Querying Complex Spatio-Temporal Sequences in Human Motion Databases

Yueguo Chen #, Shouxu Jiang *, Beng Chin Ooi #, Anthony K.H. Tung #

#*School of Computing, National University of Singapore*
Computing 1, Law Link, Singapore 117590
{chenyueg, ooibc, atung}@comp.nus.edu.sg

**School of Computer Science and Technology, Harbin Institute of Technology*
Harbin, China 150001
jsx@hit.edu.cn

***Abstract*— Content-based retrieval of spatio-temporal patterns from human motion databases is inherently nontrivial since finding effective distance measures for such data is difficult. These data are typically modelled as time series of high dimensional vectors which incur expensive storage and retrieval cost as a result of the high dimensionality. In this paper, we abstract such complex spatio-temporal data as a set of frames which are then represented as high dimensional categorical feature vectors.**

New distance measures and queries for high dimensional categorical time series are then proposed and efficient query processing techniques for answering these queries are developed. We conducted experiments using our proposed distance measures and queries on human motion capture databases. The results indicate that significant improvement on the efficiency of query processing of categorical time series (more than 10,000 times faster than that of the original motion sequences) can be achieved while guaranteeing the effectiveness of the search.

I. Introduction

Content-based retrieval of spatio-temporal patterns from human motion databases is inherently nontrivial since finding effective distance measures for such data is difficult. Defining similarities (or distances) on such data is difficult because of three reasons: 1) it's hard to extract effective numerical features from them such that an effective distance can be aggregated based on the pair-wise differences of the features; 2) the transformed data (features) are typically in high dimensional space, while the similarities between two such sequences may only exist in some subspace of the overall spatio-temporal space. The simple aggregation of distances may not take such partial similarity into account; 3) matching between the sequences might not be perfect and variation along the spatial and temporal dimensions must be taken into account.

Many existing studies [1], [2], [3] model such data as high dimensional trajectories where each frame or segment of the human motion is represented as a numerical vector in the track of the trajectories. This essentially converts the data into a high dimensional time series and the distance between any pair sequence is then measured based on the distance of the two time series. However, existing studies on the distance measures of time series [4], [5], [6], [7] have been focused on time series with low dimensionality (usually 1-dimensional to 3-dimensional trajectories). They cannot be easily applied to high dimensional time series because of the following three reasons. 1) the efficiency of warping distances for high-dimensional time series cannot be well improved by simply extending existing lower bounding techniques [6], [8] due to the well-known curse of dimensionality. 2) the efficiency of warping distances will be further weakened by the numerous pair-wise distance computation of high dimensional numerical vectors. 3) the numerical similarities obtained from distance aggregation may not be effective enough, especially when large variations (e.g. variations in shifting, scaling, directions and rotations) exist along the spatio-temporal dimensions.

Instead of measuring numerical similarities on such complex spatio-temporal data, a logic-oriented approach has recently been proposed in [9] to perform discrete feature extraction on human motion capture data as a means to improve the efficiency of content-based retrieval on motion capture data. In this technique, a number of categorical geometric features are extracted from snapshots of the numerical motion capture data and sequences of extracted categorical features are used to describe the change of logical states over time. An example is shown in Figure 1. The efficiency is achieved through binary operations on high dimensional categorical time series.

The advantages of extracting discrete features from complex spatio-temporal data include: 1) variations within spatio-temporal patterns are partially handled through discrete transformation; 2) similarities obtained from the categorical states are intuitive, and therefore, provide operational interfaces for users to specify the subspace within which overall similarities are measured; 3) the categorical format of the data reduces the storage and computational cost of complex spatio-temporal data. As such, the model of high dimensional categorical time series provides a significant probe on efficient and effective content-based retrieval of complex spatio-temporal patterns.

Most studies on data management of time series have been focused on time series of numerical vectors [10], [11], [5], [6], [8]. These techniques cannot be applied directly to categorical time series. The studies on managing categorical time series are however still limited to one dimensional data such as gene sequences [12], [13]. To overcome this limitation, our study

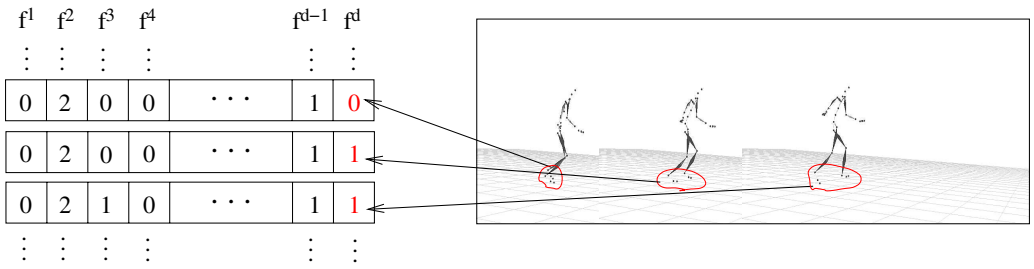

Fig. 1. Three snapshots of d dimensional categorical vectors in a categorical time series. A local geometric feature (dimension d) of distance between feet is marked in red.

here will focus on high dimensional categorical time series. In this paper, we propose new distance measures and query types for high dimensional categorical time series, which are used in our PIPA system (a database engine for interactive media, for supporting effective matches of multimedia objects). Efficient query processing techniques are developed to handle the proposed queries. Our contributions can be summarized as follow:

- We propose the sketch query, which allows users to retrieve complex spatio-temporal patterns by specifying partial dimensions on some key frames. Incremental sketch matching technique is proposed to speed up the sketch query processing.
- We propose to segment high dimensional categorical time series using convexes, and measure the distance of two categorical time series in block-wise. Efficient early abandoning technique is proposed to accelerate the subsequence matching on clip queries.
- Extensive experiments on real-life data were conducted. The results show that our solutions are not only efficient, but also yielding acceptable accuracy for content-based retrieval of spatio-temporal data.

The rest of the paper is organized as follows. Section 2 introduces some related work. Section 3 gives the problem definition. Section 4 and Section 5 introduce the distance definitions, queries, and related query processing techniques of sketch query and clip query respectively. The experimental studies are shown in Section 6, and conclusions are drawn in Section 7.

II. Related Work

A. Distances between Time Series

The distance between any two time series is essentially computed from the aggregation of pair-wise difference of data items within them. Traditionally, the Euclidean distance is used to measure the distances between time series of the same length. Warping distances such as DTW [14] and EDR [4] have been proposed to measure distances of time series with arbitrary lengths. The optimal alignments of data items between two time sequences are obtained by repeating some data items so that the lengths of two sequences can be the same. The distance is calculated by finding the best warping path in the distance matrix using dynamic programming, which has a complexity of $O(mn)$ (m and n are the lengths of time series). Lower bounds of warping distances [6], [8] have been proposed to prune real warping distance computation. Although they can be extended to high dimensional time series [15], the achieved lower bounds will be quite loose. This is because the distance between a high dimensional data item and a high dimensional bounding item (with an extension in each dimension [15]) will be much less than the real distance of two high dimensional data items without the bound.

A more common way to match time series is subsequence matching, where a short time series is compared to a long time series. Subsequence matching attempts to locate matching subsequences within a long time series to a short time series [5], and it is very useful in content-based retrieval of spatio-temporal patterns. It is however more complex than full sequence matching since the positions and lengths of the matching subsequences need to be detected. Some efficient subsequence matching algorithms [5], [16] have been proposed recently. They detect matching subsequences in a dynamic programming manner based on warping distance such as DTW [16] and SpADe [5].

B. High Dimensional Time Series

Content-based retrieval based on high dimensional numerical time series is widely used in multimedia applications [1], [2], [3], [17]. One typical example is the retrieval of motion capture data from a large motion database, which is very useful in computer animation. Many approaches for matching high dimensional numerical time series are based on numerical similarities, in which the high dimensional time series data are typically decomposed into a number of trajectories with low dimensionality (1-dimensional to 3-dimensional) [7], [18]. Warping distances such as DTW distance are preferred to match those low dimensional trajectories to handle time shifting and scaling of time series.

However, due to the variations of shifting, scaling, direction and rotation along the amplitude and temporal dimensions of sequences, the numerical similarities obtained from the distance aggregation may be far from the logical similarities [9], [18] based on human views. To address this problem, effective geometric features have been subsequently proposed to map snapshots of numerical sequences into discrete states [19], [9]. It has been shown that using logic-oriented categorical features not only significantly improves the efficiency of se-

quences matching but also achieves high accuracy in content-based retrieval of human-motion data. Studies on managing categorical time series have been focused on one dimensional gene sequences [12], [13]. To the best of our knowledge, there has been little study on query processing of high dimensional categorical time series.

C. Building Categorical Time Series

Discretization of time series has been studied in [20], [21]. However, they are focused on discretizing time series of low dimensional data. In [9], one high dimensional categorical feature vector is extracted from one frame of human motion data by thresholding the logic-oriented geometric relations. Each logic-oriented geometric relation is actually a function of spatial relationship of some local dimensions, and it is robust to variations within local dimensions; for example, the angle between humerus and radius, the distance between two feet. Discretization is applied to map a geometric relation expression to a small number of categorical states. This is very intuitive as people are sensitive to state difference instead of the quantitative difference in logical similarities. Discretization partially handles variations within numerical time series as small variation does not result in a change in the discretized states. Hysteresis-like thresholding can be applied to avoid random fluctuations around the thresholds [19].

III. Preliminaries

A. Problem Definition

A categorical time series instance S is represented as $S[1:T]$. $S[t]$, the snapshot of S at time point t, is a d-dimensional categorical vector. For those categorical vectors extracted from complex spatio-temporal data, d is typically very large, e.g., more than 40. In this paper, we focus on finding the k nearest neighbors (i.e. kNN queries) of a given query based on some distance measure as it is a very common query in the domains we are studying.

The general kNN query can be described as follow: given a small integer k, a distance function $d(\cdot,\cdot)$, a query example Q and a high dimensional categorical time series database DB, the kNN query retrieves a set M consists of k time series $R \in DB$ such that for any time series $R_1 \in M$ and any time series $R_2 \in DB - M$, $d(R_1, Q) \leq d(R_2, Q)$. Different distance measures of time series result in different query processing solutions for the kNN queries.

B. Distances Between Categorical Vectors

A commonly used distance measure of categorical vectors is Hamming distance. The Hamming distance of two categorical vectors f_1 and f_2, denoted as $H(f_1, f_2)$, is defined as the number of non-matching dimensions of two feature vectors. f_1 and f_2 are said to be matched in dimension i if $f_1^i = f_2^i$. If $H(f_1, f_2) = 0$, then f_1 and f_2 are said to be perfectly matched.

When using Hamming distance to measure the similarities of categorical vectors, users may only be interested in a subset of features by specifying some dimensions that must be matched and some dimensions which can be ignored. Therefore, instead of matching feature vectors in all dimensions, a more operable way is to match them on a user specified subspace. We denote the full dimensional space as $\Omega = \{1, \ldots, d\}$, and the subspace as $\pi \subseteq \Omega$. The Hamming distance on the subspace π can be defined as:

Definition 1 (π-Hamming distance): Given two feature vectors f_1 and f_2, the π-Hamming distance of f_1 and f_2, denoted as $H_\pi(f_1, f_2)$ is defined as the number of unmatched dimensions of two feature vectors in subspace π.

The π-Hamming distance ignores all the dimensions in $\Omega - \pi$. Hamming distance is a specific case of π-Hamming distance when $\pi = \Omega$. Based on the basic distance measures of categorical vectors, a binary decision on the matches of two categorical feature vectors f_1 and f_2 can be made. We say f_1 and f_2 are $\alpha\pi$-match, denoted as $f_1 =_{\alpha\pi} f_2$, if $H_\pi(f_1, f_2) \leq \alpha$.

Besides specifying the subspace, users may also specify some possible states on some chosen dimensions. This can be done by giving a σ clause. A σ selection clause consists of a number of specified features $\sigma = \{i_1, \ldots, i_m\} \subseteq \Omega$. For each feature $j \in \sigma$, a number of matched states $\sigma^j = \{s_{j_1}, \ldots, s_{j_p}\}$ are specified (typically $p = 1$). Given a selection clause σ, a feature vector f is a σ-match, denoted as $f \rightarrow \sigma$, if $\forall i \in \sigma$, $f^i \in \sigma^i$.

Categorical time series are extracted from semantic features. Therefore, it is quite convenient for the users to describe a σ clause by specifying some constraints on the categorical feature vectors. The Hamming distance, subspace π and σ clause form three basic elements of queries on categorical time series.

IV. Sketch Queries

A. Definition

A very common query on high dimensional time series is the sketch query, in which users describe a number of snapshots (by specifying some π and σ clauses) and the system returns those matching subsequences which fit the user-specified snapshots well. For example, in animation design, users may describe their queries by giving some sketches of the starting pose, the ending pose and some intermediate poses of an action. These sketches can be some σ clauses or some examples of snapshots, which are typically not adjacent to each other.

Formally, a sketch $\varphi = \{\sigma, f, \alpha\pi\}$ describes the snapshots users want to retrieve. It consists of a σ clause, an example of categorical vector f and a $\alpha\pi$-match filtering threshold on f. A feature vector f' is a φ-match, denoted as $f' \rightarrow \varphi$, if $f' \rightarrow \sigma$ and $f' =_{\alpha\pi} f$. The distance of f' to a sketch φ is defined as:

$$d(f', \varphi) = \begin{cases} H_\pi(f', f) & \text{if } f' \rightarrow \varphi; \\ +\infty & \text{Otherwise.} \end{cases}$$

It is obvious that $d(f', \varphi) \leq \alpha$ if f' is a φ-match. Note that σ, α, f and π are optional in a sketch φ. When α is not present, it is assumed that no $\alpha\pi$ filtering is conducted on

φ-match, i.e., $\alpha = +\infty$; If f and π are not present, the sketch φ is simply a σ clause, i.e., $f' \to \varphi \Leftrightarrow f' \to \sigma$. In this case, $d(f', \varphi) = 0$ if $f' \to \varphi$. When σ is not present, the sketch φ is simply a $\alpha\pi$ filtering. The φ-distance is a π-Hamming distance of f' to f if $f' =_{\alpha\pi} f$.

Definition 2 (γ-distance): Given a time series R and m sketches $\gamma = \{\varphi_1, \ldots, \varphi_m\}$, the γ-distance of time series R is defined as, $d(R, \gamma) = \min(\Sigma_{i=1}^{m} d(R[t_i], \varphi_i))$, where $0 < t_{i+1} - t_i \leq \Delta$, and Δ is the constraint on the temporal gaps between two consecutive qualified snapshots.

In this definition, $R[t_i]$ $(i = 1, \ldots, m)$ are snapshots of feature vectors within R. They are called qualified snapshots if the constraints $0 < t_{i+1} - t_i \leq \Delta$ (for $i = 1, \ldots, m-1$) are satisfied. The γ-distance of R is the minimum of the sum of φ-distances of m qualified snapshots among all combinations of qualified snapshots within R. Based on the γ-distance, we define the kNN sketch query on categorical time series as follows:

Definition 3 (kNN sketch query): Given a categorical time series dataset DB, a sketch query γ and a small integer k, the kNN sketch query retrieves a set M consisting of k time series $R \in DB$ such that for any time series $R_1 \in M$ and any time series $R_2 \in DB - M$, $d(R_1, \gamma) \leq d(R_2, \gamma)$.

B. φ-match Query Processing

The basis of a sketch query is the single sketch match, i.e., φ-match. A simple φ-match query retrieves all feature vectors which are φ-match in the time series database. The sketch φ is used to describe the feature vectors of interest to the users. The φ-match query can be simply processed by performing σ-match and π-Hamming distance computation on each feature vector within the database. A naïve way to achieve this is to linearly scan all feature vectors. However, it is obviously expensive since there are typically millions of categorical vectors in the database.

There are two ways of organizing categorical feature vectors in the time series database. One way is to ignore the orders of the feature vectors within sequences so that the same vectors in various positions of various time series can be summarized as one vector. A pointer list recording the positions of a feature vector is required so that the positions of the feature vectors can be traced back to the time series. Another way to organize the vectors is to store them based on the orders of vectors and sequences containing the vectors, so that fast consecutive retrieval of feature vectors for a time series can be done. The benefit of the former approach is that it saves both the memory and retrieval cost as the duplications of the feature vectors are removed within the whole database. However, since the vectors in this approach are stored in a random order, the retrieval of clips of time series is not convenient. Since queries on time series require consecutive scan on the feature vectors within time series, we choose the latter strategy in our solution.

There are many similar feature vectors, which may only differ by a few bits to each other, especially for those neighboring feature vectors. A good way to improve the efficiency of φ-match processing is to develop some lower bounding techniques on high dimensional categorical vectors, so that a tight lower bound of φ-distance of similar consecutive vectors can be efficiently calculated. We propose to use a convex to bound a number of similar categorical vectors. The convex provides a lower bound of φ-distances of all vectors within it. Therefore, efficient pruning can be achieved by using the convex.

A d-dimensional categorical feature vector f is actually a word $\bar{f}$ of d' bits in memory, where $d' \geq d$ as one feature may consume more than one bit. Figure 2 shows an example of the storage structure of a feature vector, where one feature f^i is mapped to bits $\bar{f}^{s_i}$ to $\bar{f}^{e_i}$.

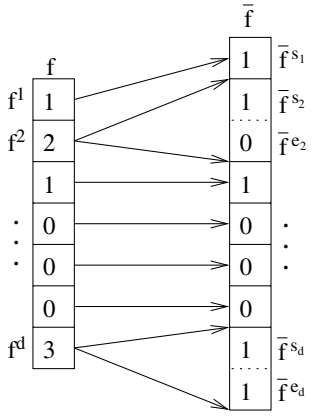

Fig. 2. Binary structure of a feature vector.

Given a number of feature vectors $f_1, \ldots, f_n$, the convex of $f_1, \ldots, f_n$ includes two d'-dimensional binary vectors, $c = (c_\top, c_\bot)$. $c_\top$ is defined as $c_\top = \bigwedge_{i,j} \overline{\bar{f}_i \oplus \bar{f}_j}$, which appears one at positions where $\bar{f}_1, \ldots, \bar{f}_n$ share common binary values. $c_\bot$ is defined as $c_\bot = \bar{f}_1$, which is the first feature vector recording the common binary values shared by $\bar{f}_1, \ldots, \bar{f}_n$ in the common positions represented by $c_\top$. Based on the convex, the lower bound of π-Hamming distance of another feature vector f to c is $LB(f, c) = D_\pi(c_\top \wedge (\bar{f} \oplus c_\bot))$. $D_\pi(\bar{f}) = \sum_{i \in \pi} \bigvee_{j=s_i}^{e_i} f^j$, which is a function checking the number of non-zero values of d features within the subspace π. It is guaranteed that $LB(f, c) \leq H_\pi(f, f_i)$, $i = 1, \ldots, n$.

Figure 3 shows an example of a convex obtained from three consecutive feature vectors f_1, f_2 and f_3. Here, π is the subspace over dimensions of d_1, d_2, d_3, d_5 and d_7. The lower bound $LB(f, c)$ is calculated based on the number of dimensions (in π) having ones (in red of the last word). In this example, it is 3.

A convex c is a σ-match if $\overline{c_\top^j} \vee (c_\bot^j \in \sigma^j)$ is true, $\forall j \in \sigma$. That means those common binary values in convex c must satisfy the σ clause. It is guaranteed that all feature vectors are not σ-matches if their convex c is not a σ-match. However, a feature vector may not be a σ-match even though its convex is a σ-match. Therefore, the σ-match of a convex actually provides a lower bound on the σ-matches of the vectors it contains.

Those bits with zeros in $c_\top$ are called flexible bits as the

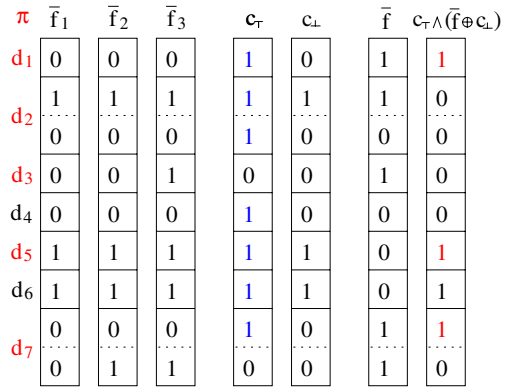

Fig. 3. An example of a convex.

convex allows variance in these bits. However, the effectiveness of convex will be affected by the number of flexible bits in $c_\top$, denoted as ρ. Of course, the smaller the number of flexible bits, the tighter the convex, and the more accurate the convex is in lower bounding φ-distances. However, ρ should not be too small because small ρ limits the number of the feature vectors that could be bounded by a convex.

We build the convexes by dividing categorical time series into a number of segments. Each segment contains a number of consecutive feature vectors, which are packaged into a convex. As shown in Algorithm 1, the convexes are built incrementally. The first feature vector is chosen as the $c_\perp$ of a convex. The convex expands until the number of flexible bits surpasses the threshold ρ. A number of convexes will be learned from a sequence by iteratively implementing the above procedure.

Algorithm 1 Convex building

Input: R, a categorical time series instance, $R[1:T]$
Input: ρ, the maximial number of flexible bits
Output: C, the convexes of features vectors in R
$i = 1$
while $i \leq T$ **do**
  $c_\perp = R[i]$ //the first vector within a convex
  $c_\top = \overrightarrow{1}$ //a vector of ones in all dimensions
  $i++$
  while $i \leq T$ **do**
    $c_\top = c_\top \bigwedge \overline{c_\perp \oplus R[i]}$ //enclose one vector
    if $NumberOfZeros(c_\top) \leq \rho$ **then**
      $aconvex.c_\top = c_\top$
    else
      $aconvex.c_\perp = c_\perp$
      $C.add(aconvex)$ //generate a convex
      break
    $i++$

C. Sketch Query Processing

In content-based retrieval of complex spatio-temporal patterns, a single snapshot is not enough to describe a pattern perfectly. Therefore, a number of sketches are used in a γ sketch query, to retrieve subsequences consisting of φ-match snapshots to the sketches in γ. A sketch query γ can be processed by executing multiple φ-match processing and joining the results of multiple φ-matches based on the temporal gap constraints (specified in Definition 2). kNN queries can be further conducted based on the γ-distances of matching subsequences generated from the spatial join.

We note that a multi-way join requires linear scan of the feature vectors (or convexes) multiple times. However, due to the spatial constraints over consecutive φ-match snapshots, the number of potential matching subsequences to γ may be quite small after the first φ-match or the first few joins of φ-matches. Therefore, subsequent φ-matches and spatial joins may not be necessary since the potential matching subsequences are already reduced to a very small number. It will be more efficient if the multi-way spatial join can be processed incrementally, so that some unnecessary linear scans and φ-match joins can be pruned.

Figure 4 shows the incremental approach of sketch query processing. Given a sketch query $\gamma = \{\varphi_1, \ldots, \varphi_5\}$, φ_1 is first processed by a linear scan of convexes and vectors in the database. A number of snapshots which are φ_1-match are retrieved after the linear scan. Next, for φ_2 clause, we may not need to retrieve all φ_2-match snapshots by linear scan over all the convexes and vectors if the number of retrieved φ_1-match snapshots is limited. The potential matching subsequences can be detected by checking the successive qualified snapshots of those φ_1-matches. Therefore, for each matching snapshot $f_1 \rightarrow \varphi_1$ ($f_1 \in R$), we try to detect successive qualified snapshots in R by only considering the temporal region from $f_1.t$ to $f_1.t + \Delta$. By doing this sketch by sketch, the matching subsequence is found cumulatively. The number of potential matching subsequences drops significantly in the incremental process. As a result, most regions are not required to be scanned during the subsequent φ-match processing. Those matching snapshots within the untouched regions are actually pruned from the computation. The incremental sketch query processing preforms γ distance computation and φ-match on the fly, providing an effective pruning approach for the φ-match processing.

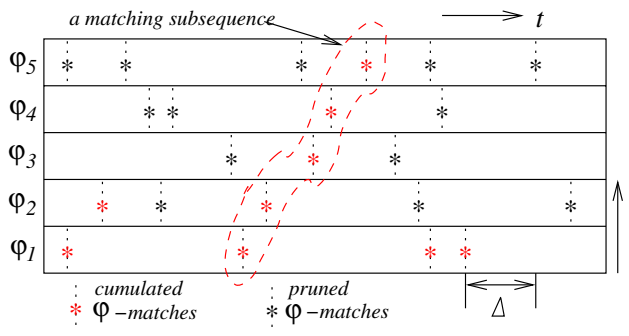

Fig. 4. Incremental sketch matching.

Along with the incremental sketch query processing, the γ-distance of a time series is also calculated incrementally.

The best fitting combination of matching snapshots can be detected by attaching a cumulative distance to each detected φ-match. The cumulative distance of a matching snapshot can be accumulated from the cumulative distance of its predecessor, i.e., $cd(f_i) = cd(f_i.predecessor) + d(f_i, \varphi_j)$. The predecessor f_k of a φ-match snapshot f_i satisfies following conditions: 1), $f_i \rightarrow \varphi_j$, $f_k \rightarrow \varphi_{j-1}$. 2), $f_k.t < f_i.t$ and $f_i.t \leq f_k.t + \Delta$. f_k is said a potential predecessor of f_i if it satisfies conditions 1 and 2. 3), f_k has the lowest cumulative distance among all potential predecessors of f_i. For the example shown in Figure 5, all φ distances of matching snapshots are given in the brackets. The γ-distance of subsequence from f_1 to f_6 is 7.

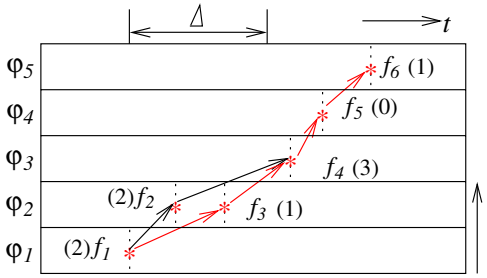

Fig. 5. Computing γ-distance.

V. Clip Queries

A. Distances Between Categorical Time Series

The distance between time series is calculated from the aggregation of pair-wise differences of data items within two time series, regardless of the factual distance used. For categorical time series, the pair-wise distance is actually the π-Hamming distance of two categorical vectors. However, the neighboring categorical vectors within one time series are typically identical or quite similar to each other as they are extracted from consecutive frames of spatio-temporal data, especially when the spatio-temporal data is in high sampling rate or low changing rate. The naive way of pair-wise distance aggregation on those near-duplicative vectors may accrue some disadvantages: 1), the distance of time series will be mainly determined by those sub-regions where intensive duplications exist, while these regions do not provide too much semantics information since the changing rate of time series is slow over these sub-regions. 2), the large number of the resultant distance aggregation (pair-wise) is not efficient; 3), the duplicated aggregation of π-Hamming distance caused by duplications of feature vectors does not make too much sense when local time scaling is present as in many cases of complex spatio-temporal data.

Many studies in content-based retrieval of videos avoid the near-duplication problem by segmenting video sequences and extracting key frames from the video sequences. The similar idea can be applied to high dimensional categorical time series by applying the convex technique. Following the Algorithm 1, a categorical time series can be partitioned into a number of segments/blocks. The similarities of vectors within one block are controlled by the threshold of flexible bits ρ. Upon measuring distances of categorical time series, instead of using pair-wise distance aggregation, we propose to use block-wise distance aggregation. Given two convexes c_1 and c_2, the block-wise distance between c_1 and c_2 is defined as $d(c_1, c_2) = D_\pi((c_1.c_\perp \oplus c_2.c_\perp) \wedge c_1.c_\top \wedge c_2.c_\top)$. The block-wise distance of two time series is then defined as the aggregation of block-wise distances of convexes within the two time series.

B. Clip Queries

We next look at a clip query which retrieves the similar categorical time series instances (or subsequences of time series) of a given query instance of time series. Depending on the matching sequences, clip queries can be classified into full clip queries and subsequence clip queries. In full clip queries, the query clip is compared to the whole time series in the database while in subsequence clip queries, the query clip may be matched to any subsequence of the time series within database. It is obvious that subsequence clip queries are more complex than full clip queries. Full clip queries have been well studied. Subsequence clip queries on the other hand are more useful since users may only provide a query example of a short clip, and the matching clips are most likely to be subsequences within the long time series instances. Consequently, we are more interested in subsequence clip queries. Given a query clip Q and a time series R, we define the τ-distance between R and Q as:

Definition 4 (τ-distance): $d_\tau(R, Q) = \min d(R', Q)$, where R' is any subsequence of R, and $d(R', Q)$ is a full clip block-wise distance between R' and Q.

Note that distance function $d(\cdot, \cdot)$ can be any distance measure of time series. However, we propose to use DTW distance as it handles temporal shifting and scaling between two time series. Moreover, it is widely used as a distance measure of time series. Based on τ-distance, the kNN clip query is defined as:

Definition 5 (kNN clip query): Given a categorical time series dataset DB, a clip query Q and a small integer k, the kNN clip query retrieves a set M consisting of k time series $R \in DB$ such that for any time series $R_1 \in M$ and any time series $R_2 \in DB - M$, $d_\tau(R_1, Q) \leq d_\tau(R_2, Q)$.

C. Clip Query Processing

Note that a time series instance R within DB may be much longer than the query clip Q. The number of subsequences of R will be extremely large as subsequences of various lengths must be allowed in order to handle temporal shifting and scaling between the matching subsequences and Q. Due to the properties of high dimensionality and categorization, indexing techniques on DTW distance [6], [8] cannot be applied to high dimensional categorical time series. It will be very expensive if all subsequences are extracted from R, and linearly compared to Q based on the full clip block-wise distances.

A better solution for matching subsequences is to use the dynamic programming approach for linear scan [5], [16]. For

the query Q with m blocks (convexes) and time series R with n blocks, a matrix of $n \times m$ is used to calculate the τ-distance between R and Q. The τ-distance $d_\tau(R, Q)$ is determined by the best matching subsequence in R. It is calculated column by column in the distance matrix. Within each column, we accumulate distances from bottom to up, based on the block-wise distances of two convexes. According to DTW distance, the cumulative distance at position $[i, j]$ is aggregated as $d[i, j] = \min(d[i, j-1], d[i-1, j], d[i-1, j-1]) + d(R.c_i, Q.c_j)$, with $d[i, 1] = d(R.c_i, Q.c_1)$. Therefore, $d[i_1, j]$ is calculated before $d[i_2, j]$ if $i_1 < i_2$ and $d[i, j_1]$ is calculated before $d[i, j_2]$ if $j_1 < j_2$.

Note that the dominant computational component of τ-distance is from the numerous block-wise distance computation and the accumulating of distances in the distance matrix. To improve the efficiency of τ-distance computation, we propose an early abandoning technique to avoid some distance computation within the distance matrix. Given a pruning threshold ε (achieved in the early stage of kNN query and refined in the later stage), the bound of a column i, denoted as h_i, is defined as:

$$h_i = \begin{cases} 1 & \text{if } i = 0; \\ \max_{j < h_{i-1},\, d[i,j] \le \varepsilon} j + 1 & \text{Else if } d[i, h_{i-1}] > \varepsilon; \\ \max_{j < m,\, d[i,j] \le \varepsilon} j + 1 & \text{Else if } d[i, m] \le \varepsilon; \\ \min_{j > h_{i-1}, d[i,j] > \varepsilon} j & \text{Otherwise.} \end{cases}$$

For each column i, the distances are first accumulated from $d[i, 1]$ to $d[i, h_{i-1}]$. If $d[i, h_{i-1}] > \varepsilon$, the distances within column i will not be accumulated further. Otherwise, the accumulation continues until $d[i, j] > \varepsilon$. The correctness of early abandoning by the bound can be proven by induction. Assume that all cumulative distances $d[i-1, j] > \varepsilon$, where $j \ge h_{i-1}$. Given a cumulative distance $d[i, l] > \varepsilon$, where $l \ge h_{i-1}$, $d[i, l+1]$ must be larger than ε as $d[i-1, l]$ and $d[i-1, l+1]$ are also larger than ε. Incrementally, $d[i, m]$ can be proven to be larger than ε. Therefore, all cells after row l in the column i can be pruned out for distance aggregation. As shown in Figure 6, the bound helps to prune the real block-wise distance computation in the regions where there are no matching subsequences as the distances will quickly surpass ε in these regions.

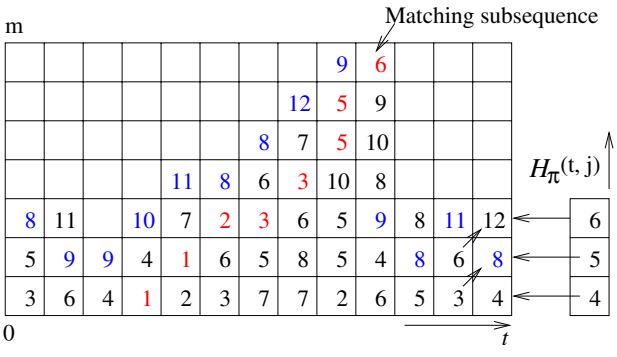

Fig. 6. Illustration of early abandoning in subsequence clip queries.

For the given example shown in Figure 6, suppose the pruning threshold of τ-distance is $\varepsilon = 7$. The cells where the bounds of columns locate are colored with blue. In the the current column of time t, three block-wise distances (4, 5, 6) are calculated. The distance stops from aggregation at the third row since the row position reaches $h_{t-1} = 3$ and $d[t, 3] = 12 > \varepsilon$. The bound of current column is updated as $h_t = 2$ where a cumulative distance of $d[t, 2] = 8$ exists.

VI. Experimental Study

A. Datasets

We generate the high dimensional categorical time series from the real-life dataset of CMU motion capture database [22]. Two thousands of human motion sequences of more than 100 subjects are used in our experiments, with the length of sequences varied from 2 to 22,948. The original motion sequences are 3-dimensional trajectories of 31 joints of human bodies. Therefore, the size of original dataset is $2.3G$. Based on the geometric features proposed in [9], 46 geometric relation features are extracted in our experiments. Among them, there are 19 pair-wise distance features (e.g., hand to hand distance), 11 angle features (e.g., the angle between femur and tibia) and 15 plane features (e.g., whether left hand is before the body). A binary vector of 56 dimensions is generated from one snapshot of 46 dimensional geometric relation features by categorizing these features. In our experiments, each feature is categorized into at most three discrete states. As a result, a categorical time series dataset of $2.4M$ is obtained. Therefore, the compression ratio of categorical time series to the original motion sequences is around **1 : 1000**. Due to its small size, once the whole transformed vector database is read into the memory, it could be kept in memory without further rereferencing. On the other hand, the original database has to be read in and the pages are paged out if there are no buffer pages available. Depending on the buffering strategy, pages may have to be read in for each scan. Our experimental platform is a PC of Pentium4 3.0G CPU with 1G RAM.

B. φ-match Query Processing

To show the effectiveness of categorizing high dimensional time series, we retrieve some pairs of matching categorical snapshots by using π-Hamming distance. Figure 7 shows 3 examples of matching snapshot pairs which have zero π-Hamming distance. We can see that these pairs of snapshots are quite similar. Some of the matching pairs have certain variation on detailed poses. However, the categorical vectors ignore small magnitude of variations, keeping those snapshots exactly matched.

One of the advantages of extracting discrete features from complex spatio-temporal data is to achieve efficiency on matching spatio-temporal data. We compare the efficiency of φ-match (0π-match) on categorical vectors, with range queries (small ε such that similar selectivity is achieved to φ-match) on original motion sequences (both disk-based and in-memory). The average computational time is shown in Table I. As expected, the results clearly show that φ-match on categorical vectors are much faster than range queries on original motion

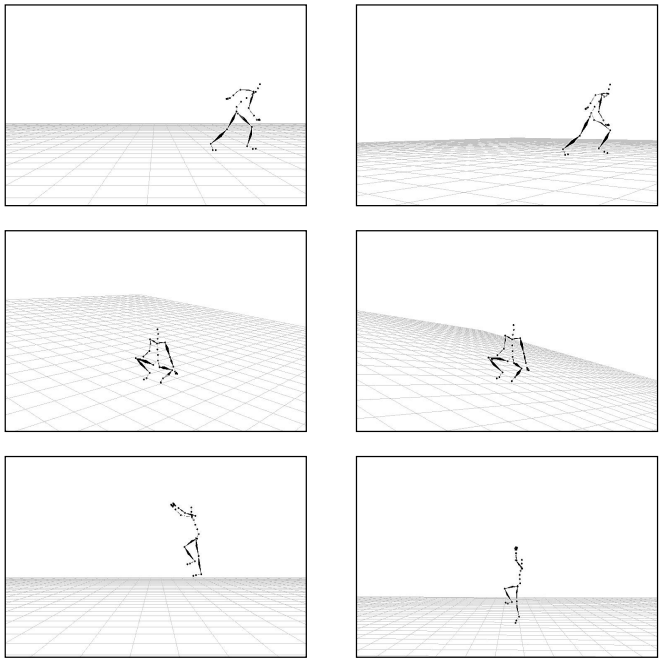

Fig. 7. Pairs of matching snapshots.

sequences, due to the efficient binary operations and small size of categorical vectors. Moveover, many results of the range queries on original motion sequences are far from logical similarities due to the different variations in human motion. While, the results of φ-match are quite matched.

	Original disk-based	Original in-memory	Categorical vectors
t(sec.)	230.1	2.66	0.116

TABLE I
EFFICIENCY ON SNAPSHOT MATCHING.

To evaluate the performance of convexes, we adjust the matching threshold α in φ-match query, and compare the efficiency of φ-match queries on categorical vectors with and without convexes in Figure 8. The number of flexible bits on building convexes is capped at $\rho = 8$. The results show that φ-match of high dimensional categorical vectors with convexes is more efficient than that without convexes when the matching threshold α is small. The performance of φ-match with convexes drops when α is enlarged, as the convex is not effective enough for pruning large distances. However, in practice, the matching threshold in $\alpha\pi$-match is very small (e.g., less than 6). The efficiency of φ-match with convex supports will be better than direct φ-match on categorical vectors.

To further study the performance of convexes, we adjust flexible bit threshold ρ. In practice, ρ should be limited to a small number compared to d' (dimensions of binary vectors) so that the derived convex can be effectively used in pruning φ-distances. However large number of convexes will be generated if ρ is set too small as it limits the number

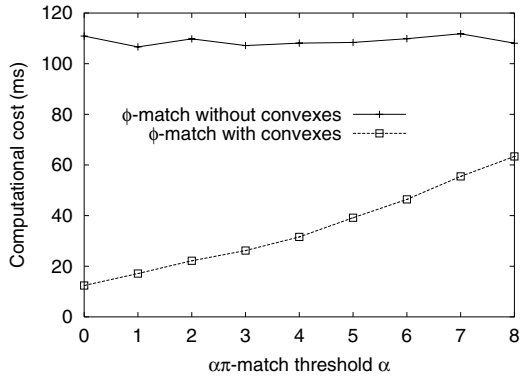

Fig. 8. Efficiency of φ-match queries.

of feature vectors contained by a convex. Consequently, the performance of convex scan will be dropped. The results in Figure 9 shows the impact of ρ on the efficiency of φ-match query processing. In our experiments, convexes achieve better pruning performance when $\rho = 7 \sim 10$.

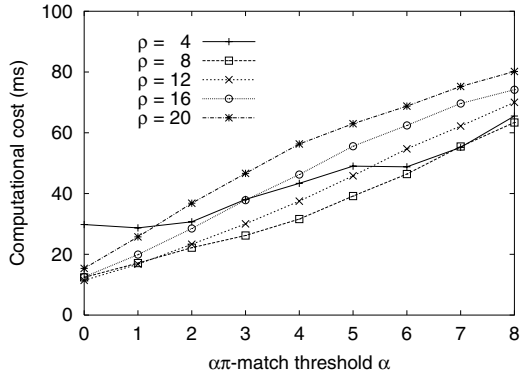

Fig. 9. Impact of ρ on the efficiency of φ-match query processing with convexes.

C. Sketch Query Processing

We evaluate the effectiveness and efficiency of the sketch query using sketches from one time series of a swing action in golf(time series 64_01[1]). As shown in Figure 10, 5 key snapshots are extracted from one query example. Assuming a user is interested in all the swing actions of golfing in the database, he may pose a sketch query consisting of these key snapshots (which may already exist or be obtained from φ-match queries). We use the 8π-match filtering on each sketch in the γ query. The results of 10 nearest neighbors of the γ sketch query are shown in Table II. The first 9 nearest neighbors of a γ sketch query contain all swing actions of golf (time series 64_02-64_10, 64_01 is excluded as it forms the sketch query) in the database. Moreover, the γ distances of these matching actions are no more than 7, which are much more less than those of the other actions (more than

[1]Videos of all time series used in our experiments are available at http://mocap.cs.cmu.edu

25). Therefore, the sketch query in our example effectively distinguishes matching sequences from unmatched sequences. Note that some sketch queries may not be good enough to distinguish sequences, however, they can be modified by refining the matching threshold α, specifying π and σ, or providing more sketches.

Rank	Results	Distances	Rank	Results	Distances
1	64_03	1	6	64_07	6
2	64_09	2	7	64_04	7
3	64_02	4	8	64_06	7
4	64_08	4	9	64_10	7
5	64_05	5	10	14_20	26

TABLE II
RESULTS OF A KNN SKETCH QUERY.

We compare the efficiency of sketch query processing of naive spatial join with that of incremental spatial join. The $\alpha\pi$-match threshold α in the γ sketch query is adjusted from 0 to 10. From the results shown in Figure 11, we can see that two sketch query processing approaches are quite efficient (no more than 0.6 seconds). Comparatively, the incremental spatial join is more efficient than the naive spatial join, especially when α is small. The smaller α filters more snapshots in φ-match. As a result, many regions of the incremental spatial join are pruned in the last several linear scans of φ-match. This is why the computational cost of the incremental spatial join is around 1/5 of that of the naive spatial join, as only one pass of linear scan is required for most of unmatched sequences. The pruning effect of the incremental spatial join will be more prominent if the number of sketches in a γ sketch query is larger.

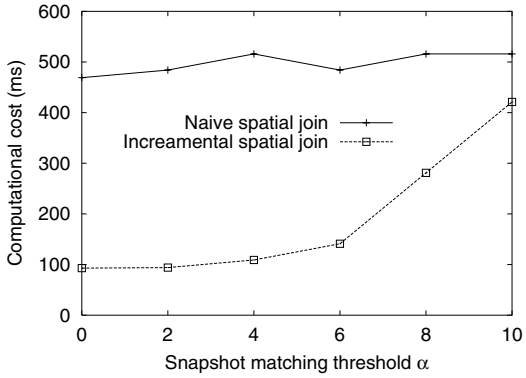

Fig. 11. Efficiency of sketch queries.

D. Clip Query Processing

We now compare the accuracy and efficiency of pair-wise distance and block-wise distance on categorical time series over a number of clip queries. The accuracy of one nearest neighbor classification has been widely used as an important measure of effectiveness of time series distance measures in many studies [4], [5], [23]. Therefore, we use it as the benchmark for accuracy comparison. 72 time series of moderate lengths (with $300 \sim 600$ snapshots) are randomly chosen as query instances. We do not use too many query examples as the class labels of the time series are not given explicitly, and many actions within long time series need to be checked manually. The average results of one nearest neighbor queries are shown in table III. We can see that block-wise distance is much more efficient than the pair-wise distance due to the summarization of categorical vectors. The accuracy of block-wise distance however can also be better than pair-wise distance when the flexible bit threshold ρ is small. For example, compared to pair-wise distance, an efficiency of 136 times faster is achieved by block-wise distance when $\rho = 4$, while the accuracy is still better than that of pair-wise distance. We should note that some error rates are generated because there are actually no matching sequences of the query clips.

ρ	Pair-wise	0	2	4	6	8
Error rate	18.1%	15.3%	15.3%	16.7%	23.6%	36.1%
Time (sec.)	66.5	4.09	1.33	0.49	0.24	0.13

TABLE III
COMPARISON OF ACCURACY AND EFFICIENCY ON ONE NEAREST NEIGHBOR CLIP QUERIES.

To improve the efficiency subsequence matching, we apply the proposed early abandoning technique on τ-distance computation, and test the efficiency of clip queries in Figure 12. The query clip is an action of kicking ball in soccer. We perform several kNN queries by adjusting the parameter k. Figure 12 shows that the less the k, the more efficiency can be achieved by the early abandoning technique because more pruning power is achieved when the pruning threshold ε is smaller. We also conduct a 4NN query of this query instance on the original motion sequences, which consumes more than 20,000 seconds. Comparatively, the categorical time series achieves more than 10,000 times faster. Moreover, the results of 4NN query on original sequences are far from accuracy due to spatial variations.

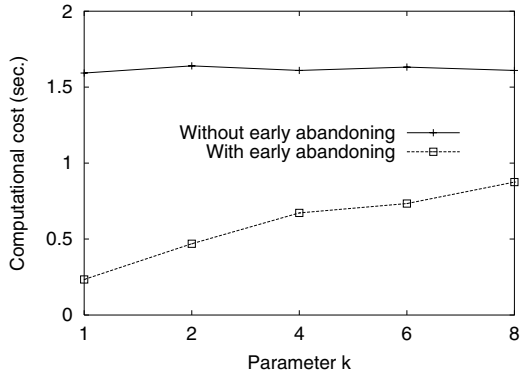

Fig. 12. Example of the effectiveness of early abandoning in clip query.

VII. CONCLUSIONS

In this paper, we have addressed the problem of query processing on high dimensional categorical time series, and

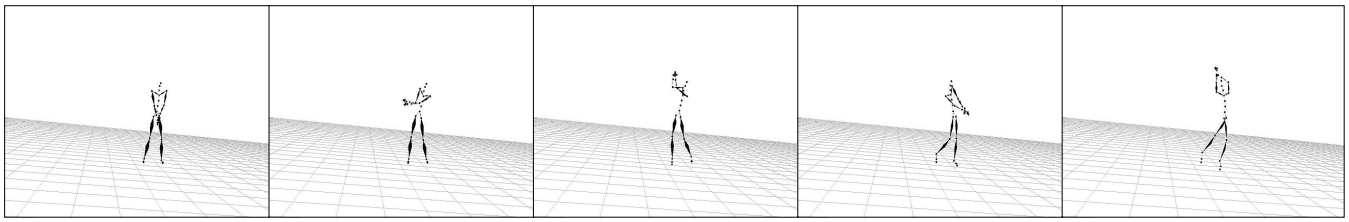

Fig. 10. Snapshot examples of a γ sketch query (a swing in golf).

to the best of our knowledge, this paper is the first to address the issue. We show that the transformation of complex spatio-temporal data to high dimensional categorical time series reduces the size of the database significantly. Effective distance measures on categorical time series are developed to perform content-based retrieval of complex spatio-temporal data efficiently and effectively.

We define the basic distance measures of high dimensional categorical time series, γ-distances and τ-distances, and subsequently propose the sketch queries and clip queries of the categorical time series. We also propose efficient query processing techniques on these queries. Experiments on the proposed distance measures and queries were conducted on a real-life spatio-temporal dataset, the CMU motion capture database [22]. The results demonstrate the efficiency and effectiveness of our proposed distance measures and queries. The proposed techniques significantly improve the efficiency of kNN queries on original motion sequences up to 10,000 times while guaranteeing the accuracy of the search. In conclusion, query processing based on categorical time series provides an efficient and effective means for content-based retrieval of complex spatio-temporal sequences.

ACKNOWLEDGMENT

This work was supported in part by the National Grant Fundamental Research 973 Program of China (Grant No. 2006CB303000), the Key Program of National Natural Science Foundation of China (Grant No. 60533110).

REFERENCES

[1] S.-C. S. Cheung and A. Zakhor, "Efficient video similarity measurement with video signature." *IEEE Trans. Circuits Syst. Video Techn.*, vol. 13, no. 1, pp. 59–74, 2003.

[2] L. Kovar and M. Gleicher, "Automated extraction and parameterization of motions in large data sets." *ACM Trans. Graph.*, vol. 23, no. 3, pp. 559–568, 2004.

[3] H. T. Shen, B. C. Ooi, and X. Zhou, "Towards effective indexing for very large video sequence database." in *SIGMOD Conference*, 2005, pp. 730–741.

[4] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories." in *SIGMOD Conference*, 2005, pp. 491–502.

[5] Y. Chen, M. A. Nascimento, B. C. Ooi, and A. K. H. Tung, "Spade: On shape-based pattern detection in streaming time series." in *ICDE*, 2007, pp. 786–795.

[6] E. J. Keogh, "Exact indexing of dynamic time warping." in *VLDB*, 2002, pp. 406–417.

[7] E. J. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle, "Indexing large human-motion databases." in *VLDB*, 2004, pp. 780–791.

[8] Y. Zhu and D. Shasha, "Warping indexes with envelope transforms for query by humming." in *SIGMOD Conference*, 2003, pp. 181–192.

[9] M. Müller, T. Röder, and M. Clausen, "Efficient content-based retrieval of motion capture data." *ACM Trans. Graph.*, vol. 24, no. 3, pp. 677–685, 2005.

[10] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases." in *FODO*, 1993, pp. 69–84.

[11] Y. Cai and R. T. Ng, "Indexing spatio-temporal trajectories with chebyshev polynomials." in *SIGMOD Conference*, 2004, pp. 599–610.

[12] S. Burkhardt, A. Crauser, P. Ferragina, H.-P. Lenhof, E. Rivals, and M. Vingron, "*Q*-gram based database searching using a suffix array (quasar)." in *RECOMB*, 1999, pp. 77–83.

[13] C. Meek, J. M. Patel, and S. Kasetty, "Oasis: An online and accurate technique for local-alignment searches on biological sequences." in *VLDB*, 2003, pp. 910–921.

[14] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD Workshop*, 1994, pp. 359–370.

[15] T. M. Rath and R. Manmatha, "Lower-bounding of dynamic time warping distances for multivariate time series." in *University of Massachusetts Amherst Technical Report MM-40*, 2002.

[16] Y. Sakurai, C. Faloutsos, and M. Yamamuro, "Stream monitoring under the time warping distance." in *ICDE*, 2007, pp. 1046–1055.

[17] Y. Sheikh, M. Sheikh, and M. Shah, "Exploring the space of a human action." in *ICCV*, 2005, pp. 144–149.

[18] A. Veeraraghavan and A. K. R. Chowdhury, "The function space of an activity." in *CVPR (1)*, 2006, pp. 959–968.

[19] M. Müller and T. Röder, "Motion templates for automatic classification and retrieval of motion capture data," in *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2006.

[20] E. J. Keogh, J. Lin, and A. W.-C. Fu, "HOT SAX: Efficiently finding the most unusual time series subsequence." in *ICDM*, 2005, pp. 226–233.

[21] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos, "A multiresolution symbolic representation of time series." in *ICDE*, 2005, pp. 668–679.

[22] *CMU Graphics Lab Motion Capture Database*, http://mocap.cs.cmu.edu/.

[23] E. J. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: a survey and empirical demonstration." in *KDD*, 2002, pp. 102–111.