Interactive Control of Avatars Animated with Human Motion Data

Jehee Lee
Carnegie Mellon University

Jinxiang Chai
Carnegie Mellon University

Paul S. A. Reitsma
Brown University

Jessica K. Hodgins
Carnegie Mellon University

Nancy S. Pollard
Brown University

Abstract

Real-time control of three-dimensional avatars is an important problem in the context of computer games and virtual environments. Avatar animation and control is difficult, however, because a large repertoire of avatar behaviors must be made available, and the user must be able to select from this set of behaviors, possibly with a low-dimensional input device. One appealing approach to obtaining a rich set of avatar behaviors is to collect an extended, unlabeled sequence of motion data appropriate to the application. In this paper, we show that such a motion database can be preprocessed for flexibility in behavior and efficient search and exploited for real-time avatar control. Flexibility is created by identifying plausible transitions between motion segments, and efficient search through the resulting graph structure is obtained through clustering. Three interface techniques are demonstrated for controlling avatar motion using this data structure: the user selects from a set of available choices, sketches a path through an environment, or acts out a desired motion in front of a video camera. We demonstrate the flexibility of the approach through four different applications and compare the avatar motion to directly recorded human motion.

CR Categories: I.3.7 [Three-Dimensional Graphics and Realism]: Animation—Virtual reality

Keywords: human motion, motion capture, avatars, virtual environments, interactive control

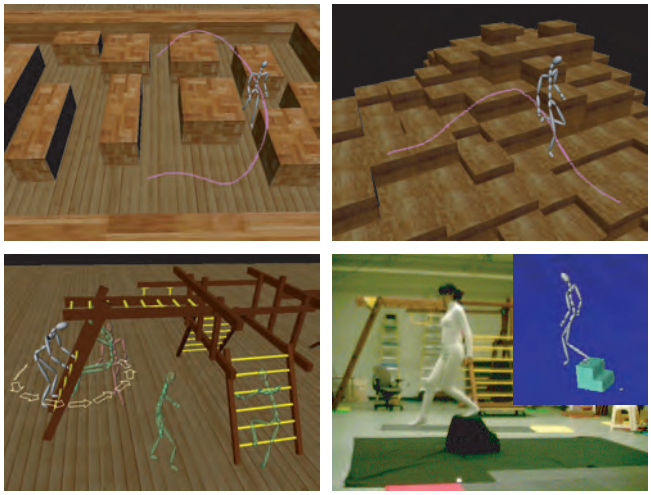


Figure 1: Real-time avatar control in our system. (Top) The user controls the avatar's motion using sketched paths in maze and rough terrain environments. (Bottom left) The user selects from a number of choices in a playground environment. (Bottom right) The user is controlling the avatar by performing a motion in front of a camera. In this case only, the avatar's motion lags the user's input by several seconds.

1 Introduction

The popularity of three-dimensional computer games with human characters has demonstrated that the real-time control of avatars is an important problem. Two difficulties arise in animating and controlling avatars, however: designing a rich set of behaviors for the avatar, and giving the user control over those behaviors. Designing a set of behaviors for an avatar is difficult primarily due to the real-time constraint, especially if we wish to make use of relatively unstructured motion data for behavior generation. The raw material for smooth, appealing, and realistic avatar motion can be provided through a large motion database, and this approach is frequently used in video games today. Preparing such a database, however, requires substantial manual processing and careful design so that the character's behavior matches the user's expectations. Such databases currently tend to consist of many short, carefully planned, labeled motion clips. A more flexible and more broadly useful approach would allow extended, unlabeled sequences of motion capture data to be exploited for avatar control. If such unstructured data is used, however, searching for an appropriate motion in an on-line fashion becomes a significant challenge.

Providing the user with an intuitive interface to control the avatar's motion is difficult because the character's motion is high dimensional and most of the available input devices are not. Input from devices such as mice and joysticks typically indicates a position (go to this location), velocity (travel in this direction at this speed) or behavior (perform this kick or pick up this object). This input must then be supplemented with autonomous behaviors and transitions to compute the full motion of the avatar. Control of individual degrees of freedom is not possible for interactive environments unless the user can use his or her own body to act out or pantomime the motion.

In this paper, we show that a rich, connected set of avatar behaviors can be created from extended, freeform sequences of motion, automatically organized for efficient search, and exploited for real-time avatar control using a variety of interface techniques. The motion is preprocessed to add variety and flexibility by creating connecting transitions where good matches in poses, velocities, and contact state of the character exist. The motion is then clustered into

{jehee|jchai|jkh}@cs.cmu.edu, {psar|nsp}@cs.brown.edu

groups for efficient searching and for presentation in the interfaces. A unique aspect of our approach is that the original motion data and the generalization of that data are closely linked; *each frame* of the original motion data is associated with a tree of clusters that captures the set of actions that can be performed by the avatar from that specific frame. The resulting *cluster forest* allows us to take advantage of the power of clusters to generalize the motion data without losing the actual connectivity and detail that can be derived from that data. This two-layer data structure can be efficiently searched at run time to find appropriate paths to behaviors and locations specified by the user.

We explore three different interfaces to provide the user with intuitive control of the avatar's motion: choice, sketch, and performance (figure 1). In choice interfaces, the user selects among a number of options (directions, locations, or behaviors) every few seconds. The options that are presented to the user are selected from among the clusters created during the preprocessing of the motion data. In the sketching interface, the user specifies a path through the environment by sketching on the terrain, and the data structure is searched to find motion sequences that follow that path. In performance interfaces, the user acts out a behavior in front of a video camera. The best fit for his or her motion is then used for the avatar, perhaps with an intervening segment of motion to provide a smooth transition. For all three interface techniques, our motion data structure makes it possible to transform possibly low-dimensional user input into realistic motion of the avatar.

We demonstrate the power of this approach through examples in four environments (figure 1) and through comparison with directly recorded human motion in similar environments. We note that the vision-based interface, due to the higher dimensional nature of the input, gives the most control over the details of the avatar's motion, but that the choice and sketch interfaces provide the user with simple techniques for directing the avatar to achieve specific goals.

2 Background

The behaviors required for animating virtual humans range from very subtle motions such as a slight smile to highly dynamic, whole body motions such as diving or running. Many of the applications envisioned for avatars have involved interpersonal communication and as a result, much of the research has focused on the subtle aspects of the avatar's appearance and motion that are essential for communication: facial expressions, speech, eye gaze direction, and emotional expression [Cassell 2000; Chopra-Khullar and Badler 1999]. Because our focus is on applications in which whole body actions are required and subtle communication is not, we review only the research related to whole body human motion.

Animated human figures have been driven by keyframed motion, rule-based systems [Bruderlin and Calvert 1989; Perlin 1995; Bruderlin and Calvert 1996; Perlin and Goldberg 1996; Chi et al. 2000; Cassell et al. 2001], control systems and dynamics [Hodgins et al. 1995; Wooten and Hodgins 1996; Laszlo et al. 1996; Faloutsos et al. 2001a; Faloutsos et al. 2001b], and, of course, motion capture data. Motion capture data is the most common technique in commercial systems because many of the subtle details of human motion are naturally present in the data rather than having to be introduced via domain knowledge. Most research on handling motion capture data has focused on techniques for modifying and varying existing motions. See Gleicher [2001] for a survey. This need may be partially obviated by the growing availability of significant quantities of data. However, adaptation techniques will still be required for interactive applications in which the required motions cannot be precisely or completely predicted in advance.

A number of researchers have shared our goal of creating new motion for a controllable avatar from a set of examples. For simple behaviors like reaching and pointing that can be adequately spanned by a data set, straightforward interpolation works remarkably well [Wiley and Hahn 1997]. Several groups explored methods for decomposing the motion into a behavior and a style or emotion using a Fourier expansion [Unuma et al. 1995], radial basis functions [Rose et al. 1998] or hidden Markov models with similar structure across styles [Brand and Hertzmann 2000]. Other researchers have explored introducing random variations into motion in a statistically reasonable way: large variations were introduced using chaos by Bradley and Stuart [1997] and small variations were introduced using a kernel-based representation of joint probability distributions by Pullen and Bregler [2000]. Domain specific knowledge can be very effective: Sun and Metaxas [2001] used principles from biomechanics to represent walking motion in such a way that it could be adapted to walking on slopes and around curves.

Lamouret and van de Panne [1996] implemented a system that was quite similar to ours albeit for a far simpler character, a hopping planar Luxo lamp. A database of physically simulated motion was searched for good transitions, based on the state of the character, local terrain, and user preferences. The selected hop is then adapted to match the terrain.

A number of researchers have used statistical models of human motion to synthesize new animation sequences. Galata and her colleagues [2001] use variable length hidden Markov models to allow the length of temporal dependencies to vary. Bowden [2000] uses principle component analysis (PCA) to simplify the motion, K-means clustering to collect like motions, and a Markov chain to model temporal constraints. Brand and Hertzmann's system allowed the reconstruction of a variety of motions statistically derived from the original dataset. Li et al. [2002] combine low level, noise driven motion generators with a high level Markov process to generate new motions with variations in the fine details. All of these systems used generalizations of the motion rather than the original motion data for synthesis, which runs the risk of smoothing out subtle motion details. These systems also did not emphasize control of the avatar's motion or behavior.

Recent research efforts have demonstrated approaches similar to ours in that they retain the original motion data for use in synthesis. Sidenbladh and her colleagues [2002] have developed a probabilistic model for human motion tracking and synthesis of animations from motion capture data that predicts each next frame of motion based on the preceeding d frames. PCA dimensionality reduction combined with storage of motion data fragments in a binary tree help to contain the complexity of a search for a matching motion fragment in the database. Pullen and Bregler [2002] allow an animator to keyframe motion for a subset of degrees of freedom of the character and use a motion capture library to synthesize motion for the missing degrees of freedom and add texture to those that were keyframed. Kovar and his colleagues [2002] generate a graph structure from motion data and show that branch and bound search is very effective for controlling a character's motion for constraints such as sketched paths where the motion can be constructed incrementally. Their approach is similar to our sketch-based interface when no clustering is used. It has the advantage over our sketch-based interface, however, that the locomotion style (or other labeled characteristics of the motion) can be specified for the sketched path. Arikan and Forsyth [2002] use a hierarchy of graphs to represent connectivity of a motion database and perform randomized search to identify motions that satisfy user constraints such as motion duration and pose of the body at given keyframes. One advantage of their work over similar techniques is flexibility in the types of constraints that can be specified by the user. To our knowledge the first paper to be published using this general approach was by Molina-Tanco and Hilton [2000], who created a system that can be controlled by the selection of start and ending keyframes. Preprocessing of the motion data included PCA dimensionality reduction and clustering. The user-specified keyframes are identified in par-

Figure 2: A subject wearing retro-reflective markers in the motion capture laboratory.

ticular clusters, a connecting path of clusters is found via dynamic programming, and the most probable sequence of motion segments passing through these clusters is used to generate the details of the new motion. Their underlying data structure is similar to ours, although our use of *cluster trees* offers more flexibility in paths available to the avatar at a given frame.

In the area of interface techniques for controlling avatars, most successful solutions to date have given the character sufficient autonomy that it can be "directed" with a low dimensional input. Besides the standard mouse or joystick, this input may come from vision (e.g. [Blumberg and Galyean 1995]) or from a puppet (e.g. [Blumberg 1998]).

Control at a more detailed level can be provided if the user is able to act out a desired motion. The infrared sensor-based "mocap" games Mocap Boxing and Police 911 by Konami are an interesting commercial example of this class of interface. The user's motion is not precisely matched, although the impact of the user's motion on characters in the environment is essential for game play. In the research community, a number of groups have explored avatar control via real time magnetic motion capture systems (e.g. [Badler et al. 1993] [Semwal et al. 1998] [Molet et al. 1996] [Molet et al. 1999]). Alternatives to magnetic motion capture are available for capturing whole body motion in real time optically [Oxford Metric Systems 2002] or via an exoskeleton [Sarcos 2002].

Vision-based interfaces are appealing because they allow the user to move unencumbered by sensors. Vision data from a single camera, however, does not provide complete information about the user's motion, and a number of researchers have used motion capture data to develop mappings or models to assist in reconstruction of three-dimensional pose and motion from video (e.g. [Rosales et al. 2001] [Brand 1999]). In the area of database retrieval, Ben-Arie and his colleagues [2001] use video data to index into a database of human motions that was created from video data and show that activity classes can be discriminated based on a sparse set of frames from a query video sequence. In our approach, the problem of controlling an avatar from vision is more similar to database retrieval than pose estimation in the sense that the system selects an action for the avatar from among a finite number of possibilities.

3 Human Motion Database

The size and quality of the database is key to the success of this work. The database must be large enough that good transitions can be found as needed and the motion must be free of glitches and other characteristic problems such as feet that slip on the ground. The human motion data was captured with a Vicon optical motion capture system. The system has twelve cameras, each of which is capable of recording at 120Hz with images of 1000x1000 resolution. We used a marker set with 43 14mm markers that is an adaptation of a standard biomechanical marker set with additional markers to facilitate distinguishing the left side of the body from the right side in an automatic fashion. The motions were captured in a working volume for the subject of approximately 8'x24'. A

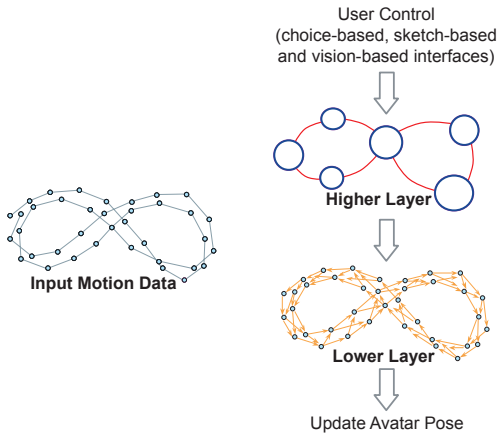


Figure 3: Two layer structure for representing human motion data. The lower layer retains the details of the original motion data, while the higher layer generalizes that data for efficient search and for presentation of possible actions to the user.

subject is shown in the motion capture laboratory in figure 2.

We captured subjects performing several different sets of motions: interacting with a step stool (stepping on, jumping over, walking around, and sitting on), walking around an empty environment (forwards, backwards, and sideways), walking over "poles and holes" rough terrain, and swinging and climbing on a piece of playground equipment. Each subject's motion is represented by a skeleton that includes his or her limb lengths and joint range of motion (computed automatically during a calibration phase). Each motion sequence contains trajectories for the position and orientation of the root node (pelvis) as well as relative joint angles for each body part. For the examples presented here, only one subject's motion was used for each example.

The motion database contains a single motion (about 5 minutes long) for the step stool example, 9 motions for walking around the environment, 26 motions on the rough terrain and 11 motions on the playground equipment. The motion is captured in long clips (an average of 40 seconds excluding the step stool motion) to allow the subjects to perform natural transitions between behaviors. Our representation of the motion data does not require hand segmenting the motion data into individual actions as that occurs naturally as part of the clustering of the data in preprocessing.

Contact with the environment is an important perceptual feature of motion, and the database must be annotated with contact information for generation of good transitions between motion segments. The data is automatically processed to have this information. The system determines if a body segment and an environment object are in contact by considering their relative velocity and proximity. For instance, feet are considered to be on the ground if one of their adjacent joints (either the ankle or the toe) is sufficiently close to the ground and its velocity is below some threshold.

4 Data Representation

Human motion is typically represented either in a form that preserves the original motion frames or in a form that generalizes those frames with a parametric or probabilistic model. Both representations have advantages; the former allows details of the original motion to be retained for motion synthesis, while the latter creates a simpler structure for searching through or presenting the data. Our representation attempts to capture the strengths of both by combining them in a two-layer structure (figure 3). The higher layer is a statistical model that provides support for the user interfaces by

clustering the data to capture similarities among character states. The lower layer is a Markov process that creates new motion sequences by selecting transitions between motion frames based on the high-level directions of the user. A unique aspect of our data structure is the link between these layers: trees of accessible clusters are stored in the lower layer on a frame by frame basis, providing a high-level and correct description of the set of behaviors achievable from each specific motion frame. The next two sections describe the details of the two layers and the link between these layers, beginning with the lower-level Markov process.

4.1 Lower Layer: Markov Process

We model motion data as a first-order Markov process, inspired by the work on creating non-repeating series of images presented in Schödl et al. [2000]. The transition from one state to the next of a first-order Markov process depends only on the current state, which is a single frame of motion.

The Markov process is represented as a matrix of probabilities with the elements P_{ij} describing the probability of transitioning from frame i to frame j. As in Schödl et al. [2000], the probabilities are estimated from a measure of similarity between frames using a exponential function:

$$P_{ij} \propto \exp(-D_{i,j-1}/\sigma), \tag{1}$$

where $D_{i,j-1}$ represents the distance between frame i and frame $j-1$, and σ controls the mapping between the distance measure and the probability of transition. The distance function is computed as

$$D_{ij} = d(p_i, p_j) + \nu d(v_i, v_j). \tag{2}$$

The first term $d(p_i, p_j)$ describes the weighted differences of joint angles, and the second term $d(v_i, v_j)$ represents the weighted differences of joint velocities. Parameter ν weights velocity differences with respect to position differences. The velocity term helps to preserve the dynamics of motion by, for example, discriminating between similar poses in forward walk and backward walk.

In our implementation, Euclidean differences are used for velocities. Position differences are expressed as

$$d(p_i, p_j) = \|p_{i,0} - p_{j,0}\|^2 + \sum_{k=1}^{m} w_k \| \log\left(q_{j,k}^{-1} q_{i,k}\right) \|^2 \tag{3}$$

where $p_{i,0} \in \mathbb{R}^3$ is the translational position of the character at frame i, $q_{i,k} \in \mathbb{S}^3$ is the orientation of joint k with respect to its parent in frame i, and joint angle differences are summed over m rotational joints. The value of $\log(q_a^{-1} q_b)$ is a vector $\mathbf{v}$ such that a rotation of $2\|\mathbf{v}\|$ about the axis $\frac{\mathbf{v}}{\|\mathbf{v}\|}$ takes a body from orientation q_a to orientation q_b. Important joints are selected manually to determine weights; weights w_k are set to one for joints at the shoulders, elbows, hips, knees, pelvis, and spine. Weights are set to zero for joints at the neck, ankle, toes and wrists, which have less impact on visible differences between poses.

The matrix of probabilities P_{ij} computed using equation 1 is dense and requires $O(n^2)$ storage space for n motion frames. Because the motion database is large (4000 to 12000 frames depending on the application), $O(n^2)$ storage space may be prohibitive. Many of the transitions are of low probability and pruning them not only reduces the required storage space but also improves the quality of the resulting motion by avoiding too frequent transitions. We use four rules to prune the transitions (table 1). The first rule is based on the observation that contact is a key element of a motion. It prunes transitions between motion segments with dissimilar contact states and is described below. The second rule sets all probabilities below a user-specified threshold to zero. The third rule favors the best transition among many similar transitions by selecting local maxima in the transition matrix and setting the probabilities of the others to zero. The fourth rule eliminates edges that are not fully contained in the single largest connected component of the graph as described below.

	total # of frames	Pruning Criteria			
		contact	likelihood	similarity	SCC
Maze	8318	394284	32229	N/A	31469
Terrain	12879	520193	60130	N/A	59043
Step Stool	4576	30058	4964	N/A	4831
Playground	5971	984152	37209	7091	5458

Table 1: The number of transition edges remaining after pruning edges present in the lower layer Markov model of motion for our examples. The total number of frames indicates initial database size for each example, and the number of possible edges is the square of this number. Edges are pruned based on contact conditions, based on a probability threshold, to eliminate similar transitions, and to remove edges that do not fall entirely within the largest strongly connected component (SCC) of the graph.

Pruning based on contact. Pruning based on contact is done by examining contact states of the two motion segments at the transition. A transition from frame i to frame j is pruned if frames i and $j-1$ or frames $i+1$ and j are in different contact states. For example, a transition is not allowed from a pose where the foot is about to leave the ground to another pose where the foot is about to touch the ground even if the configurations are similar. In practice, in all of our examples except the playground, this rule is made even more strict; transitions are allowed only during a contact change, and this same contact change must occur from frame i to frame $i+1$ and from frame $j-1$ to frame j. Pruning of similar transitions is not necessary when this stricter contact pruning rule is in force. Because contact change is a discrete event, no similar, neighboring sets of transitions exist to be pruned.

Avoiding dead ends. As pointed out by Schödl et al. [2000], transitions might lead to a portion of motion that has no exits. They suggested avoiding dead ends by predicting the anticipated future cost of a transition and giving high cost to dead ends. Although their method works well in practice, it does not guarantee that dead ends will never be encountered. We instead find the strongly connected subcomponent of the directed graph whose nodes are the frames of the motion and whose edges are the non-zero transitions. We implemented Tarjan's algorithm [Tarjan 1972], with running time linear in the number of nodes, to find all the strongly connected subcomponents. In our experiments, the algorithm usually found one large strongly connected component and a number of small components most of which consist of a single node. We set the transition probabilities to zero if the transitions leave the largest strongly connected component.

4.1.1 Blending Transitions

Although most transitions should introduce only a small discontinuity in the motion, a transition might be noticeable if the motion sequence simply jumped from one frame to another. Instead the system modifies the motion sequence after the transition to match the sequence before the transition. If the transition is from frame i to frame j, the frames between $j-1$ and $j+b-1$ are modified so that the pose and velocity at frame $j-1$ is matched to the pose and velocity at frame i. Displacement mapping techniques are used to preserve the fine details of the motion [Witkin and Popović 1995; Bruderlin and Williams 1995]. In our experiments, blend interval b ranges from 1 to 2 seconds, depending on the example. We maintain a buffer of motion frames during the blend interval. If a transition happens before the previous transition has completed, motion

frames in the buffer are used instead of the original motion frames for the overlap between the current and previous blend intervals.

Although blending avoids jerkiness in the transitions, it can cause undesirable side effects such as foot sliding when there is contact between the character and the environment during the blend interval. This problem can be addressed by using constraint-based motion editing techniques such as [Gleicher 1997; Gleicher 1998; Lee and Shin 1999]. We use the hierarchical motion fitting algorithm presented by Lee and Shin [1999].

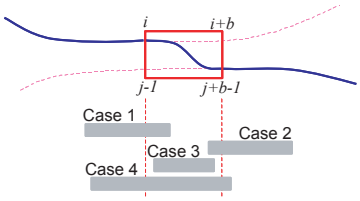
A good set of contact constraints, expressed as a desired trajectory (both translation and rotation) for each contacting body, must be provided to the motion fitting algorithm. These constraints are obtained from one of the two motion sequences involved in the blend. Suppose the transition is from frame i to frame j, and a body is in contact with the environment between frame k and frame l. If there is an overlap between the blending interval and the contact interval $[k, l]$, we establish the constraint based on the following cases:

CASE 1: $k \le i < l < i+b$

CASE 2: $j < k < j+b-1 \le l$

CASE 3: $j < k < l < j+b-1$

CASE 4: $k \le i < i+b \le l$ **or** $k \le j < j+b-1 \le l$



In **Case 1**, the constraint lies over the start of the blending interval and between frame i and frame l the foot should follow the trajectory of the motion sequence before the transition. Similarly in **Case 2** the constraint lies over the end of the blending interval and for frame k to frame $j+b-1$ the trajectory is taken from the motion sequence after the transition. In **Case 3**, the contact interval is contained within the blending interval and the trajectory of the foot can be taken from either side. Our implementation chooses the closer side. In **Case 4**, the constraint lies over both boundaries and there is no smooth transition. In this situation, the system allows the foot to slip or disables the transition by setting the corresponding probability to zero.

4.1.2 Fixed and Relative Coordinate Systems

The motion in the database is stored as a position and orientation for the root (pelvis) in every frame, along with joint angles in the form of orientation of each body with respect to its parent. The position and orientation of the root segment at frame i can be represented in a fixed, world coordinate system or as a relative translation and rotation with respect to the previous frame of motion (frame $i-1$). The decision of whether to represent the root in a fixed or relative coordinate system affects the implementation of Markov process for human motion data.

With a fixed coordinate system, transitions will only occur between motion sequences that are located nearby in three-dimensional space, while the relative coordinate system allows transitions to similar motion recorded anywhere in the capture region. The relative coordinate system effectively ignores translation on the horizontal plane and rotation about the vertical axis when considering whether two motions are similar or not.

The decision as to which coordinate system is most appropriate depends on the amount of structure in the environment created for the motion capture subjects: highly structured environments allowed less flexibility than unstructured environments. In our experiments, we used a fixed coordinate system for the playground and step stool examples, because they involved interaction with fixed objects. We used a relative coordinate system for the maze example, where motion data was captured in an environment with no obstacles. For the uneven terrain example, we ignored vertical translation of the ground plane, allowed rotations about the vertical axis in integer multiples of 90 degrees, and allowed horizontal translations that were close to integer multiples of block size. This arrangement provided flexibility in terrain height and extent while preserving step locations relative to the discontinuities in the terrain.

4.2 Higher Layer: Statistical Models

While the lower layer Markov model captures motion detail and provides the avatar with a broad variety of motion choices, the resulting data structure may be too complex for efficient search or clear presentation in a user interface. The higher layer is a generalization of the motion data that captures the distribution of frames and transitions. Our method for generalizing the data is based on cluster analysis. Clusters are formed from the original motion data. These clusters capture similarities in motion frames, but they do not capture the connections between frames (figure 4). To capture these connections, or the tree of choices available to the avatar at any given motion frame, we construct a data structure called a *cluster tree* at each motion frame. The entire higher layer is then called a *cluster forest*.

4.2.1 Cluster Analysis

Cluster analysis is a method for sorting observed values into groups. The data are assumed to have been generated from a mixture of probabilistic distributions. Each distribution represents a different cluster. Fraley and Raftery [1998] provides an excellent survey on cluster analysis. In our application, we use the isometric Gaussian mixture model in which each cluster is an isometric multivariate Gaussian distribution with variable standard deviation. Our goal is to capture similar motion states within the same cluster.

Motion state for frame i (referred to below as observation $\mathbf{x}_i$) is a vector containing root position $p_{i,0}$, weighted root orientation $q_{i,0}$, and weighted orientations $q_{i,k}$ of all bodies k with respect to their parents as follows:

$$\mathbf{x}_i = [p_{i,0} \;\; w_0 \log(q_{i,0}) \;\; w_1 \log(q_{i,1}) \;\; \ldots \;\; w_m \log(q_{i,m})]^T \quad (4)$$

where weights w_k for the shoulders, elbows, hips, knees, pelvis, and spine are set to one in our examples and all other weights are set to zero, as in equation 3. Specifying root orientation requires careful selection of the reference frame to avoid the singularity in the expression $\log(q_{i,0})$ when $q_{i,0}$ is a rotation of 2π radians about any axis. We select a reference orientation $\hat{q}$ from a series of root orientations $\{q_{i,0}\}$ such that it minimizes the distance to the farthest orientation in $\{q_{i,0}\}$. The distance between two orientations is computed as $d(q_a, q_b) = \min(\| \log(q_a^{-1} q_b) \|, \| \log(q_a^{-1}(-q_b)) \|)$.

Given a set of observations (motion frames) $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_N)$, let $f_k(\mathbf{x}_i|\theta_k)$ be the density of $\mathbf{x}_i$ from the k-th cluster, where θ_k are the parameters (the mean and the variance of the Gaussian distribution) of the k-th cluster. The Expectation Maximization (EM) algorithm is a general method to find the parameters $\theta = (\theta_1, \cdots, \theta_K)$ of clusters that maximize the mixture log-likelihood

$$\mathcal{L}(\theta_k, \tau_k, z_{ik}|\mathbf{x}) = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{ik} \log \left(\tau_k f_k(\mathbf{x}_i|\theta_k) \right), \quad (5)$$

where τ_k is the prior probability of each cluster k, and z_{ik} is the posterior probability that observation $\mathbf{x}_i$ belongs to the k-th cluster. Given initial values for the parameters of clusters, the EM algorithm iterates between the expectation step in which z_{ik} are computed from the current parameters and maximization step in which the parameters are updated based on the new values for z_{ik} (See [Fraley and Raftery 1998] for details).

For cluster analysis, we need to choose the number K of clusters. We use the Bayesian information criterion (BIC) [Fraley and

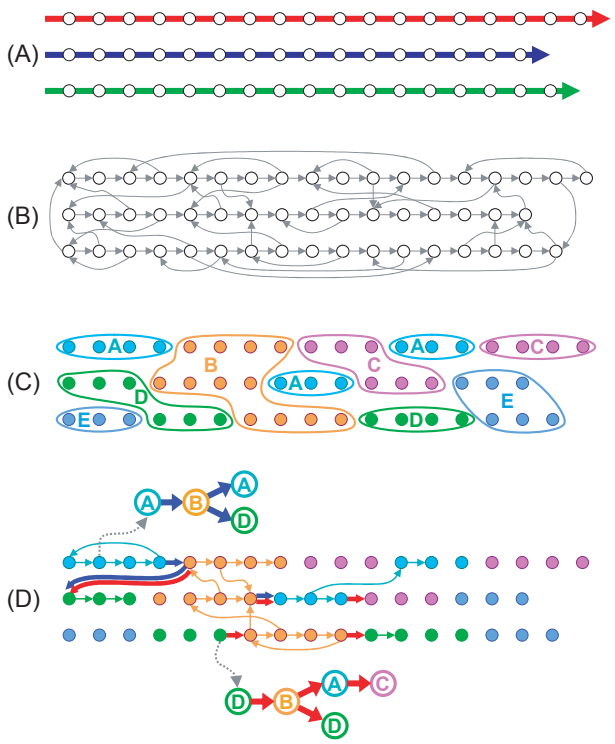


Figure 4: Motion data preprocessing. (A) The motion database initially consists of a number of motion clips containing many frames. (B) Many frame-to-frame transitions are created to form a directed graph. (C) Similar poses are clustered into groups. (D) To capture connections across cluster boundaries, we construct a cluster tree for each motion frame by traversing the graph to identify clusters that are reachable within a given depth (time) bound (6 transitions in this figure). When a transition (thick arrow) crossing cluster boundaries is encountered, a new node is added to the cluster tree. Note that frames within the same cluster may have different cluster trees.

Raftery 1998] of a mixture model $\mathcal{M} = \{\theta_k, \tau_k, z_{ik}\}$ to estimate the number of clusters:

$$BIC(\mathcal{M}) \equiv 2\mathcal{L}(\mathcal{M}|\mathbf{x}) - m_{\mathcal{M}} \log N, \quad (6)$$

where $\mathcal{L}(\mathcal{M}|\mathbf{x})$ is the maximized mixture log-likelihood for the model, and $m_{\mathcal{M}}$ is the number of independent parameters to be estimated in the model (the number of clusters $\times$ the dimension of the clusters). A larger value of $BIC(\mathcal{M})$ provides stronger evidence for the model. To estimate the number of clusters, we run the EM algorithm for a series of models with different numbers of clusters and choose the highest value of $BIC(\mathcal{M})$.

The result of expectation maximization depends greatly on the initial values for the parameters. A typical method is to repeat the EM algorithm with different (randomly generated) initial conditions and select the best result that gives the maximum log-likelihood. An alternative method is to select several keyframes by hand and take those frames as initial values for the means of clusters. This is the approach we used in practice. In our examples, 15 to 25 keyframes were selected by hand to provide a starting point for the EM algorithm. This small amount of manual input appeared to result in much better statistical models.

4.2.2 Cluster Forest

After the motion frames are clustered, we construct a *cluster forest* to encode the choices available to the avatar. The cluster forest consists of a collection of *cluster trees* representing sets of avatar behaviors. Each frame in the database has its own cluster tree, which reflects the behaviors immediately available to the avatar from that frame. This arrangement allows the interface techniques to consider only relevant behaviors. For example, if the avatar is far from the step stool, the behaviors that involve interacting with the step stool are not relevant although they may become important as the avatar moves nearer.

A cluster tree for a given motion frame r is computed as follows. The algorithm begins with a one-node tree consisting of the cluster to which r belongs. When a new frame j is visited from frame i that belongs to the k-th cluster, the algorithm checks if frame j belongs to the k-th cluster. If not, the cluster including frame j is added to the tree as a child of the current node. In either case, the children of frame j are traversed recursively. The recursion terminates when the depth of the spanning tree reaches a given maximum depth (time bound). In the examples presented here, the time bound was set to 15 seconds. Figure 4 shows some simple example cluster trees.

Cluster trees are constructed for all frames that have multiple out-going transitions. Frames with a single out-going transition are assigned to the first cluster tree reachable by traversing the lower layer graph from that frame. We note that because a cluster tree is assigned to each frame in the lower layer graph and because there are no dead ends in the lower layer graph, there will be no dead ends created in the resulting higher layer structure.

4.2.3 Cluster Paths and Most Probable Motions

A *cluster path* is a path from the root of a cluster tree to one of its leaves. If the cluster tree for a frame has k leaves, it will contain k cluster paths, each representing a collection of actions available to the avatar.

For any given cluster path, we can find the most probable sequence of motion frames. This sequence of frames is one of a generally large number of possible motions that pass through the given cluster path, but it provides a concrete representation of that cluster path that is a useful reference for the interface techniques used to control the avatar.

Suppose that cluster path $\mathbf{p}$, associated with frame s_0, consists of the sequence of clusters $(\theta_1, \theta_2, \cdots, \theta_p)$. The most probable sequence of motion frames $\mathbf{s} = (s_0, s_1, \cdots, s_N)$ through the sequence $\mathbf{p}$ can be defined in such a way that it maximizes

$$P(\mathbf{s}, \mathbf{p}) = P(\mathbf{s})P(\mathbf{p}|\mathbf{s}) = \prod_{i=1}^{N} P(s_{i-1} \rightarrow s_i)P(\theta(s_i)|s_i), \quad (7)$$

where $\theta(s_i)$ is the cluster to which frame s_i belongs. The most probable path can be found by traversing all possible paths that start from s_0, are contained within $\mathbf{p}$, and are within a given time bound. The joint probability $P(\mathbf{s}, \mathbf{p})$ is evaluated for each of these paths, and the most probable one is selected.

4.2.4 Practical Use of Clustering

Clustering was not used in all of our examples; the playground and step stool examples made use of clustering, while the maze and rough terrain examples did not. We found that the value of clustering depended on both the richness of the dataset and on the type of interface used. For sketch-based interfaces in the maze and rough terrain examples, clustering was not necessary, and we could simply search the lower layer graph for the motion that best matched the sketch. For the choice-based interface in the playground example, clustering was needed to limit the number of options presented to

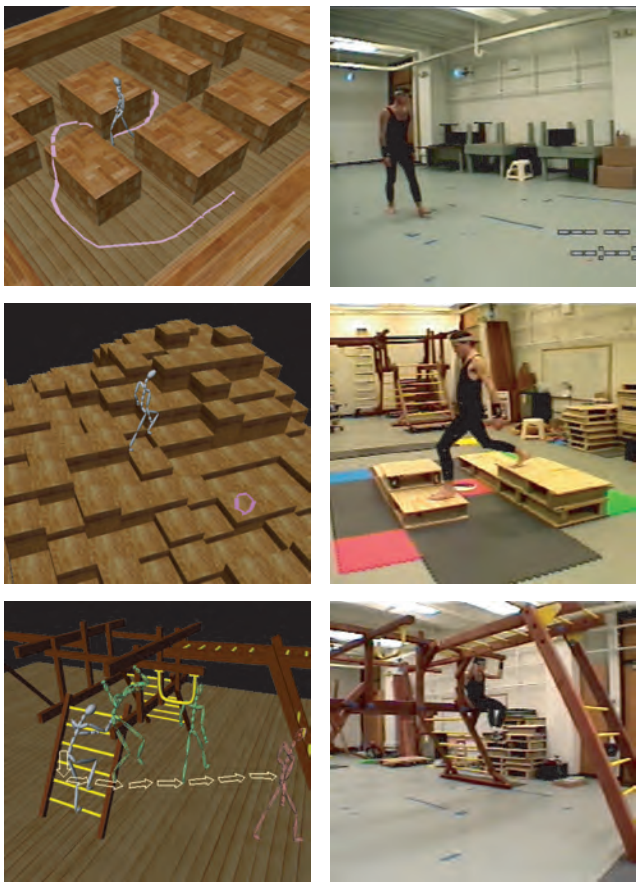


Figure 5: (Top to bottom) Maze, terrain, and playground examples. The left column shows the avatar being controlled in synthetic environments, and the right column shows the original data captured in our motion capture studio. Avatar motion in the maze and terrain environments is controlled using the sketch-based interface, and avatar motion in the playground environment is controlled using choice.

the user. For the vision-based interface used in the step stool example, clustering was required for efficiency in matching user actions to possible avatar actions. User actions were compared against tens of paths in a cluster tree instead of millions of paths in the lower layer graph.

5 Controlling the Avatar

Users of our system can control an avatar interactively through interfaces based on choice, sketching of paths, and performance (figure 5). This section describes the interfaces we explored and the process of mapping from user input to avatar motion.

5.1 Choice

In a choice-based interface, the user is continuously presented with a set of actions from which to choose (figure 5, bottom). As the avatar moves, the display changes so that the choices remain appropriate to the context. When the user selects an action, the avatar performs that action. Whenever there is no currently selected action, the avatar's motion is probabilistically determined.

One of the primary requirements for the choice-based interface is that the display should be uncluttered to avoid confusing the user.

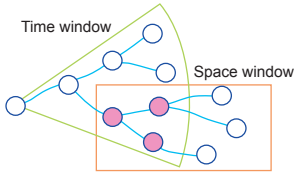


Figure 6: Windows in space and time can be used to constrain the set of clusters considered for presentation of possible actions to the user. In this figure, the pink nodes are selected, because they fall in the intersection of the time bounds and the space bounds set by the user. The two rightmost of these nodes, the leaves of the selected subtree, will be portrayed as possible avatar actions.

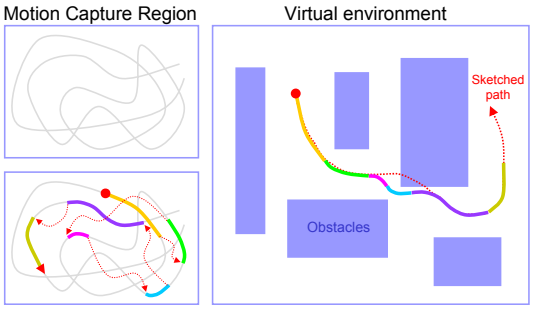


Figure 7: The sketch interface for the maze example. (Top left) Motion data was recorded from a small, empty capture region in which a motion capture subject walked around. (Bottom left) Through graph search, we select a series of motion segments that are connected through the lower layer graph. (Right) These motion segments are translated and rotated to match the sketched path in an environment with obstacles. Note that paths with collisions were eliminated during the search, and the selected path differs significantly from the sketched path where necessary to avoid the obstacle.

In practice, this means that roughly three or four actions should be presented to the user at any given time. We use the cluster tree at the current frame to obtain a small set of actions for display that are typically well-dispersed. By default, leaves of the currently active cluster tree are portrayed as example poses. Additional visual information such as the path that would be traced by the character root or footprints along that path can provide more detail about available actions when desired. The user has some ability to control how many and which actions are displayed by setting bounds on lookahead time and bounds on the desired location of the avatar in the environment (figure 6).

5.2 Sketching

The sketch-based interface allows the user to draw paths on the screen using a mouse. The two-dimensional paths are projected onto the surfaces of environment objects to provide three-dimensional coordinates. In our examples, the sketched paths are assumed to represent the trajectory of the center of mass projected onto the ground plane. We have implemented two different ways to determine avatar motion from a sketched path. In the first approach, used in the step stool environment (figure 8, top right), the most appropriate action currently available to the avatar is selected. In the second approach, used in the maze and rough terrain environments (figure 5, top and middle), avatar motion is constructed incrementally.

Our first approach to sketch-based avatar control exploits the fact that it is often possible to find a good match to a sketched path based on the cluster tree at the current frame. For a given cluster tree, the

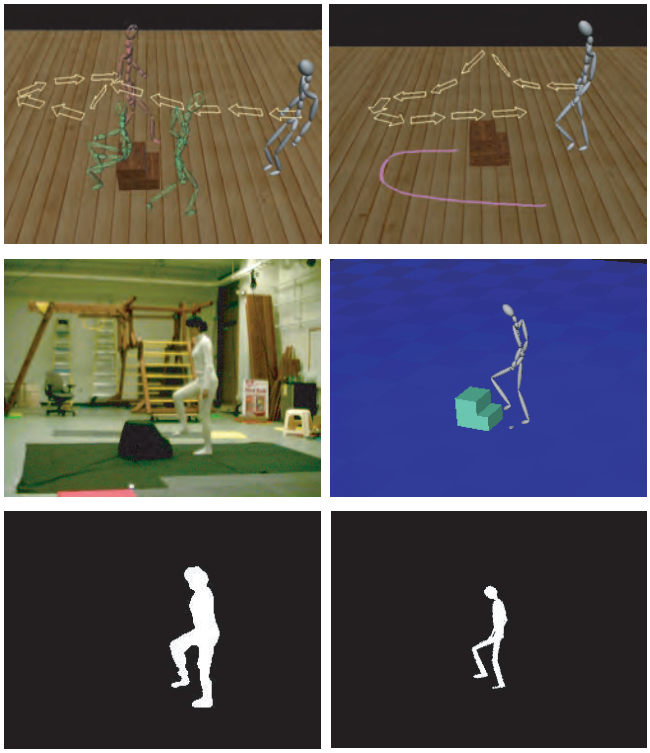


Figure 8: Step stool example. (Top left) Choice-based interface. (Top right) Sketch-based interface. (Middle and bottom left) The user performing a motion in front of a video camera and her silhouette extracted from the video. (Middle and bottom right) The avatar being controlled through the vision-based interface and the rendered silhouette that matches the user's silhouette.

actions available are represented by cluster paths through that tree. We compute a score for each cluster path m based on the most probable motion s_m as follows. First, the center of mass of the character is projected onto the surfaces of environment objects for each frame in s_m. Second, the distance between the resulting center of mass path and the sketched path is computed using dynamic time warping (described in [Bruderlin and Williams 1995]). This distance is the score assigned to cluster path m, and the cluster path with the lowest score is selected as the best match to the user sketch.

Our second approach allows us to track longer paths and to use a sketch-based interface in an environment where clustering has not been performed. In either case, the sketched path is considered as a sequence of goal positions to be traversed, and our tracking algorithm is based on best-first search through the lower layer graph. The following simple objective function is used:

$$E(p) = \|\mathbf{p}_g - \mathbf{p}\| - c((\mathbf{p}_g - \mathbf{p}) \cdot \mathbf{u}), \quad (8)$$

where $\mathbf{p}_g$ and $\mathbf{p}$ are the goal position and the actual position of the avatar respectively, and $\mathbf{u}$ is a vector of the avatar's facing direction. The first term of the objective function rewards motion toward the goal. The second term rewards facing directions that point toward the goal. At each time instance, our algorithm traverses a fixed number of frames to maintain a constant rate of motion. Note that this approach could result in local minima, where the avatar can no longer make progress toward the goal. The possibility of local minima is not a practical problem, however, because the user is doing the high-level planning and has already identified an appropriate path.

5.3 Vision

In our vision-based interface, the user acts out the desired motion in front of a camera. Visual features are extracted from video and used to determine avatar motion (figure 8, middle and bottom). The cost of an avatar's action is measured based on the difference between visual features present in rendered versions of that action and visual features present in the video.

Visual features for a single video image are obtained from a silhouette of the human body, which is extracted from the image using foreground segmentation. We use nine different features drawn from this silhouette. The first seven features are the Hu moments [Hu 1962], which are used because they yield reasonable shape discrimination in a scale and rotation invariant manner. The remaining two features are the coordinates of the central location of the silhouette region in the image plane. When measured over time, these coordinates provide information about translational motion of the user.

Visual features for a motion frame are obtained during preprocessing of the motion database. Because we do not know in advance the orientation of the user with respect to the camera, each motion frame in the database is rendered from a number of different camera viewpoints. In our examples, 18 different camera viewpoints were used. Visual features are then extracted from the rendered images just as for real camera images and stored with each motion frame.

For avatar control, we maintain a buffer of video frames as the user performs. For every time step of avatar motion, the video buffer is compared to the motions available to the avatar. If the avatar is currently at frame i, a cost is computed for each cluster path m associated with frame i and for each camera viewpoint using dynamic timewarping. The cost of cluster path m is the minimum cost over all camera viewpoints. The cluster path with minimum cost determines the avatar's next motion, and the process repeats. In our examples, the system selects an action for the avatar based on the current video buffer every 0.1 seconds as the avatar moves through the environment.

Vision-based action selection can be done in real-time, but selecting an action for the avatar requires matching avatar actions in the future to user actions in the past, which means that there will always be some lag between user motion and avatar motion. In our examples, this lag is currently three seconds.

6 Results

All of the motion data used in our experiments was collected at the rate of 120 frames/second and then down-sampled to 15 frames/second for real-time display. In the maze and terrain examples, motion data are represented in a relative coordinate system. In the other two examples, a fixed coordinate system is employed.

Maze. The maze example in figure 5 (top) shows a path sketched by a user and an avatar being controlled to follow the path and avoid obstacles. The avatar is animated with a motion database of 9 minutes (4.5 minutes duplicated by mirror reflection) in which our motion capture subject walked in a straight line, stopped and turned, side-stepped, and walked backwards. This example demonstrates that the avatar can select an appropriate motion at any time depending on user input and the environment. For example, side steps are selected when it passes through a narrow passage.

To assess the results, we recorded a subject moving in an environment with physical obstacles and then compared her motion to the motion of the avatar moving in a similar environment (figure 9). The database for the avatar's motion was recorded from a different subject in an environment without obstacles. The avatar's motion lacks context-dependent details of the recorded motion (for example, in the recorded motion the subject looked front and back to see if she would clear the obstacles, but the motion database did

Figure 9: Comparison to directly recorded motion data. (Left) A subject passing through obstacles in an environment laid out to match the virtual world. (Middle) Recorded human motion. (Right) Avatar's motion created from the database.

not include this head movement), but is still acceptable as a natural human motion.

Terrain. For the rough terrain example, we recorded motions of about 15 minutes duration on small sections of rough terrain constructed from ten wooden blocks (24 by 24 by 7.5 inches) with three height levels: the ground level, one block height, and two block height (figure 5, middle). Those motions are sufficient to allow an avatar to navigate in an extended rough terrain environment with much more height variation. The user can either sketch a path or mark a target position to direct the avatar.

Playground. For the playground example (figure 5, bottom), 7 minutes of motion were collected, and the motion data was grouped into 25 clusters. The playground example demonstrates that the avatar can be controlled in a complex environment in which dynamic, full-body interaction is required. Although the sketch interface is flexible and intuitive, drawing a path was not appropriate for avatar control in a playground environment because many distinct actions can be mapped to the same projected path. Instead, a choice-based interface was used, with key poses providing sufficient information for the user to select the desired action.

Step stool. In the step stool example, a 5-minute sequence of motion was captured in which our motion capture subject walked around, stepped on, sat on, and jumped over the step stool. This data was grouped into 14 clusters. All three interfaces were tested for the step stool example (figure 8). The advantage of the choice-based interface is that users are informed about the options available at any time. The sketch-based interface is very easy to use and responds instantly to the user input by finding a matching action and updating the display while the path is being drawn. Sometimes, the sketch-based interface was not able to distinguish different styles; jumping over and stepping over the step stool result in similar center of mass paths when they are projected on the ground. For the vision-based interface, the user performs desired motions in front of a high speed video camera that can capture 60 images per second. The silhouettes of the captured images are extracted from the video and compared to the preprocessed silhouettes of the avatar to select a matching action. The silhouette comparison usually can discriminate different styles. If the user acts out a motion that is not available in the database, the system selects motion that looks similar. For example, if the user jumps down from the step stool, no exact matching motion is available, and the system generates motion where the avatar simply steps down.

7 Discussion

We have presented a two layer structure that allows a database of unsegmented, extended motion sequences to be used effectively to control an avatar with three quite different interfaces. Although we have not yet attempted to assess the effectiveness of the interfaces through user studies, our intuition is that each interface plays a useful role and that the best interface may well be determined by the application. The choice-based interface does a good job of showing the user what the options are and therefore may be most useful in applications such as a complicated but slow moving video game where the user might not be aware of all possibilities. However this interface is only effective for relatively slow moving behaviors as the user must have the time to make a decision before the option is no longer valid. Sketching a path is a natural way to guide the avatar around the space but does not generalize to much more complicated environments where the two-dimensional mouse path projected to three dimensions does not adequately specify what the avatar should do.

The vision-based interface allows the most complete control over the motion but an interface that requires whole body motions will certainly not be appropriate for all applications. The latency of the current system would be a factor in many applications and may be necessary if a precise match among many similar actions is required. However, in applications such as ours with fairly small numbers of similar actions or where precise matches are not required, we believe that additional video cameras and the computation of more sophisticated visual cues such as motion and edge detection should allow us to reduce latency substantially.

Although the motion databases used here are relatively large from a research perspective, they are much smaller than motion databases used in video games. Because our databases are of limited size, we need to be careful to retain generality wherever possible by using a local coordinate frame to express the motion of the character root, for example. A larger database for any of our examples would result in more variety and flexibility in the avatar's motions. As database size grows, the ability to rapidly search for an appropriate action will become more and more critical. The two-layer structure proposed here—where the cluster tree at each frame provides a generalization of the set of actions available to the avatar at that frame—should scale well to larger databases, because the data structure stored at each frame contains only clusters immediately relevant to the avatar and grows in size only with the branching factor in transitions between those clusters. Keeping database size manageable is a concern, however, and determining when a database has sufficient variety for a given space of motions is an open and interesting research problem.

The Markov process on the lower level of the data representation has proven to be very effective for databases of the sizes we have explored. Smaller databases might not span the space of possible motions sufficiently to provide the necessary transitions. A much larger database might require more aggressive pruning of transitions to prevent the computation time of the preprocessing step from scaling as the square of the number of frames.

We made somewhat arbitrary decisions in deciding when to use cluster trees as part of the search process. The maze and rough terrain examples do not use clustering at all. This decision was partially motivated by the fact that clustering in a relative coordinate system is more difficult than clustering in a fixed coordinate system, and we found that clustering was not in fact necessary when the sketch-based interface was used. We assume, however, that as the size of the database increases further, clustering will become more important.

The effectiveness of clustering for avatar control is influenced by the distribution of motion capture data collected. If we captured action A 100 times and action B once, it would be difficult to expect the EM algorithm to identify two separate actions. We found that it was important to ensure that actions were captured several times; our subjects were always asked to repeat their actions at least three times.

We picked the initial member of each cluster by hand. Although this was not a burdensome chore, it does require that the application designer know the motion database sufficiently well to select widely dispersed poses as cluster seeds. It should be possible to automate this process. The quality of clustering without hand seeding did improve with computation time spent, but estimation of reliably

good clusters required several hours even for the small datasets.

The three interfaces and four behavior domains shown clearly demonstrate the power of a large database of unsegmented motion recorded in extended sequences. We anticipate that as the size of publically available motion databases grows, we will see very compelling examples developed that build on statistical techniques for preprocessing the data such as those presented here.

Acknowledgments

The authors would like to thank Rory and Justin Macey for their assistance collecting and cleaning the motion capture data. We would also like to thank all of our motion capture subjects. This work was supported in part by NSF EIA CADRE Grant #0196217.

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. In *Proceedings of SIGGRAPH 2002*.

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

BEN-ARIE, J., PANDIT, P., AND RAJARAM, S. S. 2001. Design of a digital library for human movement. In *Proceedings of the first ACM/IEEE-CS International Conference on Digital Libraries*, 300–309.

BLUMBERG, B. M., AND GALYEAN, T. A. 1995. Multi-level direction of autonomous creatures for real-time virtual environments. In *Proceedings of SIGGRAPH 95*, 47–54.

BLUMBERG, B. 1998. Swamped! Using plush toys to direct autonomous animated characters. In *SIGGRAPH 98 Conference Abstracts and Applications*, 109.

BOWDEN, R. 2000. Learning statistical models of human motion. In *IEEE Workshop on Human Modelling, Analysis and Synthesis, CVPR2000*.

BRADLEY, E., AND STUART, J. 1997. Using chaos to generate choreographic variations. In *Proceedings of the Experimental Chaos Conference*.

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

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

BRUDERLIN, A., AND CALVERT, T. W. 1989. Goal-directed, dynamic animation of human walking. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, vol. 23, 233–242.

BRUDERLIN, A., AND CALVERT, T. 1996. Knowledge-driven, interactive animation of human running. In *Graphics Interface '96*, 213–221.

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *Proceedings of SIGGRAPH 95*, 97–104.

CASSELL, J., VILHJÁLMSSON, H. H., AND BICKMORE, T. 2001. Beat: The behavior expression animation toolkit. In *Proceedings of SIGGRAPH 2001*, 477–486.

CASSELL, J. 2000. Embodied conversational interface agents. *Communications of the ACM*, 4 (April), 70–78.

CHI, D. M., COSTA, M., ZHAO, L., AND BADLER, N. I. 2000. The emote model for effort and shape. In *Proceedings of SIGGRAPH 2000*, 173–182.

CHOPRA-KHULLAR, S., AND BADLER, N. I. 1999. Where to look? Automating attending behaviors of virtual human characters. In *Proceedings of the third annual conference on Autonomous Agents*, 16–23.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. The virtual stuntman: dynamic characters with a repertoire of autonomous motor skills. *Computers & Graphics 25*, 6 (December), 933–953.

FALOUTSOS, P., VAN DE PANNE, M., AND TERZOPOULOS, D. 2001. Composable controllers for physics-based character animation. In *Proceedings of SIGGRAPH 2001*, 251–260.

FRALEY, C., AND RAFTERY, A. E. 1998. How many clusters? Which clustering method? Answers via model-based cluster analysis. *Computer Journal 41*, 8, 578–588.

GALATA, A., JOHNSON, N., AND HOGG, D. 2001. Learning variable length markov models of behaviour. *Computer Vision and Image Understanding (CVIU) Journal 81*, 3 (March), 398–413.

GLEICHER, M. 1997. Motion editing with spacetime constraints. In *1997 Symposium on Interactive 3D Graphics*, ACM SIGGRAPH, 139–148.

GLEICHER, M. 1998. Retargeting motion to new characters. In *Proceedings of SIGGRAPH 98*, 33–42.

GLEICHER, M. 2001. Comparing constraint-based motion editing methods. *Graphical Models 63*, 2, 107–123.

HODGINS, J. K., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. In *Proceedings of SIGGRAPH 95*, 71–78.

HU, M. K. 1962. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory 8*, 2, 179–187.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *Proceedings of SIGGRAPH 2002*.

LAMOURET, A., AND VAN DE PANNE, M. 1996. Motion synthesis by example. In *EGCAS '96: Seventh International Workshop on Computer Animation and Simulation*, Eurographics.

LASZLO, J. F., VAN DE PANNE, M., AND FIUME, E. L. 1996. Limit cycle control and its application to the animation of balancing and walking. In *Proceedings of SIGGRAPH 96*, 155–162.

LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 99*, 39–48.

LI, Y., WANG, T., AND SHUM, H.-Y. 2002. Motion texture: A two-level statistical model for character synthesis. In *Proceedings of SIGGRAPH 2002*.

MOLET, T., BOULIC, R., AND THALMANN, D. 1996. A real-time anatomical converter for human motion capture. In *EGCAS '96: Seventh International Workshop on Computer Animation and Simulation*, Eurographics.

MOLET, T., AUBEL, T., GAPIN, T., CARION, S., AND LEE, E. 1999. Anyone for tennis? *Presence 8*, 2, 140–156.

MOLINA TANCO, L., AND HILTON, A. 2000. Realistic synthesis of novel human movements from a database of motion capture examples. In *Proceedings of the Workshop on Human Motion*, 137–142.

OXFORD METRIC SYSTEMS, 2002. www.vicon.com.

PERLIN, K., AND GOLDBERG, A. 1996. Improv: A system for scripting interactive actors in virtual worlds. In *Proceedings of SIGGRAPH 96*, 205–216.

PERLIN, K. 1995. Real time responsive animation with personality. *IEEE Transactions on Visualization and Computer Graphics 1*, 1 (Mar.), 5–15.

PULLEN, K., AND BREGLER, C. 2000. Animating by multi-level sampling. In *Computer Animation 2000*, IEEE CS Press, 36–42.

PULLEN, K., AND BREGLER, C. 2002. Motion capture assisted animation: Texturing and synthesis. In *Proceedings of SIGGRAPH 2002*.

ROSALES, R., ATHITSOS, V., SIGAL, L., AND SCLAROFF, S. 2001. 3D hand pose reconstruction using specialized mappings. In *IEEE International Conference on Computer Vision*, 378–385.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics & Applications 18*, 5 (September - October), 32–40.

SARCOS, 2002. www.sarcos.com.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *Proceedings of SIGGRAPH 2000*, 489–498.

SEMWAL, S., HIGHTOWER, R., AND STANSFIELD, S. 1998. Mapping algorithms for real-time control of an avatar using eight sensors. *Presence 7*, 1, 1–21.

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 (ECCV)*.

SUN, H. C., AND METAXAS, D. N. 2001. Automating gait animation. In *Proceedings of SIGGRAPH 2001*, 261–270.

TARJAN, R. 1972. Depth first search and linear graph algorithms. *SIAM Journal of Computing 1*, 146–160.

UNUMA, M., ANJYO, K., AND TAKEUCHI, R. 1995. Fourier principles for emotion-based human figure animation. In *Proceedings of SIGGRAPH 95*, 91–96.

WILEY, D., AND HAHN, J. K. 1997. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications 17*, 6 (November), 39 –45.

WITKIN, A. P., AND POPOVIĆ, Z. 1995. Motion warping. In *Proceedings of SIGGRAPH 95*, 105–108.

WOOTEN, W. L., AND HODGINS, J. K. 1996. Animation of human diving. *Computer Graphics Forum 15*, 1, 3–14.