Segmenting Motion Capture Data into Distinct Behaviors

Jernej Barbič   Alla Safonova   Jia-Yu Pan   Christos Faloutsos
Jessica K. Hodgins   Nancy S. Pollard
Computer Science Department
Carnegie Mellon University

Abstract

Much of the motion capture data used in animations, commercials, and video games is carefully segmented into distinct motions either at the time of capture or by hand after the capture session. As we move toward collecting more and longer motion sequences, however, automatic segmentation techniques will become important for processing the results in a reasonable time frame.

We have found that straightforward, easy to implement segmentation techniques can be very effective for segmenting motion sequences into distinct behaviors. In this paper, we present three approaches for automatic segmentation. The first two approaches are online, meaning that the algorithm traverses the motion from beginning to end, creating the segmentation as it proceeds. The first assigns a cut when the intrinsic dimensionality of a local model of the motion suddenly increases. The second places a cut when the distribution of poses is observed to change. The third approach is a batch process and segments the sequence where consecutive frames belong to different elements of a Gaussian mixture model. We assess these three methods on fourteen motion sequences and compare the performance of the automatic methods to that of transitions selected manually.

Key words: human motion, motion capture, motion segmentation, PCA

1 Introduction

Motion capture is frequently used in movies and video games because of the naturalness and subtle detail contained in the motion. Currently, motion data are often stored in small clips to allow for easy hand sequencing and searches based on keywords describing the behavior.

Extended sequences of motion have many advantages over small motion clips. Longer shots may be more comfortable for the actors and will contain natural transitions from one behavior to the next. Collecting long sequences of motion is also the only way to capture natural behavior over extended periods of time (e.g., in an office setting). However, segmentation of these sequences for indexing, retrieval, and other processing can be tedious.

The present work suggests that automatic segmentation of human motion data based on statistical properties of the motion can be an efficient and quite robust alternative to hand segmentation. Our goal is to segment motion into distinct high-level behaviors (e.g., walking, running, punching). The problem falls into the category of unsupervised learning in the sense that no prior or training models are available—we want to be able to create a segmentation even when the behaviors have not been seen before. We focus on efficient techniques that are easy to implement and scale well with the size of the input motion data.

Given these goals, we chose three alternative segmentation techniques to examine. These techniques treat the motion as an ordered sequence of poses assumed by the character (i.e., an ordered sequence of motion frames) and segment the motion where there is a local change in the distribution of poses. The first approach chooses segments using an indication of intrinsic dimensionality from Principal Component Analysis (PCA), the second approach creates segments using a probabilistic model of motion obtained from Probabilistic PCA, and the third approach generates segments based on a Gaussian mixture model representation. We have found that very good performance can be obtained from these simple and fast techniques. The best of our methods (Probabilistic PCA) achieves over 90% precision for 95% recall, that is, very few false alarms and false dismissals.

2 Related Work

In the vision community, model-based approaches to recognition, tracking, and segmentation of high-level behaviors are widely used (e.g., [9], [21]). In graphics, Arikan, Forsyth, and O'Brien [3] describe a model-based approach to motion annotation that could also be employed for segmentation. These approaches, however, rely on the presence of hand-annotated training data. In contrast, our goal is to segment motion based only on information available in the current motion sequence (without prior models), and we focus on related work that shares this goal.

A number of researchers have found that low-level motion segmentation can be achieved in a straightforward manner. Fod, Mataric, and Jenkins [10] segment motion data by detecting zero crossings of angular velocities. Li, Wang, and Shum [19] explore a more sophisticated technique where low-level segments (*textons*) are represented as the output of linear dynamic systems. A segmentation of motion is also implicit in a state machine or motion graph representation [5, 16, 17, 2]. We are interested in segmenting motion into higher-level behaviors, such as walking, running, and sitting. Many low-level behavior components would be observed within any one of these behaviors, and so a different approach is needed.

Segmentation of video sequences has been extensively studied. Unsupervised approaches to modeling and segmentation include entropy minimization to construct Hidden Markov Models (HMMs), with high-level behaviors mapped to states of the HMM [6], clustering based on distance metrics developed over a variety of temporal scales [28], and online segmentation based on considerations of information loss [23]. Motion capture data is much simpler than video data. We hypothesize that fast online techniques for segmenting video data (e.g., as in [23]) would work very well for segmentation of motion into higher-level behaviors, and we suggest that the power of HMMs for example, may not be required for good performance.

From the data mining community subspace clustering (e.g., [1]) is of particular interest. This approach is designed to identify low-dimensional clusters in high-dimensional data. A straightforward implementation of this technique, however, would require us to consider motion as an unordered set of poses. Our experience with Gaussian Mixture Models (Section 3.3) leads us to believe that clustering on unordered poses would not work well, because it is often easier to locally capture a transition between two behaviors than it is to tease apart a large number of behaviors after they have been mapped into the same space.

Clustering has been used to improve the efficiency of searching through motion graphs (e.g., [17]), but to our knowledge it has not been explored for the purpose of high-level segmentation of human motion capture data. We evaluate a similar clustering technique for the purpose of motion segmentation (Section 3.3) and describe two online techniques that achieve better performance on our dataset.

Finally, we note that many researchers in computer graphics, robotics, computer vision, machine learning and biomechanics have explored the use of Principal Component Analysis and other dimensionality reduction techniques to aid in clustering, modeling, and other processing of motion (see, e.g., [14, 7, 25]).

3 Proposed Methods

The goal of our algorithm is to segment long motion sequences automatically into distinct behaviors. Informally, we are looking for high-level behaviors, which would be described with distinct verbs, possibly with objects (e.g., walk, run, jump, or wash the window). Motion sequences which can be described with the same verb but with changes in coordinate frame (e.g., walk straight, curve left, then walk straight some more) *should not* result in distinct segments. Using the same verb with different objects, on the other hand (e.g., clean the window, clean the floor) *should* result in distinct segments.

To state the problem mathematically, given a long motion sequence M, we wish to segment that sequence into distinct behaviors $M_1, \ldots, M_S$, where the boundaries of the behaviors and the number of behaviors S are to be determined. Each motion sequence M is represented as a sequence of frames, with 120 frames per second. Each frame is represented by specifying the rotations (relative to the parent in the body hierarchy) for all the joints in the body at that particular time. Rotations are specified by quaternions. We omit the absolute body position and body orientation information, so that the approach will be independent of the specific body position and orientation in world coordinates.

3.1 PCA Approach to Segmentation

The PCA approach is based on the observation that simple motions exhibit lower dimensionality than more complex motions. Each frame x_i ($i = 1, 2, \ldots, n$) is represented as a point in 56-dimensional space (denoted by $\mathbb{R}^{56}$), because there are 14 joints in our body hierarchy, and one quaternion is specified per joint. We treat quaternions as vectors of length 4. The motion sequence corresponds to the trajectory in $\mathbb{R}^{56}$ and the *center of motion* can be defined as

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \, . \qquad (1)$$

For a simple motion, the frames form a cluster, that is spread around the center of motion. Frames lie mostly in some low-dimensional hyperplane containing $\overline{x}$, since the 56 dimensions are highly correlated. For example, when the right hip moves the right leg forward during walking, the left hip is usually moving backwards. Similar correlations exist for most simple motions. Thus, for a given dimensionality r, we can approximate frames as

$$x'_i = \overline{x} + \alpha_{i1} v_1 + \alpha_{i2} v_2 + \ldots + \alpha_{ir} v_r \, , \qquad (2)$$

where $v_1, v_2, \ldots, v_r$ are unit orthogonal vectors forming the basis of the linear subspace corresponding to the hy-

perplane, and $\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{ir}$ are coefficients determining the specific frame. The less correlated the motion, the higher the dimensionality that will be required to accurately represent the motion.

For any hyperplane containing $\overline{x}$, we can orthogonally project the frames x_i on the hyperplane and call the resulting projections x'_i. The projection introduces the error

$$e = \sum_{i=1}^{n} \|x_i - x'_i\|^2 , \qquad (3)$$

where $\|x\|$ denotes the standard Euclidean norm in $\mathbb{R}^{56}$. Given the set of frames x_i and dimensionality r, a r-dimensional hyperplane exists for which the projection error is minimized. This hyperplane can be found by applying the widely used statistical technique of PCA [15]. To perform PCA, the singular value decomposition (SVD) is used. The center of motion is first subtracted from the frames, and the centered frames are then organized in a $n \times 56$ matrix D, where $n \gg 56$ equals the number of frames. The SVD decomposition yields matrices U, V, and Σ, such that

$$D = U\Sigma V^T. \qquad (4)$$

Columns of U and V are orthogonal unit vectors, and the matrix Σ is a 56×56 diagonal square matrix, with non-negative decreasing *singular values* σ_i on its diagonal. The first r columns of V give the basis $v_1, v_2, \ldots v_r$ of the optimal hyperplane of dimension r. Projecting the frames on the optimal hyperplane is equivalent to discarding all singular values except the largest r, and the projection error is

$$e = \sum_{i=1}^{n} \|x_i - x'_i\|^2 = \sum_{j=r+1}^{56} \sigma_j^2. \qquad (5)$$

The ratio

$$E_r = \frac{\sum_{j=1}^{r} \sigma_j^2}{\sum_{j=1}^{56} \sigma_j^2} \qquad (6)$$

is an indicator of how much information is retained by projecting the frames on the optimal r-dimensional hyperplane. The dimensionality r can be determined by selecting the smallest r such that $E_r > \tau$, where $\tau < 1$ is some specified parameter. Such determination of dimensionality r is typical for PCA analysis [11]. Our experiments show that choosing $\tau = 0.9$ introduces very little change to the appearance of motion and preserves the features that a human might likely use to tell one motion apart from another. Note that dimensionality reduction is performed only on the joints of the body, while the body position and orientation remain unaltered. A number of

more sophisticated approaches for computing intrinsic dimensionality of the data are available [20, 12, 4], but this simple method worked well in our experiments.

According to our experiments, when $\tau = 0.9$, we have $r \leq 6$ for 85% of simple motions from a database of 203 simple motions, obtained by manually segmenting a part of a larger motion capture database. If a motion consists of two or more simple motions, more dimensions are necessary to achieve the same projection error, because the data exhibits more variety. Put another way, for a fixed r, the projection error will be greater. This observation forms the basis of our algorithm: transition from one behavior to another will coincide with the point along the motion where, for a fixed r, the projection error begins to increase rapidly.

First, we use a fixed number of frames k to determine the number of dimensions r to sufficiently represent the first k frames of motion, using the rule $E_r > \tau$. Parameter k is set to 240 (2 secs) for our experiments. For every value of i, where i is steadily increasing by one, we compute the SVD decomposition of frames 1 through i, and evaluate the total error e_i using Equation 5. For a simple motion, the error e_i rises at an approximately constant slope, because the hyperplane does not change much within a single simple motion, and new frames will on average introduce about the same error. When transitioning into a new motion, however, e_i will rise much more quickly. To detect the transition, we observe the discrete derivative $d_i = e_i - e_{i-\ell}$, where parameter ℓ must be large enough to avoid noise in the data. We set it to $\ell = 60$. The initial value of i is k. To avoid the initial oscillation of the average due to the small number of frames and data points d_j, we begin testing for a cut only after i has become larger than some specified parameter i_0. In our experiment, we use $i_0 = k+\ell = 300$. Note that these choices of parameters imply that any simple motion must last at least 2.5 seconds for the algorithm to detect it. Almost all of simple motions in our motion database satisfy this criterion.

For a simple motion, the derivative d_i is more or less constant, with minor oscillations around the average after an initial period of noise due to the small number of frames. When a motion transition occurs, the derivative d_i rises sharply above the constant value (Figure 1). At any given value of i, we can check for a possible cut by computing the average and standard deviation of all the previous data points d_j, where $j < i$. If the current data point d_i is more than $k_\sigma = 3$ standard deviations from the average, we assign a motion cut to that frame. To process the rest of the motion, we restart the algorithm at the cut frame.
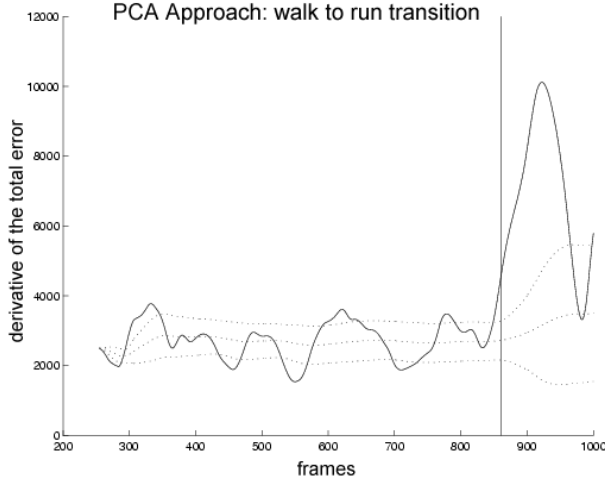
Figure 1: Transition from walking to running. The vertical line at the frame 861 corresponds to the cut detected by the algorithm. The solid line shows the derivative d_i. The central dotted line represents the average of the derivative. The two outer dotted lines show the range corresponding to one standard deviation from the average. Derivative average and standard deviation at frame i are computed over all previous data points d_j, i.e., over all $j < i$.

3.2 Probabilistic PCA Approach to Segmentation

In the second approach, we use Probabilistic PCA (PPCA) to estimate the distribution of the motion data. PPCA is an extension of the traditional PCA [24, 26] and defines a proper probability model for PCA.

In PCA the directions "outside" the subspace are simply discarded, whereas in PPCA they are modeled with noise. First, the average square of discarded singular values can be defined as

$$\sigma^2 = \frac{1}{56 - r} \sum_{i=r+1}^{56} \sigma_i^2. \qquad (7)$$

The entire data set can then be modeled by a Gaussian distribution, where the mean equals the center of motion $\overline{x}$ and the covariance matrix C is determined by the equations:

$$W = V_r (\Sigma_r^2 - \sigma^2 I)^{1/2} \qquad (8)$$

$$C = \frac{1}{n-1}(WW^T + \sigma^2 I) = \frac{1}{n-1} V \tilde{\Sigma}^2 V^T. \qquad (9)$$

Here, we follow the notation of Equation 4, and introduce V_r to denote the first r columns of matrix V, and Σ_r to denote the upper-left $r \times r$ block of matrix Σ. Matrix $\tilde{\Sigma}$ is a 56×56 diagonal matrix, obtained from Σ by replacing all discarded singular values by σ.
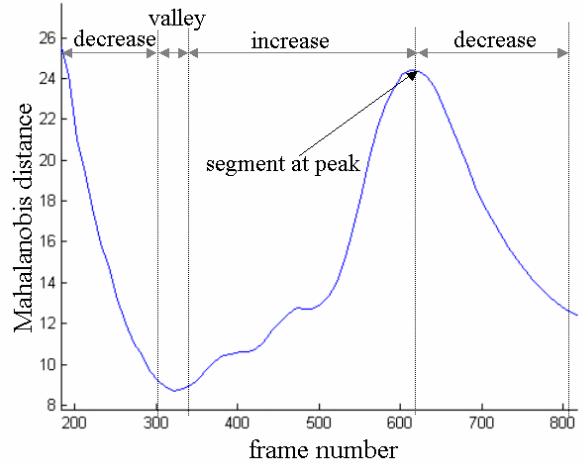

Figure 2: Plot of H as K is repeatedly increased by Δ.

We use PPCA to model the first K frames of the motion as a Gaussian distribution, represented by mean $\overline{x}$ and covariance C, where C is defined based on the the estimated intrinsic dimensionality of the motion. We estimate intrinsic dimensionality using the technique specified in the PCA approach. We set $\tau = 0.95$. This approach provides an accurate model of a particular behavior because it captures the correlation in the motion of different joint angles as well as the variance of all joint angles. After we compute $\overline{x}$ and C, we estimate how likely are motion frames $K + 1$ through $K + T$ to belong to the Gaussian distribution defined by $\overline{x}$ and C. We do this by computing an average Mahalanobis distance [8], H for frames $K + 1$ through $K + T$:

$$H = \frac{1}{T} \sum_{i=K+1}^{K+T} (x_i - \overline{x})^T C^{-1} (x_i - \overline{x}). \qquad (10)$$

An advantage of using the Mahalanobis measurement for discrimination is that distances are calculated in units of standard deviation from the mean and are therefore data independent.

We next increase K by a small number of frames, Δ, and repeat the estimation of distribution for the first K frames ($K := K + \Delta$), and of the average Mahalanobis distance, H, for frames $K + 1 : K + T$ with respect to the new distribution. For our experiments, we used $T = 150$ frames, $\Delta = 10$ frames, and the initial value of $K = T$. A reasonable estimate of T is half of the anticipated number of frames in the smallest behavior in the database.

Figure 2 shows a characteristic pattern of H: it initially decreases, then forms a valley, increases and decreases again, forming a peak. The first decrease in H happens

when frames $1 : K$ and $K + 1 : K + T$ both belong to the same behavior. As we increase K, the algorithm estimates progressively more accurate distributions of this behavior, making frames $K + 1 : K + T$ more likely. When the model converges, the valley is reached. The increase in H occurs when the new behavior enters frames $K + 1 : K + T$. The subsequent decrease in H begins when the frames of the new behavior start appearing in frames $1 : K$, and as a result the distribution begins to accommodate the new behavior. Thus, a reasonable choice for the segmentation takes place when H forms a peak. Then, frames $1 : K$ contain the old motion and frames $K + 1 : K + T$ contain the first T frames of the new motion. The algorithm declares a cut when a valley in H is followed by a peak, and the difference between the two is at least some threshold R. For our experiments, we set R to 15. In general, increasing R results in fewer segments that correspond to more distinct behaviors, whereas decreasing R results in a finer segmentation. The next cut is found by repeating the algorithm on the rest of the motion.

In rare cases the first behavior is characterized by a "wide" distribution, whereas the following behavior is characterized by a narrow distribution, that lies almost completely within the first distribution. In this case, frames from the second behavior are considered likely to have come from the first distribution and the algorithm will fail to detect the segmentation. However, if we do a backward pass of the algorithm, the segmentation will be detected because frames from the first distribution are unlikely to come from the second distribution. Therefore, we run our algorithm several times, alternating forward and backward direction, and adding new cuts at each pass until convergence (no new cuts are found). In practice, a single forward pass detects almost all behaviors and at most one backward pass is required.

3.3 GMM Approach to Segmentation

In the third approach we employ a Gaussian Mixture Model (GMM) to model the entire sequence and then segment the sequence whenever two consecutive sets of frames belong to different Gaussian distributions. The underlying assumption is that the frames from different simple motions form separate clusters, and each cluster can be described reasonably well by a Gaussian distribution. Figure 3 shows the distributions of frames from two motion sequences, each of which consisted of three simple motions concatenated together. Each sequence has been projected onto the first two principal components computed for the full sequence. The frames of the simple motions form clusters that can be modeled by Gaussian distributions, suggesting that GMM is a reasonable model for segmentation.
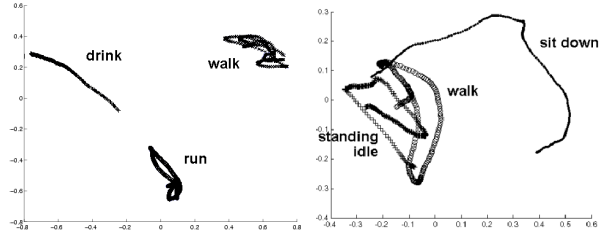

Figure 3: Two-dimensional projection of groups of simple motions: (a) Walking, running, and drinking; (b) walking, sitting down, and standing idle. Frames of each motion form their own clusters. The projection is done on the first two principal components for each sequence. In (a), the natural continuous transitions among different clusters are not shown.

We use the Expectation Maximization (EM) algorithm to estimate the Gaussian Mixture Model of the data. In a pre-processing step, we use PCA to project the frames onto a lower dimensional subspace. This dimensionality reduction is meant to speed up the EM algorithm. The number of principal components is chosen so that 90% of the variance of the original data distribution is preserved. The number of principal components here is about 32, as the complete sequences are not simple. In the ideal case, each cluster is represented by a single Gaussian distribution. Thus, a collection of k clusters can be represented by a mixture of k Gaussian distributions. The EM algorithm is used to estimate the parameters of the GMM such as mean, m_j, covariance matrix, Σ_j, and prior, π_j, for each of the Gaussians in the mixture. Note that the clusters are not of equal size. For example, longer motion segments have more points and thus form larger clusters. To capture these size differences, EM also estimates the probability π_j of a point belonging to a cluster C_j. The EM algorithm for GMM is one of the classical techniques in machine learning, and we refer the reader to [8] for algorithm details. After the GMM parameters are estimated, we can compute a most likely cluster for each frame of the motion. A change of a behavior is then detected at time i if frames x_i and x_{i+1} belong to different clusters.

In practice, if a segment is shorter than 1 second (120 frames), it is merged with its neighboring segments, i.e., the segment is split into two halves and its cluster labels are changed to those of the segments before and after. This heuristic worked well and it often removed small segments that occur on the transitions between behaviors.

Figure 4 shows the segmentation of a motion capture data sequence composed of 7 simple motions in the following order: walking, jumping forward, walk-
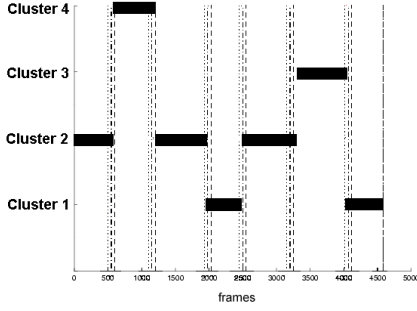
Figure 4: *The clusters for a motion capture sequence of 7 simple motions (walking, jumping forward, walking, punching, walking, kicking, punching). GMM is used to find a mixture of $k = 4$ clusters. GMM provides the correct segmentation and also identifies similar simple motions.*

ing, punching, walking, kicking, and punching. The vertical lines are the ground truth of the segmentations. Cut points between two motions are given as intervals covering the time period of the transition. Each cut point is presented by three vertical lines, indicating the start, middle, and end of the motion transition as determined by a human observer. Figure 4 also shows the cluster labels $(1, 2, 3, 4)$ assigned to the frames along the vertical axis of the graph. In this case we fit a GMM model of $k = 4$ clusters. The segmentation assignment of cluster labels also successfully identified similar simple actions. Here, the three walking motions are assigned the cluster label 2, and the two punching motions are assigned the cluster label 1.

Unfortunately, we have to specify the number of clusters k for each execution of the GMM, and we usually do not know the number of clusters in a data set. As an alternative, GMM is often estimated for various values of k, and the mixture model that maximizes some criterion, such as the Bayesian Information Criterion (BIC) [22] is chosen. This approach, however, did not seem to be useful for our data because it often returned a smaller than optimal number of clusters. Instead, fixing k at a small value such as 4 seemed to produce reasonable results without additional complexity.

We initialize the GMM clusters with random frames. Repeated experiments show that after removing short (noisy) segments, major segmentation points could always be identified, independently of the initialization of the clusters. This insensitivity to the initial conditions may be due to the characteristics of our database (see Figure 3). Alternatively, we also performed experiments where we partitioned the entire motion sequence into K consecutive parts, and created the initial guess by picking

	X_1^a	X_1^b	X_2^a	X_2^b	X_3^a	X_3^b	X_4^a	X_4^b	X_5^a	X_5^b	X_6^a	X_6^b	X_7^a	X_7^b
X_1^a	2	5	15	14	7	8	22	22	12	12	7	8	18	15
X_1^b	3	2	8	8	6	7	16	15	8	9	6	6	11	10
X_2^a	15	17	5	5	19	19	37	36	32	28	21	20	29	28
X_2^b	14	15	5	4	17	18	29	30	26	24	17	17	25	24
X_3^a	16	20	36	28	2	10	49	45	29	28	20	22	42	36
X_3^b	15	19	34	29	10	3	37	33	28	23	17	18	31	25
X_4^a	37	42	59	50	49	37	22	27	56	61	97	92	120	116
X_4^b	35	40	54	45	45	33	27	24	55	60	98	94	127	122
X_5^a	22	25	51	41	29	28	56	55	6	12	33	34	45	38
X_5^b	18	22	46	35	28	23	61	60	12	5	35	34	40	34
X_6^a	85	91	67	57	84	83	184	189	89	92	10	14	169	161
X_6^b	74	80	67	56	77	75	179	185	86	89	14	10	180	174
X_7^a	29	33	45	37	42	31	120	127	45	40	49	44	9	10
X_7^b	92	96	116	109	109	109	248	255	115	123	161	174	37	28

Figure 5: *The error matrix for the PCA algorithm.*

	X_1^a	X_1^b	X_2^a	X_2^b	X_3^a	X_3^b	X_4^a	X_4^b	X_5^a	X_5^b	X_6^a	X_6^b	X_7^a	X_7^b
X_1^a	7	31	187	190	89	71	154	163	151	152	246	270	170	175
X_1^b	31	7	152	156	77	74	134	143	125	128	176	201	156	157
X_2^a	187	152	7	19	115	196	117	109	126	116	163	183	117	122
X_2^b	190	156	19	7	137	191	184	175	128	129	211	238	156	162
X_3^a	89	77	115	137	7	48	147	163	156	154	200	220	161	164
X_3^b	71	74	196	191	48	7	143	153	204	209	280	297	206	206
X_4^a	154	134	117	184	147	143	7	15	180	149	155	141	79	96
X_4^b	163	143	109	175	163	153	15	7	174	149	159	143	83	100
X_5^a	151	125	126	128	156	204	180	174	7	41	188	218	149	149
X_5^b	152	128	116	129	154	209	149	149	41	7	155	176	105	105
X_6^a	246	176	163	211	200	280	155	159	188	155	7	23	99	101
X_6^b	270	201	183	238	220	297	141	143	218	176	23	7	86	88
X_7^a	170	156	117	156	161	206	79	83	149	105	99	86	7	14
X_7^b	175	157	122	162	164	206	96	100	149	105	101	88	14	7

Figure 6: *The error matrix for the PPCA algorithm.*

one random frame from every part. In this case, the initial points are less likely to fall into the same final cluster, which helped the convergence rate and in our case produced a slightly better result. However, the improvement over fully random initialization was small.

4 Experiments

In this section, we present the results of two experiments conducted on a motion capture database containing 14 sequences, each consisting of approximately 8000 frames. Each sequence is a series of about 10 simple motions. Typical human activities are represented, such as walking, running, sitting, standing idle, exercising, climbing, performing martial arts, washing a window, and sweeping a floor.

The first experiment illustrates the fundamental assumptions behind our algorithms. The PCA approach is based on the idea that the intrinsic dimensionality of a motion sequence containing a single behavior should be smaller than the intrinsic dimensionality of a motion sequence containing multiple behaviors. To illustrate this idea, we selected 7 distinct simple behaviors: walking, running, sitting down, forward jumping, climbing, arm stretching and punching. For each behavior, we extracted two distinct occurrences from the database. The length of

each occurrence was uniformly chosen to be 240 frames. Let us denote the extracted sequences by X_i^a, X_i^b, for $i = 1, \ldots, 7$, where X_i^a and X_i^b, are distinct representatives of the same type of a motion. Given two arbitrary extracted 240-frame motions X, X', we first estimated the number of dimensions r required to represent the first motion X. Then, we combined frames of motion X and X' to obtain a set of 480 frames. For the purposes of this experiment, the particular ordering of frames is irrelevant, and identical results would have been obtained had the frames been permuted by a random permutation. For the set of 480 frames, we computed the total projection error when these frames were reduced to r dimensions, as directed by Equation 5. These steps essentially mimic the approach taken by the PCA method. We repeated this process for the 14×14 possible ordered pairs of motions X, X' and stored the projection errors in a 14×14 matrix, with row index corresponding to X and column index corresponding to X'. The resulting matrix can be seen in Figure 5. For the PCA method to segment successfully, each motion must be much closer to itself and to other representatives of the same kind of motion than it is to other distinct motions. For every row of the matrix, the smallest element is on the diagonal, corresponding to joining two identical motions. The next smallest element corresponds to joining the motion to its similar, but distinct motion. A much larger error is obtained when concatenating two completely different behaviors, because the number of dimensions that we estimated is just enough to represent the first behavior, but too small to represent two different behaviors.

The PPCA approach is based on the assumption that Gaussian models of two separate behaviors will be quite different. We illustrate this assumption in a similar fashion. To compare the Gaussian models for two behaviors X and X', we estimated the distribution for the first behavior using Equation 9, and computed the average Mahalanobis distance of the second behavior relative to the distribution of the first motion using Equation 10. Because the PPCA-based segmentation algorithm is doing both a forward and a backward pass, it also computes the Mahalanobis distance of the first behavior relative to the distribution of the second motion and uses it for segmentation. To include this effect in our experiment, we also estimated the distribution for the second behavior and computed the average Mahalanobis distance of the first behavior relative to the distribution of the second behavior. The resulting symmetric 14×14 matrix in Figure 6 shows the maximum of the two Mahalanobis distances for each pair of behaviors. As with PCA, the element size is much smaller on the diagonal and for first off-diagonal elements than it is elsewhere in the matrix.

This comparison demonstrates the assumption behind the PPCA approach.

When capturing motion sequences for the database, the actor was asked to perform a number of distinct behaviors in a specified order (overall 35 distinct behaviors were used throughout the database). To test the segmentation of continuous motion sequences in the database we first processed the database manually to determine the correct positions for motion transitions. These correct positions were implicitly defined by the choice of particular behaviors to be performed by the actor in a particular motion sequence. Then, we ran the three algorithms proposed in this paper and compared the results to the manually determined transitions. Figure 7 shows the results of these experiments. Figure 8 shows representative poses for the behaviors that comprise one of the motion sequences in the database.

We compared the three algorithms using the standard precision/recall framework. Precision is defined as the ratio of reported correct cuts versus the total number of reported cuts. Recall is defined as the ratio of reported correct cuts versus the total number of correct cuts. The closer precision and recall are to 1, the more accurate the algorithm is. Figure 9 gives precision and recall scores for the three algorithms.

5 Discussion

We have described and tested three methods for motion segmentation. Of the three, probabilistic PCA provided the best performance on our dataset. We hypothesize that this superior performance occurs because PPCA tracks the changes in the distributions that characterize motions to find cut points while PCA considers only the change in dimensionality. In other words, there are joint range of motion limits implicit in PPCA, while PCA encodes only correlations between joints. PPCA also estimates a single distribution for each individual behavior. GMM, in contrast, treats all frames as independent and, as a result, may construct a single distribution that covers different behaviors and may also estimate two separate distributions for a single behavior. Nevertheless, both PCA and GMM provided good performance.

We assessed the performance of these three methods on a database in which the transitions between behaviors had been hand labeled. This process is imperfect because two observers will certainly pick different frames for a transition and may even disagree on what constitutes a transition. For example, should three consecutive forward jumps be considered three separate behaviors or three cycles of a forward hopping gait? The answer probably depends on the application. As we analyzed the transitions that were mistakenly identified or missed by the
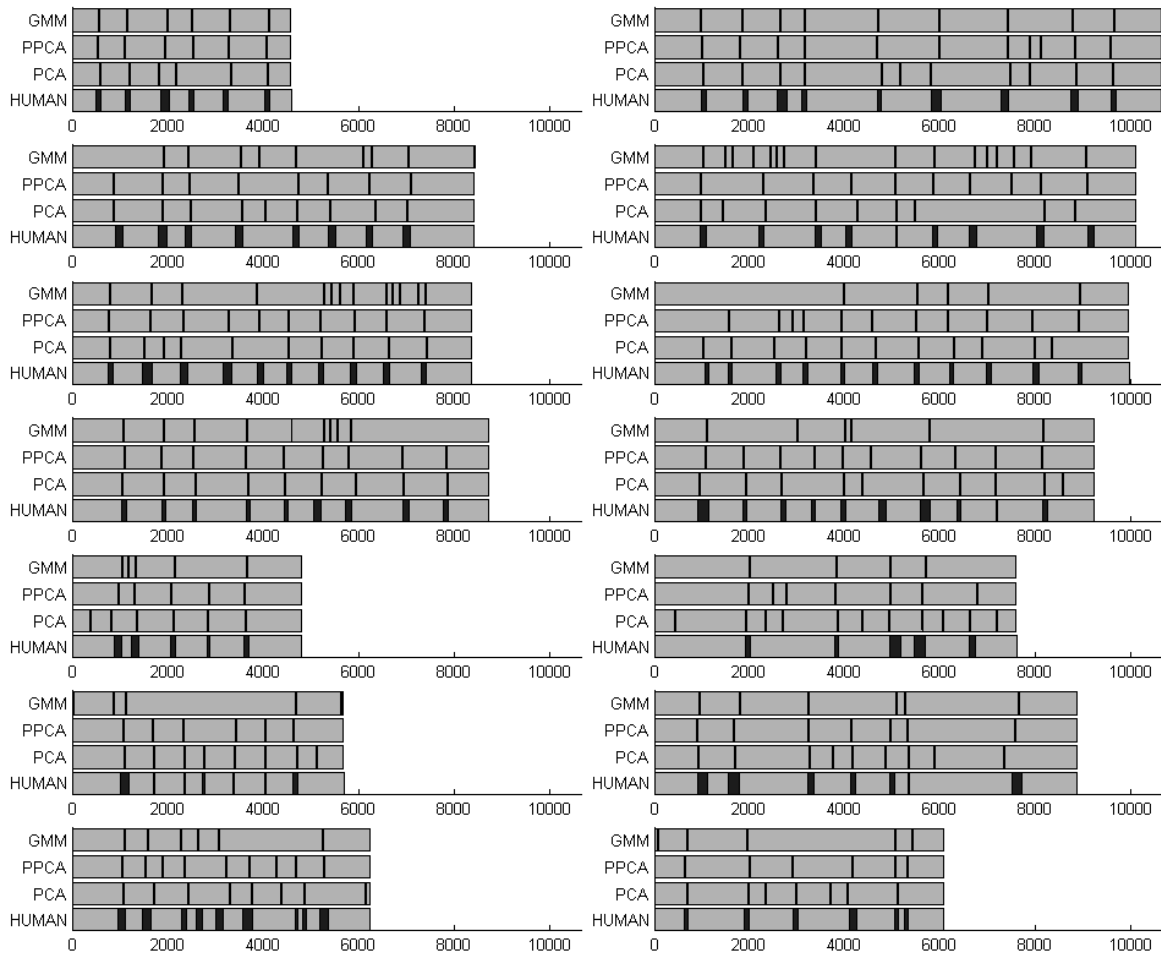
Figure 7: Motion separation points as assigned by a human observer and the three algorithms. Each chart corresponds to one motion from the database, the x-axis corresponds to the frame number, and the vertical bars specify the cut points assigned by the algorithms. For the human observer, the range (instead of a single frame) in which the transition occurred is given, as all the frames in the range are acceptable positions for a motion cut. The sequences corresponding to the upper-left chart (person taking exercise), and the next-to-last chart in the right column (janitor cleaning a room) can also be found on our webpage at: `http://graphics.cs.cmu.edu` .
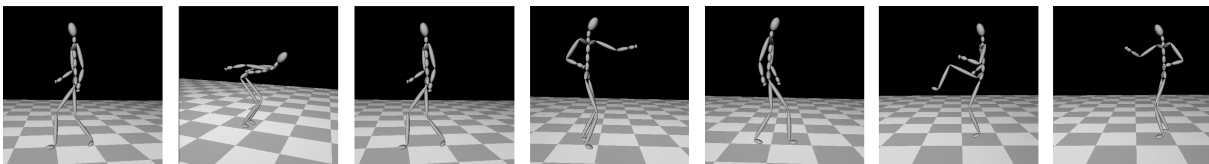

Figure 8: Representative poses for the simple motions that comprise the sequence in the upper-left chart in Figure 7. The seven simple motions are: walking, forward jumping, walking, arm punching, walking, leg kicking, and arm punching.

Method	Precision	Recall
PCA	0.79	0.88
PPCA	0.92	0.95
GMM	0.77	0.71

Figure 9: Precision and recall scores for the PCA, PPCA and GMM algorithms.

three methods, we found the following examples: walking straight and then turning while walking; jumping forward repeatedly; sitting down on a chair, and then immediately standing up; punching with the right hand and then with the left; washing a window in a left-to-right hand motion, and then switching to circular motion. Also note that our approaches won't detect changes in velocity, as velocity information is not used in our algorithms.

The exact frame of the transition is particularly difficult to determine when the motion capture subject makes a smooth transition from one behavior to another. Consequently, we allowed a range of frames to be specified as the ground truth by the human subjects.

Our metrics for detecting transitions weights all degrees of freedom equally. This assumption might not always be correct. For example, if the motion capture subject jumped on his right foot several times in a row and then continued jumping but added a swinging motion of the arms, the human observer would probably classify the two behaviors as both jumping rather than separating them with a cut point. However, the addition of arm motion might cause the automatic segmentation algorithms to declare that a transition has occurred because the arms are of equal statistical importance to the legs. The degrees of freedom of the motion trajectory could be scaled by the effective inertia at each joint. In this way, joints where movement required more effort would be given a greater weight. Also, global position and orientation information could be included in the segmentation criteria. This would likely change the semantics of our segments, as, for example, locations of turns during walking would become candidates for transitions.

Each of the three methods requires the selection of a small number of parameters. These were chosen based on the performance of the methods on a small set of motions and then applied without alteration to the full database. For PCA, we need to set the energy threshold τ, the discontinuity threshold k_σ, and the window size parameters k, ℓ, and i_0. For PPCA, we need to set the peak threshold R and the window parameters T, Δ. The threshold R is independent of the data but can be adjusted if finer or coarser segmentation is desired. For GMM, we needed to set the number of clusters k. Thus, that method is most useful when all the sequences in the database contain ap-

proximately the same number of behaviors and a constant k gives good performance.

We were pleasantly surprised that the PCA-based methods worked so well. Typical human motions apparently lead to highly correlated motion capture sequences. This phenomenon suggests that PCA needs very few components to capture most of the motion. Had this not been the case, the curse of dimensionality would make the problem very difficult because in high dimensions, most points are far away from each other and will look like outliers or discontinuities.

Our PCA-based methods worked well because the data we were operating on are apparently well modeled by Gaussian clouds. Of course, more sophisticated tools, like Independent Component Analysis (ICA) [13, 18] or Locally Weighted Precision Recall [27] may well achieve even better cut detection. Nevertheless, the Gaussian assumption and PCA provided surprisingly good results on the motions in our database.

As motion capture databases grow, segmenting them manually either at the time of capture or during data clean-up will become more difficult. Furthermore, no one segmentation will necessarily be right for all applications. A method with tunable parameters such as ours may be able to provide both the short segments required for learning a transition graph (e.g., for interactive control of a character) and the longer segments that provide a statistical description of a particular behavior. HMMs and other clustering techniques bring added power to the problem, but require an expensive search process and/or a very good initial guess. We were pleased to find that we didn't need the extra power of these techniques.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. of ACM SIGMOD*, pages 94–105, 1998.

[2] O. Arikan and D. A. Forsyth. Synthesizing constrained motions from examples. *ACM Transactions on Graphics*, 21(3):483–490, July 2002.

[3] O. Arikan, D. A. Forsyth, and J. F. O'Brien. Motion synthesis from annotations. *ACM Transactions on Graphics*, 22(3):402–408, July 2003.

[4] C. Bishop. Bayesian PCA. *Proc. of Neural Information Processing Systems*, 11:382–388, 1998.

[5] M. Brand and A. Hertzmann. Style machines. In *Proc. of SIGGRAPH*, pages 183–192. ACM Press/Addison-Wesley Publishing Co., 2000.

[6] M. E. Brand and V. Kettnaker. Discovery and segmentation of activities in video. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8):844–851, 2000.

[7] F. De la Torre and M. J. Black. A framework for robust subspace learning. *International Journal of Computer Vision*, 54(1-3):117–142, 2003.

[8] R.O. Duda, P.E.Hart, and D.G.Stork. *Pattern Classification*. Wiley & Sons, New York, 2001.

[9] D. J. Fleet, M. J. Black, Y. Yacoob, and A. D. Jepson. Design and use of linear models for image motion analysis. *International Journal of Computer Vision*, 36(3):171–193, 2000.

[10] A. Fod, M. J. Mataric, and O. C. Jenkins. Automated derivation of primitives for movement classification. *Autonomous Robots*, 12(1):39–54, 2002.

[11] K. Fukunaga. *Statistical Pattern Recognition, 2nd Edition*. John Hopkins University Press, Baltimore, 1989.

[12] K. Fukunaga and D. Olsen. An algorithm for finding intrinsic dimensionality of data. *IEEE Trans. on Computers*, 20(2), 1971.

[13] A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Components Analysis*. John Wiley & Sons, New York, 2001.

[14] O. C. Jenkins and M. J. Mataric. Deriving action and behavior primitives from human motion data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2551–2556, 2002.

[15] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, New York, 1986.

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

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

[18] T.-W. Lee and M.S. Lewicki. Unsupervised image classification, segmentation, and enhancement using ICA mixture models. *IEEE Trans. on Image Processing*, 11(3):270–279, 2002.

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

[20] T. P. Minka. Automatic choice of dimensionality for PCA. In *Proc. of Neural Information Processing Systems*, pages 598–604, 2000.

[21] N. Oliver, E. Horvitz, and A. Garg. Layered representations for human activity recognition. In *Fourth IEEE Int. Conf. on Multimodal Interfaces*, pages 3–8, 2002.

[22] D. Pelleg and A. Moore. X-means: Extending K-means with efficient estimation of the number of clusters. In *Proc. of the Seventeenth Int. Conf. on Machine Learning*, pages 727–734, 2000.

[23] N. Peyrard and P. Bouthemy. Content-based video segmentation using statistical motion models. In *Proc. of British Machine Vision Conf. BMVC'02, Cardiff*, volume 2, pages 527–536, 2002.

[24] S. Roweis. EM algorithms for PCA and SPCA. In *Proc. of Neural Information Processing Systems*, volume 10, pages 626–632. The MIT Press, 1998.

[25] M. Santello, M. Flanders, and J. F. Soechting. Patterns of hand motion during grasping and the influence of sensory guidance. *Journal of Neuroscience*, 22(4):1426–1435, 2002.

[26] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *J. of the Royal Statistical Society, Series B*, 61(3):611–622, 1999.

[27] S. Vijayakumar and S. Schaal. LWPR : An o(n) algorithm for incremental real time learning in high dimensional space. In *Proc. of Int. Conf. on Machine Learning (ICML2000)*, pages 1079–1086, 2000.

[28] L. Zelnik-Manor and M. Irani. Event-based video analysis. In *Proc. of IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 123–130. IEEE, 2001.