Hierarchical Aligned Cluster Analysis for Temporal Clustering of Human Motion

Feng Zhou, *Student Member*, *IEEE*, Fernando De la Torre, and Jessica K. Hodgins

Abstract—Temporal segmentation of human motion into plausible motion primitives is central to understanding and building computational models of human motion. Several issues contribute to the challenge of discovering motion primitives: the exponential nature of all possible movement combinations, the variability in the temporal scale of human actions, and the complexity of representing articulated motion. We pose the problem of learning motion primitives as one of temporal clustering, and derive an unsupervised hierarchical bottom-up framework called hierarchical aligned cluster analysis (HACA). HACA finds a partition of a given multidimensional time series into m disjoint segments such that each segment belongs to one of k clusters. HACA combines kernel k-means with the generalized dynamic time alignment kernel to cluster time series data. Moreover, it provides a natural framework to find a low-dimensional embedding for time series. HACA is efficiently optimized with a coordinate descent strategy and dynamic programming. Experimental results on motion capture and video data demonstrate the effectiveness of HACA for segmenting complex motions and as a visualization tool. We also compare the performance of HACA to state-of-the-art algorithms for temporal clustering on data of a honey bee dance. The HACA code is available online.

Index Terms—Temporal segmentation, time series clustering, time series visualization, human motion analysis, kernel k-means, spectral clustering, dynamic programming

1 Introduction

Systems that can detect, recognize, and synthesize human motion are of interest in both research and industry due to the large number of potential applications in virtual reality, smart surveillance systems for advanced user interfaces, and motion analysis (see [1], [2], [3] for a review). The quality of the detection, recognition, or synthesis in these applications greatly depends on the spatial and temporal resolution of motion databases, as well as the complexity of the models. Unsupervised techniques to learn motion primitives from training data have recently attracted the interest of many scientists in computer vision [4], [5], [6], [7], [8], [9], [10], [11], [12] and computer graphics [13], [14], [15], [16], [17], [18], [19], [20]. Fig. 1 illustrates the problem addressed in this paper: Given a sequence of a person walking and running, the first level of the hierarchy provided by our algorithm (HACA) is able to group the frames into two classes: running and walking. In a finer level of the hierarchy, HACA decomposes each of the actions (e.g., running, walking) into motion primitives of smaller temporal scale.

The inherent difficulty of temporally decomposing human motion stems from the large number of possible movement combinations, a relatively large range of temporal scales for different behaviors, the irregularity in the periodicity of human actions, and the intraperson motion variability. To address these challenges, this paper frames the problem of hierarchical temporal decomposition of human motion as an unsupervised learning problem, and proposes a hierarchical aligned cluster analysis (HACA). HACA is a generalization of kernel k-means (KKM) and spectral clustering (SC) for time series clustering and embedding.

Over the last few years, several approaches for unsupervised segmentation of activities have been proposed (see, for example, [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15]). HACA presents several advantages:

- The temporal clustering problem is posed as an energy minimization.
- HACA provides a natural embedding for clustering and visualizing time series data.
- HACA provides a hierarchical decomposition at several temporal scales (see Fig. 1). The time granularity of the motion primitives is specified manually.
- Minimizing HACA is an NP problem. This paper proposes an efficient coordinate descent minimization via dynamic programming.

2 Previous Work

We build on prior research in human motion analysis and temporal clustering.

2.1 Human Motion Analysis

Extensive literature in graphics and computer vision addresses the problem of grouping human actions. In the computer graphics literature, Barbic et al. [15] proposed an algorithm to decompose human motion into distinct actions based on probabilistic principal component analysis, which places a cut when the distribution of human poses changes. Jenkins et al. [13], [14] used the zero-velocity crossing points of the angular velocity to segment the stream of motion capture data into short sequences. Jenkins and Matarić [21]

• *The authors are with the Robotics Institute, Carnegie Mellon University, Smith Hall, 5000 Forbes Ave, Pittsburgh, PA 15232. E-mail: zhfe99@gmail.com, {ftorre, jkh}@cs.cmu.edu.*

Manuscript received 27 Apr. 2011; revised 19 Apr. 2012; accepted 12 May 2012; published online 14 June 2012.
Recommended for acceptance by C. Sminchisescu.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2011-04-0268.
Digital Object Identifier no. 10.1109/TPAMI.2012.137.

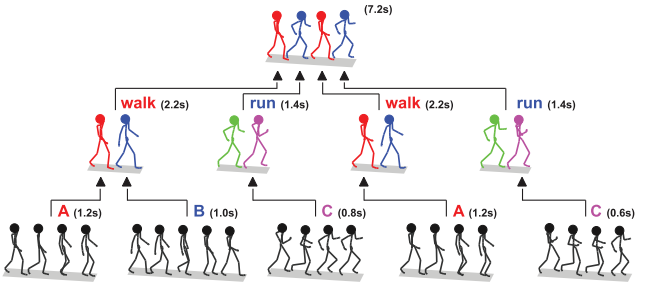


Fig. 1. Hierarchical decomposition of human motion. Each level of the figure corresponds to one hierarchy found by HACA at different temporal resolutions. The top row shows some samples of motion capture data of a person walking and then running (7.2 seconds). The second row shows the first level of the decomposition found by HACA. Each temporal pattern contains samples for walking or running. The bottom row shows the lower level, which contains subcycles of running and walking.

further extended the work by finding a nonlinear embedding using Isomap [22] that reveals the temporal structure of segmented motion. Beaudoin et al. [20] developed a string-based motif-finding algorithm to decompose motion into action primitives and interpret actions as a composition on the alphabet of these action primitives.

In the computer vision literature, Zhong et al. [23] used a bipartite graph co-clustering algorithm to segment and detect unusual activities in video. Zelnik-Manor and Irani [5] extracted spatiotemporal features at multiple temporal scales to isolate and cluster events. An outcome of the clustering process is the temporal segmentation of long video sequences into event subsequences. De la Torre et al. [9] proposed a geometric-invariant clustering algorithm to decompose a stream of facial behavior into facial gestures. Unusual facial expressions can be detected through the analysis of outlying temporal patterns. De la Torre and Agell [24] decomposed a multimodal stream of human behavior into several activities using semi-supervised temporal clustering. Recently, Guerra-Filho and Aloimonos [8], [25] presented a linguistic framework for modeling and learning human activity representations from video. To obtain a low-level representation, they segmented the movement by estimating the velocity and acceleration of the actuator attached to the joint. Minnen et al. [26] discovered motifs in real-valued, multivariate time series data by locating regions of high density in the space of all time series subsequences.

2.2 Temporal Clustering

Segmentation and clustering of time series data is a topic that has been explored in fields other than computer vision and graphics. In particular, there is a substantial amount of work in the field of data mining [27], [28], speech processing [29], [30], animal behavior analysis [12], [31], and signal processing [32], [33].

Two of the most popular approaches are change-point detection and switching linear dynamical systems (SLDSs). The goal of change-point detection [32], [33], [34] is to identify changes at unknown times and estimate the location of changes in stochastic processes. Unlike previous work on change-point detection, HACA finds the change points that minimize the error across several segments (not only two) that belong to one of k clusters.

SLDSs describe the dynamics of the time series by switching several linear dynamical systems over time. The switching states in SLDS inference implicitly provide the segmentation of an input sequence. Because exact inference in SLDS is intractable, Pavlović et al. [35] proposed approximate inference algorithms by casting the SLDS model as a dynamic Bayesian network (DBN). Oh et al. [12] introduced a data-driven MCMC (DD-MCMC) inference method to identify the exact posterior of SLDSs in the presence of intractability. In their framework [12], the standard SLDS has been improved by incorporating a duration model, thereby yielding a more accurate result in segmentation. To address the problem of learning SLDSs with an unknown number of modes, Fox et al. [31] proposed a nonparametric Bayesian method that utilizes the hierarchical Dirichlet process (HDP) as a prior on the parameters of SLDSs. Recently, Fox et al. [36] further extended this work by adding the beta process prior to discover and model dynamical behaviors that are shared among multiple related time series.

3 Aligned Cluster Analysis (ACA)

This section describes ACA and hierarchical ACA (HACA), an extension of kernel k-means and spectral clustering for clustering time series. Section 3.1 reviews the matrix formulation for k-means, KKM and SC. Section 3.2 describes the properties of the frame kernel matrix that are key to understanding ACA and HACA. Section 3.3 reviews the dynamic time alignment kernel (DTAK) that is used as a similarity measure between segments. Section 3.4 proposes the ACA energy function and its matrix formulation is discussed in Section 3.5. Section 3.6 describes a coordinate-descent strategy for optimizing ACA. Section 3.7 presents an efficient optimization strategy for ACA. Section 3.8 describes HACA.

3.1 k-Means, KKM and SC

Clustering refers to the partition of n data points into k disjointed clusters. Among various approaches to unsupervised clustering, k-means [37] is favored for its simplicity. k-means clustering splits a set of n samples into k groups by minimizing the within-cluster variation. k-means clustering finds the partition of the data that is a local optima of the following energy function [38], [39]:

$$J_{km}(\mathbf{Z},\mathbf{G}) = \sum_{c=1}^{k}\sum_{i=1}^{n} g_{ci}\|\mathbf{x}_i - \mathbf{z}_c\|^2 = \|\mathbf{X}-\mathbf{Z}\mathbf{G}\|_F^2, \quad \text{s.t. } \mathbf{G}^T\mathbf{1}_k = \mathbf{1}_n, \tag{1}$$

where $\mathbf{x}_i \in \mathbb{R}^d$ (see notation[1]) is a vector representing the ith data point and $\mathbf{z}_c \in \mathbb{R}^d$ is the geometric centroid of the data points for class c. $\mathbf{G} \in \{0,1\}^{k\times n}$ is a binary indicator matrix such that $g_{ci} = 1$ if the sample $\mathbf{x}_i$ belongs to cluster c and zero otherwise. The k-means algorithm performs

1. Bold capital letters denote a matrix $\mathbf{X}$, bold lower-case letters a column vector $\mathbf{x}$. $\mathbf{x}_i$ represents the ith column of the matrix $\mathbf{X}$. x_{ij} denotes the scalar in the ith row and jth column of the matrix $\mathbf{X}$. All nonbold letters represent scalars. $\mathbf{1}_{m\times n}, \mathbf{0}_{m\times n} \in \mathbb{R}^{m\times n}$ are matrices of ones and zeros. $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ is an identity matrix. $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}}$ denotes the euclidean distance. $\|\mathbf{X}\|_F^2 = \mathrm{tr}(\mathbf{X}^T\mathbf{X}) = \mathrm{tr}(\mathbf{X}\mathbf{X}^T)$ designates the Frobenius norm of a matrix. $\mathbf{X}\circ\mathbf{Y}$ is the Hadamard product of matrices. $\mathrm{diag}(\mathbf{x})$ is a diagonal matrix whose diagonal elements are $\mathbf{x}$. $[i,j]$ and $[i,j)$ list the integers $\{i, i+1,\ldots,j-1,j\}$ and $\{i,i+1,\ldots,j-1\}$, respectively. $\mathbf{X}_{[i,j]} = [\mathbf{x}_i,\mathbf{x}_{i+1},\ldots,\mathbf{x}_j]$ is composed of the columns of $\mathbf{X}$ indexed by the integers in $[i,j]$. $\dot{\mathbf{X}}$ denotes the previous value of $\mathbf{X}$ in an updating scheme.

coordinate descent in the energy function $J_{km}(\mathbf{Z},\mathbf{G})$. Given the actual value of the means $\mathbf{Z}\in\mathbb{R}^{d\times n}$, the first step finds $\mathbf{g}_i\in\{0,1\}^k$ for each data point $\mathbf{x}_i$ such that one of the rows is one and the others are zero, while minimizing (1). The second step computes $\mathbf{Z}=\mathbf{X}\mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}$, which is equivalent to calculating the mean of each cluster. These alternating steps are guaranteed to converge to a local minimum of $J_{km}(\mathbf{Z},\mathbf{G})$ [40].

A major limitation of the k-means algorithm is that it is only optimal for spherical clusters. To overcome this limitation, kernel k-means [41] implicitly maps the data to a higher dimensional space using kernels. KKM [41], [38] minimizes

$$J_{kkm}(\mathbf{G})=\sum_{c=1}^{k}\sum_{i=1}^{n}g_{ci}\underbrace{\|\phi(\mathbf{x}_i)-\mathbf{z}_c\|^2}_{dist^2_\phi(\mathbf{x}_i,\mathbf{z}_c)}=\|\phi(\mathbf{X})-\mathbf{Z}\mathbf{G}\|_F^2,\quad \text{s.t.}\quad \mathbf{G}^T\mathbf{1}_k=\mathbf{1}_n, \tag{2}$$

where $dist^2_\phi(\mathbf{x}_i,\mathbf{z}_c)$ is the squared distance between the ith sample and the center of class c in the feature space, that is:

$$dist^2_\phi(\mathbf{x}_i,\mathbf{z}_c)=\kappa_{ii}-\frac{2}{n_c}\sum_{j=1}^{n}g_{cj}\kappa_{ij}+\frac{1}{n_c^2}\sum_{j_1,j_2=1}^{n}g_{cj_1}g_{cj_2}\kappa_{j_1j_2}, \tag{3}$$

where $n_c=\sum_{j=1}^{n}g_{cj}$ is the number of samples that belong to class c. The kernel function κ is defined as $\kappa_{ij}=\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$.

Similarly to the first step in the k-means algorithm, KKM assigns the sample to the closest cluster mean ($\dot{\mathbf{Z}}$) computed in the previous step:

$$g_{c_i^*i}=1,\quad \text{where}\quad c_i^*=\arg\min_c dist^2_\phi(\mathbf{x}_i,\dot{\mathbf{z}}_c). \tag{4}$$

In KKM, in general, the mean cannot be computed explicitly. However, there is no need to compute the mean because $dist^2_\phi(\mathbf{x}_i,\dot{\mathbf{z}}_c)$ can be calculated from the kernel matrix.

Spectral clustering also minimizes a weighted version of (2), but $\mathbf{G}$ is relaxed to be continuous. See [41] and [38] for a more detailed explanation of the relation between SC and KKM. In this paper, we will extend KKM and SC to cluster and find a low-dimensional embedding of a given time series.

3.2 Frame Kernel Matrix

This section describes some properties of the frame kernel matrix, $\mathbf{K}=\phi(\mathbf{X})^T\phi(\mathbf{X})\in\mathbb{R}^{n\times n}$, where $\mathbf{X}\in\mathbb{R}^{d\times n}$ is a multidimensional time series of length n. Each entry, κ_{ij}, defines the similarity between two frames, $\mathbf{x}_i$ and $\mathbf{x}_j$, by means of a kernel function $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. The linear kernel, $\kappa_{ij}=\mathbf{x}_i^T\mathbf{x}_j$, and the Gaussian kernel, $\kappa_{ij}=\exp(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2})$, are perhaps the most commonly used kernels. In the literature of dynamical systems, the frame kernel matrix ($\mathbf{K}$) is alternatively called the recurrence matrix [42], [43], and its structure reveals important information about the dynamics.

To illustrate the properties of this matrix, consider the 1D time series shown in Fig. 2a. In this case, we compute the frame kernel matrix using the exponential kernel, $\kappa_{ij}=\exp(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\sigma^2})$. We choose an infinitely small bandwidth ($\sigma\to 0$) to make a binary frame kernel matrix (Fig. 2b). In the following, we highlight one property, period ambiguity, that is relevant to ACA. Fig. 2a (second and third row) plots two

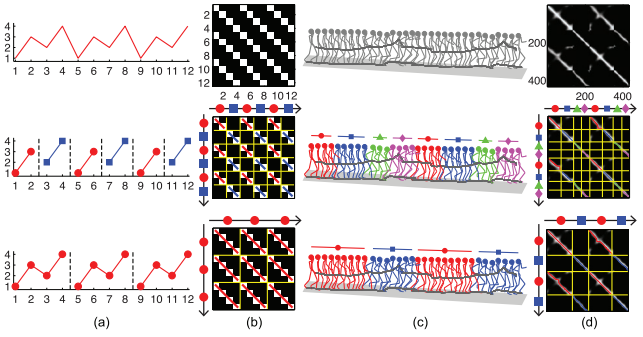


Fig. 2. Decomposition of time series into two different temporal scales. (a) Temporal clustering of 1D time series. Vertical black dotted lines denote the segment's boundaries. (b) Frame kernel matrix. Each segment corresponds to a rectangular block (yellow line). (c) Temporal clustering of motion capture data. (d) Frame kernel matrix.

different, but valid, decompositions of the same time series at two different temporal scales. To avoid this temporal ambiguity, we introduce a parameter $n_{\max}$ to constrain the length of the segments. In this case, we set $n_{\max}=2$ and $n_{\max}=4$ for the second and third rows, respectively. Similarly, Fig. 2c shows an example of a multidimensional time series of motion capture data of a subject doing two activities. Fig. 2d illustrates the corresponding frame kernel matrix at two different temporal resolutions levels.

3.3 Dynamic Time Alignment Kernel

A temporal clustering algorithm needs to define a distance between segments of different length. Ideally, this distance should be invariant to the speed of the human action. This section reviews the DTAK that extends dynamic time warping (DTW) to satisfy the properties of a distance.

A frequent approach to aligning time series has been DTW. A known drawback of using DTW as a distance is that it fails to satisfy the triangle inequality [44]. To address this issue, Shimodaira et al. [45] proposed the DTAK. Given two sequences $\mathbf{X}=[\mathbf{x}_1,\ldots,\mathbf{x}_{n_x}]\in\mathbb{R}^{d\times n_x}$ and $\mathbf{Y}=[\mathbf{y}_1,\ldots,\mathbf{y}_{n_y}]\in\mathbb{R}^{d\times n_y}$ DTAK computes the similarity using dynamic programming. DTAK uses the cumulative kernel matrix $\mathbf{U}\in\mathbb{R}^{n_x\times n_y}$ (as in DTW), computed in a recursive manner as

$$\tau(\mathbf{X},\mathbf{Y})=\frac{u_{n_xn_y}}{n_x+n_y},\ u_{ij}=\max\begin{cases}u_{i-1,j}+\kappa_{ij}\\ u_{i-1,j-1}+2\kappa_{ij}\\ u_{i,j-1}+\kappa_{ij}.\end{cases} \tag{5}$$

$\mathbf{U}$ is initialized at the upper left, i.e., $u_{11}=2\kappa_{11}$.

$$\kappa_{ij}=\phi(\mathbf{x}_i)^T\phi(\mathbf{y}_j)=\exp\left(-\frac{\|\mathbf{x}_i-\mathbf{y}_j\|^2}{2\sigma^2}\right)$$

is the frame kernel that constitutes the kernel matrix $\mathbf{K}\in\mathbb{R}^{n_x\times n_y}$. Fig. 3c illustrates the procedure to build $\mathbf{U}$ for two short sequences (Fig. 3a). Fig. 3b shows the binary frame kernel matrix $\mathbf{K}$ when $\sigma\to 0$. The final value of DTAK, $\tau(\mathbf{X},\mathbf{Y})=\frac{11}{13}$, is computed by normalizing the bottom right of $\mathbf{U}$ with the sum of sequence lengths.

A more revealing mathematical expression to understand DTAK can be obtained using matrix notation. Observe that DTAK computes a monotonic trajectory (the red curve in Fig. 3b) starting from the top-left corner to the bottom-right corner of the frame kernel matrix $\mathbf{K}$. This

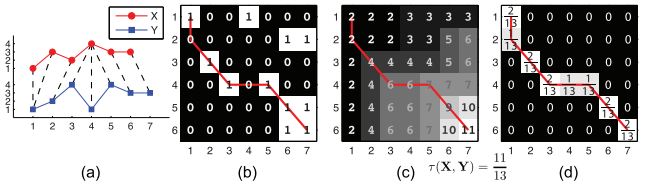


Fig. 3. Computation of DTAK. (a) Alignment examples for two 1D sequences. (b) Frame kernel matrix ($\mathbf{K}$). (c) Cumulative kernel matrix ($\mathbf{U}$). (d) Normalized correspondence matrix ($\mathbf{W}$).

monotonic trajectory can be mathematically parameterized by two frame indexes vectors $\mathbf{p} \in \{1:n_x\}^l$ and $\mathbf{q} \in \{1:n_y\}^l$, where l is the optimal number of steps that need to align $\mathbf{X}$ and $\mathbf{Y}$ by DTAK (e.g., $l = 8$ in the case of Fig. 3). Using these indexes, we can define a new normalized correspondence matrix $\mathbf{W} = [w_{ij}]_{n_x \times n_y} \in \mathbb{R}^{n_x \times n_y}$, where $w_{ij} = \frac{1}{n_x+n_y}(p_c - p_{c-1} + q_c - q_{c-1})$ if there exist $p_c = i$ and $q_c = j$ for some c (i.e., for every one of the l steps we have two indexes that encode the correspondence between the two time series). Otherwise, $w_{ij} = 0$. See Fig. 3d for an example of $\mathbf{W}$. Using this new matrix, DTAK can be rewritten in a more compact way as follows:

$$\tau(\mathbf{X},\mathbf{Y}) = \mathrm{tr}(\mathbf{K}^T\mathbf{W}) = \psi(\mathbf{X})^T\psi(\mathbf{Y}), \tag{6}$$

where $\psi(\cdot)$ denotes a mapping of the sequence into a feature space. By the Mercer theorem [46] this mapping exists when $\tau(\mathbf{X},\mathbf{Y})$ is a positive definite kernel. Unfortunately, DTAK is not necessarily a strictly positive definite kernel [47], [48], and a regularization of the kernel matrix needs to be done (see Section 3.5).

3.4 Energy Function for ACA

Given a sequence $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ with n samples, ACA decomposes $\mathbf{X}$ into m disjointed segments, each of which corresponds to one of k classes. The ith segment, $\mathbf{Y}_i \doteq \mathbf{X}_{[s_i,s_{i+1})} = [\mathbf{x}_{s_i}, \ldots, \mathbf{x}_{s_{i+1}-1}] \in \mathbb{R}^{d \times n_i}$, is composed of samples that begin at position s_i and end at $s_{i+1} - 1$. The length of the segment is constrained as $n_i = s_{i+1} - s_i \le n_{\max}$, where $n_{\max}$ is the maximum length of the segment and controls the temporal granularity of the factorization. An indicator matrix $\mathbf{G} \in \{0,1\}^{k \times m}$ assigns each segment to a class; $g_{ci} = 1$ if $\mathbf{Y}_i$ belongs to class c, otherwise $g_{ci} = 0$. For instance (see Fig. 4), the 1D sequence (Fig. 4a) with 23 frames has been segmented into seven segments that belong to three clusters (Fig. 4b).

A major limitation of standard k-means clustering for analysis of time series data [49] is that the temporal ordering of the frames is not taken into account. This section combines KKM and SC with the DTAK to achieve temporal clustering. ACA extends previous work on KKM and SC by minimizing:

$$\begin{aligned} J_{aca}(\mathbf{G},\mathbf{s}) &= \sum_{c=1}^{k}\sum_{i=1}^{m} g_{ci} \underbrace{\|\psi(\mathbf{X}_{[s_i,s_{i+1})}) - \mathbf{z}_c\|^2}_{dist^2_\psi(\mathbf{Y}_i,\mathbf{z}_c)} \\ &= \|[\psi(\mathbf{Y}_1), \ldots, \psi(\mathbf{Y}_m)] - \mathbf{Z}\mathbf{G}\|_F^2, \\ \text{s.t.} \quad & \mathbf{G}^T\mathbf{1}_k = \mathbf{1}_m \text{ and } s_{i+1} - s_i \in [1, n_{\max}], \end{aligned} \tag{7}$$

where $\mathbf{G} \in \{0,1\}^{k \times m}$ is a class indicator matrix and $\mathbf{s} \in \mathbb{R}^{m+1}$ is the vector that contains the start and end of each segment. $\mathbf{Y}_i \doteq \mathbf{X}_{[s_i,s_{i+1})}$ denotes a segment. Similar to KKM, $dist^2_\psi(\mathbf{Y}_i,\mathbf{z}_c)$ is the squared distance between the ith segment and the center of class c in the feature space defined by the nonlinear mapping $\psi(\cdot)$, which is

$$dist^2_\psi(\mathbf{Y}_i,\mathbf{z}_c) = \tau_{ii} - \frac{2}{m_c}\sum_{j=1}^{m} g_{cj}\tau_{ij} + \frac{1}{m_c^2}\sum_{j_1,j_2=1}^{m} g_{cj_1}g_{cj_2}\tau_{j_1j_2},$$

where $m_c = \sum_{j=1}^{m} g_{cj}$ is the number of segments that belong to class c. The dynamic kernel function τ is defined as $\tau_{ij} = \psi(\mathbf{Y}_i)^T\psi(\mathbf{Y}_j) = \mathrm{tr}(\mathbf{W}_{ij}^T\mathbf{K}_{ij})$ based on (6).

The differences between ACA (7) and KKM (2) are worth pointing out:

1. ACA clusters variable length features, that is, each segment $\mathbf{Y}_i$ might have a different number of samples (columns of $\mathbf{Y}_i$), whereas standard KKM has a fixed number of features (rows of $\mathbf{x}_i$).
2. A new variable, $\mathbf{s}$, is introduced to represent the starting and ending of each segment.
3. The distance used in ACA, $dist_\psi(\mathbf{Y}_i,\mathbf{z}_c)$, is based on DTAK, which is robust to noise and invariant to temporal scaling factors.
4. A DP-based approach is used to efficiently solve ACA.

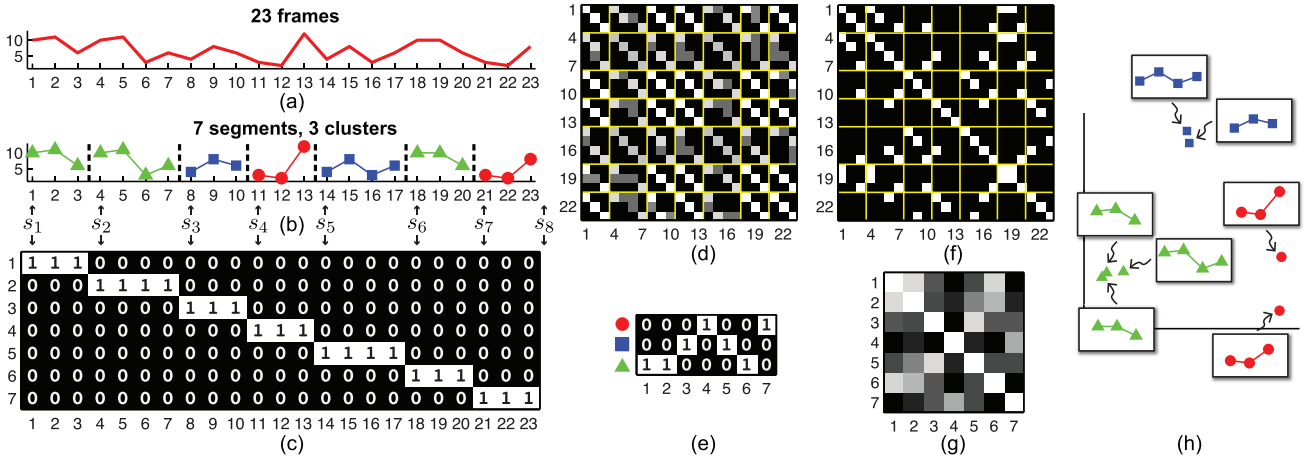


Fig. 4. A synthetic example of temporal clustering. (a) An example of 1D time series. (b) Temporal clustering of the 1D time series. (c) Sample-segment indicator matrix ($\mathbf{H}$). (d) Normalized correspondence matrix ($\mathbf{W}$). (e) Segment-cluster indicator matrix ($\mathbf{G}$). (f) Frame kernel matrix ($\mathbf{K}$). (g) Segment kernel matrix ($\mathbf{T}$). (h) 2D embedding computed using the top two eigenvectors of $\mathbf{T}$.

3.5 Matrix Formulation for ACA

A more enlightening formulation for ACA is the matrix form. Suppose that a sequence $\mathbf{X} \in \mathbb{R}^{d\times n}$ of length n has been segmented into m segments, $\{\mathbf{Y}_i \in \mathbb{R}^{d\times n_i}\}_{i=1}^{m}$ and $\sum_{i=1}^{m} n_i = n$. A key insight to understanding ACA is that we can define two kernel matrices: $\mathbf{T} = [\tau_{ij}]_{m\times m} \in \mathbb{R}^{m\times m}$, the segment kernel matrix (kernel between segments), and $\mathbf{K} = [\kappa_{ij}]_{n\times n} \in \mathbb{R}^{n\times n}$, the frame kernel matrix (kernel between frames). Each element of the segment kernel matrix ($\mathbf{T}$), $\tau_{ij} = \tau(\mathbf{Y}_i, \mathbf{Y}_j) = \mathrm{tr}(\mathbf{K}_{ij}^T \mathbf{W}_{ij})$, is the DTAK between the ith and jth segments ($\mathbf{Y}_i$ and $\mathbf{Y}_j$) computed using (6), where $\mathbf{K}_{ij} \in \mathbb{R}^{n_i\times n_j}$ and $\mathbf{W}_{ij} \in \mathbb{R}^{n_i\times n_j}$ are the frame kernel matrix and the normalized correspondence matrix between segments $\mathbf{Y}_i$ and $\mathbf{Y}_j$, respectively.

After some linear algebra, it can be shown that $\mathbf{T} \in \mathbb{R}^{m\times m}$ can be expressed as the product of a global correspondence matrix ($\mathbf{W}$), a global kernel frame matrix ($\mathbf{K}$), and a sample-segment indicator matrix ($\mathbf{H}$) as follows:

$$\mathbf{T} = \big[\mathrm{tr}\big(\mathbf{K}_{ij}^T \mathbf{W}_{ij}\big)\big]_{m\times m} = \mathbf{H}\big(\mathbf{K} \circ \mathbf{W}\big)\mathbf{H}^T, \tag{8}$$

where $\mathbf{W} = [\mathbf{W}_{ij}]_{m\times m} \in \mathbb{R}^{n\times n}$ and $\mathbf{K} = [\mathbf{K}_{ij}]_{m\times m} \in \mathbb{R}^{n\times n}$ are obtained by rearranging the $m \times m$ blocks of $\mathbf{W}_{ij}$ and $\mathbf{K}_{ij}$,respectively. $\mathbf{H} \in \{0,1\}^{m\times n}$ is a matrix that encodes the correspondence between samples and segments such that $h_{ij} = 1$ if the jth sample belongs to the ith segment. See Fig. 4 for an example of these matrices.

Unfortunately, DTAK is not a strictly positive definite kernel [47], [48]. Thus, we add a scaled identity matrix to $\mathbf{K}$, that is, $\mathbf{K} \leftarrow \mathbf{K} + \sigma\mathbf{I}_n$, where σ is chosen to be the absolute value of the smallest eigenvalue of $\mathbf{T}$ if it has negative eigenvalues.[2]

After substituting the optimal value of

$$\mathbf{Z} = [\psi\big(\mathbf{Y}_1\big), \ldots, \psi\big(\mathbf{Y}_m\big)]\mathbf{G}^T\big(\mathbf{G}\mathbf{G}^T\big)^{-1}$$

in (7), a more understandable form of J_{aca} results in:

$$\begin{aligned} J_{aca}(\mathbf{G},\mathbf{H}) &= \mathrm{tr}((\mathbf{I}_m - \mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{G})\mathbf{T}) \\ &= \mathrm{tr}((\mathbf{I}_m - \mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{G})\mathbf{H}(\mathbf{K}\circ\mathbf{W})\mathbf{H}^T) \\ &= \mathrm{tr}((\mathbf{L}\circ\mathbf{W})\mathbf{K}), \\ \text{where}\quad \mathbf{L} &= \mathbf{I}_n - \mathbf{H}^T\mathbf{G}^T\big(\mathbf{G}\mathbf{G}^T\big)^{-1}\mathbf{G}\mathbf{H}. \end{aligned} \tag{10}$$

Recall $\mathbf{H}$ depends on $\mathbf{s}$ and $\mathbf{G} \in \{0,1\}^{k\times m}$ is the segment-cluster indicator matrix such that $g_{ij} = 1$ if the jth segment belongs to the ith temporal cluster. See Fig. 4 for an example of temporal clustering and the role of the matrices $\mathbf{K}, \mathbf{W}, \mathbf{H}$.

Consider the special case when each segment is one frame, i.e., $m = n$ and $\mathbf{H} = \mathbf{I}_n$. The segment kernel matrix becomes simply the frame kernel matrix, i.e., $\tau_{ij} = \kappa_{ij}$ and $\mathbf{W} = \mathbf{1}_{n\times n}$. In this case, the energy function of ACA is equivalent to the function minimized by KKM [41], [50], [51]:

$$J_{kkm}(\mathbf{G}) = \mathrm{tr}(\mathbf{L}\mathbf{K}), \text{ where } \mathbf{L} = \mathbf{I}_n - \mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{G}. \tag{11}$$

KKM finds the binary matrix $\mathbf{G} \in \{0,1\}^{k\times n}$ (i.e., the indicator matrix between samples and clusters) which makes $\mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{G}$ as correlated as possible with the sample kernel matrix $\mathbf{K}$. On the other hand, ACA has two indicator matrices: $\mathbf{G}$, the segment-cluster indicator matrix, that solves for the correspondence between segments and clusters, and $\mathbf{H}$, the sample-segment indicator matrix that encodes the correspondence between samples and segments. ACA finds the two binary matrices $\mathbf{G}$ and $\mathbf{H}$ that, after applying DTAK between all pairwise segments, make the matrix $\mathbf{H}^T\mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{G}\mathbf{H} \circ \mathbf{W}$ as correlated as possible with the frame kernel $\mathbf{K}$. Fig. 4 illustrates the role of different matrices in a synthetic temporal clustering example. Notice that once the matrices $\mathbf{K}, \mathbf{W}, \mathbf{H}$ are computed by ACA, the eigenvectors of the matrix $\mathbf{T}$ (8) provide a natural embedding for visualizing the seven segments of the time series in a low-dimensional space (see Fig. 4h).

3.6 Coordinate-Descent Optimization for ACA

In the previous section, we have formulated the problem of temporal clustering as an integer programming problem (7) over two variables ($\mathbf{G}$ and $\mathbf{s}$). Recall that $\mathbf{G}$ encodes the segment-cluster correspondence and $\mathbf{s}$ (or, equivalently, $\mathbf{H}$) encodes the sample-segment correspondence. Optimizing over $\mathbf{G}$ and $\mathbf{s}$ is NP-hard. This section proposes an efficient coordinate-descent scheme that alternates between computing $\mathbf{s}$ using dynamic programming and $\mathbf{G}$ with a winner-take-all strategy [52].

We solve the following subproblem at each iteration:

$$\mathbf{G}, \mathbf{s} = \arg\min_{\mathbf{G},\mathbf{s}} J_{aca}(\mathbf{G},\mathbf{s}) = \arg\min_{\mathbf{G},\mathbf{s}} \sum_{c=1}^{k}\sum_{i=1}^{m} g_{ci} dist_{\psi}^2\big(\mathbf{Y}_i, \dot{\mathbf{z}}_c\big),$$

where $\dot{\mathbf{z}}_c$ is the cluster mean implicitly computed from the segmentation $(\dot{\mathbf{G}}, \dot{\mathbf{s}})$ derived in the previous step. Given a sequence $\mathbf{X}$ of length n, however, the number of all possible segmentations is exponential, i.e., $O(2^n)$, which makes a brute-force search for $\mathbf{s}$ infeasible. We used a DP-based algorithm to exhaustively examine all possible segmentations in polynomial time. Observe that the matrix $\mathbf{H}$ (see Fig. 4c) has a monotonic structure and can be optimally optimized using DP.

Recall that we could rewrite (7) as a sum of the following distances:

$$J_{aca}(\mathbf{G},\mathbf{s}) = \sum_{c=1}^{k}\sum_{i=1}^{m} g_{ci} dist_{\psi}^2\big(\mathbf{Y}_i, \mathbf{z}_c\big).$$

To further leverage the relationship between $\mathbf{G}$ and $\mathbf{s}$, we introduce an auxiliary function, $J(\cdot) : [1,n] \rightarrow \mathbb{R}$,

$$J(v) = \min_{\mathbf{G},\mathbf{s}} J_{aca}(\mathbf{G},\mathbf{s})|_{\mathbf{X}_{[1,\,v]}}, \tag{12}$$

to relate the minimum energy directly with the tail position v of the subsequence $\mathbf{X}_{[1,v]} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_v]$. We can further justify that $J(\cdot)$ satisfies the principle of optimality [53], i.e.,

$$J(v) = \min_{1<i\le v} \Big(J(i-1) + \min_{\mathbf{G},\mathbf{s}} J_{aca}(\mathbf{G},\mathbf{s})|_{\mathbf{X}_{[i,\,v]}}\Big), \tag{13}$$

2. It can be proven that adding $\sigma\mathbf{I}_n$ to $\mathbf{K}$ has the same effect as adding $\sigma\mathbf{I}_m$ to $\mathbf{T}$, that is,

$$\mathbf{H}((\mathbf{K} + \sigma\mathbf{I}_n) \circ \mathbf{W})\mathbf{H}^T = \mathbf{T} + \sigma\mathbf{H}\bar{\mathbf{W}}\mathbf{H}^T \equiv \mathbf{T} + \sigma\mathbf{I}_m, \tag{9}$$

where $\bar{\mathbf{W}} = \mathbf{I}_n \circ \mathbf{W} \in \mathbb{R}^{n\times n}$ is a diagonal matrix. Notice that the diagonal of $\mathbf{W}$ is composed of m blocks of $\mathbf{W}_{ii}$ and that $\mathbf{W}_{ii} = \frac{1}{n_i}\mathbf{I}_{n_i}$ according to (5). Therefore, we can conclude $\bar{\mathbf{W}} = \mathrm{diag}(\frac{1}{n_1}\mathbf{1}_{n_1}; \ldots; \frac{1}{n_m}\mathbf{1}_{n_m})$. In addition, because $\mathbf{H} = [\mathbf{h}^1, \ldots, \mathbf{h}^m]^T \in \{0,1\}^{m\times n}$ is binary and its rows are orthogonal, we can conclude $\mathbf{h}^{iT}\mathbf{h}^i = n_i$ and $\mathbf{h}^{iT}\mathbf{h}^j = 0, i \neq j$. Combining these results for $\bar{\mathbf{W}}$ and $\mathbf{H}$, we prove that $\sigma\mathbf{H}\bar{\mathbf{W}}\mathbf{H}^T \equiv \sigma\mathbf{I}_m$.

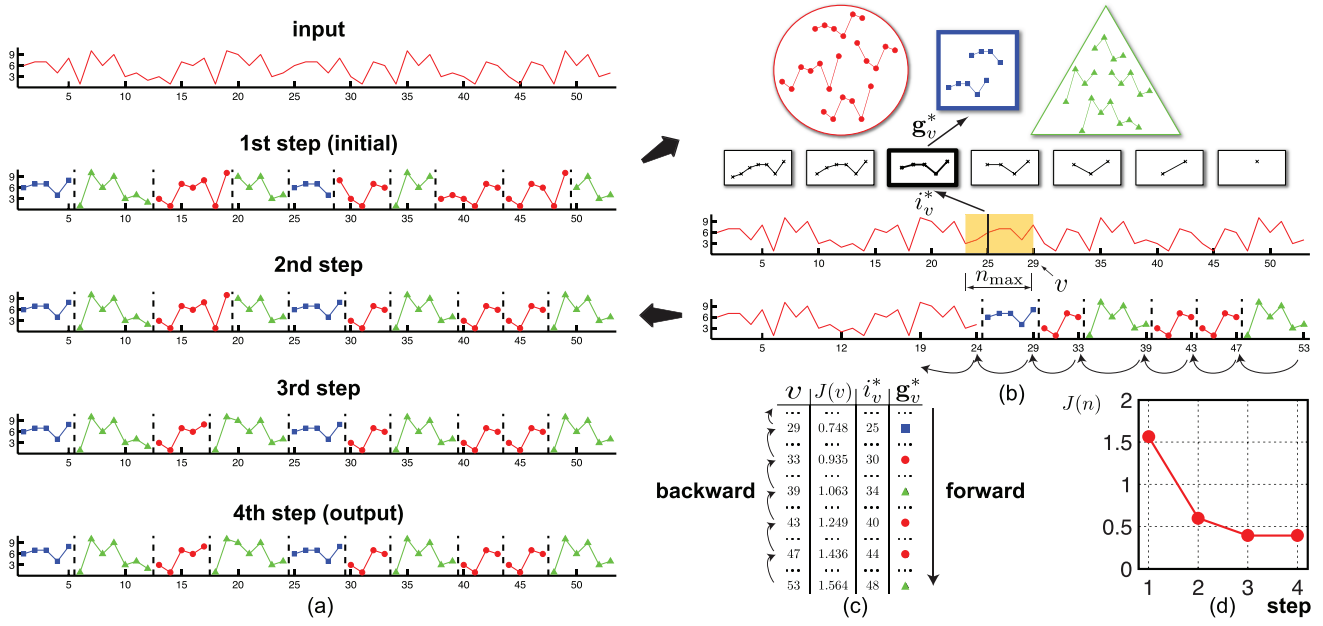


Fig. 5. Coordinate-descent optimization for ACA. (a) Optimization of a 1D sequence that has converged in four steps. (b) Illustration of the DP-based search between first step and second step. (c) Data structure used in DP-based search. (d) ACA error in each step.

which implies that the optimal decomposition of the subsequence $\mathbf{X}_{[1,v]}$ is achieved only when the segmentations on both sides of $\mathbf{X}_{[1,i-1]}$ and $\mathbf{X}_{[i,v]}$ are optimal and their sum is minimal. Although the number of possible ways to decompose the sequence $\mathbf{X}$ is exponential in n, dynamic programming [53] offers an efficient approach to minimize $J(\cdot)$ by using Bellman's equation, which is

$$J(v) = \min_{v-n_{\max}<i\leq v} \left(J(i-1) + \min_{\mathbf{g}} \sum_{c=1}^{k} g_c dist_{\psi}^2(\mathbf{X}_{[i,v]}, \dot{\mathbf{z}}_c) \right), \quad (14)$$

where $dist_{\psi}^2(\mathbf{X}_{[i,v]}, \dot{\mathbf{z}}_c)$ is the squared distance between the segment $\mathbf{X}_{[i,v]}$ and the center of class c:

$$dist_{\psi}^2(\mathbf{X}_{[i,v]}, \dot{\mathbf{z}}_c) = \tau(\mathbf{X}_{[i,v]}, \mathbf{X}_{[i,v]}) - \frac{2}{\dot{m}_c} \sum_{j=1}^{\dot{m}} \dot{g}_{cj} \tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}}_j) + \frac{1}{\dot{m}_c^2} \sum_{j_1,j_2=1}^{\dot{m}} \dot{g}_{cj_1} \dot{g}_{cj_2} \tau(\dot{\mathbf{Y}}_{j_1}, \dot{\mathbf{Y}}_{j_2}).$$

When $v = n$, $J(n)$ is the optimal cost of the segmentation that we seek. The inner values, i_v^*, $\mathbf{g}_v^* = \arg\min_{i,\mathbf{g}} J(v)$, are the head position and label for the last segment, respectively, which lead to the minimum. Equation (14) unifies KKM and segment-based ACA clustering based on the length constraint $n_{\max}$. If $n_{\max} = 1$, each segment consists of a single frame and (14) is equivalent to KKM.

Fig. 5 illustrates the procedure for optimizing ACA. Given an n-length sequence $\mathbf{X}$ with an initial segmentation (Fig. 5a), ACA applies the following forward-backward algorithm to cluster the sequence (Figs. 5b and 5c):

- **Forward step**. Scan from the beginning ($v = 1$) of the sequence to its end ($v = n$). For each v, $J(v)$ is computed according to (14). That is, for every position $i - n_{max} < i < v - 1$, we compute the DTAK between the segment $\mathbf{X}_{[i,v]}$ and each of the segments for each of the classes, $\mathbf{Y}_j$, precomputed in the last iteration. Fig. 4b illustrates this process. Recall that we cannot explicitly compute the mean of each class. We store the head position i_v^*, label $\mathbf{g}_v^*$, and $J(v)$ that has the lowest error in the table, see Fig. 4c.
- **Backward step**. Trace back from the end of sequence ($v = n$) and cut off the segment whose head $s = i_v^*$. The indication vector $\mathbf{g} = \mathbf{g}_v^*$ can be indexed from the stored positions (see Fig. 5c for an example). Repeat this operation on the left part of the sequence ($v = i_v^* - 1$).

These steps are repeated until $J(n)$ converges (Fig. 5d).

3.7 Efficient Dynamic Programming for Kernel Calculation

A straightforward implementation of (14) is prohibitively expensive, i.e., $O(n^2 n_{\max}^2)$, due to the bottleneck of computing $\tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}}_j)$ for all i, v, j. This section describes an efficient dynamic programming solution to solve J which has a time complexity of $O(n^2 n_{\max})$.

The computation of $\tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}}_j)$ involves the construction of the cumulative kernel matrix $\mathbf{U} \in \mathbb{R}^{n_v \times \dot{n}_j}$, where $n_v = v - i + 1$ and $\dot{n}_j = \dot{s}_{j+1} - \dot{s}_j$ are segment lengths of $\mathbf{X}_{[i,v]}$ and $\dot{\mathbf{Y}}_j$, respectively. Observe that we do not need to recompute the whole matrix $\mathbf{U}$ because some columns of $\mathbf{U}$ have been previously calculated in the forward scan of $J(v)$. The computation of $\tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}}_j)$ is illustrated in Fig. 6. $\tau(\mathbf{X}_{[i,v-1]}, \dot{\mathbf{Y}}_j)$ has been previously calculated (the top-left matrix in Fig. 6), and to compute $\tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}}_j)$ DP has to compute the matrix $\mathbf{U}$ by adding a new column (the top-right matrix in Fig. 6). Using this simple observation, we can reduce the computational complexity to $O(n^2 n_{\max})$.

To make the implementation efficiently, we maintain an active cumulative kernel matrix $\mathbf{U} \in \mathbb{R}^{n_{\max} \times n_{\max} \times n}$ that is used in the kernel calculation:

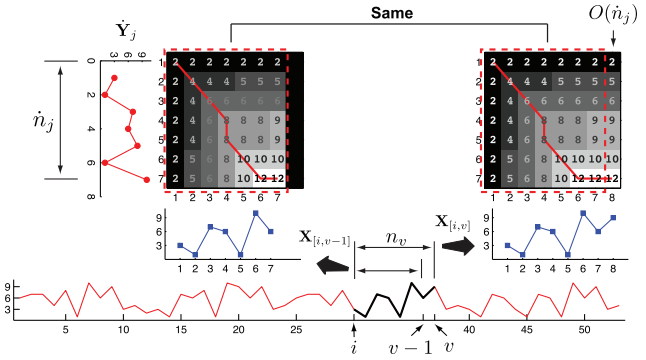


Fig. 6. Efficient DP-based calculation of the dynamic kernel, $\tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}}_j)$.

$$\tau\big(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}}_j\big) = \frac{\mathbf{U}\big(n_v, \overline{v}, \dot{s}_{j+1} - 1\big)}{n_v + \dot{n}_j},$$

$$\text{where } \mathbf{U}\big(n_v, \overline{v}, \dot{s}\big) = \max \begin{cases} \mathbf{U}\big(n_v, \overline{v}, \dot{s} - 1\big) + \kappa_{v\dot{s}} \\ \mathbf{U}\big(n_v - 1, \overline{v-1}, \dot{s} - 1\big) + 2\kappa_{v\dot{s}} \\ \mathbf{U}\big(n_v - 1, \overline{v-1}, \dot{s}\big) + \kappa_{v\dot{s}}, \end{cases} \tag{15}$$

where $\dot{s} \in [\dot{s}_j, \dot{s}_{j+1})$ and $\mathbf{U}$ is initialized at $\mathbf{U}(1, \overline{v}, \dot{s}_j) = 2\kappa_{v\dot{s}_j}$ for each $v \in [1, n]$ and $j \in [1, \dot{m}]$. We used $\overline{v} = v \pmod{n_{\max}}$ instead of v to further reduce the cost in space from $O(n^2 n_{\max})$ to $O(n n_{\max}^2)$. Without any loss in accuracy, $\tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}}_j)$ can be calculated by filling with an $\dot{n}_j$-by-1 vector rather than the original $\dot{n}_j$-by-n_v matrix.

The algorithm to optimize ACA w.r.t. $\mathbf{G}$ and $\mathbf{s}$ is summarized in DPSearch (see Algorithm 1). This algorithm only requires two parameters: length constraint $n_{\max}$ and the number of clusters k.

Algorithm 1: DPSearch

parameter: $n_{\max}$, k
input : $\dot{\mathbf{G}} \in \{0,1\}^{k \times \dot{m}}$, $\dot{\mathbf{s}} \in \mathbb{R}^{(\dot{m}+1)}$, $\mathbf{K} \in \mathbb{R}^{n \times n}$
output : $\mathbf{G} \in \{0,1\}^{k \times m}$, $\mathbf{s} \in \mathbb{R}^{(m+1)}$

1 **begin**
2 **for** $v = 1$ *to* n **do** *forward construction*
3 $J(v) \leftarrow \infty$;
4 **for** $n_v = 1$ *to* $n_{\max}$ **do**
5 *Current segment* $\mathbf{X}_{[i,v]}$ *headed by* $i = v - n_v + 1$;
6 **for** $j = 1$ *to* $\dot{m}$ **do**
7 **for** $\dot{s} = \dot{s}_j$ *to* $\dot{s}_{j+1} - 1$ **do**
8 *Compute* $\tau(\mathbf{X}_{[i,v]}, \dot{\mathbf{Y}}_j)$ *by updating* $\mathbf{U}(n_v, \overline{v}, \dot{s})$ *with* $\mathbf{K}$;
9 $c^* \leftarrow \arg\min_c dist_\psi(\mathbf{X}_{[i,v]}, \dot{\mathbf{z}}_c)$;
10 $J \leftarrow J(i-1) + dist_\psi(\mathbf{X}_{[i,v]}, \dot{\mathbf{z}}_{c^*})$;
11 **if** $J < J(v)$ **then** *update the record*
12 $J(v) \leftarrow J, \mathbf{g}_v^* \leftarrow \mathbf{e}_{c^*}, i_v^* \leftarrow i$;
13 **while** $v > 0$ **do** *backward segmentation*
14 *Create a segment* $\mathbf{Y} = \mathbf{X}_{[i_v^*, v]}$ *with the label* $\mathbf{g}_v^*$;
15 $v \leftarrow i_v^* - 1$;

The computational cost in time is mainly determined by the deepest nest of iterations, which in DPSearch corresponds to the update of $\mathbf{U}$ (line 8). Because the sum of all the segments' length equals the total number of frames, i.e., $\sum_{j=1}^{\dot{m}} \dot{s}_{j+1} - \dot{s}_j \equiv n$, we can estimate the time cost of DPSearch as $O(n^2 n_{\max})$. The overall time complexity is $O(n^2 n_{\max} t)$, where t is the number of iterations.

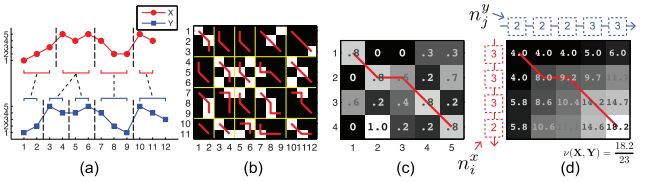


Fig. 7. Generalized time alignment kernel. (a) Segmented sequences where black dotted lines indicate the GDTAK alignment. (b) Frame kernel matrix ($\mathbf{K}$). (c) Segment kernel matrix ($\mathbf{T}$). (d) Cumulative kernel matrix ($\mathbf{U}$).

3.8 Hierarchical Aligned Cluster Analysis

This section describes hierarchical ACA, a hierarchical extension of ACA. HACA reduces the computational complexity of ACA and provides a hierarchical decomposition at different temporal scales.

As discussed in Section 3.7, ACA's computational cost is linear in the length constraint $n_{\max}$ and quadratic in the length of the sequence n. Unlike ACA, HACA starts the search with small temporal scales and propagates the result to larger temporal scales. The computational complexity is $O(n^2 n_{\max})$ and HACA is more efficient because it starts using smaller temporal scales (i.e., $n_{\max}$). For instance, if the length constraint for ACA is $n_{\max}$, the equal setting for a two-level HACA involves two pairs, $n_{\max}^{(1)}$ and $n_{\max}^{(2)}$, where $n_{\max}^{(1)}$ and $n_{\max}^{(2)}$ denote the constraints used in the first and second level, respectively. In the following, we show how to compute HACA using the same algorithm as ACA, but replacing the kernel DTAK with the generalized DTAK (GDTAK).

We extend the definition of DTAK to propagate the solution at different levels (i.e., temporal scales). Previous work by Keogh and Pazzani [54] used this strategy to speed up the computation of DTW. Unlike [54], we used the values from the lower level to compute the similarity in a bottom-up way.

Given two sequences $\mathbf{X} \in \mathbb{R}^{d \times n_x}, \mathbf{Y} \in \mathbb{R}^{d \times n_y}$ and their segmentations $\mathbf{s}_x \in \mathbb{R}^{m_x+1}, \mathbf{s}_y \in \mathbb{R}^{m_y+1}$, where n_x, n_y and m_x, m_y represent the number of frames and segments, respectively. Our generalized dynamic time alignment kernel is defined (at the second level) as

$$\nu(\mathbf{X}, \mathbf{Y}) = \frac{u_{m_x m_y}}{n_x + n_y}, \; u_{ij} = \max \begin{cases} u_{i-1,j} + n_i^x \tau_{ij} \\ u_{i-1,j-1} + \big(n_i^x + n_j^y\big)\tau_{ij} \\ u_{i,j-1} + n_j^y \tau_{ij}, \end{cases}$$

where n_i^x is the length of segment $\mathbf{X}_i = \mathbf{X}_{[s_i^x, s_{i+1}^x)}$ (similarly for n_j^y) and τ_{ij} is the DTAK defined (at the first level) for segment $\mathbf{X}_i$ and $\mathbf{Y}_j$. Similarly to DTAK, the recursive computation is initialized as $u_{11} = (n_1^x + n_1^y)\tau_{11}$. Fig. 7 shows two sequences, $\mathbf{X}$ and $\mathbf{Y}$, aligned using GDTAK.

Fig. 8 shows the process of using HACA to learn motion primitives from motion capture data. In the first level, the input motion is represented as a sequence of frames. After extracting the features and calculating the frame kernel matrix (see Section 4), ACA is applied to obtain and refine the temporal segmentation result. In the first iteration, we usually set a small length constraint ($n_{\max}^{(1)}$) to extract short primitives, such as bending a leg. To propagate the result to the next level, we recalculate the frame kernel matrix in the

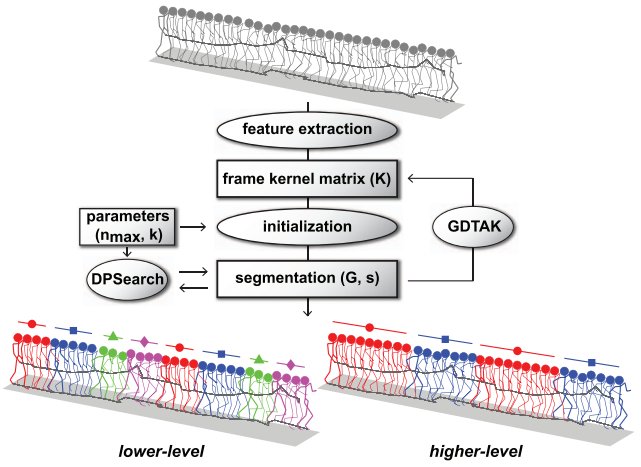


Fig. 8. The main processing steps to learn motion primitives with HACA.

second iteration by performing GDTAK on the segmented frames. After that, ACA is applied to obtain a new segmentation with longer segments.

4 Experiments

This section evaluates the performance of ACA and HACA on synthetic time series, motion capture data, and video of humans performing multiple actions. Additionally, we run ACA on the honey bee dance data, and compare it with state-of-the-art algorithms for temporal segmentation. The HACA code is available online at http://humansensing.cs.cmu.edu/aca.

4.1 Preprocessing and Evaluation Metric

In all experiments, we compared ACA and HACA with SC, which has recently become one of the most popular clustering algorithms [55]. In our implementation, we used the version of Ng et al. [56]. To temporally smooth the time series, following [15] we removed short segments as follows: Suppose that we are given a sequence $\mathbf{X} \in \mathbb{R}^{d \times n}$ and its associated frame kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$. We first run SC on $\mathbf{K}$ to cluster the n frames into k clusters. We then merge consecutive frames that have the same label. Later, we remove the segments that have less frames than $\frac{1}{2} n_{\max}$.

To evaluate the clustering accuracy, we computed the confusion matrix between the segmentation $(\mathbf{G}_{alg}, \mathbf{H}_{alg})$ provided by the algorithm (e.g., ACA, HACA, SC) and the ground truth $(\mathbf{G}_{tru}, \mathbf{H}_{tru})$, that is,

$$\mathbf{C} = \mathbf{G}_{alg} \mathbf{H}_{alg} \mathbf{H}_{tru}^T \mathbf{G}_{tru}^T \in \mathbb{R}^{k \times k}, \tag{16}$$

where $\mathbf{G}_{alg}$ and $\mathbf{H}_{alg}$ are, respectively, the segment-cluster and sample-segment indicator matrices returned by ACA, HACA, or SC. Each entry c_{c_1,c_2} in the confusion matrix represents the total number of frames that belong to the cluster segment c_1 that are shared by the cluster segment c_2 in the ground truth. Once the confusion matrix is computed, we apply the Hungarian algorithm [57] to find the optimum cluster correspondence, and compute the accuracy as follows:

$$\text{acc} = \max_{\mathbf{P}} \frac{\text{tr}(\mathbf{CP})}{\text{tr}(\mathbf{C1}_{k \times k})}, \tag{17}$$

subject to the constraint that $\mathbf{P} \in \{0,1\}^{k \times k}$ is a permutation matrix.

4.2 Synthetic Data

We generated two experiments on synthetic time series: a synthetic time series without hierarchical structure (top row of Fig. 9, single level) and one with hierarchical structure (bottom row of Fig. 9, multiple levels). We run all our synthetic experiments with different levels of temporal random noise. A noise level of 0.1 indicates that, on average, a noisy frame is inserted every 10 frames. The frame kernel matrix is computed as $\kappa_{ij} = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$. σ is set to be the average distance from the 20 percent closest neighbors.

In the first case (top row of Fig. 9), each time series, $\mathbf{X} \in \mathbb{R}^{2 \times n}$, is generated by randomly sampling 10 2D Gaussian distributions that have been grouped into three temporal clusters. An example of synthetic time series of length 96 without hierarchical structure is illustrated in Fig. 9a. Fig. 9b shows the 10 Gaussians and the underlying

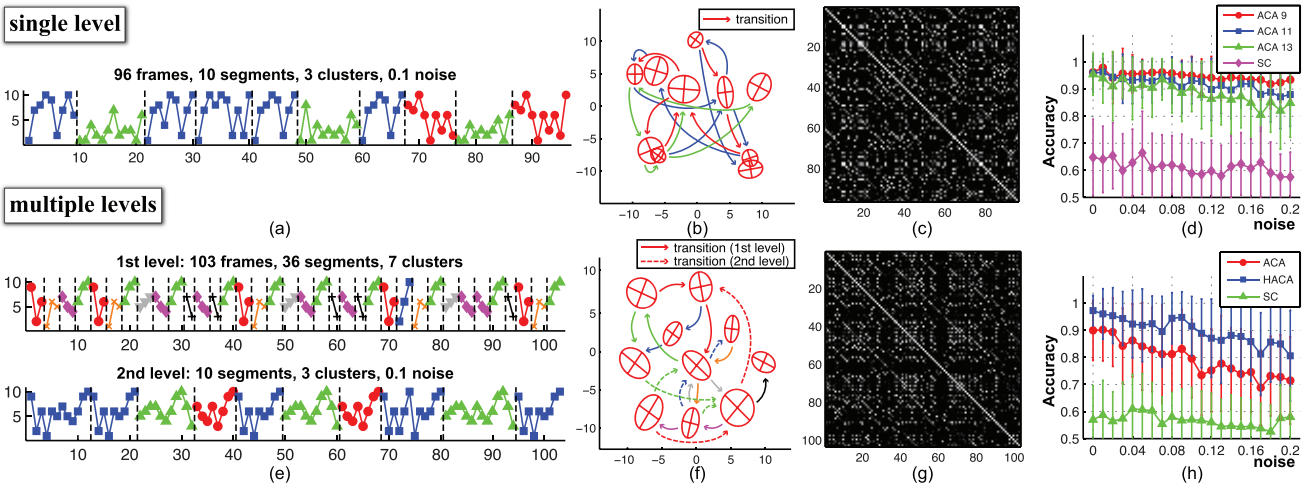


Fig. 9. Temporal clustering computed by ACA and HACA on synthetic sequences. (a) An example of 1D time series. (b) Ten Gaussians clusters used to generate the 1D time series. (c) Frame kernel matrix ($\mathbf{K}$). (d) Mean accuracy and one standard deviation as a function of noise. The length constraint ($n_{\max}$) is plotted in the legend. (e) An example of 1D time series with two-level hierarchical structure. (f) Ten Gaussians used to generate the time series. (g) Frame kernel matrix ($\mathbf{K}$). (h) Mean accuracy and one standard deviation.

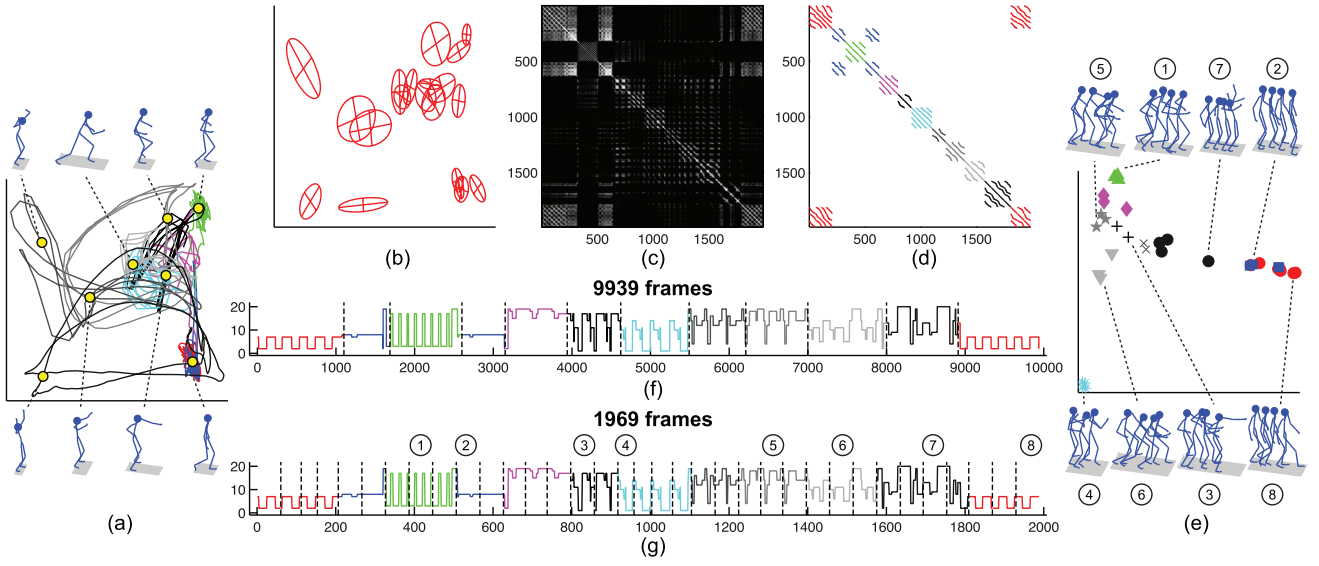


Fig. 10. Temporal clustering computed by ACA on motion capture data (subject 86, trial 2). This sequence contains 9,939 frames of a subject performing eight actions. (a) Embedding frames by PCA (different colors indicate clusters of actions). (b) Clustering frames into 20 clusters. (c) Frame kernel matrix ($\mathbf{K}$). (d) The matrix $(\mathbf{H}^T\mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{G}\mathbf{H}) \circ \mathbf{W}$ computed by ACA. (e) Embedding segments by ACA. (f) Original sequence with ground-truth segmentation. (g) Reduced sequence ($n_{reduce} = 5$) with segmentation obtained by ACA.

transition between Gaussians. Each color for the arrows in Fig. 9b indicates one temporal cluster. Fig. 9c shows its frame kernel matrix $\mathbf{K}$. We tested three different length constraints on segments, $n_{\max} = 9, 11, 13$, for ACA. ACA is initialized using a random segmentation.

In the second case (bottom row of Fig. 9), the sequences have a two-level hierarchical structure. As in the first case, each time series is generated by randomly sampling 10 2D Gaussian distributions that have been grouped into seven temporal clusters. The main difference from the first case is that the seven clusters can be grouped into three clusters. For instance, Fig. 9e illustrates a synthetic time series composed by 103 frames sampled from 10 spatial Gaussian clusters (Fig. 9f). Notice that the generated time series has a two-level structure. In the first level (Fig. 9e top), the 103 frames can be grouped into seven temporal clusters. And in the second level (Fig. 9e bottom), the seven temporal clusters can be further grouped into three larger temporal clusters. In this case, we set the length constraint $n_{\max} = 9$ for ACA and $n^{(1)}_{\max} = n^{(2)}_{\max} = 3$ for the two HACA levels, respectively. ACA and HACA are initialized using a random segmentation.

Fig. 9d shows the accuracy of ACA and HACA on the two synthetic datasets. For each case, we generated 100 time series to evaluate the performance. We run SC, ACA, and HACA 10 times with random initializations, and choose the solution with minimum energy (7). In the first dataset (the first row of Fig. 9), we investigated the performance of ACA under different length constraints (i.e., $n_{\max} = 9, 11, 13$) for the segments. The second row of Fig. 9 shows that HACA outperforms ACA to discover a sequence that has a hierarchical structure. Moreover, HACA is less computationally expensive than ACA. The SC algorithm designed for conventional clustering problems performed much worse than ACA and HACA.

4.3 Motion Capture Data

We used the Carnegie Mellon University Motion Capture database [58] to discover motion primitives. The human motion data were captured with a Vicon optical motion capture system of 12 MX-40 cameras at 120 Hz. The subjects wore 41 markers, and the motion data include absolute root position and orientation and the relative joint angles of 29 joints. In order to provide a continuous representation, the 3D Euler angles were transformed to 4D quaternions.

There are many possible distances between body configurations that can provide a good similarity to match human motion (see [59], [60] for an extensive discussion). Similarly to the method of Barbic et al. [15], we only considered the 14 most informative joints and computed the distance between frames as

$$dist^2_{ij} = \sum_{k=1}^{14} \| \log\left(\mathbf{q}^{-1}_{jk}\mathbf{q}_{ik}\right)\|^2, \tag{18}$$

where $\mathbf{q}_{ik} \in \mathbf{S}^3$ is the complex form of the quaternion for the kth joint in the ith frame. $\| \log(\mathbf{q}^{-1}_{jk}\mathbf{q}_{ik})\|$ is the geodesic norm which gives the shortest distance from $\mathbf{q}_{ik}$ to $\mathbf{q}_{jk}$ on $\mathbf{S}^3$.

The frame kernel matrix,

$$\kappa_{ij} = \exp\left(-\frac{dist^2_{ij}}{2\sigma^2}\right),$$

is shown in Fig. 10c. σ is set to be the average distance from the 20 percent closest neighbors.

It is not computationally practical to run HACA on large amounts of motion capture data. Recent work [61], [62] has shown that human motion is locally linear and we can therefore assume human motion is typically smooth. It is thus possible to reduce the number of frames without losing information relevant to temporal clustering. Following previous work on temporal clustering [9], [20], we first apply a clustering step to group frames into k_{reduce} classes. Later, we reduce to one sample each set of n_{reduce} consecutive frames of the same class. Fig. 10a shows frames projected onto two principal components for subject 86. Fig. 10b shows

2D PCA projections of 20 clusters obtained by clustering 9,939 frames using k-means. Observe that there are some overlapping Gaussians in the 2D PCA space; however, these Gaussians do not necessarily overlap in the high-dimensional space. After clustering, every five consecutive frames that belong to the same class are reduced to one frame. The original sequence had 9,939 frames (Fig. 10f) and was compressed to 1,988 frames (Fig. 10g).

We selected 14 sequences performed by subject 86, each of which is a combination of roughly 10 natural actions (e.g., walking, punching, drinking, running). Typically, each sequence contains almost 8,000 frames (70 secs). Due to the quadratic complexity of our algorithm with respect to the number of frames, we reduce the length of each sequence by a factor of five in order to make the experiments scalable. Please refer to [63, Section 4] for the details of temporal reduction. Quaternions are used as features to group the frames into 20 clusters. The length for each activity ranges from 160 and 300 frames. Therefore, we set the length constraint on segments to $n_{\max} = 60$ for ACA and $n_{\max}^{(1)} = 10$ and $n_{\max}^{(2)} = 6$ for the two HACA levels, respectively. ACA and HACA are initialized with the algorithm described in [63].

Fig. 11a shows the results of temporal clustering for ACA, HACA, SC, and manual labeling (ground truth), respectively, on the 14 sequences. Different actions are marked with different colors. Given a segmentation, we calculated its accuracy with the expression given in (17). We run ACA and HACA five times and selected the solution with the minimum objective function (7). The clustering accuracy is shown in Fig. 11b. ACA and HACA identify the distinct actions by providing a segmentation closer to the one provided by the human observer. The motions that are almost cyclic were more clearly detected. Fig. 11c shows the time (in seconds) spent by ACA and HACA.

Fig. 10e shows the 2D embedding provided by ACA. There are 35 points that correspond to 35 segments grouped into eight actions. The embedding is computed with the two eigenvectors of the kernel segment matrix ($\mathbf{T}$) defined in (8). Common dimensionality reduction methods (e.g., PCA, LDA, Isomap, LLE) find embeddings from a data sample in the high-dimensional space to a point in the embedded space. Unlike these methods, ACA finds an embedding for time series such that each point in the embedding corresponds to a segment of different length in the sequence. Moreover, the embedding found by ACA incorporates information about the dynamics that are key to defining actions. In the PCA embedding (Fig. 10a), the eight actions are not clearly separated, but in the ACA embedding (Fig. 10e) they are.

Additionally, we can derive a motion tree (Fig. 12) consisting of segments based on the results given by HACA. For instance, both walking and standing have an initial gait with both feet on the ground (movement A in Fig. 12b). What makes them different is that the double-support gait within the walking motion should be followed by the leg movement. In another example, this double-support gait also appears as one phase in the duration of raising arms (movement A in Fig. 12a).

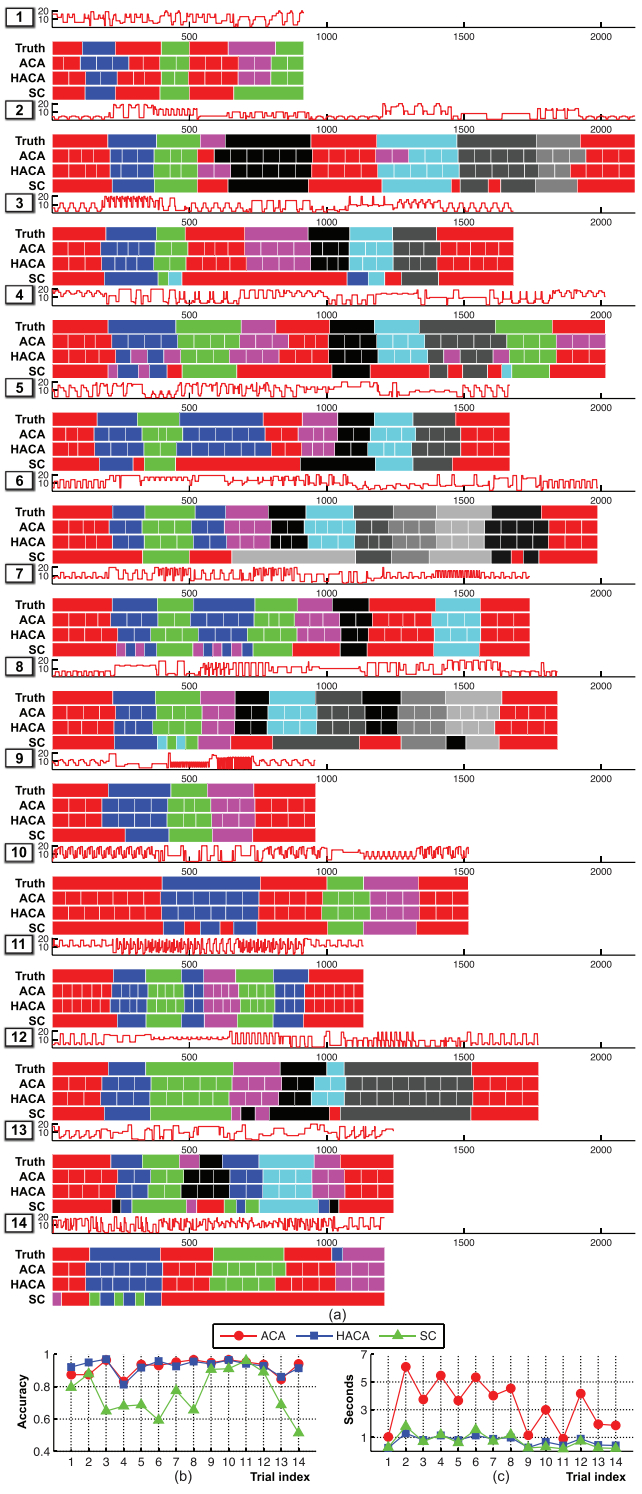


Fig. 11. Comparison of different temporal clustering methods on motion capture data. (a) The 14 motions are performed by subject 86 from the CMU Motion Capture Database. In the first row of each block, the frames are expressed as one of 20 labels given by k-means. The next four rows illustrate the human label and results given by ACA, HACA, and SC, respectively. White lines indicate the boundaries of actions, while the different colors correspond to distinct actions. (b) Accuracy. (c) Time.

4.4 Video Data

The third experiment shows how ACA and HACA can robustly segment a video sequence of several subjects performing different actions. We used two standard video databases, Weizmann [64] and KTH [65].

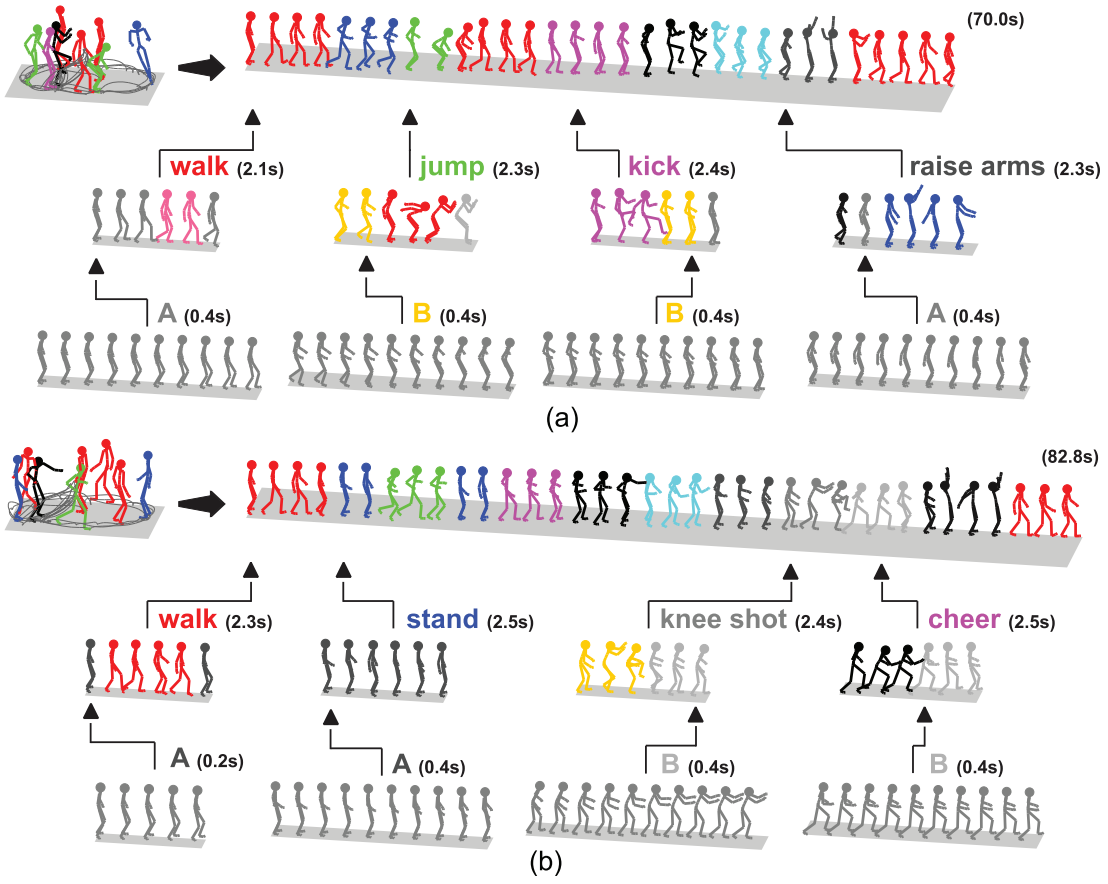


Fig. 12. Hierarchical decomposition of several motion capture sequences. The original sequence is plotted in the upper left corner, and it has been retargeted to a line for easier visualization. Several actions, such as walking, jumping, and kicking (second row), share common components at the lower temporal granularity (third row). (a) Subject 86, trial 3. (b) Subject 86, trial 6.

The Weizmann dataset contains 90 videos of 10 individuals performing nine different actions (walking, siding, skipping, running, performing jumping jacks, jumping, jumping in place, one-hand waving, and two-hand waving). The KTH dataset contains six types of human actions (walking, jogging, running, boxing, hand-waving, and hand-clapping) performed by 25 subjects in different scenarios. In the experiment, we used the subset s_1 that contains 150 videos recorded outdoors.

We first describe two algorithms to extract dynamic features from video. In the case of the Weizmann dataset, we compute the silhouette with background subtraction because the background is provided. We represent the person as a binary mask (Fig. 13b) which does not include color or texture. The silhouette mask is scale-normalized, preserving the aspect ratio. Additionally, we compute a second mask that only contains the legs and hands to remove the torso pixels from the similarity measure. In order to define a meaningful similarity between two images of different sizes, we used the measure proposed by Cutler and Davis [42], in which the bounding box is shifted in an area to search for the maximum overlap, that is,

$$dist_{ij} = \min_{|dx,dy|<r} \sum_{(x,y)\in B_i} |O_i(x+dx, y+dy) - O_j(x,y)|, \quad (19)$$

where B_i is the bounding box for the ith frame, and $O_i(x,y)$ is the binary value of the mask at position (x,y). $dist_{ij}$ represents the minimum number of pixels for frame i that are not matched by frame j (Fig. 13c). The final similarity (Fig. 13 d) is defined as $\kappa_{ij} = \exp(-\frac{dist_{ij}^2}{2\sigma^2})$.

In the case of the KTH database [65], it is not easy to get a good estimate of the background or the silhouette. In this case, optical flow can provide a more robust representation for human motion. Similar to the motion descriptors proposed

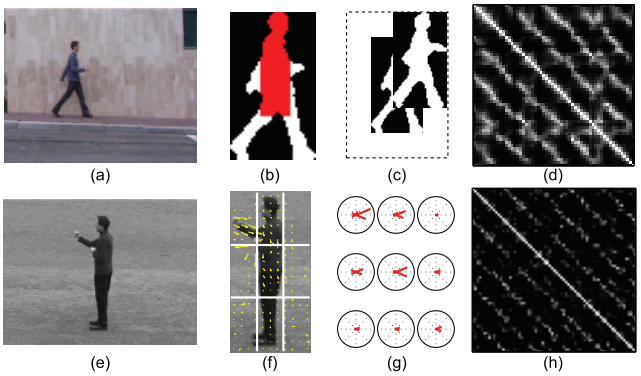
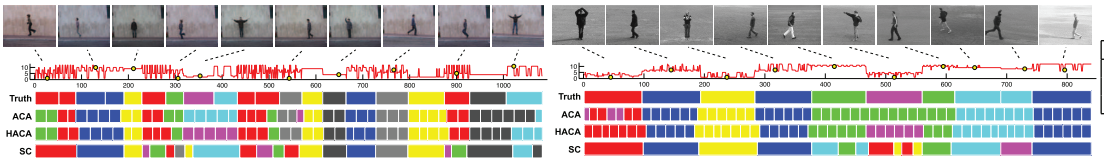


Fig. 13. Video features. Top row: Silhouette-based feature. (a) An image of a person walking from the Weizmann database. (b) Binary silhouette and the estimated torso mask (red). (c) Search for the best match between two frames. (d) Frame kernel matrix ($\mathbf{K}$). Bottom row: Flow-based feature. (e) An image of a person boxing from the KTH database. (f) Optical flow divided in 3×3 blocks. (g) Spatial and directional histogram. (h) Frame kernel matrix ($\mathbf{K}$).



	Weizmann	KTH
ACA	0.76	0.80
HACA	**0.77**	**0.83**
SC	0.63	0.72

(c)

Fig. 14. Comparison of different temporal clustering methods on video data. (a) Example of the Weizmann dataset. (b) Example of the KTH dataset. (c) Accuracy.

by Efros et al. [66], we used a normalized-correlation-based tracker to estimate the subject's location in each image. After scaling the human-centric figure (Fig. 13f) to the same size, the velocity at each point (x, y) is calculated with the standard Lucas-Kanade algorithm [67]. In order to deal with the noise and uncertainty in pose, we represent the optical flow with a spatial ($n_{row} \times n_{col}$) and directional (n_{bin}) binning. Each bin $h^b_{i,j}$ is computed as

$$h^b_{i,j} = \sum_{x,y,\theta} w_i(x) w_j(y) w_b(\theta) mag(x, y), \tag{20}$$

where $w_i(x)$, $w_j(y)$, and $w_b(\theta)$ are the weights for the point's position and orientation with respect to the bin. $mag(x, y)$ is the magnitude of optical flow at position (x, y). The resulting flow feature is a concatenated histogram of all the bins (Fig. 13g). We used the χ^2 distance (Fig. 13 h), and the final similarity is defined as $\kappa_{ij} = \exp(-\frac{\chi^2}{2\sigma^2})$. σ is set to be the average distance from the 20 percent closest neighbors.

The set of testing videos is synthesized[3] by concatenating clips that are randomly selected from these two databases. We generated 10 testing videos for the Weizmann dataset and 10 videos for the KTH dataset. Each of the videos contains 10-20 clips of different actions. The kernel matrix is constructed by using the silhouette-based (for Weizmann) and flow-based features (for KTH), respectively. After that, we initialized ACA and HACA as described in Section [63], and ran ACA, HACA, and SC 10 times, one for each concatenated video. We set the length constraint of the actions to $n_{\max} = 16$ for ACA, while setting $n^{(1)}_{\max} = 4$ and $n^{(2)}_{\max} = 4$ for the first and second level of HACA, respectively. Fig. 14c shows the average accuracy of each method.

As can be observed, ACA and HACA outperform SC in the task of clustering actions in video. SC algorithms make use of frame similarity, which allows distinguishing motions that are different. However, motions like walking and running are only similar in certain phases. In contrast, both ACA and HACA can discriminate walking from running by integrating temporal information. Notice that HACA is usually more efficient for temporal clustering than ACA. Furthermore, HACA returns a hierarchical decomposition of motion where different actions on the top levels (large temporal scale) share the same component in the lower levels (short temporal scale), see Fig. 15. In the past few years, KTH and Weizmann have been extensively used as benchmark datasets for evaluating supervised algorithm for action recognition. Currently, the best supervised algorithm achieves an accuracy of 90 percent for the Weizmann dataset [68] and 94 percent for the KTH dataset [69] using presegmented videos and classifying all the videos. Recently, Hoai et al. [70] achieved 87.7 percent in the task of temporal segmentation of human motion in the KTH dataset. Their approach is also supervised. Interestingly, algorithms such as ACA and HACA are able to achieve competitive detection performances (77 percent) for human actions in a completely unsupervised fashion. Moreover, ACA and HACA allow a temporal decomposition at different temporal scales.

3. To the best of our knowledge, there are no publicly available labeled video databases adequate to test the temporal clustering problem. By randomly concatenating the clips into a long sequence, we can generate ground-truth labels.

4.5 Honey Bee Dance Data

We have applied ACA to segment a honey bee dance from video, which is a classical example of animal behavior and communication. The dataset [12] consists of six video sequences with 1,058, 1,125, 1,054, 757, 609, and 814 frames, respectively. The bee motion in each sequence can be classified into three different regimes: waggle, left turn, and right turn.

The position of the bee (Fig. 16a) is provided by Oh et al. [12]. Fig. 16b shows the time series of the position and angle of the bee over time. $\mathbf{p}_i = [x_i, y_i]^T \in \mathbb{R}^2$ denotes the 2D coordinates and $\mathbf{a}_i = [\sin\theta_i, \cos\theta_i]^T \in \mathbb{R}^2$ the heading angle at the ith frame. To normalize the location of the bee in the box, we computed the relative position, $x_i = x^0_i - x^m_i$ and $y_i = y^0_i - y^m_i$, where x^0_i, y^0_i are the position in the box coordinate frame and x^m_i, y^m_i is the mean position averaged

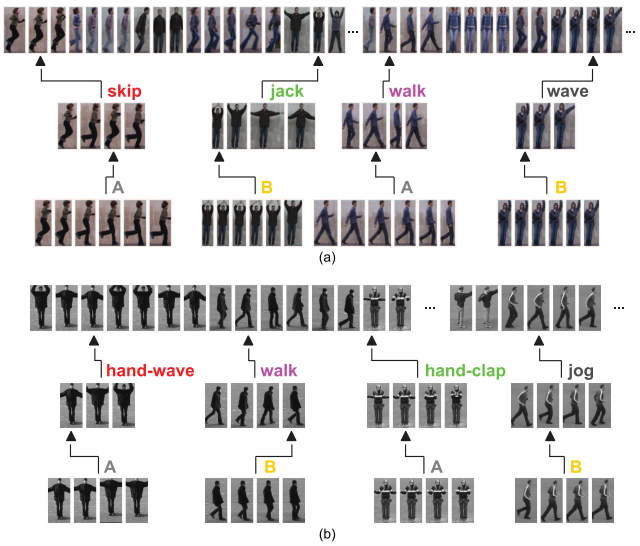


Fig. 15. Hierarchical decomposition of video sequences. This motion tree is derived by HACA. Observe that several actions in the second row (walk, jog) share components from lower levels (smaller temporal scale). For instance, jog and walk share the B component. (a) Example from the Weizmann dataset. (b) Example from the KTH dataset.

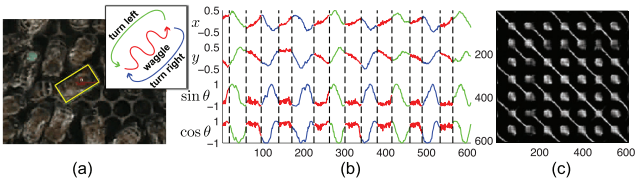


Fig. 16. (a) A frame showing the box to track the bee and the trajectory (green). (b) 4D features including position (x, y) and head orientation ($\sin\theta$, $\cos\theta$). (c) Frame kernel matrix ($\mathbf{K}$).

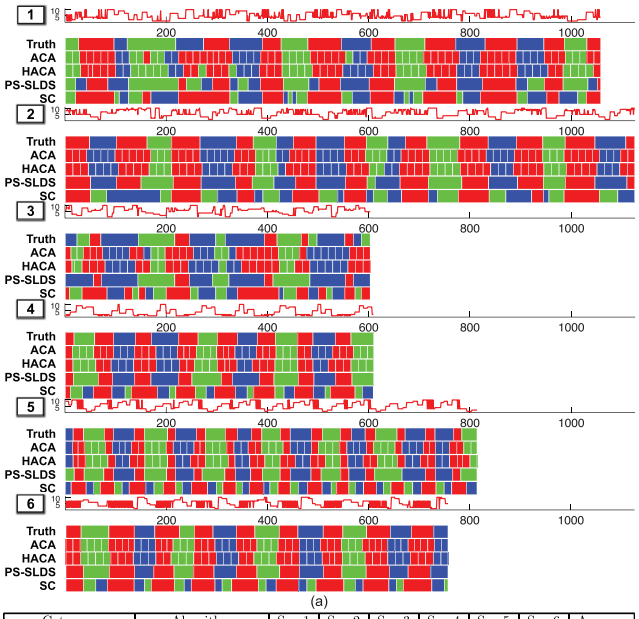
over the last 50 frames. We normalized the position in the range of $[-1, 1]$ and computed the frame kernel matrix as

$$\kappa_{ij} = \exp\left(-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{2\sigma_p^2} - \frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2\sigma_a^2}\right)$$

(Fig. 16c). σ_p and σ_a are set to be the average distance from the 50 and 20 percent closest neighbors, respectively.

We compare ACA to SLDS with a hierarchical Dirichlet process prior (HDP-SLDS) [31], switching vector autoregressive (VAR) process with a HDP prior using unsupervised Gibbs sampling (HDP-VAR (I)) [31], partially supervised Gibbs sampling (HDP-VAR (II)) [31], and parametric segmental SLDS (PS-SLDS) [12]. In this experiment, we set the length constraint to $n_{\max} = 15$ for ACA because it corresponds to the average cycle of the bee dance. In HACA, we set $n_{\max}^{(1)} = 5, n_{\max}^{(2)} = 3$ for the first and second level, respectively. Fig. 17 shows the segmentation results for the six sequences for ACA, HACA, PS-SLDS [12], and SC. We used the results reported in [31] for HDP-SLDS, HDP-VAR (I), and HDP-VAR (II), and the results in [12] for PS-SLDS.

As observed in the table of Fig. 17b, ACA and HACA outperform other unsupervised methods such as SC and HDP-VAR (I), in all six sequences except for the fourth one. In comparison with weakly supervised approaches, HDP-VAR (II), ACA and HACA achieve higher accuracy in four sequences (first, second, fourth, and sixth). Furthermore, ACA and HACA outperform the supervised methods, HDP-SLDS and PS-SLDS, in three sequences (first, second, and sixth) and underperform in the other three (third, fourth, and fifth). HDP-SLDS and PS-SLDS learn the parameters of a mixture of linear dynamic systems using a leave-one-out strategy, whereas ACA and HACA are unsupervised techniques.



Category	Algorithm	Seq 1	Seq 2	Seq 3	Seq 4	Seq 5	Seq 6	Average
Unsupervised	ACA	.845	**.925**	.600	.922	.878	**.928**	.850
	HACA	**.853**	.900	.629	.917	.845	.908	.842
	SC	.698	.631	.509	.671	.577	.649	.623
	HDP-VAR (I) [31]	.465	.441	.456	.832	**.932**	.886	.669
Weakly Supervised	HDP-VAR (II) [31]	.659	.885	.792	.869	.923	.891	.837
Supervised	HDP-SLDS [31]	.740	.861	.813	**.934**	.902	.904	.859
	PS-SLDS [12]	.759	.924	**.831**	**.934**	.904	.910	**.877**

(b)

Fig. 17. Comparison of different temporal clustering methods on the honey bee dance data. (a) Segmentation on the six honey bee dance sequences. Red, blue, and green indicate the regimes of waggling, turning left, and turning right, respectively. (b) Clustering accuracy.

5 Conclusion and Future Work

In this paper, we have proposed ACA and HACA, an extension of KKM and SC for hierarchical temporal clustering and embedding of time series. HACA combines standard vector-space approaches for clustering with DTAK and dynamic programming. We have shown how HACA can robustly temporally cluster human motion in motion capture data and video. Additionally, we have compared the performance of ACA and HACA to state-of-the-art algorithms for supervised and weakly supervised temporal clustering, achieving comparable performance using an unsupervised technique.

Although HACA has shown promising results, there are a number of limitations. First, the computational complexity of ACA is $O(n^2 n_{\max})$, which limits its applicability to long sequences. The HACA algorithm could be sped up by pruning some DP solutions; however, the computational complexity in space would remain $O(n^2)$ due to the computation and storage of the n-by-n frame kernel matrix ($\mathbf{K}$). We are currently extending HACA using subsampling techniques to start the clustering using a decimated version of the time series and propagate it to another layer with higher temporal scales. This enhancement will improve the computational complexity in space and time. On the other hand, the success of ACA partially depends on the choice of the kernel parameters and the functional form of the kernel. Currently, we are exploring approaches to learning the kernel matrix for temporal clustering from a set of manually labeled sequences.

Acknowledgments

This work was partially supported by the US National Science Foundation (NSF) under Grant Nos. EEEC-0540865, RI-1116583, and CPS-0931999. 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] D. Gavrila, "The Visual Analysis of Human Movement: A Survey," *Computer Vision and Image Understanding*, vol. 73, no. 1, pp. 82-98, 1999.

[2] T.B. Moeslund, H. Adrian, and V. Krüger, "A Survey of Advances in Vision-Based Human Motion Capture and Analysis," *Computer Vision and Image Understanding,* vol. 104, nos. 2/3, pp. 90-126, 2006.

[3] R. Poppe, "Vision-Based Human Motion Analysis: An Overview," *Computer Vision and Image Understanding,* vol. 108, nos. 1/2, pp. 4-18, 2007.

[4] Y. Rui and P. Anandan, "Segmenting Visual Actions Based on Spatio-Temporal Motion Patterns," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2000.

[5] L. Zelnik-Manor and M. Irani, "Statistical Analysis of Dynamic Actions," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 9, pp. 1530-1535, Sept. 2006.

[6] D.D. Vecchio, R.M. Murray, and P. Perona, "Decomposition of Human Motion Into Dynamics-Based Primitives with Application to Drawing Tasks," *Automatica,* vol. 39, no. 12, pp. 2085-2098, 2003.

[7] C. Lu and N.J. Ferrier, "Repetitive Motion Analysis: Segmentation and Event Classification," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 2, pp. 258-263, Feb. 2004.

[8] G. Guerra-Filho and Y. Aloimonos, "Understanding Visuo-Motor Primitives for Motion Synthesis and Analysis," *J. Visualization and Computer Animation,* vol. 17, pp. 207-217, 2006.

[9] F. De la Torre, J. Campoy, Z. Ambadar, and J.F. Cohn, "Temporal Segmentation of Facial Behavior," *Proc. 11th IEEE Int'l Conf. Computer Vision,* 2007.

[10] P.K. Turaga, A. Veeraraghavan, and R. Chellappa, "Unsupervised View and Rate Invariant Clustering of Video Sequences," *Computer Vision and Image Understanding,* vol. 113, no. 3, pp. 353-371, 2009.

[11] T. Kobayashi, F. Yoshikawa, and N. Otsu, "Motion Image Segmentation Using Global Criteria and DP," *Proc. IEEE Eighth Int'l Conf. Automatic Face & Gesture Recognition,* 2008.

[12] S.M. Oh, J.M. Rehg, T. Balch, and F. Dellaert, "Learning and Inferring Motion Patterns Using Parametric Segmental Switching Linear Dynamic Systems," *Int'l J. Computer Vision,* vol. 77, nos. 1-3, pp. 103-124, 2008.

[13] A. Fod, M.J. Matarić, and O.C. Jenkins, "Automated Derivation of Primitives for Movement Classification," *Autonomous Robots,* vol. 12, no. 1, pp. 39-54, 2002.

[14] O.C. Jenkins and M.J. Matarić, "Deriving Action and Behavior Primitives from Human Motion Data," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems,* 2002.

[15] J. Barbic, A. Safonova, J.-Y. Pan, C. Faloutsos, J.K. Hodgins, and N.S. Pollard, "Segmenting Motion Capture Data into Distinct Behaviors," *Proc. Graphics Interface,* 2004.

[16] M. Müller, T. Röder, and M. Clausen, "Efficient Content-Based Retrieval of Motion Capture Data," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 677-685, 2005.

[17] G. Liu and L. McMillan, "Segment-Based Human Motion Compression," *Proc. ACM Siggraph/Eurographics Symp. Computer Animation,* 2006.

[18] F. Lv and R. Nevatia, "Recognition and Segmentation of 3-D Human Action Using HMM and Multi-Class AdaBoost," *Proc. European Conf. Computer Vision,* 2006.

[19] R. Hamid, S. Maddi, A. Bobick, and I. Essa, "Structure from Statistics-Unsupervised Activity Analysis Using Suffix Trees," *Proc. 11th IEEE Int'l Conf. Computer Vision,* 2007.

[20] P. Beaudoin, S. Coros, M. van de Panne, and P. Poulin, "Motion-Motif Graphs," *Proc. ACM Siggraph/Eurographics Symp. Computer Animation,* 2008.

[21] O.C. Jenkins and M.J. Matarić, "A Spatio-Temporal Extension to Isomap Nonlinear Dimension Reduction," *Proc. Int'l Conf. Machine Learning,* 2004.

[22] J.B. Tenenbaum, V. de Silva, and J.C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science,* vol. 290, no. 5500, pp. 2319-2323, 2000.

[23] H. Zhong, J. Shi, and M. Visontai, "Detecting Unusual Activity in Video," *Proc. IEEE Conf. Computer Vision Pattern Recognition,* 2004.

[24] F. De la Torre and C. Agell, "Multimodal Diaries," *Proc. IEEE Int'l Conf. Multimedia and Expo,* 2007.

[25] G. Guerra-Filho and Y. Aloimonos, "A Language for Human Action," *Computer,* vol. 40, no. 5, pp. 42-51, May 2007.

[26] D. Minnen, C.L. Isbell, I.A. Essa, and T. Starner, "Discovering Multivariate Motifs Using Subsequence Density Estimation and Greedy Mixture Learning," *Proc. 22nd Int'l Conf. Artificial Intelligence,* 2007.

[27] E.J. Keogh, S. Chu, D. Hart, and M.J. Pazzani, "An Online Algorithm for Segmenting Time Series," *Proc. IEEE Int'l Conf. Data Mining,* 2001.

[28] X. Xuan and K. Murphy, "Modeling Changing Dependency Structure in Multivariate Time Series," *Proc. 24th Int'l Conf. Machine Learning,* 2007.

[29] M. Ostendorf, V.V. Digalakis, and O.A. Kimball, "From HMM's to Segment Models: A Unified View of Stochastic Modeling for Speech Recognition," *IEEE Trans. Speech and Audio Processing,* vol. 4, no. 5, pp. 360-378, Sept. 1996.

[30] F. Desobry, M. Davy, and C. Doncarli, "An Online Kernel Change Detection Algorithm," *IEEE Trans. Signal Processing,* vol. 53, no. 8, pp. 2961-2974, Aug. 2005.

[31] E. Fox, E. Sudderth, M. Jordan, and A. Willsky, "Nonparametric Bayesian Learning of Switching Linear Dynamical Systems," *Proc. Neural Information Processing Systems,* 2008.

[32] S.M. Kay, *Fundamentals of Statistical Signal Processing, Volume 2: Detection Theory.* Prentice-Hall, Inc., 1993.

[33] Z. Harchaoui, F. Bach, and E. Moulines, "Kernel Change-Point Analysis," *Proc. Neural Information Processing Systems,* 2009.

[34] P. Fearnhead, "Exact and Efficient Bayesian Inference for Multiple Changepoint Problems," *Statistics Computing,* vol. 16, no. 2, pp. 203-213, 2006.

[35] V. Pavlović, J.M. Rehg, and J. MacCormick, "Learning Switching Linear Models of Human Motion," *Proc. Neural Information Processing Systems,* 2000.

[36] E. Fox, E. Sudderth, M. Jordan, and A. Willsky, "Sharing Features among Dynamical Systems with Beta Processes," *Proc. Neural Information Processing Systems,* 2009.

[37] J.B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proc. Fifth Berkeley Symp. Math. Statistical Probability,* 1967.

[38] F. De la Torre and T. Kanade, "Discriminative Cluster Analysis," *Proc. 23rd Int'l Conf. Machine Learning,* 2006.

[39] H. Zha, X. He, C.H.Q. Ding, M. Gu, and H.D. Simon, "Spectral Relaxation for *k*-Means Clustering," *Proc. Neural Information Processing Systems,* 2001.

[40] S.Z. Selim and M.A. Ismail, "*k*-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 6, no. 1, pp. 81-87, Jan. 1984.

[41] I.S. Dhillon, Y. Guan, and B. Kulis, "Kernel *k*-Means: Spectral Clustering and Normalized Cuts," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* 2004.

[42] R. Cutler and L.S. Davis, "Robust Real-Time Periodic Motion Detection, Analysis, and Applications," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 781-796, Aug. 2000.

[43] N. Marwan, M.C. Romanoa, M. Thiela, and J. Kurthsa, "Recurrence Plots for the Analysis of Complex Systems," *Physics Reports,* vol. 438, pp. 237-329, 2007.

[44] B.-K. Yi, H.V. Jagadish, and C. Faloutsos, "Efficient Retrieval of Similar Time Sequences under Time Warping," *Proc. 14th Int'l Conf. Data Eng.,* 1998.

[45] H. Shimodaira, K.-I. Noma, M. Nakai, and S. Sagayama, "Dynamic Time-Alignment Kernel in Support Vector Machine," *Proc. Neural Information Processing Systems,* 2001.

[46] B. Schölkopf and A.J. Smola, *Learning with Kernels.* MIT Press, 2002.

[47] B. Haasdonk, "Feature Space Interpretation of SVMs with Indefinite Kernels," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 4, pp. 482-492, Apr. 2005.

[48] M. Cuturi, J.-P. Vert, O. Birkenes, and T. Matsui, "A Kernel for Time Series Based on Global Alignments," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing,* 2007.

[49] T.W. Liao, "Clustering of Time Series Data—A Survey," *Pattern Recognition,* vol. 38, no. 11, pp. 1857-1874, 2005.

[50] F. De la Torre, "A Least-Squares Framework for Component Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 34, no. 6, pp. 1041-1055, June 2012.

[51] R. Zass and A. Shashua, "A Unifying Approach to Hard and Probabilistic Clustering," *Proc. 10th IEEE Int'l Conf. Computer Vision,* 2005.

[52] S. Roweis and Z. Ghahramani, "A Unifying Review of Linear Gaussian Models," *Neural Computation,* vol. 11, no. 2, pp. 305-345, 1999.

[53] D.P. Bertsekas, *Dynamic Programming and Optimal Control.* Athena Scientific, 1995.
[54] E.J. Keogh and M.J. Pazzani, "Scaling Up Dynamic Time Warping for Datamining Applications," *Proc. Sixth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* 2000.
[55] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.
[56] A.Y. Ng, M.I. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," *Proc. Neural Information Processing Systems,* pp. 849-856, 2001.
[57] R. Burkard, M. DellAmico, and S. Martello, *Assignment Problems.* SIAM, 2009.
[58] "Carnegie Mellon University Motion Capture Database," http://mocap.cs.cmu.edu, 2012.
[59] J. Lee, J. Chai, P.S.A. Reitsma, J.K. Hodgins, and N.S. Pollard, "Interactive Control of Avatars Animated with Human Motion Data," *ACM Trans. Graphics,* vol. 21, no. 3, pp. 491-500, 2002.
[60] J. Wang and B. Bodenheimer, "An Evaluation of a Cost Metric for Selecting Transitions between Motion Segments," *Proc. ACM Siggraph/Eurographics Symp. Computer Animation,* 2003.
[61] R. Bowden, "Learning Statistical Models of Human Motion," *Proc. IEEE Workshop Human Modeling, Analysis, and Synthesis,* 2000.
[62] K. Forbes and E. Fiume, "An Efficient Search Algorithm for Motion Data Using Weighted PCA," *Proc. ACM Siggraph/Eurographics Symp. Computer Animation,* 2005.
[63] F. Zhou, F. De la Torre, and J.K. Hodgins, "Aligned Cluster Analysis for Temporal Segmentation of Human Motion," *Proc. Eighth IEEE Int'l Conf. Automatic Face & Gesture Recognition,* 2008.
[64] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as Space-Time Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 12, pp. 2247-2253, Dec. 2007.
[65] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing Human Actions: A Local SVM Approach," *Proc. 17th Int'l Conf. Pattern Recognition,* 2004.
[66] A.A. Efros, A.C. Berg, G. Mori, and J. Malik, "Recognizing Action at a Distance," *Proc. Ninth IEEE Int'l Conf. Computer Vision,* 2003.
[67] B.D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proc. Seventh Int'l Joint Conf. Artificial Intelligence,* 1981.
[68] J.C. Niebles, H. Wang, and L. Fei-Fei, "Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words," *Int'l J. Computer Vision,* vol. 79, no. 3, pp. 299-318, 2008.
[69] Q.V. Le, W.Y. Zou, S.Y. Yeung, and A.Y. Ng, "Learning Hierarchical Invariant Spatio-Temporal Features for Action Recognition with Independent Subspace Analysis," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2011.
[70] M. Hoai, Z.-Z. Lan, and F. De la Torre, "Joint Segmentation and Classification of Human Actions in Video," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2011.

Feng Zhou received the BS degree in computer science from Zhejiang University in 2005, the MS degree in computer science from Shanghai Jiao Tong University in 2008, and the MS degree in robotics from Carnegie Mellon University in 2011. He is now working toward the PhD degree in robotics at Carnegie Mellon University. His research interests include machine learning and computer vision. He is a student member of the IEEE.

Fernando de la Torre received the BSc degree in telecommunications, and the MSc and PhD degrees in electronic engineering from the La Salle School of Engineering at Ramon Llull University, Barcelona, Spain in 1994, 1996, and 2002, respectively. He is an associate research professor in the Robotics Institute at Carnegie Mellon University. His research interests are in the fields of computer vision and machine learning. Currently, he is directing the Component Analysis Laboratory (http://ca.cs.cmu.edu) and the Human Sensing Laboratory (http://humansensing.cs.cmu.edu) at Carnegie Mellon University. He has more than 100 publications in refereed journals and conferences. He has organized and co-organized several workshops and has given tutorials at international conferences on the use and extensions of component analysis.

Jessica K. Hodgins received the PhD degree in computer science from Carnegie Mellon University (CMU) in 1989. She joined the Robotics Institute and Computer Science Department at Carnegie Mellon University as a associate professor in the fall of 2000. She is now a professor in computer science and robotics, associate director for the faculty in the Robotics Institute as well as the director of Disney Research, Pittsburgh Laboratory. Prior to moving to CMU, she was an associate professor and assistant dean in the College of Computing at the Georgia Institute of Technology. She has received a US National Science Foundation (NSF) Young Investigator Award, a Packard Fellowship, and a Sloan Fellowship. She was editor-in-chief of the *ACM Transactions on Graphics* from 2000 to 2002 and papers chair for ACM Siggraph 2003.