Using SAX representation for human action recognition

Imran N. Junejo*, Zaher Al Aghbari

University of Sharjah, P.O. Box 27272, Sharjah, United Arab Emirates

A R T I C L E   I N F O

Article history:
Received 13 December 2011
Accepted 27 April 2012
Available online 11 May 2012

Keywords:
Action recognition
Computer vision
Pattern recognition
Image understanding
SAX representation
Data mining
Intelligent systems
Video surveillance

A B S T R A C T

Human action recognition is an important problem in Computer Vision. Although most of the existing solutions provide good accuracy results, the methods are often overly complex and computationally expensive, hindering practical application. In this regard, we introduce Symbolic Aggregate approximation (SAX) to address the problem of human action recognition. Given motion trajectories of reference points on an actor, SAX efficiently converts this time-series data to a symbolic representation. Moreover, the distance between two time series is approximated by the distance between their SAX representation, which is straightforward and very simple. Requiring only trajectories of reference points, our method requires neither structure recovery nor silhouette extraction. The proposed method is validated on two public datasets. It has an accuracy comparable to related works and it performs well even in varying conditions, in addition to being faster compared to the existing methods.

© 2012 Elsevier Inc. All rights reserved.

1. Introduction

Visual recognition and understanding of human actions has attracted much attention over the past three decades [1,2] and remains an active research area of Computer Vision. A good solution to the problem holds a yet unexplored potential for many applications such as the searching and the structuring of large video archives, video surveillance, human–computer interaction, gesture recognition and video editing. The difficulty associated with this problem has to do with the large variation of human action data due to the individual variations of people in expression, posture, motion and clothing; perspective effects and camera motions; illumination variations; occlusions and disocclusions; and distracting effects of scenes surroundings. As a consequence, current methods often resort to complex solutions, which might provide a good accuracy, but render the solutions impractical.

Various approaches using different constructs have been proposed over the years for action recognition. These approaches can be roughly categorized on the basis of representation used by the authors. Time evolution of human silhouettes was frequently used as an action descriptor. For example, [3] proposed to capture the history of shape changes using temporal templates and [4] extends these 2D templates to 3D action templates. Similarly, the notions of action cylinders [5], and space–time shapes [6–8] have been introduced based on silhouettes. Recently, space–time approaches analyzing the structure of local 3D patches in the video have shown promising [9–13]. Almost all of the mentioned works above rely mostly on an effective feature extraction technique, which is then combined with machine learning or pattern recognition techniques. These feature extraction methods can be roughly categorized into: motion-based [14,15], appearance based [8], space–time volume based [6–8], and space–time interest points or local features based [9–13,16]. Motion-based methods generally compute the optical-flow from a given action sequence, followed by appropriate feature extraction. However, optical-flow based methods are known to be very susceptible to noise and are easily led to inaccuracies. Appearance based methods are prone to differences in appearance between the already seen sequences (i.e. the training dataset) and the new sequences (i.e. the testing sequence). Volume or shape based methods require highly detailed silhouette extraction, which may not be possible in a given real-world noisy video datasets. In comparison to these approaches, the space–time interest point (STIP) based methods [9,11,12] have received considerable attention and thus are extremely popular. STIP-based methods are more robust to noise and camera movement and also seem to work quite well with low resolution inputs. However, these methods rely solely on the discriminative power of individual local space–time descriptors. Information related to the global spatio-temporal distribution is ignored. Thus due to the lack of this temporal information, smooth motions cannot be captured using STIP methods. In addition, issues like optimal space–time descriptor selection and codebook clustering algorithm selection have to be addressed, with fine-tuning various parameters, which is highly data dependent [17].

* Corresponding author.
E-mail addresses: iJunejo@sharjah.ac.ae (I.N. Junejo), zaher@sharjah.ac.ae (Z.A. Aghbari).
Most of above mentioned techniques perform reasonably well for standard action recognition. However, for large-scale action recognition, where the dataset consists of thousands of labeled action videos, such a scheme requires a tremendous amount of time for (i) computing the image/video descriptors, and (ii) computing the (dis)similarities between the sequences – thus the complexity increases dramatically. Dimensionality reduction helps somewhat, but at the cost of reduced accuracy. In this regards, a simple and an efficient method is desired that scales well to large datasets, and yet provides comparable or better accuracy than existing methods.

In this work, while using trajectories of tracked body-joints, we use a new time series representation: Symbolic Aggregate Approximation (SAX) [18–20] in order to effectively address the action recognition problem. SAX tremendously reduces the dimensionality of time series that represents an action sequence. Other symbolic representations, such as Discrete Fourier Transform (DCT) [21], and Discrete Wavelet Transform (DWT) [22], may be used to reduce the memory space required to store the time series but the intrinsic dimensionality of the symbolic representation is not reduced [20]. Another desirable feature of SAX is its lower bounding property. That is, the Euclidean distance between two time series in the SAX space is guaranteed to be less than or equal to the distance between them in the original space [20]. Due to the above two features, SAX symbolic representation is used in this paper to efficiently and effectively recognize human actions from large sequences of video. Thus using this method, in contrast to the geometry-based methods [5,23–25] (to be discussed in Section 2), we do not require estimation of corresponding points between video sequences. Differently to the prevailing methods, we do not extract any features from SAX representation and rely solely on the Euclidean distance; thus we are able to avoid resorting to expensive machine learning techniques. We believe that this is the key to a fast and an efficient solution: avoiding complex computations and achieving good and comparable action recognition accuracy.

The proposed method is illustrated in Fig. 1. A sample from a golf sequence is shown in Fig. 1(a) from the CMU motion capture (mocap) dataset. Here we show trajectory of only one body-joint, for clarity of presentation. In this figure, we project the 3D mocap data onto a 2D image plane using synthetic cameras. As a trajectory can be represented by $(x,y,t)$, for the sake of simplicity, Fig. 1(b) shows the hand trajectory projected onto the $x$-axis, where $t$ is time, and the length of the trajectory is 350. First, treating the hand trajectory as a time series, we compute the SAX-based representation, an example is shown in Fig. 1(c), where the horizontal axis represents the SAX breakpoints that map trajectory values into SAX symbols and the vertical axis represents the SAX words that quantize the trajectory. This particular example is only of length 8 and an alphabet size (i.e. distinct values) of only 10. This process thus not only reduces the trajectory size, but it also makes the proposed method computationally feasible, as we shall demonstrate in Section 5. Second, once we have the SAX representation of the training and the test sequence, we used...
the distance measure defined in Section 4 to compare corresponding tracked joints and perform the Nearest Neighbor (NN) classification. Although the method does require known correspondences between joints but time correspondences are not required. We show recognition results comparable to the state of the art methods, and additionally the method is shown to be very fast as well.

The paper is organized as follows: Section 2 reviews some of the work relevant to the action recognition and the time series representation. We introduce the SAX representation of human action recognition in Section 3. We describe our action recognition scheme in Section 4, along with distance measure used to compare different SAX representations. Finally, we test the proposed method on two public datasets and demonstrate strong results in Section 5, followed by conclusion.

2. Related work

This paper addresses action recognition, a topic which has received considerable attention from researchers recently. The input to our method is the tracked position of body joints of an actor performing a particular action sequence – we will also call these as the reference points. This is an input to various existing techniques as well, especially view-independent action recognition methods using epipolar geometry as in [5,23–27]. In [5,23,24] point correspondences between actions are assumed to be known for imposing fundamental matrix constraints and performing view-invariant action recognition. Rao et al. [25] show that the maxima in space–time curvature of a 3D trajectory are preserved in 2D image trajectories, and are also view-invariant. Parameswaran and Chellappa [27] proposed a quasi view-invariant approach, requiring at least 5 body points lying on a 3D plane or that the limbs trace a planar area during the course of an action. Recently [26] showed that for a moving plane the fundamental ratios, i.e. the ratios among the elements in the upper left 2 × 2 submatrix of the fundamental matrix, are invariant to the camera parameters as well as its orientation and can be used for action recognition. Such methods, however, rely either on existing point correspondences between image sequences or/and on many videos representing actions in multiple views. Both of these assumptions, are limiting in practice due to (i) the difficulty of estimating non-rigid correspondences in videos and (ii) the difficulty of obtaining sufficient video data spanning view variations for many action classes. Thus, obtaining automatic and reliable point correspondences across videos for natural human actions is a very challenging and currently unsolved problem, which limits the application of above mentioned methods in practice. Given an action sequence and some low-level features, [28] compute distances between extracted features for all pairs of time frames and store results in a Self-Similarity Matrix (SSM). The authors claim SSM to be stable across view changes. Histograms of gradient orientations are accumulated at local patches in regions around the diagonal of these SSMs, which have a log-polar structure. Finally Nearest Neighbor classifiers or the SVMs are used for the final classification. Although we do not claim our SAX representation to be view-independent, but we shall demonstrate experimentally that the performance decreases gracefully under 3D view changes, which indicates the robustness of the proposed method.

Treating motion trajectory of reference points on an actor as time-series data has a close relation to the works [29–32] (a full review of all the related work is beyond the scope of the current work). Using velocity history of (KLT) tracked points, [30] present a generative mixture model for the video sequences. A log-polar uniform quantization is performed on these tracks, with eight bins for direction, and five for magnitude. Activity classes are modeled as weighted mixture of bags of augmented trajectory sequences, where each mixture component models a velocity history feature with a Markov Chain. The model is trained by using the EM algorithm. In addition, to choose the best parameters for quantization, depending on the number of mixture components, the approach can be computationally expensive. Employing a probabilistic framework, [29] present a method for augmenting local features with spatio-temporal relationships. Using STIP-HOG based and trajectory-based features, probabilities are accumulated in a Naïve-Bayes like fashion into a reduced number of bins, and SVMs are used to perform the final classification. Using concepts from the theory of chaotic systems [31], the trajectories of reference points are treated as non-linear dynamical system that are generating an action. Using delay-embedding scheme, each such reference trajectory is used to construct a phase space of appropriate dimension. Final representation of each action is a feature vector, containing dynamical and metric invariants of the constructed phase space, namely Lyapunov exponent, correlation integral and correlation dimension. Along the same lines, based on extracted silhouettes, [32] introduces a manifold embedding method, called Local Spatio-Temporal Discriminant Embedding (LSTDE), that projects points lying in a local neighborhood into the embedding space. The idea is that the points of the same class are close together in this space and the data points of different classes are far apart. The authors aim to find an optimal embedding by maximizing the principal angles between temporal subspaces associated with data points of different classes. However, the embedding is a costly affair, often relying on fine-tuning of many parameters such an embedding dimension. In this work, we show recognition results comparable to [31] and a significant improvement over [32], in addition to proposing a simple and computationally less complex solution that is very efficient.

Typically, since the size of time series is too large to fit in main memory, reducing the size of time series using an appropriate representation emerged as an issue of interest in the data mining community. Many representation techniques have been proposed in the literature, but we will address in this section the most commonly used ones. Faloutsos et al. introduced Discrete Fourier Transform, DFT, [21] to reduce the dimensionality of subsequences which are then indexed using the reduced representation. The indexing was then improved in [22] by using the Discrete Wavelet Transform, DWT, to reduce the dimensionality of time series; however, DWT is effective for time series of lengths that are an integer power of two. The authors of [33] used the Piecewise Aggregate Approximation, PAA, in efficient data mining techniques. Then, the Singular Value Decomposition, SVD, was proposed by [34] to improve the accuracy of time series mining however at the expense of the computational cost. In contrast to the above mentioned techniques, the SAX representation uniquely has both desirable properties [20]: dimensionality reduction of time series and lower bounding of Euclidean distance between time series in the original space.

Recently, although a different approach altogether, [16] introduces Prototype Trees to solve action recognition. We thus compare results of the proposed method with the Prototype Trees approach and demonstrate the effectiveness and efficiency of our tool.

3. SAX – time series representation

We briefly review the SAX (Symbolic Aggregate approXimation) representation of a time series as presented in [20]. Although there are many representations for time series in the literature, SAX is the only one that reduces the dimensionality and lower bounds the Lp norms (distance functions) [35], which guarantees that
the distance between any two vectors in the SAX representation is smaller than, or equal to, the distance between these two vectors in the original space. Table 1 explains the symbols used in this paper.

We begin by a formal definition of time series:

**Definition.** A time series $T = \{t_1, t_2, \ldots, t_n\}$, $t_i$ is an ordered set of $n$ real-valued variables.

Before dealing with time series, we have to remove the distortions, namely the offset translation and the amplitude scaling, that could greatly affect the results of the action recognition tasks. Thus, we normalize each time series to have a mean of zero and standard deviation of one. Breakpoints are a set of numbers that divide the curve into $p$ equi-probable regions, where $T_o$ is the original time series, $\text{mean}(T_o)$ is the mean value of time series variables, and $\text{std}(T_o)$ is the standard deviation of the time series variable.

### 3.1. Converting time series to SAX representation

After normalizing the time series, it is converted to a Piecewise Aggregate Approximation (PAA) and then to the symbolic representation SAX (see Fig. 2).

The conversion process is performed by the *time series to SAX representation algorithm*, which takes as input the time series, $T$, its length, $n$, the number of PAA segments, $w$, and the number of alphabets used to represent the time series, $p$.

The details of the *time series to SAX representation algorithm* are explained below:

**Algorithm.** Time series to SAX representation

**Input:** $T$, $n$, $w$, $p$

**Output:** SAX representation

1. Normalize $T$ using Eq. (1).
2. Set the number of PAA segments i.e. $w$.
3. Transform $T$ into PAA coefficients using Eq. (2).
4. Convert the PAA coefficients into SAX symbols.

- An input time series $T$ of length $n$ is normalized using Eq. (1).
- Choose an appropriate value for $w$.
- Transform $T$ into segments using PAA by dividing the length $n$ of $T$ into $w$ equal-sized “frames”. The mean value of the data falling within a frame is calculated using Eq. (2) and a vector of these values (PAA coefficients) becomes the data-reduced representation.

$$X_i = \frac{w}{n} \sum_{j=\lfloor i\cdot w \rfloor+1}^{\lfloor (i+1)\cdot w \rfloor} x_j$$

- Since normalized time series follow the Gaussian distribution [19,20], breakpoints are determined so that they produce equal-sized areas under the Gaussian curve. The Gaussian curve used to compute the breakpoints has been defined to represent the distribution of the normalized time series and thus it has a mean of zero and standard deviation of one. Breakpoints are a sort of numbers $B = \{b_1, b_2, \ldots, b_{p-1}\}$ such that the area under a Gaussian curve from $b_i$ to $b_{i+1} = 1/p\cdot b_0$ and $b_0$ are defined as $-\infty$ and $+\infty$, respectively, see Fig. 2. These breakpoints can be taken from a breakpoint lookup table [19,20] and they divide the amplitude values of the time series into $p$ equi-probable regions as follows:
  - Depending on the value of $p$, the breakpoints are determined.
  - All the PAA coefficients that are below the smallest breakpoint are mapped to the first symbol, say $a$, all coefficients greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped to the second symbol, say $b$, and so on.
- After mapping all PAA coefficients to their corresponding symbols, we get a SAX representation of the input time series. For example, in Fig. 2 the SAX representation of the time series $T_i$ is bbcbaacb.

### 4. SAX-based action recognition

For Computer Vision applications, we look for a representation that is general enough to work with trajectories of different lengths, is easy to compute and does not introduce computational overhead. In addition, Computer Vision also applications requires a representation with fewer parameters to optimize. This is where the SAX representation shines considerably. Based on extensive experiments, the authors of [20,35] show that the parameters alphabet size, $p$, and word size, $w$, are not too critical and a word size in the range of $5$–$8$ and an alphabet size of $3$–$10$ seem reasonable for the task at hand. However, the appropriate value of $w$ and $p$ depends on the data. That is, smaller values of $w$ are more suitable for relatively smooth and slowly varying trajectories and on the other hand larger value of $w$ are appropriate for fast varying trajectories. Also, smaller values of $p$ would result in clustering.
trajectory values in few alphabets and larger values of \( p \) works against the SAX goal to reduce the dimensionality of trajectories.

4.1. Features

We aim to match action sequences with similar characteristics. Therefore, in addition to the reference point trajectories, which we refer to as \( \text{traj} \) feature, we also explore three additional features:

4.1.1. Motion characteristics:

The action trajectory whose velocity is similar to the velocity characteristics of another action trajectory should be considered similar. Velocity for a trajectory \( T(x_i, y_i, t_i), i = 0,1,N - 1 \), is calculated as:

\[
\mathbf{v}_i = \left( \frac{x_{i+1} - x_i}{t_{i+1} - t_i}, \frac{y_{i+1} - y_i}{t_{i+1} - t_i} \right), \quad i = 0,1, \ldots N - 1
\] (3)

where \( \mathbf{v}_i \) is velocity information of the action trajectory. In addition, we also extract the acceleration from the action trajectory. We convert the velocity and the acceleration to the SAX representation, which we denote as \( \text{vel} \) and \( \text{acc} \), respectively.

4.1.2. Spatio-temporal curvature feature:

We would like to capture the discontinuity in the velocity, acceleration and position of an action trajectory. This allows us to discriminate between a straight arm motion and a motion involving elbow bending etc. The curvature is defined as:

\[
\kappa = \frac{\sqrt{y''(t)^2 + x''(t)^2 + (x'(t)y''(t) - x''(t)y'(t))^2}}{(\sqrt{x'(t)^2 + y'(t)^2 + 1})^3}
\] (4)

where \( x' \) and \( y' \) are the velocity components in \( x \) and \( y \) direction, and \( x'' \) and \( y'' \) are the acceleration components in \( x \) and \( y \) direction.

<table>
<thead>
<tr>
<th>Method</th>
<th>cam1</th>
<th>cam2</th>
<th>cam3</th>
<th>cam4</th>
<th>cam5</th>
<th>cam6</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSM-based</td>
<td>92.7%</td>
<td>92.7%</td>
<td>95.7%</td>
<td>95.7%</td>
<td>91.5%</td>
<td>93.9%</td>
</tr>
<tr>
<td>Propose Method</td>
<td>94.9%</td>
<td>93.3%</td>
<td>92.3%</td>
<td>91.9%</td>
<td>94.3%</td>
<td>96.1%</td>
</tr>
<tr>
<td>( \text{traj} )</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>( w )</td>
<td>14</td>
<td>6</td>
<td>6</td>
<td>21</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>Propose Method</td>
<td>95.2%</td>
<td>94.8%</td>
<td>94.0%</td>
<td>96.8%</td>
<td>96.8%</td>
<td>96.8%</td>
</tr>
<tr>
<td>( \text{traj}+\text{curv} )</td>
<td>8</td>
<td>8</td>
<td>11</td>
<td>23</td>
<td>14</td>
<td>6</td>
</tr>
<tr>
<td>( p )</td>
<td>8</td>
<td>14</td>
<td>14</td>
<td>4</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>Propose Method</td>
<td>91.7%</td>
<td>91.0%</td>
<td>89.2%</td>
<td>89.8%</td>
<td>96.0%</td>
<td>90.8%</td>
</tr>
<tr>
<td>( \text{traj}+\text{vel} )</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>( w )</td>
<td>6</td>
<td>17</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>Propose Method</td>
<td>98.1%</td>
<td>91.8%</td>
<td>93.5%</td>
<td>94.8%</td>
<td>97.1%</td>
<td>94.0%</td>
</tr>
<tr>
<td>( \text{traj}+\text{acc} )</td>
<td>8</td>
<td>7</td>
<td>12</td>
<td>19</td>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>( w )</td>
<td>11</td>
<td>14</td>
<td>14</td>
<td>9</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>Propose Method</td>
<td>93.9%</td>
<td>91.2%</td>
<td>90.3%</td>
<td>93.0%</td>
<td>97.3%</td>
<td>92.6%</td>
</tr>
<tr>
<td>( \text{traj}+\text{vel}+\text{acc}+\text{curv} )</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>13</td>
<td>6</td>
</tr>
<tr>
<td>( p )</td>
<td>11</td>
<td>11</td>
<td>23</td>
<td>11</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>Propose Method</td>
<td>95.2%</td>
<td>93.5%</td>
<td>94.6%</td>
<td>95.2%</td>
<td>97.4%</td>
<td>94.2%</td>
</tr>
<tr>
<td>( \text{traj}+\text{acc}+\text{curv} )</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>9</td>
<td>23</td>
<td>6</td>
</tr>
<tr>
<td>( w )</td>
<td>14</td>
<td>6</td>
<td>13</td>
<td>6</td>
<td>24</td>
<td>12</td>
</tr>
</tbody>
</table>

Fig. 3. SAX-based action recognition on CMU mocap data. (a) Accuracy obtained for different camera view with their corresponding parameters. (b) For cam5, and using \( \text{traj}+\text{acc}+\text{curv} \) features, we test with different parameters and plot the accuracy obtained. The figure shows that different combinations of \( w \) and \( p \) do not greatly affect the overall accuracy.
The distance between two SAX representations of trajectories requires looking up the corresponding breakpoints in the breakpoint lookup table between each pair of SAX symbols, squaring them, summing them, taking the square root and finally multiplying by the square root of the compression rate ².

The distance, \( \text{dist}(\mathbf{r}, \mathbf{c}) \), values, between two SAX symbol values \( r \) and \( c \) is calculated by the following expression:

\[
\text{dist}(r, c) = \begin{cases} 
0 & \text{if } |r - c| \leq 1 \\
\beta_{\max(r,c)} - \beta_{\min(r,c)} & \text{otherwise}
\end{cases}
\]  

(6)

where \( \beta_{\max(r,c)} \) is the breaking point value \([35]\). The first case of Eq. (6) simply says that if the two symbols \( r \) and \( c \) fall in some region, or two consecutive regions, under the Gaussian curve then the distance between these two symbols will be zero. That is the distance between neighboring SAX symbols is zero. In the second case of Eq. (6), what \( \beta_{\min(r,c)} \) means, for example, is the breakpoint corresponding to the minimum of the two symbols \( r \) and \( c \). For example, if we refer to Fig. 2, the distance between symbol \( s \) and \( c \) would be \( \beta_{\max(s,c)} - \beta_{\min(s,c)} = \beta_s - \beta_c = -0.43 - (-0.43) = 0.86 \). For two action sequences, denoted by \( \mathcal{H}(\mathcal{I}) \) where \( m \) body joint trajectories are available, the total SAX distance between the two sequences is defined as:

\[
\text{MINDIST}(\mathcal{H}(\mathcal{I}), \mathcal{H}(\mathcal{J})) = \sum_{j=1}^{m} \text{dist}(\mathcal{H}_i(\mathcal{I}), \mathcal{H}_j(\mathcal{J}))
\]  

(7)

which is the sum of the distances between each corresponding pair of body parts.

5. Experiments and results

In this section we evaluate the SAX-based trajectory representation for the task of human action recognition. The input to the method is the location of the reference points throughout an action sequence, given in the form of \( xy \)-trajectories. For this purpose, we use trajectories of only 13 joints on the human body. We also experiment with the \text{vel}, \text{acc}, and \text{curv} features, as described above.

In the following we consider the Nearest Neighbor Classifier (NNC). We simply assign to test sequence \( \mathcal{H}(\mathcal{I}) \) the action label of the training sequence \( \mathcal{J} \) which minimizes distance \( \text{MINDIST}(\mathcal{H}(\mathcal{I}), \mathcal{H}(\mathcal{J})) \) over all training sequences. The distance \( \text{MINDIST} \) is as defined in (7).

We evaluate SAX-based action recognition on two public datasets. For all recognition experiments we report results for \( n \)-fold cross-validation and make sure the actions of the same person do not appear in the training and in the test sets simultaneously. In Section 5.1 we validate the approach in controlled multi-view settings using motion capture data. In Section 5.2 we demonstrate and compare the discriminative power of our method on a standard action dataset \([7]\). We compare our results with the state of the art methods using the same dataset.

5.1. Experiments with CMU MoCap dataset

In order to be able to compare our results, we adopt the same experimental setup as \([28]\); we use 3D mocap data from the CMU dataset (http://mocap.cs.cmu.edu). Trajectories of 13 reference points on the human body were projected to six cameras with pre-defined orientations with respect to the human body (see Fig. 4(a)). We have used 164 sequences in total corresponding to 12 action classes (\text{bend, cartwheel, drink, fjump, flystroke, golf, jjack, jump, kick, run, walk, walkturn}). For data from all views, the obtained accuracy results using the different features are shown in Fig. 3(a). We compare our results to the state of the art method on this.
The second row shows the results obtained by [28], and the subsequent rows show the accuracy obtained by our method using the traj, traj + curv, traj + vel, traj + acc, traj + vel + acc + curv, and traj + acc + curv, respectively. We have put the highest accuracy for each column in bold typeset. As the figure shows, we outperform [28] in four of the six views while using only traj, and five of the six views using the traj + curv feature, and have comparable accuracy in the remaining views. From this figure, we observe the vel feature does not contribute towards a higher accuracy, and the best overall results are obtained using just the traj + curv feature.

As discussed above in Section 1, the parameters \( w \) and \( p \) do not greatly affect the accuracy results. This can be seen in Fig. 3(b), which shows the recognition accuracy for different combinations dataset [28]. The second row shows the results obtained by [28], and the subsequent rows show the accuracy obtained by our method using the traj, traj + curv, traj + vel, traj + acc, traj + vel + acc + curv, and traj + acc + curv, respectively. We have put the highest accuracy for each column in bold typeset. As the figure shows, we outperform [28] in four of the six views while using only traj, and five of the six views using the traj + curv feature, and have comparable accuracy in the remaining views. From this figure, we observe the vel feature does not contribute towards a higher accuracy, and the best overall results are obtained using just the traj + curv feature.

As discussed above in Section 1, the parameters \( w \) and \( p \) do not greatly affect the accuracy results. This can be seen in Fig. 3(b), which shows the recognition accuracy for different combinations...
of w and p, ranging from 4 to 24 each. The average accuracy across this entire range is 92.7%. Thus, the recognition rate is fairly stable across different values of w and p.

Fig. 4(b) demonstrates the best action recognition results, obtained from cm4a5. The average accuracy here is 97.4%, compared to 93.9% obtained by [28]. The parameters for this setup are: w = 23 and p = 24, and the running time for the entire training and testing is approximately 24 s implemented on MATLAB. The action of jack is confused with walk, and cartwheel with jack, which is understandable as the actions are greatly similar.

Although we make no claims that the SAX-based representation is view-invariant, Fig. 5 demonstrates results of action recognition when training and testing on different views. For this setup, the parameters were set to w = 8 and p = 14, while using the traj + acc + curv features. As observed from the diagonal of the confusion matrix, the recognition accuracy is highest when training and testing on the same views while the best accuracy 95.2% is achieved for cm4a1. Interestingly, the recognition accuracy degrades gracefully, i.e. the regions near the diagonal still show good accuracy results, which indicates the method is somewhat robust to angle changes.

5.2. Experiments with Weizmann action dataset

To assess the discriminative power of our method on real video sequences, we apply it to a standard single-view video dataset with nine classes of human actions performed by nine subjects [7](see Fig. 6(top)). The dataset contains ninety-three action sequences, each having a resolution of 180 x 144. The nine different action classes are: bending down, jumping back, jumping in place, galloping sideways, running, walking, waving one hand and waving two hands. Given the low resolution of image sequences in this dataset, the trajectories were acquired by [31] via semi-automatic tracking of 13 body joints.

By using the NNC classifier, the best average recognition accuracy achieved by our method is 88.6% using the traj feature alone. The corresponding confusion matrices are illustrated in Fig. 6(bottom). Some errors occur when jump action is confused with the skip action, or when the walk is confused with the running action. Although the obtained accuracy is not state of the art on this dataset, it is comparable, if not better, than the methods that treat trajectory data as time series, i.e. 89.7% for [31], and 80% for [32]. Similarly, [16] recently introduced prototype trees to solve action recognition and reports an accuracy of 88.8% (when using motion information only). However, these methods are computationally expensive. Whereas higher recognition rates on this dataset have been reported, e.g., in [36], the main strength of our method is the fast and the simple solution: where the entire process of cross validation takes less than 50 s on MATLAB. Such an efficient response time and comparable accuracy makes our method suitable for various action recognition applications.

5.3. Time comparison

We compare speed of the proposed method with that of [28] and [31]. Since we are only interested in comparing the running times, in order to simulate conditions of working with a large dataset, we take the CMU dataset (164 sequences), and duplicate it uniformly until it reaches to 1312 files (i.e. 800% of the original size). We choose to compare with [28,31], for availability of code and ease of use, and because [28] is the only method, to our knowledge, that is using the particular CMU dataset for action recognition, thus enabling us to make a fair comparison.

The three methods are implemented on MATLAB (although some parts of [31] are coded in C), using an Intel Core2 2.4 GHz CPU. The final comparison results are shown in Fig. 7. To make a fair comparison, we start the three methods with input features already computed, so the figure compares only the classification (or the training/testing) phases of the two methods, while ignoring the feature extraction costs. We then perform leave-one-out cross validation for the different data sizes, as shown in the figure. For the case when all 1312 sequences are used, the time taken for method [28] is >15,000 s (250 min), compared to the 654.23 s (10.9 min) for our method, which is a considerable improvement.

6. Conclusion

In conclusion, we have proposed an effective and an efficient method to address the human action recognition problem by employing the SAX representation. The SAX representation tremendously reduces the dimensionality of the time-series data, thus reducing the strain on the memory space and computational power required. The conversion is simple and straight-forward, and the distance to compare two SAX representations is shown to be fast as well. We believe the proposed method is very practical and considering the existing solo dynamic very suitable for various action recognition applications. We test on two public datasets and report encouraging results. In addition to its simplicity and low computational costs, we have shown that the results obtained are comparable, if not better, to the existing methods.

References