2 \end{aligned} \qquad (9)$$

We initialize the optimization with the closest example in the database and optimize using the Levenberg-Marquardt programming method [Bazaraa et al. 1993]. The solution converges rapidly because of a good starting point and a low-dimensional optimization space (generally less than seven). In our experiments, α is set to 0.8 and β is set to 0.2.



(a) small walking database (b) medium-size boxing database (c) large heterogeneous database

Figure 5: Comparison of four dimensionality reduction methods: each curve shows the average reconstruction error with increasing number of dimensions. We cannot compute the complete GPLVM error curves for the medium and large databases because of the computational cost.

7 Numerical Comparison

In this section, we compare alternative techniques for the constituent elements of our performance animation system: dimensionality reduction for the human poses and local models for synthesis.

7.1 Dimensionality Reduction

The performance of the system depends on the ability to represent human motion in a low-dimensional space. Without this low-dimensional representation, the mapping from the control signals to the motion database would be one to many. The low dimensional space also reduces the time required for optimization.

In this section, we compare the performance of our online dimensionality reduction algorithm with other dimensionality reduction methods. Previous work in dimensionality reduction can be divived into three categories: global principal component analysis (PCA) [Bishop 1996], nonlinear dimensionality reduction [Mardia et al. 1979; Scholkopf et al. 1999; Roweis and Saul 2000; Tenenbaum et al. 2000; Lawrence 2004], and mixtures of local linear models (LLM) [Fukunaga and Olsen 1971; Bregler and Omohundro 1995; Hinton et al. 1995]. To study the performance of these algorithms on human motion data, we constructed three human motion databases: a 6-second walking database (700 frames), a 10-minute boxing database (71597 frames), and a 1-hour heterogeneous database (433622 frames) that includes walking, running, jumping, hopping, dancing, basketball, boxing, Kendo, and climbing on playground equipment.

Figure 5 plots the performance of one algorithm from each of these classes as well as the approach described in this paper. Each curve shows the average reconstruction error per joint with increasing number of dimensions. The average reconstruction error is the L_2 distance between the original motion and the motion reconstructed from the low-dimensional space.

Principal Component Analysis (PCA). PCA finds a global linear subspace approximating the nonlinear manifold of the motion capture database. Global PCA is widely used because of its simplicity and computational efficiency [Bishop 1996]. The number of dimensions required increases dramatically as the size and heterogeneity of the human motion database increase. For the walking database, 14 dimensions are required to obtain a reconstruction error of less than one degree. For the boxing database, 27 dimensions are required and the large heterogeneous database requires at least 38 dimensions.

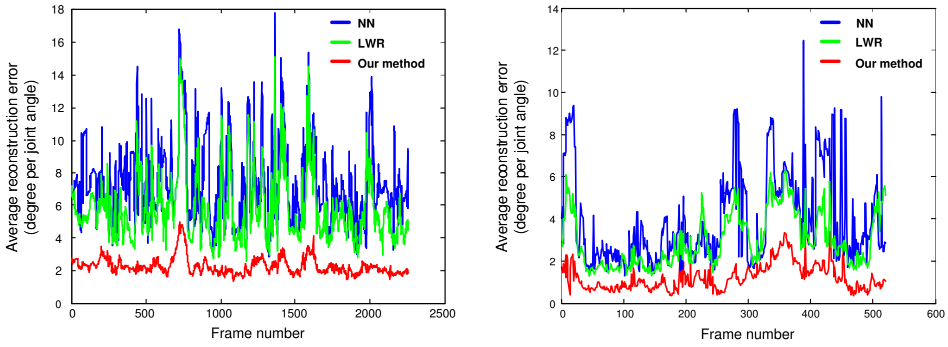
Nonlinear dimensionality reduction. Direct synthesis of human motions in the low-dimensional space requires an explicit mapping from the low-dimensional space to the original configuration space. Most previous research in nonlinear dimensionality reduction [Mardia et al. 1979; Scholkopf et al. 1999; Roweis and Saul 2000; Tenenbaum et al. 2000], therefore, is not appropriate for our work. An exception is the work of Lawrence [2004], where a Gaussian Process Latent Variable Model (GPLVM) was proposed to compute a global nonlinear map from the low-dimensional latent space to a high-dimensional space. Recently, Grochow and his colleagues [2004] applied GPLVM to human motion data to animate a character interactively. GPLVM works well for the small walking data set (figure 5(a)). The average reconstruction error in the 2D latent space is about 2.7 degrees per joint; however, like global PCA, the performance of the GPLVM deteriorates as the size and heterogeneity of the database increase.

Mixtures of local linear models (LLM). LLM first partitions the data space into disjoint regions with a clustering technique and then performs PCA for each cluster [Fukunaga and Olsen 1971; Bregler and Omohundro 1995; Hinton et al. 1995]. LLM performs well for all three databases (figure 5). For the walking database, six dimensions are required to obtain a reconstruction error of less than one degree. For the boxing database, eighteen dimensions are required and the large heterogeneous database requires at least 27 dimensions. The performance of LLM is dependent on the number and quality of the clusters.

Our method provides better performance for our application than the three other methods because it is a local method where the model is constructed at run-time using only the current set of nearest neighbors. LLM has better performance than either global PCA or GPLVM because LLM models clusters of the data. Our local modeling method is more efficient than LLM because we build the local model based on the neighbor set of the current query data rather than a precomputed cluster center. Our method scales well with the size and heterogeneity of the database; the algorithm creates similar error curves for the three testing databases. For the walking database, four dimensions are required to obtain a reconstruction error of less than one degree. For the boxing database, six dimensions are required and the large heterogeneous database requires at least five dimensions.

7.2 Online Motion Synthesis Using Local Models

The local model used for online motion synthesis is different from previous lazy learning methods such as locally weighted regres-



(a) Boxing

(b) Walking

Figure 6: Comparison of methods for synthesizing motions from low-dimensional continuous control signals. (a) Average errors for boxing motion: 7.67 degrees/joint per frame for nearest neighbor synthesis (NN), 6.15 degrees/joint per frame for locally weighted regression (LR), and 2.31 degrees/joint per frame for our method. (b) Average errors for walking motion: 4.46 degrees/joint per frame for NN, 3.32 degrees/joint per frame for LWR, and 1.30 degrees/joint per frame for our method. None of the testing sequences are in the database and boxing and walking motions are synthesized from the same set of markers used for the two camera system (figure 1).

sion [Atkeson et al. 1997a] because we synthesize the motion in a low-dimensional parametric space constructed from a set of closest examples rather than directly interpolating the local examples. In figure 6, we compare our method with two popular local learning methods: nearest neighbor (NN) and locally weighted regression (LWR). Nearest neighbor synthesis simply chooses the closest example. Locally weighted regression interpolates the nearby points by their distance to the query point and has proven very effective in such problem domains as motor learning [Atkeson et al. 1997b] and speech synthesis [Daelemans and van de Bosch 2001]. Figure 6 shows that our method creates more accurate results than either nearest neighbor synthesis or locally weighted regression.

8 Results

We test the effectiveness of our algorithm on different behaviors and different users using a large and heterogeneous human motion database and evaluate the quality of the synthesized motions by comparing them with motion capture data recorded with a full marker set. Our results are best seen in video form[1], although we show several frames of a few motions in figure 7.

We tested our system by controlling and animating a virtual character using two synchronized streams of video data and a small set of markers. Figure 7.1–7.5 shows sample frames of the results. The users control boxing, Kendo, walking, running, jumping and hopping. In the accompanying video, we also demonstrate that the users can transition from one behavior to another, for example from walking to running and that the system can synthesize motions in which the user is not fully facing forward. The video also illustrates a one camera system which uses a slightly larger set of markers.

We also compare the reconstructed motion with motion capture data recorded with a full marker set.

Leave-one-out evaluation. First, we evaluate the reconstructed motions by leaving out one sequence of motion capture data from each database as the testing sequence (figure 7.6). The 3D trajectories from the control markers used for the two camera system (as captured by the Vicon system) are then input to our online motion synthesis system to construct an animation (figure 7.7). This test, however, does not include the effect of errors in the tracking of markers in the two camera vision system.

End-to-end evaluation. To perform an end-to-end evaluation, we synchronize the Vicon system and our two camera system and capture the movement of the user wearing the full set of markers. We then compare the motion from the Vicon system using the full marker set and the motion from the two camera system using the small set of markers. The result of this test for boxing is shown in the video. The reconstruction error for the boxing sequence is 2.54 degrees per joint angle.

9 Discussion

We have presented an approach for performance animation that uses a series of local models created at run time from a large and heterogeneous human motion database to reconstruct full-body human motion from low-dimensional control signals. We demonstrate the power and flexibility of this approach with different users wearing a small set of markers and controlling a variety of behaviors in real time by performing in front of one or two video cameras. Given an appropriate database, the results are comparable in quality to those obtained from a commercial motion capture system; however, our performance animation system is far less expensive and requires less time to suit up the user.

Because the models are local, the system handles a heterogeneous database without difficulty. In our experiments, combining databases containing different behaviors had no effect on the performance of the local models or on the quality of the reconstructed motion. When we used a database in which each clip was labelled according to its behavior, we observed that the nearest neighbor search would rarely pick up a sequence of poses from a behavior other than the one the user was performing. A global method such as PCA or GPLVM has a much more difficult time modeling a heterogeneous database because it must compute a global model of the entire database rather than consecutive local models of the region around the user's current pose. Global models might be appropriate for applications such as synthesis of motion without a continuous driving signal (for example, [Safonova et al. 2004]), but given the temporal coherence of the control signals of our performance animation system, they were not required.

[1] http://graphics.cs.cmu.edu/projects/performance-animation

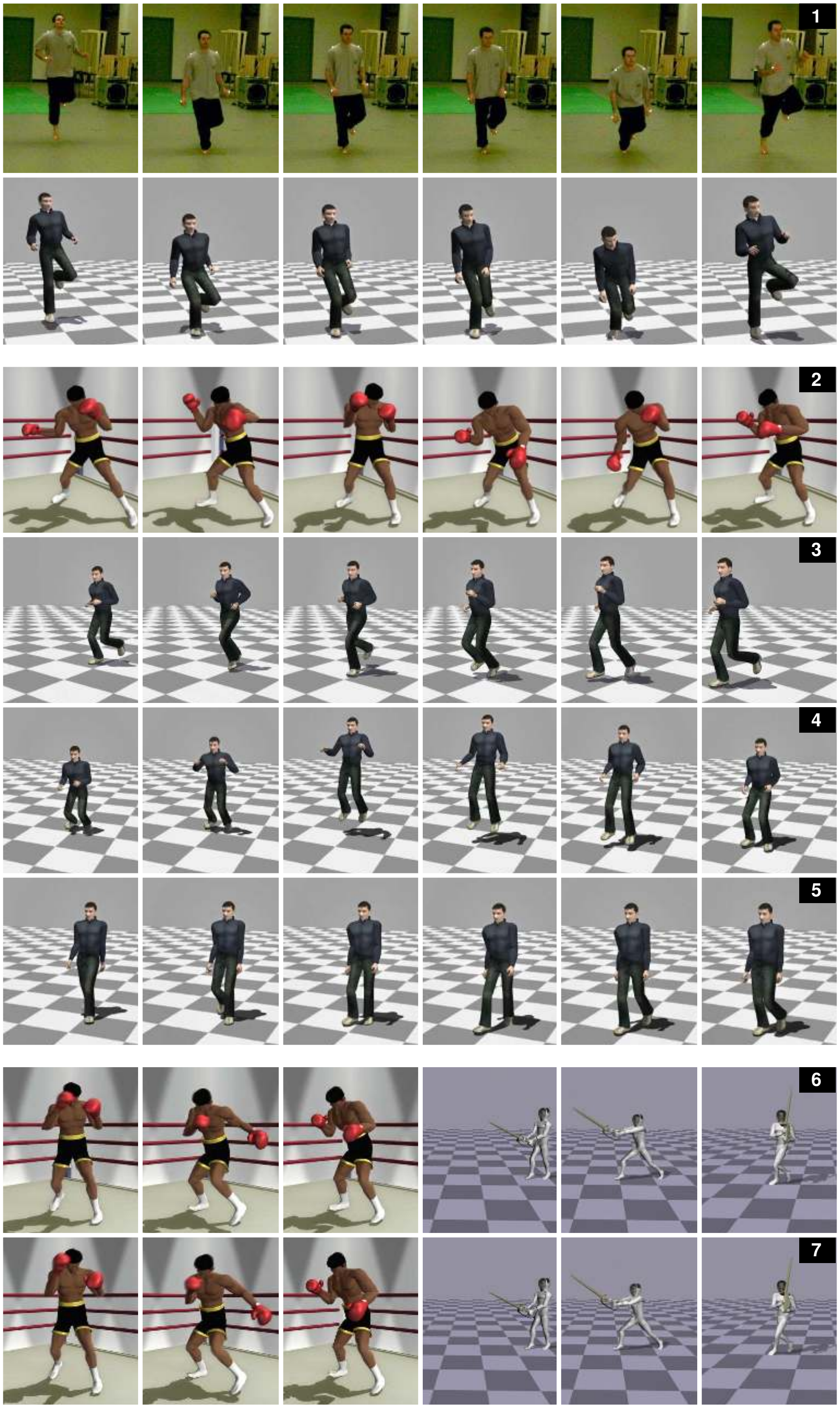


Figure 7: Performance animation from low-dimensional signals. (1) The input video and corresponding output animation. (2)–(5) Animations created by users using the two camera system. (6) Ground truth motion capture data. (7) Synthesized motion from the same marker set as that used for the two camera system.

The performance of the system scales well with the size of the database because the nearest neighbor search is independent of the size of the database; however, it is dependent on the density of the database because an ε ball is searched to find the nearest neighbors at run time. If repeated poses in the database became an issue, the size of the ε ball could be reduced or the database could be culled for duplicate sequences as a pre-computation step.

The system achieves some generality beyond the database particularly with respect to small changes in style (speed and exact posture). However, we do find nearest neighbors for the entire pose on each frame and therefore novel combinations of behaviors (hopping while performing Kendo, for example) will likely not yield reasonable results. We have not yet attempted to assess how far the user's motions can stray from those in the database before the quality of the resulting animation declines to an unacceptable level.

We have tested the system with users whose motion was not part of the database and found that the quality of the reconstructed motion was still good. We have not yet attempted to rigorously assess the dependence of the system on the body type of the user. A larger set of prototype skeleton models would likely result in a better match to the user's skeleton as would a more sophisticated pose for calibration (such as a "motorcycle" pose with all joints slightly bent rather than a "T" pose).

We made somewhat an arbitrary decision in choosing where to place the markers for a specific behavior, although we always put several markers on the torso to compute the root position and orientation. For locomotion, we placed markers only on the hands, feet, and shoulders, which allowed a user wearing street clothes to control the character's motion. For boxing, we added a marker on the head because head motion is a key element of boxing. For Kendo, we placed markers on the lower arms rather than on the hands in order to reduce occlusion by the sword. We also added one marker on the sword. A more principled analysis of marker placement could be performed using synthetic data rendered with a graphical model of the character performing the target behavior.

We have tested the system with two synchronized cameras and a small set of markers. One limitation of the current system is that it does not allow the user to move freely in the space because of the requirement that most markers be seen by at least one camera. A third camera would reduce the constraints on the user's facing direction. For any given number of cameras, a larger set of markers should reconstruct the motion more accurately but will increase the level of the intrusiveness of the animation interface. Similarly, adding more cameras to the system could improve the performance; the system, however, will become more expensive and cumbersome. The number of cameras and markers should probably be determined by the application.

We chose to use two cameras and a small marker set because a user might be able to create such a simple and cheap setup at home. Other sensors also fit this description and might provide the types of control signals we seek. For example, inertial measurement units (IMUs) are now sold in small packages and could include a wireless link to the animation computer [Xsens MT-9 2004; MicroStrain 3DM-G 2004]. A standard performance animation system for human characters would require at least 18 IMUs for full-body motion control. The local models should allow us to significantly reduce the number of IMUs and thus the cost of the system.

Another limitation of the system is that an appropriate database must be available. That should not be a problem for a sports video game because the virtual players are often animated using motion capture. We believe that the range of behaviors expected of the user is sufficiently limited in many other applications and that this approach will be widely applicable. For example, local models could be used to constrain the search space for markerless human motion capture [Cheung et al. 2003] as well as motion planning [Yamane et al. 2004]. Commercial motion capture systems could use the local model to filter noisy data and fill in missing values automatically. Another potential application is the automatic synthesis of detailed full-body animation based on a small set of trajectories keyframed by an animator [Pullen and Bregler 2002].

Acknowledgments

The authors would like to thank Moshe Mahler for his help in modeling and rendering the images for this paper and Justin Macey for his assistance in collecting and cleaning the motion capture data. The authors would like to thank Alias/Wavefront for their donation of Maya software. This material is based upon work supported by the National Science Foundation under Grants No. CNS-0196217, IIS-0205224, and IIS-0326322.

References

AHA, D. 1997. Editorial, special issue on lazy learning. In *Artificial Intelligence Review*. 11(1-5):1–6.

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. In *ACM Transactions on Graphics*. 21(3):483–490.

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2003. Motion synthesis from annotations. In *ACM Transactions on Graphics*. 22(3):402–408.

ATKESON, C. G., MOORE, A. W., AND SCHAAL, S. 1997a. Locally weighted learning. In *Artificial Intelligence Review*. 11(1-5):11-73.

ATKESON, C. G., MOORE, A. W., AND SCHAAL, S. 1997b. Locally weighted learning for control. In *Artificial Intelligence Review*. 11(1-5):75–113.

BADLER, N. I., HOLLICK, M., AND GRANIERI, J. 1993. Real-time control of a virtual human using minimal sensors. In *Presence*. 2(1):82–86.

BAZARAA, M. S., SHERALI, H. D., AND SHETTY, C. 1993. *Nonlinear Programming: Theory and Algorithms*. John Wiley and Sons Ltd. 2nd Edition.

BISHOP, C. 1996. *Neural Network for Pattern Recognition*. Cambridge University Press.

BRAND, M., AND HERTZMANN, A. 2000. Style machines. In *Proceedings of ACM SIGGRAPH 2000*. 183–192.

BRAND, M. 1999. Shadow puppetry. In *Proceedings of IEEE International Conference on Computer Vision*. 1237–1244.

BREGLER, C., AND OMOHUNDRO, S. 1995. Nonlinear image interpolation using manifold learning. In *Advances in Neural Information Processing Systems 7*. 973–980.

CHEUNG, G., BAKER, S., AND KANADE, T. 2003. Shape-from-silhouette of articulated object and its use for human body kinematics estimation and motion capture. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 77–84.

Daelemans, W., and van de Bosch, A. 2001. Treetalk: Memory-based word phonemisation. In *Data-Driven Techniques in Speech Synthesis*, Kluwer. 149-172.

Dontcheva, M., Yngve, G., and Popovic, Z. 2003. Layered acting for character animation. In *ACM Transactions on Graphics*. 22(3):409–416.

Freeman, W. T., Anderson, D., Beardsley, P., Dodge, C., Kage, H., Kyuma, K., Miyake, Y., Roth, M., Tanaka, K., Weissman, C., and Yerazunis, W. 1998. Computer vision for interactive computer graphics. In *IEEE Computer Graphics and Applications*. 18(3):42–53.

Fukunaga, K., and Olsen, D. 1971. An algorithm for finding intrinsic dimensionality of data. In *IEEE Transactions on Computers*. C-20:176–183.

Grochow, K., Martin, S. L., Hertzmann, A., and Popovic, Z. 2004. Style-based inverse kinematics. In *ACM Transactions on Graphics*. 23(3):522–531.

Guo, S., and Roberge, J. 1996. A high level control mechanism for human locomotion based on parametric frame space interpolation. In *Eurographics Workshop on Computer Animation and Simulation'96*. 95–107.

Hinton, G., Revow, M., and Dayan, P. 1995. Recognizing handwritten digits using mixtures of linear models. In *Advances in Neural Information Processing Systems 7*. 1015–1022.

Howe, N., Leventon, M., and Freeman, W. 1999. Bayesian reconstruction of 3d human motion from single-camera video. In *Advances in Neural Information Processing Systems 12*. 820–826.

Konami Boxing and Police 911 Game, 2001. http://www.konami.com.

Kovar, L., and Gleicher, M. 2004. Automated extraction and parameterization of motions in large data sets. In *ACM Transactions on Graphics*. 23(3):559–568.

Kovar, L., Gleicher, M., and Pighin, F. 2002. Motion graphs. In *ACM Transactions on Graphics*. 21(3):473–482.

Lawrence, N. D. 2004. Gaussian process latent variable models for visualization of high dimensional data. In *Advances in Neural Information Processing Systems 16*. 329–336.

Lee, J., Chai, J., Reitsma, P., Hodgins, J., and Pollard, N. 2002. Interactive control of avatars animated with human motion data. In *ACM Transactions on Graphics*. 21(3):491–500.

Li, Y., Wang, T., and Shum, H.-Y. 2002. Motion texture: A two-level statistical model for character synthesis. In *ACM Transactions on Graphics*. 21(3):465–472.

Mardia, K., Kent, J., and Bibby, M. 1979. *Multivariate Analysis*. Academy Press.

MicroStrain 3DM-G, 2004. http://www.microstrain.com.

Oore, S., Terzopoulos, D., and Hinton, G. 2002. A desktop input device and interface for interactive 3d character. In *Proceedings of Graphics Interface 2002*. 133–140.

Pullen, K., and Bregler, C. 2002. Motion capture assisted animation: Texturing and synthesis. In *ACM Transactions on Graphics*. 21(3):501–508.

Ren, L., Shakhnarovich, G., Hodgins, J. K., Pfister, H., and Viola, P. A. 2004. Learning silhouette features for control of human motion. In *Computer Science Technical Reports 2004, Carnegie Mellon University*. CMU-CS-04-165.

Rose, C., Cohen, M. F., and Bodenheimer, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. In *IEEE Computer Graphics and Applications*. 18(5):32–40.

Roweis, S., and Saul, L. 2000. Nonlinear dimensionality reduction by locally linear embedding. In *Science*. 290(5500):2323–2326.

Safonova, A., Hodgins, J., and Pollard, N. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *ACM Transactions on Graphics*. 23(3):514–521.

Scholkopf, B., Smola, A., and Muller, K.-R. 1999. Kernel principal component analysis. In *Advances in Kernel Methods–SV Learning*, MIT Press. 327–352.

Semwal, S., Hightower, R., and Stansfield, S. 1998. Mapping algorithms for real-time control of an avatar using eight sensors. In *Presence*. 7(1):1–21.

Shin, H. J., Lee, J., Gleicher, M., and Shin, S. Y. 2001. Computer puppetry: An importance-based approach. In *ACM Transactions on Graphics*. 20(2):67–94.

Sidenbladh, H., Black, M. J., and Sigal, L. 2002. Implicit probabilistic models of human motion for synthesis and tracking. In *European Conference on Computer Vision*. 784–800.

Sony Eye Toy Systems, 2003. http://www.eyetoy.com.

Stone, M. 1974. Cross-validatory choice and assessment of statistical predictions. In *Journal of the Royal Statistical Society*. 36:111–147.

Tenenbaum, J., Silva, V., and Langford, J. 2000. A global geometric framework for nonlinear dimensionality reduction. In *Science*. 290(5500):2319–2323.

Vicon Systems, 2004. http://www.vicon.com.

Wiley, D. J., and Hahn, J. K. 1997. Interpolation synthesis of articulated figure motion. In *IEEE Computer Graphics and Applications*. 17(6):39–45.

Xsens MT-9, 2004. http://www.xsens.com.

Xu, G., and Zhang, Z. 1996. *Epipolar Geometry in Stereo, Motion, and Object Recognition: A Unified Approach*. Kluwer.

Yamane, K., and Nakamura, Y. 2003. Natural motion animation through constraining and deconstraining at will. In *IEEE Transactions on Visualization and Computer Graphics*. 9(3):352–360.

Yamane, K., Kuffner, J. J., and Hodgins, J. K. 2004. Synthesizing animations of human manipulation tasks. In *ACM Transactions on Graphics*. 23(3):532–539.

Yin, K., and Pai, D. K. 2003. Footsee: An interactive animation system. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 329–338.

Zhang, Z. 1999. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the International Conference on Computer Vision*. 666–673.