Aligned Cluster Analysis for Temporal Segmentation of Human Motion

Feng Zhou, Fernando De la Torre, Jessica K. Hodgins
Robotics Institute, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, Pennsylvania 15213
zhfe99@gmail.com ftorre@cs.cmu.edu jkh@cs.cmu.edu

Abstract

Temporal segmentation of human motion into actions is a crucial step for understanding and building computational models of human motion. Several issues contribute to the challenge of this task. These include the large variability in the temporal scale and periodicity of human actions, as well as the exponential nature of all possible movement combinations. We formulate the temporal segmentation problem as an extension of standard clustering algorithms. In particular, this paper proposes Aligned Cluster Analysis (ACA), a robust method to temporally segment streams of motion capture data into actions. ACA extends standard kernel k-means clustering in two ways: (1) the cluster means contain a variable number of features, and (2) a dynamic time warping (DTW) kernel is used to achieve temporal invariance. Experimental results, reported on synthetic data and the Carnegie Mellon Motion Capture database, demonstrate its effectiveness.

1. Introduction

In the past two decades, motion capture systems were able to track and record human motion with high spatial and temporal resolution. The extensive proliferation of motion databases urges the development of efficient techniques to index and build models of human motion. One key aspect to understand and build better models of human motion is to develop unsupervised algorithms for decomposing human motion into a set of actions [10]. The problem of factorizing human motion into actions, and more generally the temporal segmentation of human motion, is an unsolved problem in human motion analysis. The inherent difficulty of human motion segmentation stems from the large intra-person physical variability, wide range of temporal scales, irregularity in the periodicity of human actions, and the exponential nature of possible movement combinations. To partially address these problems, we formulate the temporal segmentation of human behavior as a temporal clustering problem.

Fig. 1 illustrates the main goal of this paper: given a sequence of motion capture data, we are able to find temporally coherent clusters of actions (e.g. walking, rotating, jumping). We propose Aligned Cluster Analysis (ACA), an extension of kernel k-means clustering that allows unsupervised clustering of temporal patterns. Compared to previous literature, our approach has multiple advantages: (1) The temporal *granularity* of the human action can be controlled by the user. (2) A robust temporal matching metric is defined by means of the Dynamic Time Alignment Kernel (DTAK) [20]. (3) The temporal segmentation problem is posed as a versatile energy minimization problem. An efficient coordinate descent algorithm solves ACA.

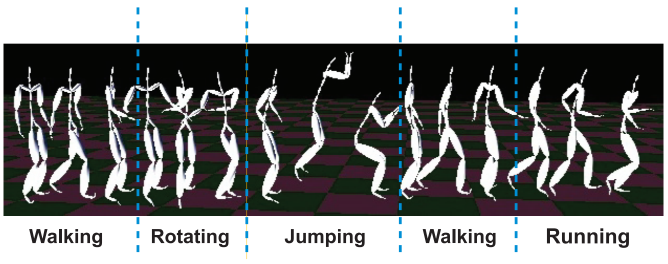


Figure 1. Segmentation of motion capture data into actions.

The remainder of the paper is organized as follows. Section 2 reviews previous work on temporal segmentation of motion capture data. Section 3 introduces theoretical foundations for ACA. Section 4 presents the details of preprocessing and initializing stages for human motion segmentation. Section 5 demonstrates experimental results on synthetic and motion captured data. Finally, Section 6 concludes the paper and outlines future work.

2. Previous work

This section describes previous work on temporal segmentation of human motion and standard clustering techniques.

2.1. Segmenting motion capture data

Temporal segmentation of human actions is an emerging topic in the field of computer vision. Zhong *et al.* [22] use a bipartite graph co-clustering algorithm to segment and detect unusual activities in video. Zelnik and Irani [21] define a flow based matching between actions. De la Torre *et al.* [7] propose a geometric-invariant clustering algorithm to decompose a stream of facial behavior into facial gestures. Unusual facial expressions can be detected through comparisons with clusters of facial gestures.

In the graphics literature, Barbic *et al.* [2] proposed an algorithm to decompose human motion into distinct actions by detecting sudden changes in the intrinsic dimensionality of the Principal Component Analysis (PCA) model. Jenkins *et al.* [8] [12] use the zero-velocity crossing points of the angular velocity to segment the stream of motion capture data. Li *et al.* [17] fit a mixture of linear dynamical systems to a sequence using maximum likelihood approach. Each of the linear systems considers a motion texton that can be used to synthesize new motion sequences. Recently, Beaudoin *et al.* [3] developed a string-based motif-finding algorithm which allows for a user-controlled compromise between motif length and the number of motions in a motif.

In the field of data-mining, the problem of segmentation of time series is well known [14]. Dynamic Time Warping (DTW) is one of the most popular measures between temporal sequences. The idea of dynamic warping has been successfully used in motion blending [6] and clip locating [16] to overcome high fluctuations of articulated movements. Due to the high cost of exact DTW matching, approximate distances, such as the Minimum Bounding Rectangles (MBR) [1], are applied to segment the trajectories.

This paper differs from previous work in the vision, graphics and data-mining literature in the way in which the temporal segmentation problem is formulated. We propose to frame the temporal segmentation problem as an energy-based temporal clustering, providing an elegant mathematical solution via Dynamic Programming (DP).

2.2. k-means clustering and kernel extensions

Clustering refers to the partition of n data points into k disjointed clusters. Among various approaches to unsupervised clustering, k-means [11] is favored for its simplicity. k-means clustering splits a set of n objects into k groups by minimizing the within-cluster variation. That is, k-means clustering finds the partition of the data that is a local optimum of the energy function [7]:

$$J_{km}(\mathbf{M}, \mathbf{G}) = \sum_{c=1}^{k}\sum_{i=1}^{n} g_{ci}||\mathbf{d}_i - \mathbf{m}_c||_2^2 = ||\mathbf{D} - \mathbf{M}\mathbf{G}||_F \quad (1)$$
$$s.t.\ \mathbf{G}^T\mathbf{1}_k = \mathbf{1}_n \ \ and \ \ g_{ij} \in \{0, 1\}$$

where $\mathbf{d}_i \in \Re^{d\times 1}$ (see notation[1]) is a vector representing the i-th data point and $\mathbf{m}_c$ is the geometric centroid of the data points for class c. $\mathbf{G} \in \Re^{k\times n}$ is a binary indicator matrix, such that $g_{ci} = 1$ if sample $\mathbf{d}_i$ belongs to cluster c, and zero otherwise.

A major limitation of the k-means algorithm is that it is optimal only when applied to spherical clusters. To overcome this limitation, kernel k-means [19] implicitly maps the data to a higher dimensional space using kernels. The kernel k-means minimizes:

$$J_{kkm}(\mathbf{G}) = \sum_{c=1}^{k}\sum_{i=1}^{n} g_{ci}\underbrace{||\phi(\mathbf{d}_i) - \phi(\mathbf{m}_c)||_2^2}_{dist_c(\mathbf{d}_i)} \quad (2)$$

where $\phi(\cdot)$ is the mapping, $dist_c(\mathbf{d}_i)$ is the distance between i^{th} point and the center of class c, i.e.

$$dist_c(\mathbf{d}_i) = \kappa_{ii} \underbrace{- \frac{2}{n_c}\sum_{j=1}^{n} g_{cj}\kappa_{ij}}_{f_{ci}} + \underbrace{\frac{1}{n_c^2}\sum_{j_1,j_2=1}^{n} g_{cj_1}g_{cj_2}\kappa_{j_1 j_2}}_{h_c} \quad (3)$$

where n_c is the number of samples that belong to class c. The kernel function κ is defined as $\kappa_{ij} \triangleq \phi(\mathbf{d}_i)^T\phi(\mathbf{d}_j)$.

Similar to the first step in the k-means algorithm, the kernel k-means assigns the sample to the closest cluster:

$$g_{\hat{c}i} = 1, \quad \hat{c} = \arg\min_{c=1}^{k} dist_c(\mathbf{d}_i) \quad (4)$$

It is worth noting that in kernel k-means there is no need to explicitly recompute the mean for each cluster.

3. Temporal segmentation

In this section, we formulate the temporal segmentation problem as a clustering one.

3.1. Temporal segmentation with ACA

Given a sequence $\mathbf{X} \in \Re^{d\times n}$ of motion capture data with n frames, we want to decompose $\mathbf{X}$ into m disjointed segments, each of which corresponds to one of k actions (i.e. classes). The segment itself, $\mathbf{Y}_i \triangleq \mathbf{X}_{[s_i, s_{i+1})}$, is composed by the frames that begin at position s_i and end[2] at $s_{i+1} - 1$. We constrain the length of the segment to the range $w_i \in [w_{min}, w_{max}]$, in order to control the temporal granularity of actions. A k-by-1 indicator vector $\mathbf{g}_i$ is used to assign each segment to an action. $g_{ci} = 1$ if $\mathbf{Y}_i$ belongs to class c, otherwise $g_{ci} = 0$.

[1]Bold capital letters denote a matrix $\mathbf{D}$, bold lower-case letters a column vector $\mathbf{d}$. $\mathbf{d}_i$ represents the i^{th} column of the matrix $\mathbf{D}$. d_{ij} denotes the scalar in the i^{th} row and j^{th} column of the matrix $\mathbf{D}$. All non-bold letters represent scalars. $||\mathbf{x}||_2 = \sqrt{\mathbf{x}^T\mathbf{x}}$ denotes the Euclidean distance.

[2]There is a dummy position $s_{m+1} = n + 1$ kept for the last segment.

3.2. Energy function for ACA

There are two major challenges in framing temporal segmentation as a clustering problem: (1) modeling the temporal variability of human actions, and (2) defining a robust metric between temporal actions. To address these problems, ACA extends previous work on kernel k-means (eq. 2) by minimizing:

$$J_{ACA}(\mathbf{G},\mathbf{s}) = \sum_{c=1}^{k}\sum_{i=1}^{m} g_{ci}\,\underbrace{dist_c(\mathbf{X}_{[s_i,s_{i+1})})}_{dist_c(\mathbf{Y}_i)} \quad (5)$$

It is worth pointing out the differences between ACA, eq. 5 and kernel k-means eq. 2: (1) ACA clusters variable features, that is, each segment $\mathbf{Y}_i$ might have a different number of frames, whereas standard kernel k-means has fixed number of features (rows of $\mathbf{d}_i$). (2) The kernel used in ACA, $dist_c(\mathbf{Y}_i)$, uses DTW that is robust to noise and invariant to the speed of the action. DTW is not a properly defined metric because it does not satisfy the triangular inequality [13]. This limitation has lead to substantial efforts to seek other distances that satisfy the metric requirements. Recently Shimodaira *et al.* [20] proposed the Dynamic Time Alignment Kernel (DTAK), which satisfies the Cauchy-Schwartz inequality, which effectively makes DTAK a metric between time sequences.

DTAK, $\kappa_{dtak}(\hat{\mathbf{X}},\tilde{\mathbf{X}})$, is defined between two time series $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1,\cdots,\hat{\mathbf{x}}_{n_1})$ and $\tilde{\mathbf{X}} = (\tilde{\mathbf{x}}_1,\cdots,\tilde{\mathbf{x}}_{n_2})$:

$$\kappa_{dtak}(\hat{\mathbf{X}},\tilde{\mathbf{X}}) = \frac{p_{n_1,n_2}}{n_1+n_2} \quad (6)$$

$$p_{i,j} = \max\begin{cases} p_{i-1,j} + \kappa_{ij} \\ p_{i-1,j-1} + 2\kappa_{ij} \\ p_{i,j-1} + \kappa_{ij}\end{cases} \quad (7)$$

where κ_{ij} is $\kappa_{ij} = e^{-\frac{1}{2\sigma^2}||\hat{\mathbf{x}}_i - \tilde{\mathbf{x}}_j||^2}$, and $p_{0,0} = 0$.

For instance, let us consider two short sequences, $\hat{\mathbf{X}} = [1,2,1,2]$ and $\tilde{\mathbf{X}} = [1,1,2,2]$. The kernel with $\sigma = \infty$ is used for simplicity, i.e. $\kappa(1,1) = \kappa(2,2) = 1$, $\kappa(1,2) = \kappa(2,1) = 0$. To construct $\mathbf{P}$ (tab. 1), we start from the upper-left corner, where $p_{1,1} = p_{0,0} + 2\kappa(1,1) = 2$. The remaining entries are gradually filled by increasing the subscripts of $p_{i,j}$ along the rows and columns. Finally, the value of DTAK is calculated by dividing the bottom-right $p_{4,4}$ by the sum of the sequence lengths, $\kappa_{dtak}(\hat{\mathbf{X}},\tilde{\mathbf{X}}) = \frac{7}{8}$.

3.3. Coordinate descent optimization

In this section, we describe a Dynamic Programming (DP)-based algorithm to perform coordinate descent to solve for ACA (i.e. $\mathbf{G},\mathbf{s}$).

To optimize over $\mathbf{s}$ and $\mathbf{G}$ with DP, we introduce an auxiliary function, $L(u,v) \triangleq \min_{\mathbf{G},\mathbf{s}:\mathbf{X}_{[u,v]}} J_{ACA}$, to store

Table 1. Example to calculate $\mathbf{P}$ with DTAK

$\mathbf{P}$	1	1	2	2
1	2	3	3	3
2	2	3	5	6
1	3	4	5	6
2	3	4	6	7

the minimum cost J_{ACA} to all segmentations on the subsequence $(\mathbf{x}_u,\mathbf{x}_{u+1},\cdots,\mathbf{x}_v)$. Observe, that $L(u,v)$ contains the minimum J_{ACA} for the best $\mathbf{s}$, $\mathbf{G}$ within the (u,v) range. Note that the function L depends on the range of segmentation in the sequence $\mathbf{X}$, allowing us to use the traditional DP *divide-and-conquer* paradigm:

$$L(u,v) = \min_{u<i\leq v} L(u,i-1) + L(i,v). \quad (8)$$

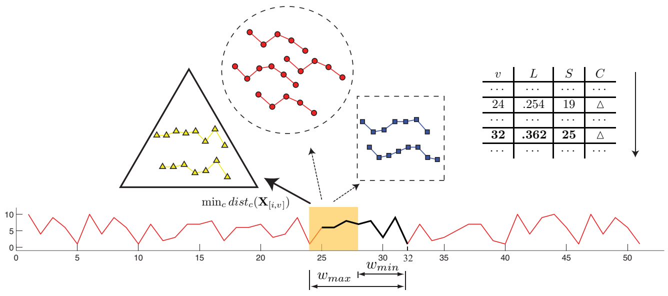
The above equation implies that the optimal decomposition of the subsequence $\mathbf{X}_{[u,v]}$ is achieved only when the segmentations on both sides $\mathbf{X}_{[u,i-1]}$ and $\mathbf{X}_{[i,v]}$ are optimal and their sum is minimal. Moreover, this recursive decomposition could be repeated until encountering the *granular segment* (i.e. w is in the range $w \in [w_{min}, w_{max}]$). In fact, previous decomposition (eq. 8) is equivalent to:

$$L(v) = \min_{w_v\in[w_{min},w_{max}]}\left(L(i-1) + \min_{\mathbf{g}}\sum_{c=1}^{k} g_c dist_c(\mathbf{X}_{[i,v]})\right) \quad (9)$$

in this case, $L(v) = L(1,v)$, and $i = v - w_v + 1$ is the head position of the granular segment $\mathbf{X}_{[i,v]}$. When $v = n$, the $L(n)$ is actually the optimal cost of the segmentation that we seek. The inner values i and $\mathbf{g}$ that lead to the minima are the head position and label for the last segment respectively. Eq. 9 unifies both point-based k-means and segment-based ACA clustering by the constraint of length $[w_{min}, w_{max}]$. If $w_{min} = w_{max} = 1$, where each segment consists of one single frame, this is equivalent to kernel k-means. Based on the recursive equation (eq. 9), we compute our algorithm with a forward and backward step to obtain $\mathbf{G}^{new}, \mathbf{s}^{new}$ based on $\mathbf{G}^{old}, \mathbf{s}^{old}$:

1. Forward step: Scan from the beginning ($v = 1$) of the sequence to its end ($v = n$), see fig. 2. $L(v)$ is assigned as $L(v) = \min_{w_v\in[w_{min},w_{max}]}(L(i-1) + dist_{\hat{c}}(\mathbf{X}_{[i,v]}))$, where $\hat{c}$ is the closest cluster of the previous segmentation ($\mathbf{G}^{old}, \mathbf{s}^{old}$) for the segment $\mathbf{X}_{[i,v]}$. A record is kept of the optimal head position i and label $\hat{c}$ for each v.

2. Backward step: Trace back from the end of sequence $v = n$. Cutting off the segment whose head position $\mathbf{s}^{new}$ and the label $\mathbf{g}^{new}$ could be indexed from the stored values in the step 1. Repeat this operation on the left part of the sequence.



v	L	S	C
$\cdots$	$\cdots$	$\cdots$	$\cdots$
24	.254	19	△
$\cdots$	$\cdots$	$\cdots$	$\cdots$
32	**.362**	**25**	△
$\cdots$	$\cdots$	$\cdots$	$\cdots$

Figure 2. Forward ACA step. To construct $L(v)$ ($v = 32$), the starting position ($i = 25$) of the best segment (bold) is selected from the pool of candidates in the shadow area. The label for the segment is determined to be the closest model (triangle). The segments of three classes, which are marked in triangle, square and circle respectively, are the segmentation of the sequence in the last step.

3.4. The Algorithm

In this section, we give further details on an efficient implementation of the DP algorithm.

The calculation of $dist_c(\mathbf{Y}_i)$ in eq. 9 involves three components: κ_{ii}, h_c, f_{ci}. The term κ_{ii} is a constant term and it does not affect the optimization of $\mathbf{G}, \mathbf{s}$. The latter two terms depend on DTAKs. However, storing all possible kernel pairs between segment is prohibitive in space $O(n^2w^2)$ and time $O(n^2w^4)$.

To make the algorithm efficient in practice, we need to reduce the computational cost of h_c and f_{ci} . First, it is possible to directly calculate the DTAK used for h_c. Given a segmentation ($\mathbf{G}^{old}, \mathbf{s}^{old}$), the number of segments are $O(\frac{n}{m})$, where the space and time needed for h_cs is $O(\frac{n^2}{m^2})$ and $O(n^2)$ respectively. For f_{ci}s, we instead maintain a relatively smaller *active* kernel matrix, $\mathbf{A}$, to reuse the previous calculations of DTAK (eq. 6) during the optimization.

The optimization process mainly consists of two algorithms, $DPSearch$ (alg. 1) and $ActiveUpdate$ (alg. 2). $DPsearch$ starts by forward constructing L storing in each position the minimum value, C and S for labels and position respectively. The components of the updated parameter will be obtained by back-tracking. During the processing, $\mathbf{A}(v, w_v, j, w_j)$, which stores the kernel between the segments $\mathbf{X}_{(v-w_v,v]}$ and $\mathbf{X}_{[s_j^{old}, s_{j+1}^{old})}$, is updated from its neighbors according to the definition in eq. 6. In order to reuse the space, a circularly-linked list of w_{max} length is implemented to index the position of segment $\mathbf{X}_{(v-w_v,v]}$, i.e. $pos_v = v \mod w_{max}$.

3.5. Complexity analysis

Given a sequence with n frames and an average segment width w, we need $O(n^2)$ space to store the kernel matrix of frames. At the beginning of each step in the iterative procedure, h_cs and the active $\mathbf{A}$ are created as a block of $O(\frac{n^2}{w^2})$ and $O(nw^2)$ respectively. As v increases, each evaluation takes $O(nw)$ to calculate the distance from $\mathbf{G}^{old}, \mathbf{s}^{old}$, and therefore it takes $O(n^2w)$ to scan through the whole sequence. To sum up, the space complexity is $O(n^2)$, which makes it possible to process sequences with thousands of frames. The overall time complexity is $O(n^2wt)$, where t is the number of iterative steps.

Algorithm 1: $\mathbf{G}^{new}, \mathbf{s}^{new} = DPSearch(\mathbf{G}^{old}, \mathbf{s}^{old})$

```
Obtain the $h_c$s from $\mathbf{G}^{old}, \mathbf{s}^{old}$;
Initialize all $L(v) \leftarrow \infty$ except $L(0) \leftarrow 0$;
for $v \leftarrow 1$ to $n$ do
    for $w_v \leftarrow 1$ to $\min(w_{max}, v)$ do
        $ActiveUpdate(\mathbf{A}, v, w_v, \mathbf{G}^{old}, \mathbf{s}^{old})$;
        if $w_v \geq w_{min}$ then
            Head position $i \leftarrow v - w_v + 1$;
            $\hat{c} \leftarrow \arg\min_c dist_c(\mathbf{X}_{[i,v]})$;
            $\hat{l} \leftarrow L(i-1) + dist_{\hat{c}}(\mathbf{X}_{[i,v]})$;
            if $\hat{l} < L(v)$ then
                $L(v), C(v), S(v) \leftarrow \hat{l}, \hat{c}, i$;
            end
        end
    end
end
while $v > 0$ do
    Insert $C(v), S(v)$ into the top of $\mathbf{G}^{new}, \mathbf{s}^{new}$;
    $v \leftarrow S(v) - 1$;
end
```

Algorithm 2: $ActiveUpdate(\mathbf{A}, v, w_v, \mathbf{G}^{old}, \mathbf{s}^{old})$

```
$pos_v \triangleq v \mod w_{max}$;
for $j = 1$ to $m$ do
    for $w = 1$ to $w_j$ do
        Update $\mathbf{A}(pos_v, w_v, j, w)$ from
        $\mathbf{A}(pos_v, w_v, j, w-1)$,
        $\mathbf{A}(pos_{v-1}, w_v - 1, j, w)$ and
        $\mathbf{A}(pos_{v-1}, w_v - 1, j, w-1)$;
    end
end
$f_{vc} \leftarrow \sum_{j=1}^{m} g_{jc}^{old} \mathbf{A}(pos_v, w_v, j, w_j)$;
```

4. Segmentation on motion capture data

This section describes two strategies to scale ACA to segment large collections of motion capture data: (1) temporal reduction, and (2) good initialization.

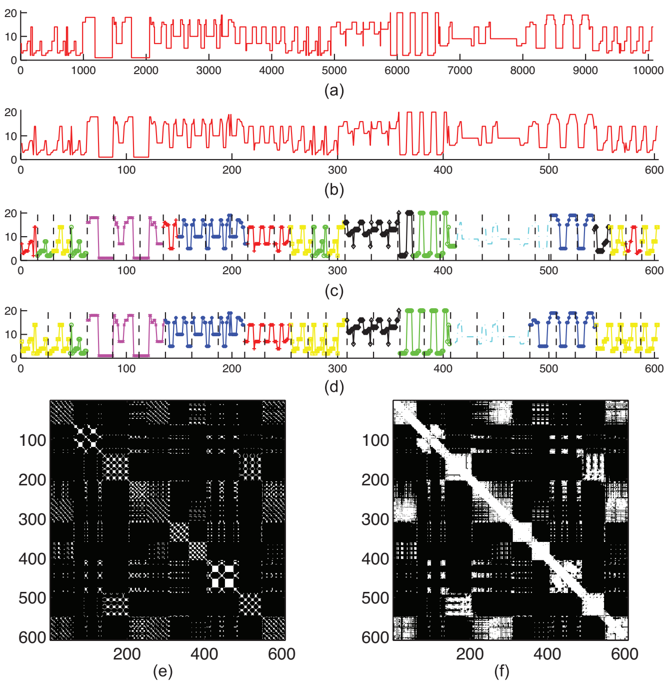


Figure 3. Temporal reduction and initialization. (a) Label indexes for 10078 frames after k-means. (b) Temporal reduction (608 frames). (c) Initialized segmentation. (d) Final segmentation after running ACA. (e) Similarity matrix computed from (b). (f) Improved similarity matrix.

4.1. Temporal reduction

It is not computationally practical to run ACA on large amounts of motion capture data. Because human motion is typically smooth, and recent work [4,9] has shown evidence that human motion is locally linear, it is possible to temporally reduce the number of frames without losing important information relevant for temporal segmentation. Following previous work on temporal segmentation [7], we first apply a clustering step to group frames into k_{reduce} classes, and remove irrelevant consecutive frames within the same class.

We use the Carnegie Mellon Motion Capture database, which contains 149 subjects performing several activities. The motion capture system uses 41 markers per subject. Similar to the method of Barbic *et al.* [2], we only consider the 14 most informative joints out of 29. The 3-D Euler angles are transformed to 4-D quaternions to provide a smoother and continuous representation of motion. We apply the k-means [11] algorithm to cluster the frames into k_{reduce} classes. A large k_{reduce} is usually preferred to capture subtle human behavior. Fig. 3(a) shows the labels ($k_{reduce} = 20$) of a 10078-frame sequence (subject 86, trial 4), which contains seven distinct actions. Every 20 consecutive frames that belong to the same class are reduced to one frame (fig. 3(b)).

4.2. ACA initialization

Minimizing ACA is a non-convex optimization problem, and the quality of the solution is highly sensitive to initial conditions [5]. In this section, we describe a coarse segmentation process based on spectral clustering methods that provides a good initialization for ACA.

Generally speaking, many human actions, such as walking or running, are periodic movements. This periodicity can be observed in the block structure of the similarity matrix between all the frames. Fig. 3(e) shows the similarity matrix of the 608 frames (after temporal reduction). The similarity matrix is computed by assigning 1 if the two frames belong to the same class given by k-means. To emphasize the frames that might belong to the same type of actions, we modified the similarity matrix by propagating the similarity between two temporally close frames that share the same cluster. More specifically, given two pairs of frames, $\mathbf{x}_{i_1}, \mathbf{x}_{j_1}$ vs $\mathbf{x}_{i_2}, \mathbf{x}_{j_2}$, we define the new similarity κ' as

$$\left.\begin{array}{l}\kappa'_{i_1 j_1}\\ \kappa'_{i_2 j_2}\\ \kappa'_{i_1 j_2}\\ \kappa'_{j_1 i_2}\end{array}\right\} \leftarrow 1, \quad \text{if} \left\{\begin{array}{c}\kappa_{i_1 i_2} = 1\\ \kappa_{j_1 j_2} = 1\\ i_1 - j_1 = i_2 - j_2\\ |i_1 - j_1| \leq w_{max}\end{array}\right. \tag{10}$$

Fig. 3(f) shows the frames lying in the similar actions are linked together after the propagation.

We use spectral clustering algorithms [7, 18] to find an embedding where samples are easier to cluster. Notice that long (short) segments will be divided (merged) to satisfy the predefined length constraint $[w_{min}, w_{max}]$ for each of the actions. Fig. 3 (c) shows that this coarse initialization (fig. 3 (d)) identifies meaningful segments.

5. Experimental results

In this section, several experiments on synthetic and real dat evaluate the segmentation performance of ACA.

5.1. Synthetic data

In the first experiment, we synthetically generate a random 1-D sequence (fig. 4(a)) with four temporal clusters. The length of each segment is restricted to be between 10 and 15 samples (frames), and the value of each sample is a uniform random integer between in the range $[1, 20]$. Several artificial frames are randomly inserted (temporal noise) into the sequence. The parameter, p_{noise}, controls the amount of noise. For instance, $p_{noise} = 0.2$ indicates that one noise frame might be inserted every 5 frames.

The ACA algorithm runs 10 times with random initialization, and the solution with minimum J_{ACA} is selected. DTAK is constructed based on an exponential kernel κ with $\sigma = \infty$ (fig. 4(c)). Observe that in this case, no temporal reduction or good initialization is used.

To quantify the segmentation accuracy, we need to compare the segmentation provided by ACA and the ground-truth[3]. To compute the accuracy we use a confusion matrix (fig. 4(e)) between the ACA and ground-truth. The confusion matrix is calculated as follows:

$$\mathbf{C}(c_1, c_2) = \sum_{i=1}^{m_{ACA}} \sum_{j=1}^{m_{truth}} g_{c_1 i}^{ACA} g_{c_2 j}^{truth} |\mathbf{Y}_i^{ACA} \cap \mathbf{Y}_j^{truth}| \tag{11}$$

where $\mathbf{Y}_i^{ACA}$ and $\mathbf{Y}_j^{truth}$ are two segments given by the ACA algorithm and ground-truth data respectively, and $|\mathbf{Y}_i^{ACA} \cap \mathbf{Y}_j^{truth}|$ denotes the number of frames they share. Fig. 4(e) illustrates the confusion matrix for the synthetic problem. The classical Hungarian algorithm [15] is applied in order to find the optimum solution for the cluster correspondence problem.

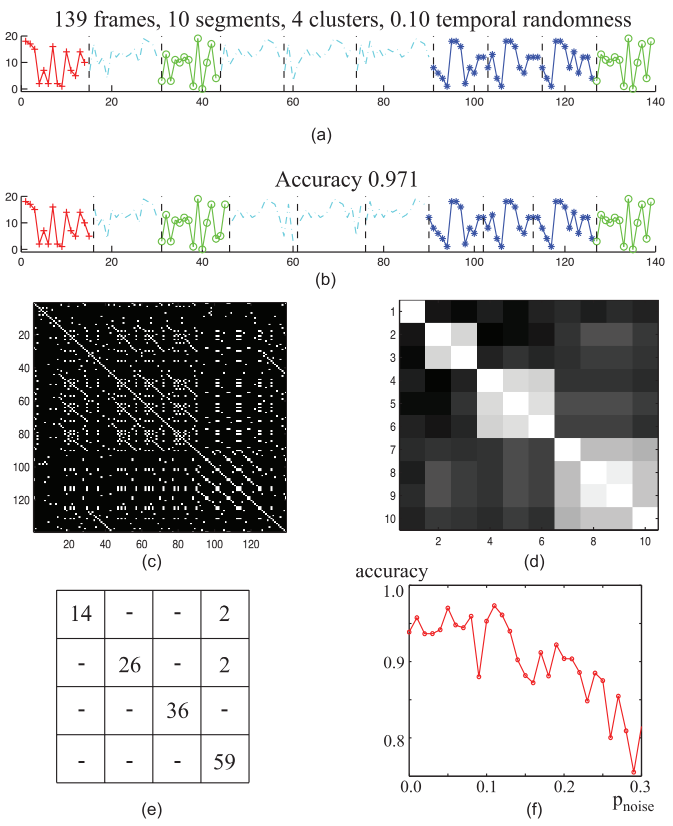
Fig. 4(d) depicts the DTAK matrix for the segments given by the ACA algorithm. Good segmentations tend to have large within-class and low between-class connectivity. Fig. 4(f) shows the accuracy results of our algorithm for different levels of noise ($p_{noise} = 0.0$-0.3). For each p_{noise}, we repeated the above generation of data 10 times to average the results.

5.2. Motion capture data

In the second experiment, we choose the 15 sequences performed by subject 86, each of which is a combination of 10 natural actions (e.g. walking, punching, drinking, running). Typically each sequence contains 8000 frames (70 secs). Quaternions are used as features to group the frames into 20 clusters, and reduce the length of the sequence as explained in section 4.1. The length for each activity ranges from 50 to 200 frames. After initializing with the algorithm described in section 4.2, ACA is optimized until convergence. For each sequence, ACA would usually converge in 3-5 iterations. Each iteration took average 30 seconds in an unoptimized Matlab code with Intel Core 2 Duo 2.4 GHz and 2 GB memory.

Fig. 5 shows the segmentation obtained through ACA, manual labeling, and the method proposed by Barbic *et al.* method [2] respectively, in four sequences. Different actions are marked with different colors. The black stripes in the human label sequences indicate areas where the judgments vary among labelers, while areas in the PCA bars indicate the 2-sec preparation period used for estimating the underlying quaternion distribution [2]. We should mention that the PCA approach works in an on-line procedure, while ACA is an off-line approach. Moreover, ACA identifies the distinct actions by providing a segmentation closer to the one provided by the human observer. In fact, the motions that are almost cyclic were more clearly detected (dark lines

[3]Recall that for the synthetic data, the ground-truth segmentation is known in advance.



14	-	-	2
-	26	-	2
-	-	36	-
-	-	-	59

Figure 4. A synthetic example: (a) ground-truth segmentation, (b) segmentation given ACA, (c) similarity matrix, (d) similarity matrix computed on segments given by ACA (the segments are reordered according to their labels), (e) confusion matrix, and (f) clustering accuracy versus temporal noise.

whose both sides have the same color) by ACA than PCA-based approaches or human labeling.

6. Conclusions

In this paper, we have presented ACA, an extension of kernel k-means for temporal segmentation. ACA combines standard vector-space approaches for clustering with Dynamic Time Alignment Kernel (DTAK) and Dynamic Programming (DP). The main contributions of our paper are: (1) formulation of temporal segmentation with ACA, (2) temporal reduction and initialization strategies for ACA, and (3) efficient computation of ACA. ACA has been applied to temporal decomposition of motion capture data into a set of actions, but it is a generic algorithm and can be applied to other data (e.g. facial expression, speech). Although ACA has shown promising preliminary results, there is still the need for algorithms to automatically select the optimal number of actions and avoid local minima in the optimization.

Acknowledgements This work was partially supported by the National Science Foundation under Grant No.

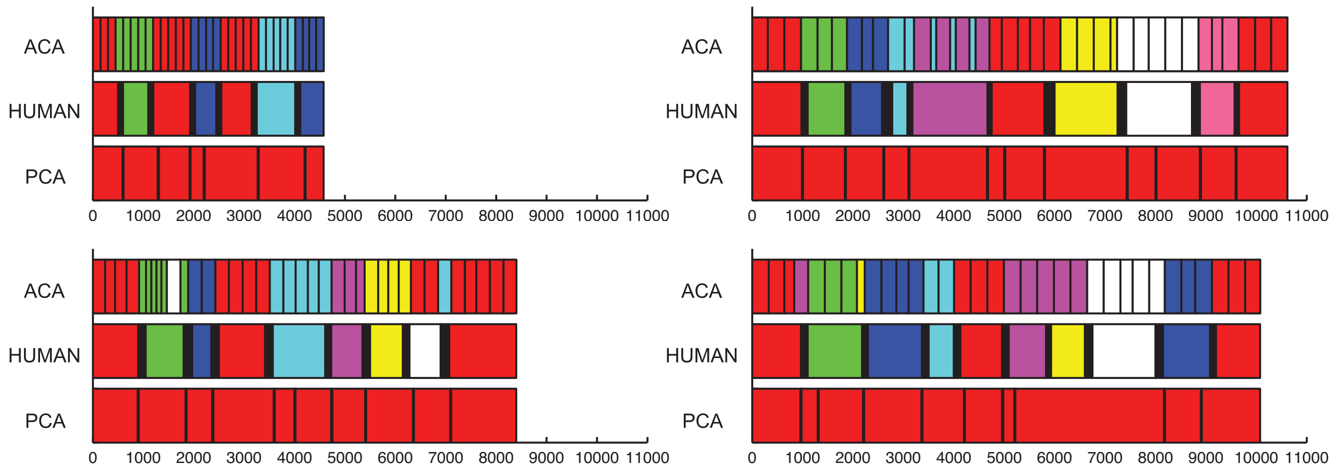


Figure 5. Motion segmentation provided by human observers, PCA and ACA. Black lines indicate the boundaries of actions. The different colors used in HUMAN and ACA correspond to distinct actions. For ACA, the black lines within the area of same color show the composition of cyclic movements.

EEEC-0540865. The data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] A. Anagnostopoulos, M. Vlachos, M. Hadjieleftheriou, E. J. Keogh, and P. S. Yu. Global distance-based segmentation of trajectories. In *KDD*, pages 34–43, 2006.

[2] J. Barbic, A. Safonova, J.-Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard. Segmenting motion capture data into distinct behaviors. In *Graphics Interface*, pages 185–194, 2004.

[3] P. Beaudoin, S. Coros, M. van de Panne, and P. Poulin. Motion-motif graphs. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2008.

[4] R. Bowden. Learning statistical models of human motion. *IEEE Workshop on Human Modeling, Analysis and Synthesis, CVPR*, 2000.

[5] P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In *ICML*, pages 91–99, 1998.

[6] A. Bruderlin and L. Williams. Motion signal processing. In *ACM SIGGRAPH*, pages 97–104, 1995.

[7] F. De la Torre, J. Campoy, Z. Ambadar, and J. F. Cohn. Temporal segmentation of facial behavior. In *ICCV*, pages 1–8, 2007.

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

[9] K. Forbes and E. Fiume. An efficient search algorithm for motion data using weighted PCA. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 67–76, 2005.

[10] G. Guerra-Filho and Y. Aloimonos. Understanding visuo-motor primitives for motion synthesis and analysis. *Comp. Anim. Virtual Worlds*, 17:207–217, 2006.

[11] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[12] O. C. Jenkins and M. J. Matarić. Deriving action and behavior primitives from human motion data. In *IROS*, volume 3, pages 2551–2556, 2002.

[13] E. J. Keogh. Exact indexing of dynamic time warping. In *VLDB*, pages 406–417, 2002.

[14] E. J. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. *Data Min. Knowl. Discov.*, 7(4):349–371, 2003.

[15] D. E. Knuth. *The Stanford GraphBase*. Addison-Wesley Publishing Company, 1993.

[16] L. Kovar and M. Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph.*, 23(3):559–568, 2004.

[17] Y. Li, T.-S. Wang, and H.-Y. Shum. Motion texture: a two-level statistical model for character motion synthesis. *ACM Trans. Graph.*, 21(3):465–472, 2002.

[18] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[19] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[20] H. Shimodaira, K.-I. Noma, M. Nakai, and S. Sagayama. Dynamic time-alignment kernel in support vector machine. In *NIPS*, pages 921–928, 2001.

[21] L. Zelnik-Manor and M. Irani. Event-based analysis of video. In *CVPR*, pages 123–130, 2001.

[22] H. Zhong, J. Shi, and M. Visontai. Detecting unusual activity in video. In *CVPR*, pages 819–826, 2004.