Learning Silhouette Features for Control of Human Motion

Liu Ren[1], Gregory Shakhnarovich[2], Jessica K. Hodgins[1]
Hanspeter Pfister[3], Paul A. Viola[4]

July 2004
CMU-CS-04-165

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[1]Carnegie Mellon University;
[2]Massachusetts Institute of Technology;
[3]Mitsubishi Electric Research Laboratories;
[4]Microsoft Research.

Funding of the first and third authors was provided in part by NSF-IIS0326322, NSF-IIS0205224, the ONR VIRTE project, and Mitsubishi Electric Research Laboratories. The first author was also partially supported by a School of Computer Science Alumni Fellowship from Carnegie Mellon University.

Keywords: performance animation, motion control, motion capture, animation interface, machine learning, computer vision

Abstract

We present a vision-based performance interface for controlling animated human characters. The system combines information about the user's motion contained in silhouettes from several viewpoints with domain knowledge contained in a motion capture database to interactively produce a high quality animation. Such an interactive system will be useful for authoring, teleconferencing, or as a control interface for a character in a game. In our implementation, the user performs in front of three video cameras; the resulting silhouettes are used to estimate his orientation and body configuration based on a set of discriminative local features. Those features are selected by a machine learning algorithm during a preprocessing step. Sequences of motions that approximate the user's actions are extracted from the motion database and scaled in time to match the speed of the user's motion. We use swing dancing, an example of complex human motion, to demonstrate the effectiveness of our approach. We compare the results obtained with our approach to those obtained with a set of global features, Hu moments, and to ground truth measurements from a motion capture system.

Figure 1: A dancing user drives the motion of two animated characters.

1 Introduction

People, even without training, can move in expressive ways to play charades, pantomime a winning touchdown, or act out a story for a child. The goal of our research is to provide untrained users with a vision-based interface for interactively controlling complex human motions. Simple vision processing supplies partial information about the user's movements; and domain knowledge from a previously captured motion database supplements this information to allow plausible movement to be inferred. The process can be performed interactively; the computation is real time with less than a second of latency between the capture of the video and the rendering of the animated motion.

The experimental scenario in this paper is a single user animating a couple swing dancing. We present a low-cost, camera-based system that allows a user to "drive" the dance of the couple interactively (Figure 1). The user can change the high-level structure of the dance, introducing a different sequence of steps, rotations, or movements. The final sequence tries to maintain the quality of the previously captured dance yet reflects the user's higher-level goals and spontaneity. We envision that a system such as this one might provide the infrastructure for teleconferencing applications, an interactive game for a dancing child, an animated karaoke performance for an adult, or with a sufficiently broad database, an animation authoring interface for a naive user.

The vision-based interface relies on silhouettes extracted from three video streams to match the body configuration of the user against a database of poses and movements. This matching process uses a set of discriminative local features that are computed on the silhouette images [42]. The best local features for estimating the yaw orientation and body configuration of the user are selected from a very broad set of possible features by training on synthetic data set. The selection is based on a variant of the AdaBoost algorithm [34; 9]. The estimation of the yaw orientation incorporates a fast search method called locality sensitive hashing (LSH) [17]. The estimation of the body configuration relies on the temporal coherence evidenced in a domain-specific database of human motion. The database is pre-processed into an augmented motion graph that extends the original motion graph [23; 22] to handle scaling in velocity and two interacting characters. The estimation of the pose of the animated character is performed with a local search in the augmented motion graph in the region around the frame currently being animated. A small buffer of input video provides sufficient information for the match and introduces a delay of less than one second.

Our experimental results include several users dancing in street clothes and the resulting motions of an animated couple. For evaluation, we capture video and motion capture data simultaneously and compare the motion computed by our system with the ground-truth motion capture data. We also compare the performance of our system with that of a commonly used set of global features, Hu moments [19], to demonstrate the effectiveness of our learned local features for mapping 2D silhouettes to 3D poses.

Alternative approaches to creating performance-based animation systems include vision-based tracking algorithms, small marker sets in concert with inverse kinematics, real-time motion capture systems, and special purpose sensors such as accelerometers or the pad used in Footsee [46]. Our system has potential advantages over these approaches because it requires no instrumentation of the user (other than reasonably tight-fitting clothing, a controlled background, and fixed lighting), produces high quality motion given an appropriate database, and requires only off-the-shelf cameras rather than special-purpose sensors or cameras. The major limitation of our approach is that it relies on a domain-specific database; however, with the increasing availability of high quality motion capture data, we believe that this will not prove to be a major impediment for applications in which the behaviors of the user can be reasonably predicted in advance.

The paper makes several technical contributions. First, we present an interactive performance animation system that allows the generation of high quality and complex human motion from only three video inputs. Unlike other approaches, we use local features that operate only on small, or local, regions of the silhouette image to estimate yaw orientation and body configuration from silhouettes. The key insight in our approach is that the information contained in a motion database can be used to learn those discriminative local features for a particular task. Second, for fast silhouette-based yaw estimation, we present a novel solution that combines AdaBoost and LSH. Finally, we propose an augmented motion graph structure that allows for smoothing the estimated motion for high quality output from the video input, handling speed variation in the user's performance and synthesizing the motion of two human characters from the performance of one user.

2 Background

We build on research in recovering motion from video, in creating animations from motion capture data, and on performance animation systems. In this section, we discuss the most relevant research from each of three areas.

Motion Recovery from Video Almost all computer vision approaches rely on tracking to analyze human motion from video input (see [16] for a comprehensive survey). Three dimensional (3D) human motion tracking is most relevant to our work [45; 5; 12; 39]. In general, these systems assume accurate initialization, and then subsequently track pose changes based on an articulated 3D human model. For example, Bregler and Malik [5] used an exponential map formalism for kinematic chains to track simplified human body models. The tracking approach of Delamarre and Faugeras [12] is closely related to our own in setup: three synchronized cameras were used to compute silhouettes. In their approach, a synthetic articulated model of the human form was first rendered online and then the synthetic silhouettes were compared with the real ones. The difference between the silhouettes was used to generate forces, which were then applied to each part of the 3D articulated model for a better match with the real silhouettes. Most 3D tracking approaches require time-consuming per frame nonlinear optimization, which is not appropriate for real-time or interactive applications such as performance animation systems. Numerical drift is also a significant problem in these approaches even with manual initialization. As a result, few papers describe systems that can track complex motions or motion with significant yaw rotation for more than a few seconds.

Motion analysis that only requires tracking coarse poses can be performed more robustly. Ramanan and Forsyth described a system that allowed multiple people to be tracked for motion annotation [31]. A robust algorithm for 2D pose tracking was first used and then the resulting 2D poses were lifted to a sequence of coarse 3D poses. This approach was quite effective for classifying human movements such as walking, running, and jumping in an indoor environment.

Alternatively, a robust vision-based interface can directly estimate the 3D body pose without initialization and incremental updates, thus avoiding problems with initialization and drift [33; 28; 35]. For example, Rosales and Sclaroff [33] trained a neural network to map each 2D silhouette to 2D positions of body joints and then applied an EM algorithm to reconstruct a 3D body pose based on 2D body configurations from multiple views. Mori and Malik [28] explored an example-based approach to recover 2D joint positions by matching extracted image features (shape context features) with those of cached examples and then directly estimating the 3D body pose from the 2D configuration considering the foreshortening of each body segment in the image. More recently, Shakhnarovich and colleagues [35] combined multiple cues (edges and appearance) and developed an extension of locality sensitive hashing to rapidly estimate the upper body poses of human figures from a large database of example images of figures viewed from the front. Efros and colleagues [15] proposed a spatio-temporal motion descriptor based on noisy optical flow measurements for pose estimation from one video camera. Their framework was designed for human action recognition from far views. Despite these advances, direct estimation of precise 3D body pose from 2D noisy image features remains a difficult problem. Such systems sometimes yield coarse, imprecise or incomplete body pose estimations. We improve on this situation by utilizing the spatial-temporal structure of pre-recorded human motion and learning efficient and effective image features from the information in the database.

We extract silhouettes from the video data and use those to drive the character's motion. Silhouettes have been widely used for motion estimation from video data [3; 12; 8; 33; 27; 23; 7]. Motion estimation from a single camera is challenging because a single silhouette is not enough to disambiguate the 3D pose. Brand [3] inferred a 3D motion sequence that was most consistent with a series of input silhouettes using a hidden Markov model that was estimated from 3D motion capture data. Due to the sparsity of their input data, the system produced motion that was not an accurate reproduction of human motion and was therefore not suitable for animation. Lee and colleagues [23] explored a single-camera vision interface and retrieved the 3D motion data from the database by extracting silhouettes and comparing global features (Hu moments) to control an avatar. Simple global features are easy to extract, but they are not discriminative enough to catch the subtle changes in the silhouette that occur during complex human motions. This limitation can lead to imprecise pose matching as discussed by Rosales and his colleagues [33]. Lee and colleagues introduced a long delay (3 seconds) to reduce the problems caused by both the imprecise matching metric and pose ambiguities from the one camera interface. Although effective for simple motions such as those they tested (walking over, onto and sitting on a step stool), their approach does not generalize to complex human motions such as dancing or to interactive environments.

Silhouettes from multiple cameras can be used to reduce the ambiguity problem in pose estimation. Such systems usually estimate motion via polygonal or volumetric modeling of the observed shape but the additional cameras increase the complexity of modeling and the computational power required [8; 26; 27]. For accurate 3D reconstruction of body shape, high quality video cameras are often required. Carranza and colleagues [7] incorporated domain knowledge in the form of a human body model but their approach was computationally expensive and therefore not suitable for interactive applications.

Domain knowledge, especially that embedded in a human motion database, has also been used by researchers to estimate 3D human motion from video input [25; 13; 4; 38; 23; 40]. Leventon and Freeman [25] applied dimensionality reduction to 3D motion capture data and then used a Bayesian approach to synthesize a 3D motion sequence from tracked 2D marker positions in a monocular image sequence. Similarly, Sidenbladh and colleagues [38] learned a low-dimensional linear model of 3D human motion from a database and presented a probabilistic tree search method for efficient synthesis and tracking. More recently, Stenger and colleagues [40] proposed a probabilistic method for tracking that searched a database that was hierarchically partitioned into areas with constant density. These approaches as well as the work reported here demonstrate that the domain knowledge inherent in a human motion database allows efficient motion synthesis from video data.

Resequencing Motion Capture Data Resequencing motion capture data is a very successful method for synthesizing new animations and controlling animated characters [23; 22; 1; 2]. In this approach, novel motions are created by selecting motion clips from a database in the form of a graph and reassembling them to generate a new motion that approximately follows a path specified by the animator or that matches a set of constraints specified by the animator. This approach produces compelling animation in part because the motion that is selected is not modified and therefore retains the subtle details of the original motion data. We build on this work by using the estimated pose of the user to find similar sequences of motion in a modified motion graph. We extend the motion graph concept to handle two interacting subjects and allow time scaling of the motion sequences to better match the user's performance.

Performance-Based Animation Performance-based animation systems have been used quite successfully for characters that respond in real time to the actions of human actors [18; 29; 36]. Although problems with retargetting, filtering, and removing collisions remain, all commercially available technologies for motion capture now have the ability to perform real time capture of a human actor. While these systems are very effective for choreographed performances, the time required to instrument the user with markers or sensors prevents their use in applications where the system will be used by many different people. Like other vision-based systems, our approach requires only a controlled background and reasonably tight-fitting clothing and does not require the additional overhead of suiting up the user with markers or sensors.

Another approach to performance-based animation is to use a less obtrusive sensing technology than markers or magnetic sensors. Yin and her colleagues [46] used a foot pressure sensor pad to interactively control avatars. The pressure patterns for the foot-to-ground contacts of the user were correlated to a motion capture database with matching footpad information to reconstruct full body motion. This interface is inherently limited in that not all poses, particularly subtle changes in upper body poses, provide unique foot pressure patterns but it does provide appealing results for a number of kicking and stepping motions.

3 EXAMPLE-BASED APPROACH

As described in Figure 2, our vision-based performance interface allows a user to control an animated human character. The system extracts silhouettes from three camera inputs. The vision-based performance interface estimates the dancer's motion from these silhouettes. Silhouette-based motion estimation is a difficult problem because the mapping from 2D silhouettes to 3D body configurations is ambiguous. In this paper, we use an example-based framework to solve this problem by relying on domain knowledge in the motion capture database. In this section, we will first present an overview of our example-based approach and then discuss the major challenges and the proposed solutions.

In our example-based framework, captured motion is stored in a database along with the corresponding synthetic silhouettes. The body configuration for a new silhouette from the cameras is estimated by searching the database for a similar silhouette and retrieving the corresponding body configuration as well as changes in global position and orientation. Despite the simplicity of the example-based approach, there are a few challenges in using this approach for motion estimation from silhouettes:

Example storage Because the database determines the set of allowable body configurations, it must cover the space of required motions for the application domain. For general human motion, an example-based approach may require a large amount of data and therefore a prohibitively large database. The example storage problem can be reduced by building a behavior-specific motion database of the 3D motion data and 2D silhouette data. We build a database that only covers the basic movements of east coast swing. Even for an application-specific database, a naive representation would still require a lot of storage. We represent each silhouette example as a pre-computed bit string based on an efficient binary local feature set [42]. All the silhouettes only require 34MB storage space.

False matches Example-based motion estimation from silhouettes requires that similar silhouettes correspond to similar poses. However, silhouettes are ambiguous because two quite

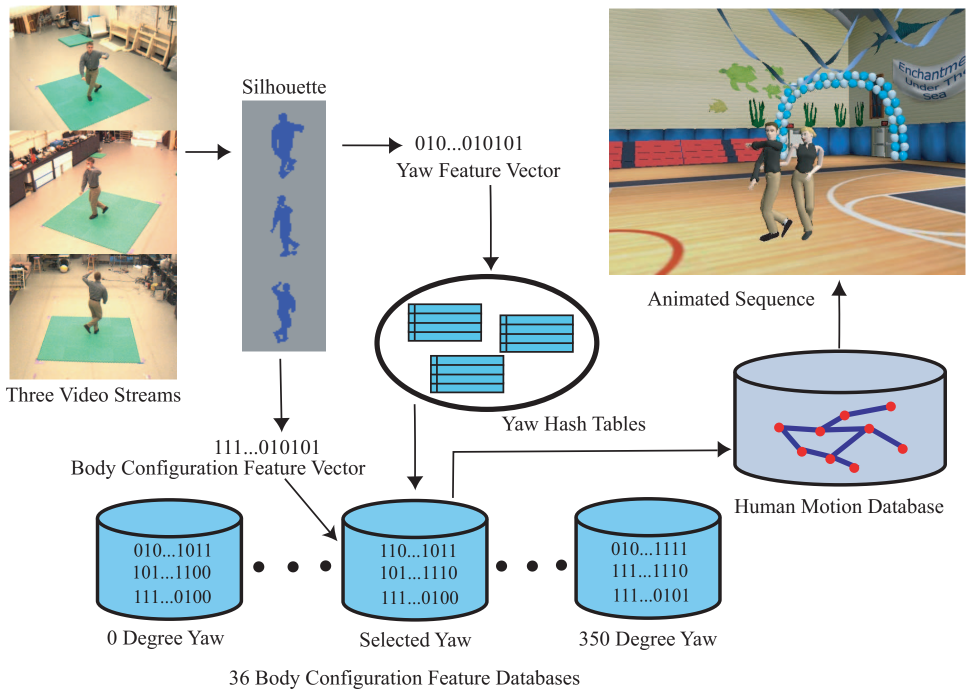


Figure 2: System diagram. Our system uses an example-based approach. In the preprocessing step, we construct one feature database for the yaw estimation and 36 feature databases of discrete yaw angles for the body configuration estimation. The examples, the binary feature vectors exacted from the synthetic silhouettes and the associated pose parameters, are stored in those databases. At run time, the three video streams are processed to silhouettes. Discriminative local features are extracted from those silhouettes and form binary feature vectors for yaw and body configuration estimation. The system first uses the yaw feature vector to search for a similar yaw orientation through a hashing algorithm. The system then searches for a similar body configuration from the feature database of estimated yaw angle. The spatial-temporal structure of human motion data facilitates this process. The animation of the user and his partner are created from the poses found in the database.

different poses may create similar silhouettes. Large scale ambiguity in the silhouettes is caused by the 3D-to-2D projection and mirror symmetry. Small scale ambiguity also exists because a small change on the boundary of the silhouette (for example, the arm) can imply a large change in the limb position (for example, the hand) that is located inside the silhouette boundary. As a result, two silhouettes may appear similar even if their underlying poses are quite different. Such ambiguities can cause false matches, in which the nearest silhouette in the database does *not* correspond to the nearest pose. We reduce the ambiguity by using three cameras and constructing silhouette similarity measures that are local and therefore reduce the problem of false matches. This goal is an explicit part of our learning algorithm (AdaBoost) because it selects a small number of good local features from a large set.

Robustness Silhouettes can be noisy because the separation between background and foreground is not precise. Our silhouette similarity measures are robust to noisy inputs because each local feature used in the similarity measure is computed based only on the statistics of a local patch and is independent of other parts of the silhouette.

Computational efficiency Example-based estimation requires frequent comparisons of silhouettes with similarity measures and this process must be computationally efficient for online applications. The rectangular local features we choose are efficient [42] and provide a small bit string representation of each silhouette. The silhouette similarity measure is then the Hamming distance metric between two bit strings, which is also computationally efficient.

Search efficiency Online example-based estimation requires an efficient K-Nearest Neighbor (KNN) search. Real-time performance is impractical with current PCs for a large silhouette database because the complexity of naive KNN search is a linear function of the size of the database. We use two methods to improve the search efficiency. We use *locality sensitive hashing* (LSH), an approximate nearest neighbor search framework, to speed up the KNN search process for yaw estimation. We then use a local search to retrieve a close body configuration from an *augmented motion graph*, a graph structure whose edges connect body configurations that are nearby in body configuration space to reduce the search space.

We now describe the details of this example-based framework and the results obtained with our system. We first describe the video and motion data processing. Then we discuss learning good similarity measures for the example-based estimation (Section 5) obtained through an offline training process (Section 6). Next we describe the online motion control in Section 7. We then compare the results obtained with our approach to those obtained with a widely used global feature, Hu moments, and to ground truth measurement from a motion capture system in Section 8. Finally, we discuss some of the engineering choices we made in the design of our system and the limitations of the system in Section 9.

4 Video and Motion Data Processing

In our real-time video capture system, three Dragonfly cameras [30] are connected to a single PC through a IEEE-1394 firewire interface. Synchronized video frames are captured at 15 fps with a resolution of 640×480. Figure 3 shows the layout of the system. We use software for multiple camera calibration to build a workspace coordinate system and determine the position of the real and the corresponding synthetic cameras [6]. This information is used for rendering of synthetic silhouettes for training.

The video frames are processed to form real silhouettes. We assume a static background and fixed lighting to simplify the extraction of the silhouettes. Shadows often cause errors in silhouettes. We avoid this problem by using a simple two-step method. We first use a low threshold to extract an initial silhouette, which may contain shadows. Then we compute the zero and first order spatial moments to determine the silhouette center and recompute the foreground pixels below the center using a higher threshold. This two-step process eliminates most shadows near the feet but

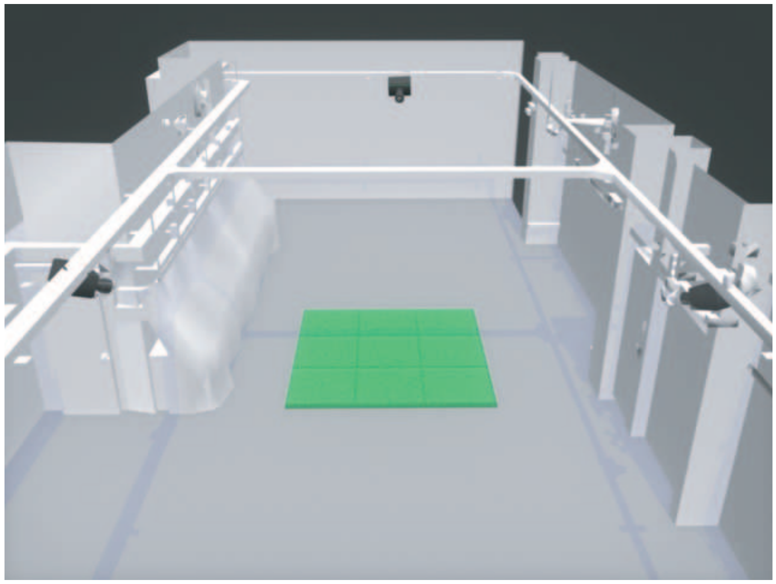


Figure 3: In our system configuration, the three Dragonfly cameras (black) point downward to capture a $8' \times 8'$ workspace (green). The three cameras form an approximate isosceles triangle with the smallest acute angle of about 30 degrees. The camera positions are not tuned for the system except that the distance between each camera and the workspace should be large enough that an affine camera model provides a reasonable approximation.

does not mislabel the foreground pixels of the face and arms. An alternative method would be to use an angular difference metric in the hue channel of pixel color but this approach is significantly more computationally expensive [8].

As the user dances, he moves around the capture region and his silhouette changes size in the image. We compensate for this by cropping and scaling silhouettes (Figure 4). The resulting silhouettes from the three views are concatenated and treated as one image in the subsequent analysis. For simplicity, we use the term silhouette to refer to this combined representation from the three cameras.

We build a human motion database by capturing the motion of two subjects dancing together. We use a Vicon optical motion capture system [41] with twelve cameras, each of which can record at 120Hz with images of 1000×1000 resolution. We use a standard 41 marker setup on each subject. The motion data were down-sampled to 15 fps to match the frame rate of our video cameras. The size of the motion capture area is $8' \times 24'$. The motion database contains approximately 6 minutes of swing dancing motion (east coast swing). The motion data were processed into a hierarchical skeleton (joint angles and pelvis position and orientation for a total of 62 degrees of freedom). Contact constraints are not enforced. As a result, the fixed or sliding contacts made by the hands and arms of the dancing couple are not always realistic.

Recorded human motion is used in several different ways in the system. The motion data are used to create synthetic examples for training by rendering a human model and extracting a silhouette from the rendered image. The motion database is also used to create a motion graph

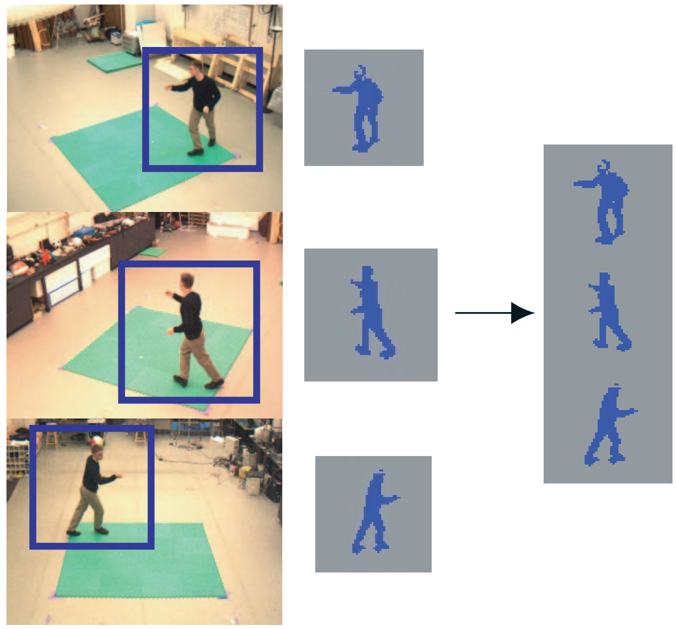


Figure 4: The silhouette is extracted from each video input and then cropped. To crop a silhouette, the system computes the first order moment and the second order central moment from the horizontal and vertical distribution of foreground pixels, respectively. The centroid of the foreground pixels and their standard deviation in the two directions are then calculated from those image moments. We use the center and 2.6 times the largest standard deviation in the two directions to determine the square bounding box for the cropping. Finally, the cropped silhouette is resampled to a 60×60 pixel window using bilinear interpolation and the three silhouettes are stacked together as a combined representation for further analysis. The silhouettes are not perfect because the background is not controlled. In particular, when the user's head passes in front of a dark object, the silhouette of the head is often incomplete as is shown in the top silhouette of this example. Similar problems occur when the color of the user's clothing is similar to the background color. Shadows sometimes affect the shape of the feet in the silhouette.

for synthesis of the motion of the dancing couple. Finally, we perform motion capture and video capture simultaneously and use motion capture data that are not included in the database as ground truth to assess the performance of the system.

5 Learning Silhouette Similarity Measures

Silhouette similarity measures are used to retrieve examples that match the pose of the input from the motion database. The matches are found based on silhouettes from the cameras and synthetic silhouettes computed from the motion capture database. The similarity measures must minimize false matches and be, computationally efficient. In this section, we describe a machine learning approach to constructing such similarity measures. The learning algorithm selects a set of efficient binary features such that the Hamming distance between the feature vectors computed from two silhouettes is smaller when the underlying poses of the silhouettes are sufficiently close. We now describe the learning task and its solution in detail.

5.1 The Learning Task

Similarity relationship in the pose space is captured by the binary neighborhood function

$$N_\theta(\pi_1, \pi_2) = 1 \text{ if } |\pi_1 - \pi_2| < \theta, \text{ and -1 otherwise.} \tag{1}$$

That is, N_θ labels a pair of poses, π_1, π_2, as positive if the two poses are within θ from each other under a distance measure $|\cdot|$ in pose space, and as negative otherwise. Our motion database consists of examples of poses with corresponding synthetic silhouettes (π_i, S_i). For pairs of such examples we can evaluate N_θ on the poses directly, thus producing "ground truth" similarity labels. The same labels are assigned to the silhouette pairs as to corresponding poses: (S_i, S_j) is positive if and only if (π_i, π_j) is also positive.

At run time our system must infer the pose from silhouettes only. That can be done by means of a *silhouette similarity measure* F. Applied on a pair of silhouettes S_1 and S_2, $F(S_1, S_2)$ infers the value of N_θ for the corresponding poses π_1 and π_2: large positive values of $F(S_1, S_2)$ imply $N_\theta(\pi_1, \pi_2) = 1$. The learning task is to construct an F which is as much in agreement with N_θ as possible.

We first consider a similarity measure based on a single binary valued feature h_j: the Hamming distance between the values of $h_j(S_1)$ and $h_j(S_2)$. Zero distance implies that the corresponding poses π_1, π_2 form a positive pair (i.e., are similar); and a distance of one implies that the poses form a negative pair. Thus, this similarity measure is associated with a *silhouette pair classifier* H_j:

$$H_j(S_1, S_2) = \begin{cases} +1 & \text{if } h_j(S_1) = h_j(S_2), \\ -1 & \text{if } h_j(S_1) \neq h_j(S_2). \end{cases} \tag{2}$$

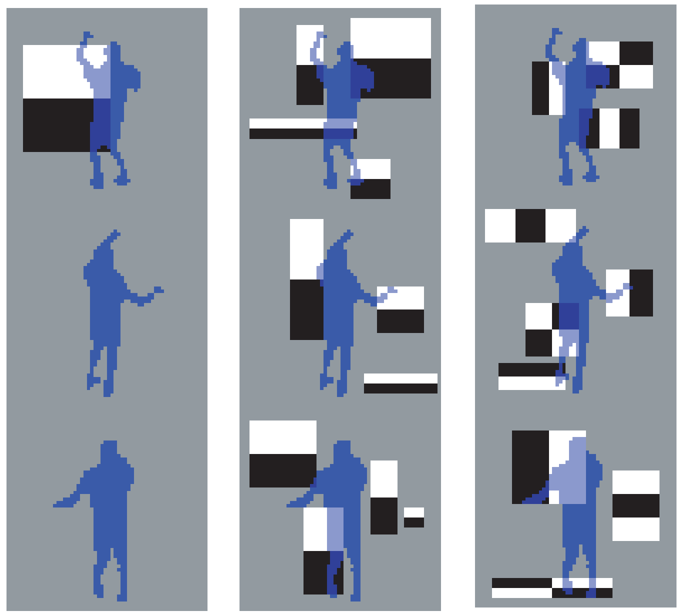
The accuracy of this similarity measure is equivalent to the classification accuracy of H_j with respect to the labels assigned by N_θ. This accuracy may be empirically evaluated for a given h_j using a set of pairs with known labels.

In general given a restrictive set of binary features, no single H_j can approximate the neighborhood function with satisfactory accuracy. Approximation of N_θ may be achieved by combining many H_js computed from a set of binary features h_j:

$$F(S_1, S_2) = \sum_{j=1}^{M} H_j(S_1, S_2). \tag{3}$$

Let $h_1, \ldots, h_M$ be M binary features; these features define an encoding of a silhouette S into a binary vector $V(S)$ of length M, in which the j-th bit is $V(S)[j] = h_j(S)$. The Hamming distance between encodings of two silhouettes is directly related to Equation (3).

Taking the sign of $F(S_1, S_2)$ defines a binary classifier on silhouette pairs, which is a combination of M simpler classifiers. In the machine learning literature such a classifier is known as an



(a) (b) (c)

Figure 5: (a) A rectangular filter on a binary silhouette image. Two rectangles are used in the filter. We compute the value of the filter by summing the intensities of all pixels in the white regions and subtracting the sum of the intensities of all pixels in the black regions. (b) The same type of filter can be defined with a variety of scales, aspect ratios, and locations. (c) Other types of filters used in our system can have different orientations and more rectangles, which can be defined in a similar way. With this definition, the potential filter set is very large although it can be pruned by considering only those filters that overlap silhouettes in the training set.

ensemble classifier. The accuracy of the Hamming distance in the space of $V(S)$ as a similarity measure is equivalent to the classification accuracy of F:

$$\sum_{i,j} N_\theta(\pi_i, \pi_j)\mathrm{sign}\left(F(S_i, S_j)\right), \tag{4}$$

and can be assessed empirically based on a training set of labelled silhouette pairs.

In this work, we select the simple binary features h_j from a large pool of local features originally proposed by Viola and Jones [42]. These features have been used successfully in such real-time object detection problems as face detection and pedestrian detection [42; 43]. They are defined as

$$h_j(S) = \begin{cases} 1 & \text{if } \phi_j(S) > T_j, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

where T_j is a threshold and a *rectangular filter* $\phi_j(S)$ is a scalar linear function of the pixels in a silhouette image S. Rectangular filters are displayed graphically in Figure 5 as rectangles with white and black regions. By using summed area tables [10], often called the integral image, we

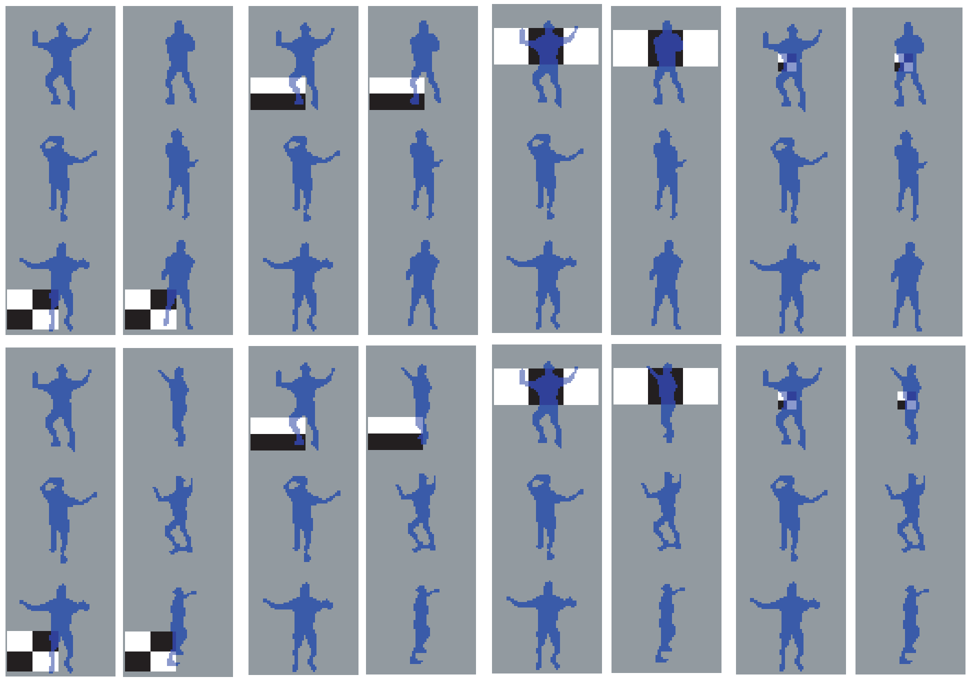


Figure 6: The figure shows examples of features that were selected by AdaBoost for the yaw similarity measure. Each feature is also associated with a threshold. The top row shows positive training examples. The second row shows negative training examples.

can compute the value of each rectangular filter in constant time independent of its scale and location [42]. The number of available features is very large, because the available filter set itself is very large and there are many choices for the threshold of each filter. We use a machine learning process to select a subset of M filters and a best threshold for each.

5.2 Feature Selection using AdaBoost

An ensemble classifier can be efficiently solved using a variant of the AdaBoost algorithm [34; 9] adapted to classify example pairs instead of single examples. In the work of Viola and Jones [42] a collection of binary features is used to classify each example image into one of two classes ("face" or "not face"). We adopt the form of simple pair classifier used by Shakhnarovich and colleagues [35] to classify example pairs as similar or not in the AdaBoost framework.

AdaBoost proceeds in rounds, adding one feature at a time. Initially each training example pair is assigned a weight of $1/C$, where C is the number of example pairs. In each round, the features are computed for each example pair and each feature is assigned an error based on the weights of the example pairs which it classified incorrectly. AdaBoost selects the feature with the lowest weighted error. Example pairs that are correctly classified by the new ensemble classifier receive a lower weight while the weights of incorrectly classified example pairs are increased. These new weights are then used to select another feature in the next round. The weight for each selected feature $h_j(S)$ is computed based on the weighted error made by that feature: lower errors receive higher weights.

In particular, the best binary feature is determined by selecting not only a filter but also a threshold for that filter in each round. The cost for any given threshold T of a filter ϕ has two terms. The first is the number of example pairs which are near in pose space but split by the threshold: $N_\theta(\pi_i, \pi_j) = 1$ but $\phi(S_i) < T < \phi(S_j)$ or $\phi(S_j) < T < \phi(S_i)$. The second is the number of examples which are distinct in pose space but on the same side of the threshold: $N_\theta(\pi_i, \pi_j) = -1$ but either $\phi(S_i) \leq T$ and $\phi(S_j) \leq T$ or $\phi(S_j) > T$ and $\phi(S_i) > T$). For a given filter ϕ, the optimal T can be computed by first sorting the examples by $\phi(S_i)$ and then traverses the sorted list while recording of pairs that are erroneously split or combined by the threshold. We select T with the smallest number of errors as the optimal threshold for filter ϕ.

Our feature selection algorithm can be summarized as follows: construct a training set of pairs of silhouettes, labeled by N_θ, and run AdaBoost for M iterations yielding M binary features. In each iteration, we determine the optimal threshold T_i for each selected filter ϕ_i, thus defining a pair classifier H_i. AdaBoost then selects the H_i with the lowest weighted error, reweights the examples (pairs), and proceeds to the next iteration.

AdaBoost produces an ensemble classifier which is a weighted combination of pair classifiers. We discard the weights found by AdaBoost and give equal weights to all the selected features because we intend to use the underlying simple features for efficient retrieval of silhouettes.

6 Training

We use synthetic silhouettes rendered from the motion capture database to create training examples. The selection of a good set of training examples is crucial for creating estimators that are robust to changes in yaw orientation (rotation of the pelvis about the vertical axis) and translation as the user spins and steps around in the workspace. The potential training set therefore includes each body configuration in the database rendered in five locations in the workspace with 36 different yaw orientations (Figure 7). These transformations create a potential training set of 921,600 silhouettes from 5120 frames of motion data.

Pairs of these silhouettes could, in principle, be used to learn only one similarity measure for pose that includes both body configuration and orientation. However, this approach appears to be infeasible due to the wide variety of silhouettes that this similarity measure would have to handle. We choose instead to separate the estimation of yaw orientation from that of the internal body configuration. We construct one yaw estimator and 36 rotation-specific body configuration estimators, one for each 10 degrees of yaw rotation. Each body configuration estimator is then trained to distinguish body configurations within a narrow range of yaw angles (± 5 degrees), which is a much easier problem than determining body configuration independent of yaw angle. To find the complete body pose, the system first estimates yaw orientation and then uses the appropriate body configuration estimator for that yaw angle to determine the rest of the pose. We describe the similarity measure for yaw estimation in the next section and then discuss the construction of body configuration similarity measures in Section 6.2.

6.1 Similarity Measure for Yaw Estimation

Yaw estimation is a multilabel classification task with 36 classes each corresponding to a 10 degree bin centered at 0, 10, 20, ..., 350 degrees. A pair of silhouettes is labelled positive if their yaw angles fall into the same bin. The body configuration is not taken into consideration. See Figure 8 for examples.

Because we use pairs of silhouettes for training, the set of 921,600 synthetic silhouettes would provide more than 23 billion positive example pairs and three orders of magnitude more negative example pairs. We use only a small, representative set of body configurations as training examples. We find the set by using the *k*-means algorithm [14] to find 50 clusters in the set of body configurations and then picking the body configuration closest to the center of each cluster. The distance metric for clustering is the L2 distance between two body configuration parameters (See Figure 9). For each of these 50 body configurations, we generate a multi-view silhouette of a synthetic human model rendered at a random orientation within each of 36 yaw bins (yaw of 0, 10, 20, ..., 350 ± 5 degrees), and at each of 5 locations (the center and the corners of the workspace). Because AdaBoost involves computing the value of each filter for every training example, the resulting set of pairs $(50 \times 36 \times 5)^2$ is still too large. We randomly select 4,000 positive and 6,000 negative training pairs from this example set and these examples are used in AdaBoost learning.

Given the richness of the moves seen in swing dancing, a training set of 10,000 is unlikely to span the space of silhouettes that will be seen when the user dances. We address this problem by *resampling* the training data set [37; 20] after a number of rounds. We compute the AdaBoost weights on a much larger set of examples based on the classification by the set of binary features selected so far, then select a new training set according to these weights, taking care to maintain a balanced representation of positive and negative examples.

6.2 Body Configuration Similarity

The silhouette similarity measure relevant for body configuration is learned separately for each of the 36 bins for yaw rotation. Using the synthetic model, we render the 5,120 body configurations at 5 locations, yielding 25,600 examples from which the training pairs for each bin can be constructed. The true label for a pair of poses is determined by a threshold on the L2 distance between the two sets of body configuration parameters (the threshold was set by inspection). The distance metric is the same as that used for *k*-means clustering. Examples of a positive and a negative training pair are shown in Figure 10.

The number of positive pairs for body configuration is much lower than it is for yaw, and we use clustering only to reduce the number of negative pairs. The binary similarity functions are selected by AdaBoost as described in Section 6.1.

7 Online Motion Control

For online motion control, the system first estimates the yaw orientation of the human body from the silhouette. Then it finds a close match for the body configuration by searching the database using the feature set selected for that yaw orientation. Those two estimation processes must be fast and must result in a smooth motion sequence. We first discuss how we preprocess the data to improve the performance for online motion control (Section 7.1). Then we explain the details of the robust online yaw estimation (Section 7.2) and the body configuration estimation (Section 7.3).

7.1 Preprocessing

Because the evaluation of the similarity function only involves comparing feature vectors, the feature vectors for the silhouette images in the database can be precomputed. The feature vector is approximately of length 300 and requires far less space to store than the full silhouette.

Given an input silhouette I from the video cameras and a yaw similarity measure of M features, the yaw estimation can be performed as follows. We first compute the M binary features from the silhouette image I and concatenate them to create a *bit vector* $[h_1(I)\dots h_M(I)]$. The K examples most similar to the input are those with the smallest Hamming distance between their bit vector representations and that of the input. The yaw angle can be estimated as the majority vote among these top K matches. However, retrieving K similar examples is a computationally challenging problem that involves the search of more than 900,000 high-dimensional bit vectors. For efficiency, we preprocess the data and use a fast algorithm for approximate similarity search: Locality-Sensitive Hashing (LSH) [17]. LSH is described in Section 7.1.1.

Given an input silhouette and an estimated yaw angle, the system then uses the selected features in the appropriate yaw bin to compute a body configuration feature vector and to find a close pose in the motion capture database. If performed globally, this search would not necessarily result in a smooth animation sequence. Instead, the system searches locally near the previously selected pose in an *augmented motion graph*. The augmented motion graph is constructed as a preprocessing step and includes two extensions to the original single-character motion graph structure [23; 22]. One extension allows the animation of two interacting characters and the other permits speed changes in the motion playback to accommodate tempo variations in the music and dancing motion. These extensions are described in detail in Section 7.1.2.

7.1.1 Locality-Sensitive Hashing

LSH proceeds by randomly selecting k functions among those features chosen by AdaBoost, thus defining a k-bit hash function:

$$g(x) = [h^{(1)}(x), h^{(2)}(x), ..., h^{(k)}(x)], \tag{6}$$

The entire database is indexed by a hash table with 2^k buckets: a silhouette I is associated with the bucket $g(I)$. In order to increase the probability of success (finding examples similar to the input) LSH constructs l hash tables with independently constructed keys $g_1,\dots,g_l$. Then, given an input I, the set of candidate matches consisting of the union of the buckets $g_1(I),\dots,g_l(I)$ is exhaustively searched for examples similar to I.

The parameters k (the number of bits per hash key) and l (the number of hash tables) affect the probability of finding a good match for the input and the expected search time. A large k makes spurious matches less likely and thus speeds up the search but also increases the probability of missing good matches. A large l increases the probability of success but also increases the cost of the exhaustive search. We set these parameters based on an empirical evaluation that uses two disjoint sets of silhouettes with known yaw: an example set and a test set. For k in a feasible range, we build a hash table from an example set with a k-bit hash key randomly constructed according to Equation (6). Then we use the test set to evaluate the probability P_k of finding an example similar to the test input and the expected number R_k of candidate examples that need to be searched per input. This procedure is repeated a number of times. For a given P_k and R_k with l independent hash tables, the probability of success is $1-(1-P_k)^l$; we choose a fixed high probability (0.96) as our target value and find the smallest number l_k for which the target is achieved. We also need to balance the probability of success with the search time. Therefore, we choose the value of k that minimizes the expected search time given by $l_k R_k$, while achieving the desired probability of success. With 260 features selected by AdaBoost for yaw estimation, the best performance was achieved with $l = 15$ hash tables and with $k = 18$ bits per hash key. These parameters led to a significant acceleration in performance over a brute force search. Instead of searching 900,000 examples for yaw estimation, the system only needs to compare about 100 examples.

7.1.2 Augmented Motion Graph

A single-character motion graph models the motion data as a first-order Markov process [23; 22]. The transition from one state to the next depends only on the current state (a single motion frame). The probability of a transition between two frames is computed from a distance function measuring the similarity of the body poses and velocities. The insight that made this approach effective is that the absolute location and orientation of the character is not important for many recorded motions and the state can be represented in a body local or relative coordinate system. The relative coordinate system then allows transitions to similar motions recorded anywhere in the workspace and requires far less data to achieve good control over the character's motion [23].

Because the motion of the two dancing partners performing east coast swing is highly correlated, we can easily build a two-character motion graph. We extend the relative coordinate system to two characters by representing the *partner*'s position and orientation as relative to that of the *driven character* (the character that the user controls). The distance function used to compute the transitions is extended to two characters by comparing the body configuration and velocity for each character, the relative external pose of the partner, and the relative velocity between the two characters. In the distance function calculation, the body configuration is a high-dimensional vector representing limb end positions in the root segment coordinate system. We use a strategy similar to that used by Lee and colleagues [23] for pruning the transition edges. Foot contact states are also used in the pruning process. However, we only use the foot contact states of the driven character not those of both characters. We blend transitions and maintain foot contact constraints for both characters using a hierarchical motion fitting algorithm [24]. We represent the partner's foot contact constraints in the root coordinate system of the driven character during the blending process.

The captured motion reflects the choreographic rules of swing dancing, that is, the motion consists of basic movements and begins and ends with specific poses. As observed by Kim and colleagues [21], the transitions learned from the example motions in the motion graph may not perfectly reflect those choreographic rules. We use the beginning and ending poses of basic movements to connect nodes that meet choreographic rules but were left out in the automatic motion graph construction. In practice, the transitions computed automatically cover most basic movements, and only 0.25% of the transitions are added manually.

The two-character motion graph is further augmented by adding extra edges and nodes to control the speed of the motion in response to the user's performance (Figure 11). Extra edges allow frames to be skipped for faster playback and extra nodes allow the motion to be slowed down. Only the feature vectors are stored for the extra nodes, and the joint angles are interpolated at runtime.

7.2 Robust Online Yaw Estimation

Online yaw estimation improves the estimate by removing outliers caused by the inherent 180 degree ambiguity in silhouettes. First, we use LSH to retrieve a set of nearest examples (20 in the results reported here) and build a histogram of their yaw angles. Then we smooth the histogram using a low pass filter. We choose the two peak values from the smoothed histogram, θ_1 and θ_2, and form a candidate set $\theta_1, \theta_2, \theta_1 \pm 180, \theta_2 \pm 180$. We compare those candidates with the yaw value estimated in previous frames and choose the closest one as new yaw estimation. Finally, we combine the yaw estimations from nearby video frames and apply a Gaussian low pass filter. In our experience, a simple majority vote from 20 nearest neighbors yields an acceptable yaw estimate most of the time, but the method described above is more robust (Figure 12).

7.3 Online Body Configuration Estimation

Given the yaw estimate, the system searches locally in the augmented motion graph and computes the corresponding 3D motion frame with a full set of parameters (global translation and orientation and body configuration). The search is performed in two stages: first using multiple video frames to perform a coarse mapping and then refining that match using the extra nodes in the augmented motion graph.

To perform the coarse mapping, the system first computes the matching cost for all the child nodes of the previous matched node, W, in the augmented motion graph (Figure 11) and selects the node $(W+1)$ with the smallest cost. This step may skip frames if the best match can be achieved with a faster motion (Figure 13). We use a look-ahead and look-back of r frames (a total of $2r+1$ video frames) and compute the Hamming distance between the two sets of body configuration feature vectors (bit vectors). The matching cost is a weighted average using a Gaussian function centered on the current node W. There will likely be multiple segments of length $2r+1$ centered at W because of branching in the augmented motion graph (Figure 13). The smallest of those costs for those segments is assigned to W. The look-ahead video buffer adds an r frame delay to the system, but improves robustness and removes much of the mapping ambiguity between the video input and the motion capture data.

The system refines this coarse match by searching the extra nodes around node $(W+1)$ for a lower cost match. This refinement uses only the current video frame for the computation of matching cost.

The extra nodes and edges in the motion graph allow the original motion to be scaled in time to match the user's motion. To prevent a sudden change in velocity, we average the scaling factors across m frames (thus introducing an additional m frame delay for a total delay of $r+m$ frames). With $r = 10$ and $m = 3$ and a frame rate of 15 fps, the delay is 0.87 seconds for the examples presented in this paper. The delay introduced by Gaussian smoothing of the yaw estimates is also included in the 0.87 seconds delay because the yaw orientation is estimated independently of body configuration.

8 Results

Using these algorithms for training and search, we built a vision-based performance interface for controlling the motion of a dancing couple interactively. We tested the performance of our system with users whose motion was not in the database and who were dancing to different music than that used when the database was recorded. Figure 14 shows three dancing sequences from two users.

To learn the yaw and body configuration features, we ran the AdaBoost algorithm until the learned ensemble classifier achieved a small testing error. Figure 15 shows the error of the yaw classifier on the test set as the number of features increases; qualitatively similar behavior was observed while learning body configurations features (Figure 16). The training was done on a Beowulf cluster with 90 nodes, each with a 1 GHz CPU and 512M RAM. Table 1 gives some of the parameters used in training, the computation time required, and the performance of the classifiers. The result of the offline learning phase was a set of 260 binary features for the yaw similarity measure and a set of 320 features for each of the 36 body configuration similarity measures.

The yaw estimation uses the 20 top matches retrieved by LSH. We compared the yaw estimate obtained by our system to a ground truth as provided by motion capture data (Figure 17). For 73% of the frames, the LSH-aided yaw estimation has an error of less than 10 degrees. The estimation error is less than 20 degrees for 92.5% of the frames and smaller than 30 degrees for 98% of the frames. Because errors in yaw estimation may affect the body configuration estimation, we tested our body configuration similarity measure in the presence of errors in yaw; the performance as a function of yaw error is plotted in Figure 18 for one yaw bin. Similar performance was seen for other yaw angles. This test demonstrates that the body configuration estimation is robust to the yaw estimation errors seen in our system.

We compare the performance of our system with that of Hu moments, a simple set of global features that have often been used in silhouette-based systems [11; 32; 23]. Hu Moments were introduced by Hu in the early sixties to carry out pattern recognition of objects [19]. Hu moments are seven moment invariants that are invariant under translation, rotation and scaling of the 2D object in the image. Hu moments are also global features because the real value of each moment invariant is computed from the coordinates of all the pixels in the body and boundary of the 2D object not just from the pixels in a local patch.

The comparison with Hu moments used motion data captured simultaneously with video data as the ground truth. To create a common reference frame, we place a black and white checkerboard calibration pattern with motion capture markers in the field of view of both video and motion

	Yaw	Body configuration (60 degree yaw)
Features	260	320
Training examples (positive/negative)	800,000/2,000,000	200,000/10,000,000
Training examples per round (positive/negative)	4,000/6,000	3,000/5,000
Resampling interval (rounds)	40	80
Testing examples (positive/negative)	300,000/37,000,000	50,000/2,000,000
Testing error rates (false negative/false positive)	4.86%/9.92%	0.5%/0.9%
Training Time (hours)	2.5	1.8

Table 1: The number of examples used in training the yaw classifier and the body configuration similarity functions and the error rates achieved. The statistics for each of the 36 body configuration similarity functions are approximately the same as those for 60 degree yaw which is represented in the table.

capture cameras. The subject then performs while both video and motion capture data are recorded. The ground truth motion capture data are not included in the database so the best match from the database will not be an exact match.

We find the best match with respect to the global Hu moment metric and with respect to the local rectangular features. To make the comparison with Hu moments fair, we do not use temporal constraints or filtering in the yaw estimation, but otherwise the approach using local features is as described earlier. To find the global best match, we compute the end positions of each limb in the ground truth motion capture data and use those 3D locations to retrieve the closest body pose from the database by exhaustive search. For each of these methods, we compute the error as the L2 distance in 3D parameter space between the retrieved match from the database and the recorded motion capture data for each testing frame. The histogram of the difference between the error obtained with Hu moments and the error with the local rectangle features is shown in Figure 19. The performance of local features is better than that of Hu moments on 86% of the frames. Figure 20 shows a sample of results from the three methods.

To evaluate whether the temporal information inherent in the augmented motion graph would correct the body configurations that are poorly estimated by the Hu moment, we create animation using Hu moments and compare it to that created using local features. Seven Hu moments are extracted from each of the three silhouettes and stored in the feature database. The process is otherwise identical except that the system computes the Hamming distance between the body configuration bits in the estimated yaw bin in the first case, and the system calculates the Euclidean distance between the Hu moments of the input silhouette and those of the matched motion frame in each yaw bin and chooses the smallest one as the similarity measure in the second case. A similar method was used in the previous approaches based on Hu moments [11;

23]. Figure 21 demonstrates the performance of the two approaches on a motion sequence of about 7 seconds (15fps). The estimation error is quite stable for rectangular local features but shows serious drifting problems for Hu moments.

9 Discussion

We have described a new silhouette-based vision interface that uses a set of learned local features to allow the user to control animated characters that reproduce his motion and that of his virtual dance partner. We have demonstrated that this approach produces visually pleasing motion for long sequences of user input. We have assessed this approach both qualitatively through animations and quantitatively through error measures and a comparison with Hu moments.

Instead of using 2D local features from a multiview silhouette, we could have based our approach on visual hulls or 3D volumes. However, the computation cost for online reconstruction and the storage cost for those 3D examples would likely be prohibitively expensive for an interactive vision-based interface. Matching the 3D representations to the motion in the database would be difficult because only a low-quality and view dependent 3D reconstruction can be obtained from three camera views. We chose to work on 2D silhouettes primarily for efficiency reasons.

Silhouette similarity measures can be implemented in many different ways. They can be based on measures such as Hamming distance between foreground pixel masks, Hausdorff distance between points on contours, or Euclidean distance between global image features such as Hu moments. However, none of these similarity measures directly addresses the problem of "false matches," where similar metric values do not always imply close poses. Because our feature set is selected from a very large set with the goal of avoiding false matches, our approach is likely to be more robust to this problem than a single, fixed metric. Direct comparison between foreground pixel masks is not memory efficient; Hausdorff distance metric is not computationally efficient; Hu moments, which are based on global statistics of silhouette pixels, are efficient but not robust because noisy silhouettes can affect the values of all components. The local feature set we chose is particularly appealing for real-time or interactive vision-based applications because it requires only a small amount of storage and is computationally efficient.

In our example-based approach, we use LSH for fast yaw estimation. The procedure has also been used in other fast parameter estimation applications. We compare our fast parameter estimation approach with the work of Shakhnarovich and his colleagues [35], where a feature or hash function was selected based on its classification performance on the training data. Their approach does not account for dependence between those hash functions. If the feature space is highly redundant, as the rectangular feature set is, many variants of the same feature may be selected. In contrast, we first use AdaBoost to learn a set of independent hash functions and then randomly select the locality sensitive hash functions among them. We also set parameters for LSH based on an empirical evaluation rather than using a closed-form formula to choose the bounds for those parameters as done by Shakhnarovich and colleagues [35].

The major limitation to this approach is that it requires a database that is appropriate for the problem domain. We believe that this is not a serious limitation for many application areas where the likely human behaviors can be predicted in advance. For example, we envision that such a system might allow a little girl to spin and twirl in her family room and see an animated version of her motion applied to a cartoon hippo, a ballerina, or a little girl of the same age. Similarly, adults might be able to see an animated version of themselves gesturing while crooning into the microphone during a karaoke session. Although less structured than dancing, expressive gestures made during meetings might also be contained in a database of sufficient size and used for teleconferencing. We have not yet tested to see how much generality we can get from a given database. For applications where precise reproduction of the user's motion is not needed, a basic database may be sufficient and a specialized one will not be necessary.

The body type of the synthetic character used for training was not specifically tuned to our subjects (Figure 22); however, we expect that the current system would not be robust to large changes in body type. We believe that this problem could be solved by training for a small set of different body types and then choosing the appropriate set of classifiers automatically based on the silhouettes extracted from a few calibration poses.

In the current system, the user must wear fairly tight-fitting clothing (Figure 22 b). This restriction is an inherent weakness of silhouette-based methods because the system must see the same silhouette each time that the user assumes a particular pose. Clothing that has "state" independent of the user's current pose will introduce errors. We have observed that the system appears robust to a loose shirt but not loose pants, which cause the silhouette to lose the separation between the legs.

Some restrictions in the setup prevent the current system from being used in the average family room. We assume a static background and reasonable color separation between the user and the background. We believe that more sophisticated silhouette extraction algorithms might improve the situation. The current implementation requires three cameras and the selection of the local features is specific to the camera positions. The framework might work reliably with two cameras, although we have not extensively tested this setup. We have also not assessed how robust the system is to changes in camera position and orientation. It should be robust to translation and rotation of the cameras about the vertical axis centered in the performance area because the performer makes movements that correspond to those changes of view point in the current system without introducing errors.

The implementation of the current system required that we select quite a few parameters: the number of local features chosen for yaw and body configuration classification, the number of positive and negative training examples, the number of rounds of training, and the frequency with which the training set is changed. All of these parameters affect the learning time of the various classifiers. That time is now between 2-3 hours per classifier on a large computing cluster (37 classifiers total). Although AdaBoost weighs the training examples to ensure that those currently misclassified receive greater emphasis, training could likely be made more rapid by fine tuning these parameters or by relying on more sophisticated algorithms for selection of the training set. Alternatively, the time required for training could be reduced at the cost of a small impact on performance by further reducing the available feature set. Wu and colleagues implemented such a system and used a simple and direct feature selection method to replace AdaBoost [44].

Although the yaw angle is estimated in the search process, it is not used in the animation of the character. The position and yaw orientation of the character is computed from the trajectory selected in the motion graph. Therefore, the character's position and yaw orientation will diverge from that of the human dancer as small differences in their motion accumulate. We have not found divergence to be a serious drawback in the dancing application as the incremental translations and rotations appear natural. For an application where the absolute facing direction is important, such as karaoke, this problem could be at least partially addressed by including yaw orientation as part of the matching function when the path through the motion graph is selected.

Both the yaw and the body configuration search take recent values for those parameters into account thus ensuring temporal consistency in the motion. However, if a wrong path is selected in the motion graph, this property could lead the system to stray quite far from the human dancer's motion. We have not seen this problem in practice, perhaps in part because swing dancing is quite structured with well-defined beginning and ending poses for each move. This structure may allow the search to recover from a less-than-ideal choice of paths in the motion graph with the start of the next dance move. If increasing error becomes a problem in other applications, we believe that locality sensitive hashing could also be used to provide a global search for body configuration and to correct errors at the cost of some discontinuity in the motion.

References

[1] Okan Arikan and David A. Forsyth. Interactive motion generation from examples. In *ACM Transactions on Graphics*, volume 21(3), pages 483–490, 2002.

[2] Okan Arikan, David A. Forsyth, and James F. O'Brien. Motion synthesis from annotations. In *ACM Transactions on Graphics*, volume 22(3), pages 402–408, 2003.

[3] M. Brand. Shadow puppetry. In *Proceedings of the International Conference on Computer Vision*, pages 1237–1244, 1999.

[4] Matthew Brand and Aaron Hertzmann. Style machines. In *Proceedings of SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 183–192, 2000.

[5] Cristof Bregler and Jitendra Malik. Tracking people with twists and exponential maps. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 8–15, 1998.

[6] California Institute of Technology. Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc/, 2002.

[7] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. In *ACM Transactions on Graphics*, volume 22(3), pages 569–577, 2003.

[8] Kong Man Cheung, Takeo Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3D voxel reconstruction of human motions. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 714 – 720, June 2000.

[9] Michael Collins, Robert Schapire, and Yoram Singer. Logistic regression, AdaBoost and Bregman distances. In *Computational Learning Theory*, pages 158–169, 2000.

[10] Franklin C. Crow. Summed-area tables for texture mapping. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 207–212, 1984.

[11] James W. Davis and Aaron F. Bobick. The representation and recognition of human movement using temporal templates. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 928–934, 1997.

[12] Quentin Delamarre and Olivier D. Faugeras. 3D articulated models and multi-view tracking with silhouettes. In *Proceedings of the International Conference on Computer Vision*, pages 716–721, 1999.

[13] J. Deutscher, Andrew Blake, , and Ian Reid. Articulated body motion capture by annealed particle filtering. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 126–133, June 2000.

[14] Richard Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley & Sons, Inc., 2000.

[15] Alexei Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *IEEE International Conference on Computer Vision*, pages 726–733, 2003.

[16] D. M. Gavrila. The visual analysis of human movement: A survey. In *Computer Vision and Image Understanding*, volume 73, pages 82–98, January 1999.

[17] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, 1999.

[18] John P. Granieri, Jonathan Crabtree, and Norman I. Badler. Production and playback of human figure motion for visual simulation. In *ACM Transactions on Modeling and Computer Simulation*, volume 5(3), pages 222–241, 1995.

[19] M. K. Hu. Visual pattern recognition by moment invariants. In *IEEE Transactions on Information Theory*, volume 8, pages 179–187, February 1962.

[20] Michael Jones and Paul Viola. Face recognition using boosted local features. In *MERL Technical Report TR2003-25*, 2003.

[21] Tae H. Kim, Sang Park, and Sung Yong Shin. Rhythmic-motion synthesis based on motion-beat analysis. In *ACM Transactions on Graphics*, volume 22(3), pages 392–401, 2003.

[22] Lucas Kovar, Michael Gleicher, and Frederic Pighin. Motion graphs. In *ACM Transactions on Graphics*, volume 21(3), pages 473–482, 2002.

[23] Jehee Lee, Jinxiang Chai, Paul Reitsma, Jessica Hodgins, and Nancy Pollard. Interactive control of avatars animated with human motion data. In *ACM Transactions on Graphics*, volume 21(3), pages 491–500, 2002.

[24] Jehee Lee and Sung Yong Shin. A hierachical approach to interactive motion editing. In *Proceedings of SIGGRAPH 1999*, Computer Graphics Proceedings, Annual Conference Series, pages 39–48, 1999.

[25] Michael E. Leventon and William T. Freeman. Bayesian estimation of a 3D human motion from an image sequence. In *MERL Technical Report TR1998-06*, 1998.

[26] Wojciech Matusik, Chris Buehler, and Leonard McMillan. Polyhedral visual hulls for real-time rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 115–126. Springer-Verlag, 2001.

[27] Ivana Mikic, Mohan Triverdi, Edward Hunter, and Pamela Cosman. Articulated body posture estimation from multicamera voxel data. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 455–461, 2001.

[28] Greg Mori and Jitendra Malik. Estimating human body configurations using shape context matching. In *Proceedings of European Conference on Computer Vision*, volume 3, pages 666–680, 2002.

[29] H. Noser and D. Thalmann. Sensor based synthetic actors in a tennis game simulation. In *Proceedings of the 1997 Conference on Computer Graphics International*, page 189, 1997.

[30] Point Grey Corporation. Dragonfly cameras. http://www.ptgrey.com, 2001.

[31] Deva Ramanan and David A. Forsyth. Automatic annotation of everyday movements. In *Advances in Neural Information Processing Systems 16 (in press)*, 2004.

[32] R. Rosales and S. Sclaroff. Specialized mappings and the estimation of body pose from a single image. In *IEEE Human Motion Workshop*, pages 19–24, 2000.

[33] Romer Rosales, Matheen Siddiqui, Jonathan Alon, and Stan Sclaroff. Estimating 3D body pose using uncalibrated cameras. In *Boston University Computer Science Department Technical Report*, number 2001-008, 5 2001.

[34] Robert E. Schapire and Yaram Singer. Improved boosting algorithms using confidence-rated predictions. In *Machine Learning*, pages 297–336, 1999.

[35] Gregory Shakhnarovich, Paul Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *IEEE International Conference on Computer Vision*, pages 750–757, 2003.

[36] Hyun Joon Shin, Jehee Lee, Sung Yong Shin, and Michael Gleicher. Computer puppetry: An importance-based approach. In *ACM Transactions on Graphics*, volume 20(2), pages 67–94, 2001.

[37] Catherine A. Shipp and Ludmila I. Kuncheva. An investigation into how adaboost affects classifier diversity. In *Proceedings of 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 203–208, 2002.

[38] Hedvig Sidenbladh, Michael J. Black, and Leonid Signal. Implicit probabilistic models of human motion for synthesis and tracking. In *Proceedings of European Conference on Computer Vision*, pages 784–800, 2002.

[39] Cristian Sminchisescu and Bill Triggs. Estimating articulated human motion with covariance scaled sampling. In *International Journal of Robotics Research*, volume 22(6), pages 371–393, 2003.

[40] B. Stenger, A. Thayananthan, P. Torr, and R. Cipolla. Filtering using a tree-based estimator. In *Proceedings of the International Conference on Computer Vision*, pages 1063–1070, 2003.

[41] Vicon Motion Systems. http://www.vicon.com/, 2002.

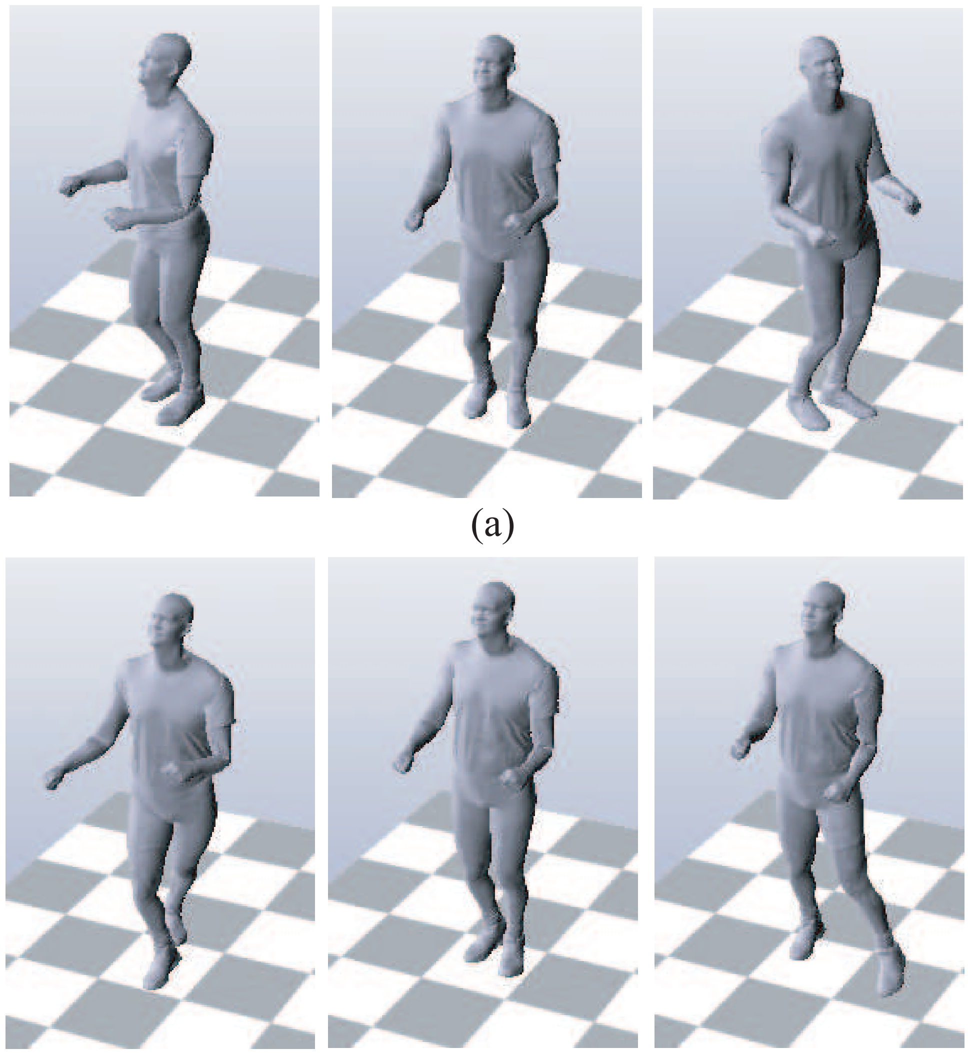
[42] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[43] Paul Viola, Michael J. Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. In *IEEE International Conference on Computer Vision*, pages 734–741, 2003.

[44] Jianxin Wu, James M. Rehg, and Matthew D. Mullin. Learning a rare event detection cascade by direct feature selection. In *Advances in Neural Information Processing Systems 16 (in press)*, 2004.

[45] M. Yamamoto, A. Sato, S. Kawada, T. Kondo, and Y. Osaki. Incremental tracking of human actions from multiple views. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2–7, 1998.

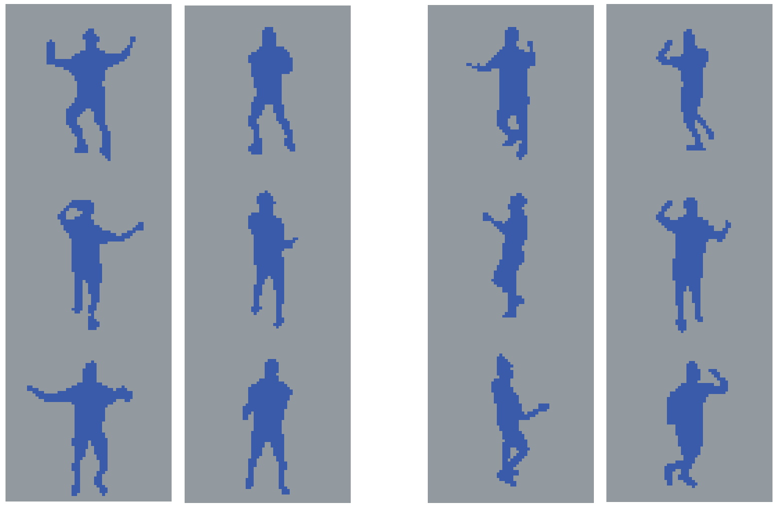
[46] KangKang Yin and Dinesh K. Pai. Footsee: an interactive animation system. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 329–338, 2003.



(a)

(b)

Figure 7: (a) A single body configuration rendered from different yaw orientations. (b) Different body configurations at the same yaw orientation. We generate synthetic silhouettes from synthetic images rendered from three virtual cameras. Because the synthetic silhouettes must be rendered using virtual cameras at fixed positions, significant movement for those cameras in the real setting would require recalibrating the cameras, regenerating these synthetic silhouettes and reselecting these features.



(a) Positive (b) Negative

Figure 8: Examples of training pairs used to select yaw similarity features. (a) Same yaw angle but different body configurations. (b) Different yaw angles but same body configuration.

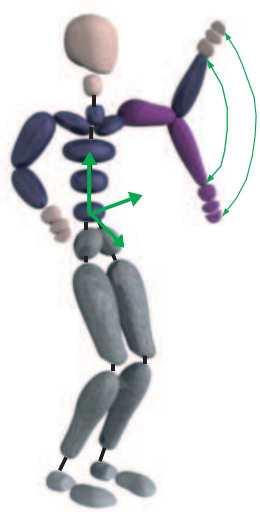
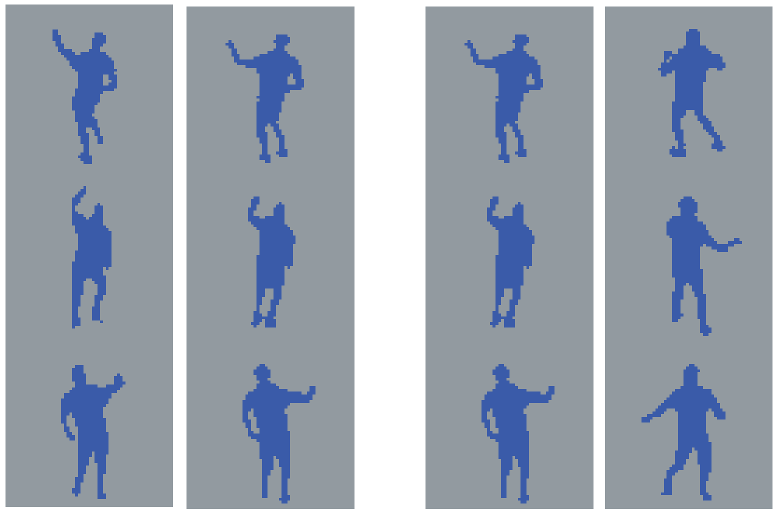


Figure 9: The body configuration parameters are a set of limb end positions in the root segment coordinate system of the human body. They are concatenated to form a vector in high dimension space. The L2 distance metric is used in the clustering.



(a) Positive (b) Negative

Figure 10: Examples of training pairs for body configuration similarity (yaw is 90 degrees).

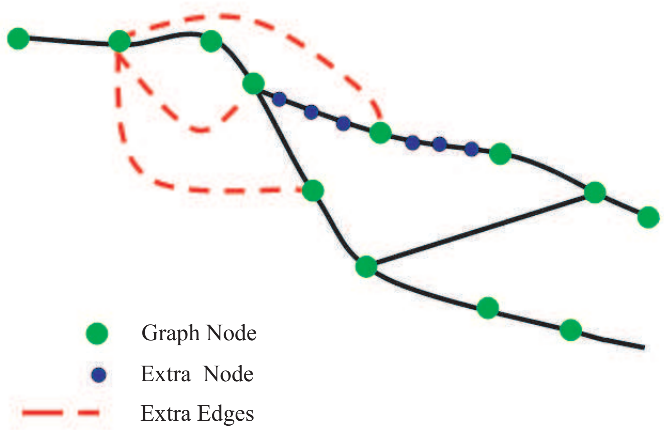


Figure 11: Extra edges and nodes in the augmented motion graph. Three extra nodes between every two original motion frames allows for the refinement of the pose estimation and the slowdown of the playback speed. The extra edges allow one or two original nodes to be skipped.

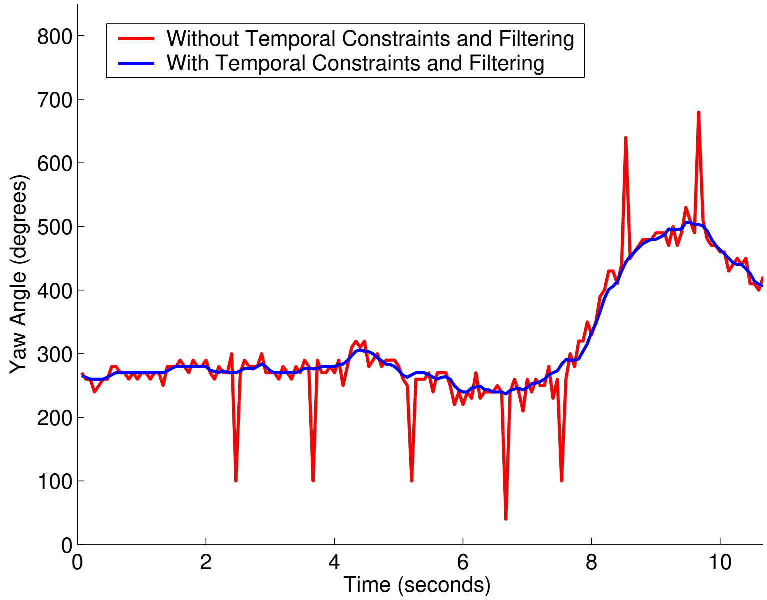


Figure 12: Yaw estimation comparison. The red line shows the result of yaw estimation using a simple majority voting scheme. Temporal constraints and filtering eliminate errors of 180 degrees and smooth the estimated yaw. The testing video trial was captured at 15 frames per second.

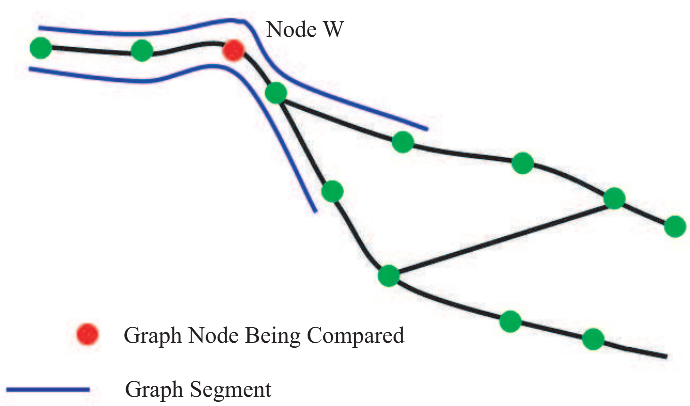


Figure 13: Two different segments through a node. For simplicity, we illustrate the use of two look-ahead and two look-back video frames for the matching. In the experiments described in the paper, ten look-ahead and ten look-back video frames were used.

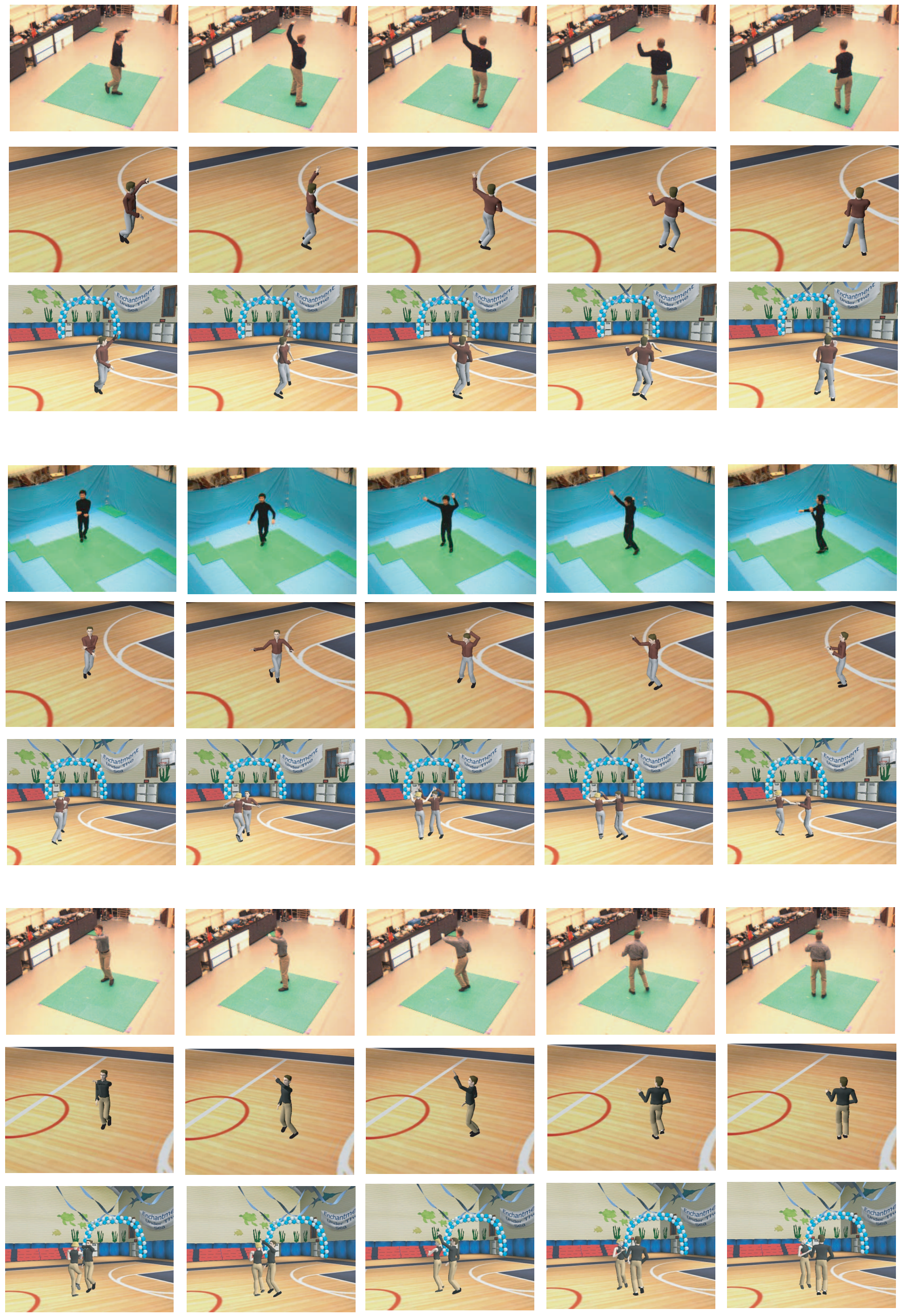


Figure 14: Three dancing sequences from two subjects with street clothes. The sample frames show the matching between the video, the male dancer and the couple.

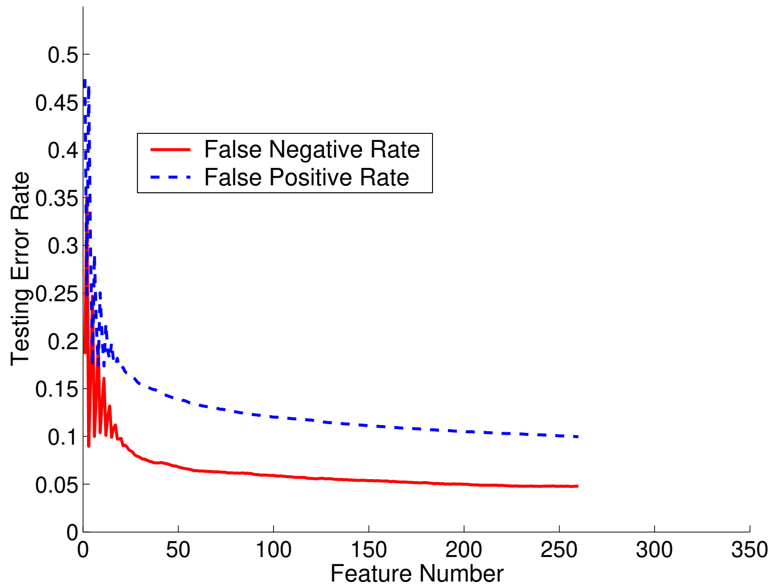


Figure 15: The error of the yaw classifier on the test set as the number of features increases.

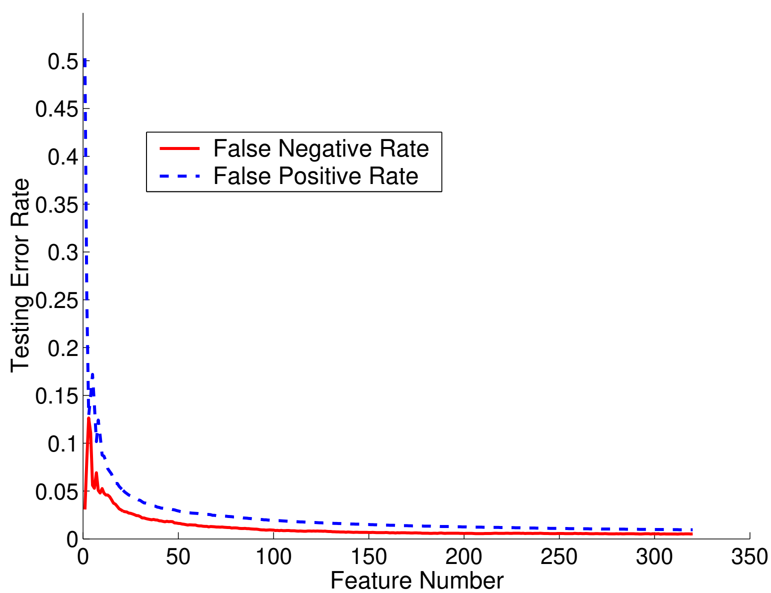


Figure 16: The error of the body configuration classifier (60 degree yaw angle) on the test set as the number of features increases.

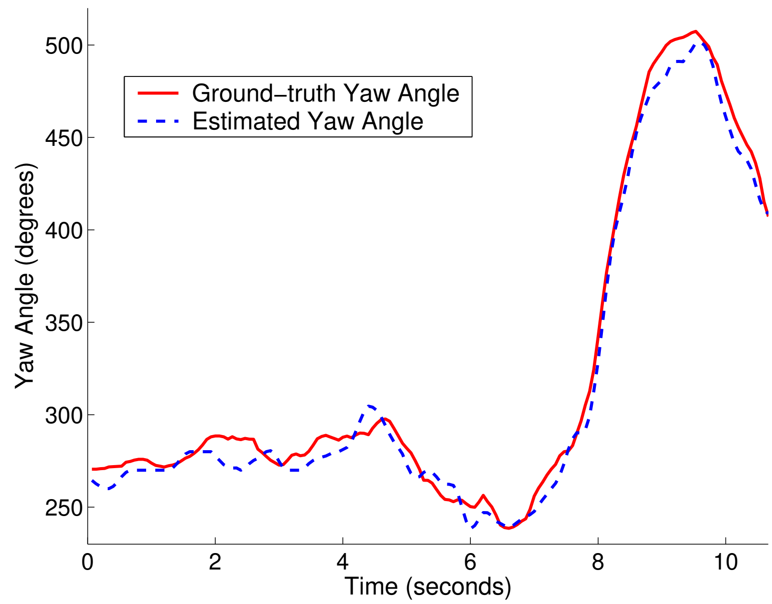


Figure 17: Comparison of yaw estimation with motion capture data that was recorded simultaneously with the input video data. The input video was captured at 15 frames per second.

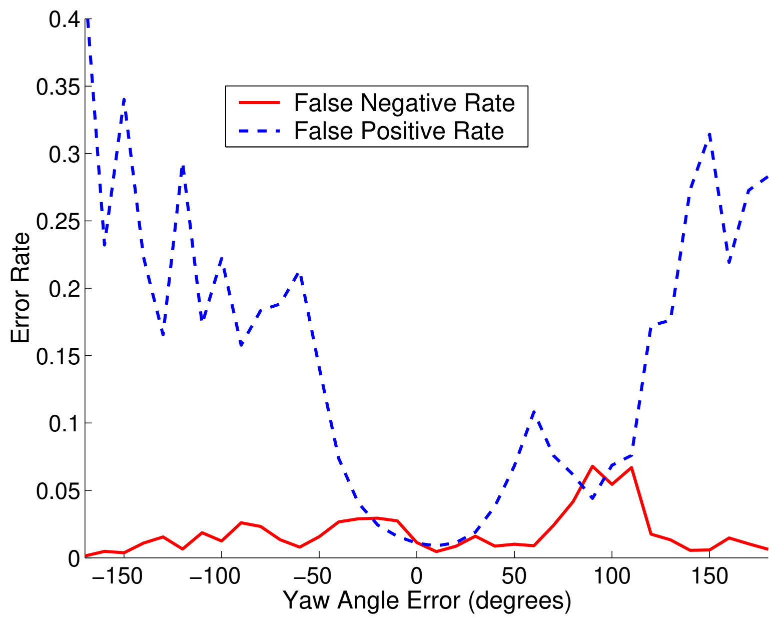


Figure 18: Performance of body configuration classifiers in the presence of errors in yaw estimation. We test the performance of each of the 36 body configuration classifiers using test data with a yaw angle of zero. The graph shows the false positive rate and the false negative rate of the body configuration classifier of each yaw bin. Similar performance was seen for testing using the data from other yaw angles. The result shows that body configuration estimation is robust to the yaw estimation errors in our system.

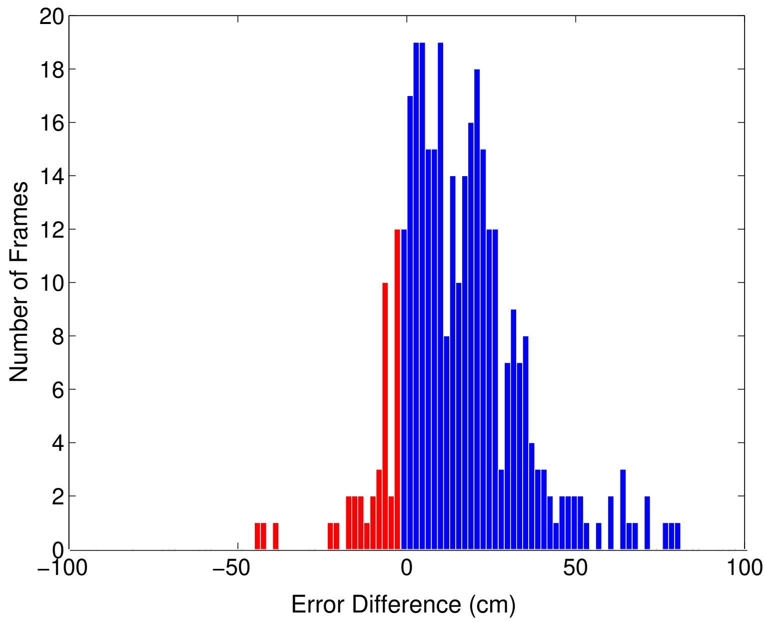


Figure 19: Performance comparison between local features (rectangle features) and global features (Hu moments) for body configuration. The graph shows a histogram of the L2 estimation error for Hu moments minus that achieved with local features. The blue bars indicate the frames on which the local features performed better, the red show the frames on which Hu moments perform better.

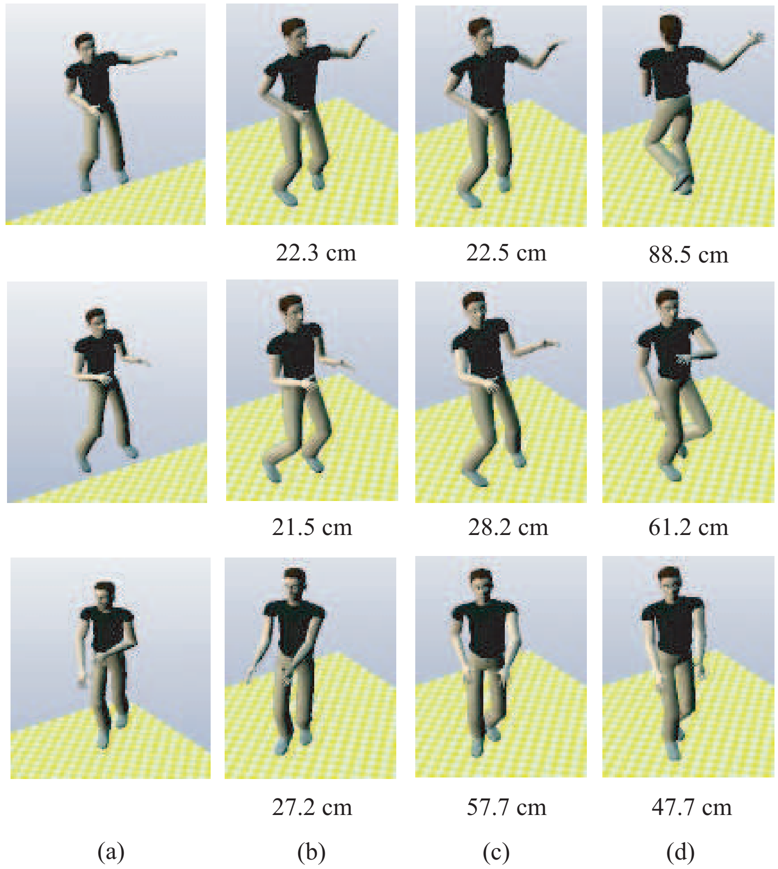


Figure 20: Comparison of the pose retrieval results from three different approaches on several frames of motion. (a) The ground truth from motion capture process. The image shows the translation, orientation, and body configuration of the motion capture subject. (b) The best available match in the database found by a brute force search on the 3D locations of the end effectors in the coordinate system of the root segment. The image visualizes the yaw angle from ground truth and the retrieved body pose. (c) The closest match found with rectangle local features. (d) The closest match found with Hu moments. For a fair comparison, temporal constraints and filtering are not applied for the yaw estimation in (c). No temporal information is used in any of the methods. Both (c) and (d) show the estimated yaw angle and body configuration. For each body configuration, we compute the L2 distance between its body configuration parameters and the ground truth shown in (a). Those errors are given below each image. Because the body configuration parameters are defined in the root coordinate system of the human body, the estimation error in yaw is ignored when computing those errors. The first row shows an example where Hu moments failed to correctly estimate the yaw orientation. In the second row, Hu moments had a good estimate for yaw orientation but failed to estimate the body configuration accurately. These two examples show typical cases when rectangle features perform better than Hu moments. The third row shows an example where Hu moments performs slightly better than rectangle features. In our experiment, the average baseline error (distance to the best match from the true 3D parameters) was 25 cm; the average error with our method was 44 cm while the Hu moments had an average error of 76 cm.

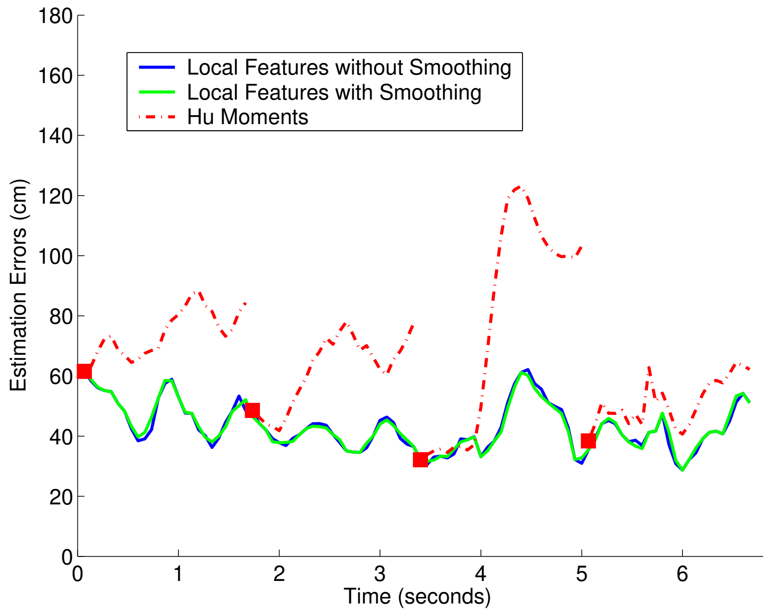


Figure 21: Performance comparison between rectangular local features and global features (Hu moments). The testing video trial was captured at 15 frames per second. The estimation error is the L2 distance between the retrieved body configuration parameters and the ground-truth parameters from the simultaneous motion capture process. The local features provide a good match to the user's motion but the Hu moments tend to drift away from the captured ground truth. Every 1.7 seconds, we manually restart the search with Hu moments from the body configuration found with local features. The graph also shows the estimation error of local features when the three-frame smoothing is disabled. The smoothing averages the scaling factors across three frames and improves the visual quality of the synthesized motion although this improvement is not visible in the computed error.

(a)

(b)

Figure 22: (a) The synthetic human model used to generate synthetic silhouettes for training. (b) Human subjects with different street clothes used for testing our system. Our system demonstrates its robustness to different body types of the user and his clothing.