Human Action Recognition Using Time Delay Input Radial Basis Function Networks

Davood Kalhor, Ishak Aris, Trifa Moaini and Izhal Abdul Halin
Dept. of Electrical and Electronic Engineering
Faculty of Engineering, Universiti Putra Malaysia
Serdang, Selangor, Malaysia
da.kalhor@gmail.com, ishak_ar@eng.upm.edu.my, t.moaini@gmail.com, izhal@eng.upm.edu.my

Abstract — This paper presents a fast, vision-based method for the problem of human action representation and recognition. The first problem is addressed by constructing an action descriptor from spatiotemporal data of action silhouettes based on appearance and motion features. For action classification, a new Radial Basis Function Network (RBF), called Time Delay Input Radial Basis Function Network (TDIRBF) is proposed by introducing time delay units to the RBF in a novel approach. A TDIRBF offers a few desirable features such as an easier learning process and more flexibility. The representational power and speed of the proposed method were explored using a publicly available dataset. Based on experimental results, implemented in MATLAB and on standard PCs, the average time for constructing a feature vector for a high-resolution video was just about 20 ms/frame (or 50 fps) and the classifier speed was above 15 fps. Furthermore, the proposed approach demonstrated good performance in terms of both execution time and overall performance (a new performance measure that combines accuracy and speed into one metric).

Keywords - action recognition; action representation; motion descriptor; neural network; radial basis function network

I. INTRODUCTION

Over the past decade, understanding human activities from a stream of video has received an increasing attention in the computer vision society due to a wide range of promising applications such as automatic visual surveillance [1] [2], robotics and Human-Computer Interaction (HCI) [3], elderly health care centers [4], training and analysis of sport activities [5], and art performances [6].

A variety of activity hierarchies and terminologies are in use by researchers in this area [7] [8]. In this paper, we adopted the one suggested in [8]: “action/motor primitives,” “actions,” and “activities.” Action primitives are defined as the smallest level of abstraction. In a higher level, an action is constructed from a sequence of these primitives. Finally, a series of actions makes the larger-scale event: activity.

Vision-based action recognition methods aim to recognize human actions from a sequence of observations captured by a single or multiple cameras. Understanding human actions is complex and challenging due to the following reasons: 1) the number of potential actions performed by humans is large; 2) human actions are dynamic in time and space and vary from person to person; 3) spatial and temporal characteristics of some actions are very close. Over the last two decades, extensive research efforts have been devoted to the problem of human action recognition and very impressive progress has been made. These researches have mostly focused on proposing/using new descriptors and classifiers to attain methods with high accuracy (improving discrimination power) either without or with one of the following abilities: robustness to parameters such as the camera view and occlusion [9] [10] [11] [12] [13] learning from few examples [14] [15], and ability to reject unseen data [15]. Early attempts to represent actions were mainly based on appearance [16], and it is still an important part of recent methods [17] [15] [18]. Motion provides very good cues about moving objects, and therefore, has been widely used for action description [19] [20]. In current action recognition systems, it plays an essential role in modeling actions [17].

In spite of these recent developments, there have been very few studies on the real-time action recognition [20]. This is mainly because the majority of existing methods, to achieve high recognition rate, have employed very complex features to describe human actions, and hence time-consuming methods. On the other hand, using simple features can improve the speed but undesirably causes accuracy reduction. Dealing with this paradox is the main
motivation of this work. The proposed strategy to resolve this problem is to find optimal descriptors, which have low dimensionality and time complexity while preserving required power of presentation to reach a high-accuracy rate. For this reason, this study uses spatiotemporal data of silhouettes for action representation. Early applications of silhouettes for action recognition were reported in the 1990s [16]. However, it is still used due to its efficiency (for a recent example, please see [21]). The other sources motivating this paper are as follows:

- Time Delay Radial Basis Function Networks (TDRBFs) has been successful applied to speech recognition [22].
- TDRBFs offer a few advantages such as the ability to cope with time series and being shift-invariant in time, good performance, fewer numbers of parameters compared with Time Delay Neural Networks (TDNNs), and simplicity and high speed in training [23].
- Despite reporting excellent performance for TDRBFs and their good characteristics, to the best of our knowledge, there is no report of using them for human action recognition.
- There have been studies showing that RBF networks are suitable for real-time applications [23], but it is not clear whether TDRBFs inherit this characteristic from them.

- TDRBFs require a simpler process of model selection, compared with TDNNs.

This paper presents a new method to represent and recognize human actions in video sequences. The processing flow of the suggested method is shown in Fig. 1. For action representation, this method uses spatiotemporal data of silhouettes and constructs a feature vector at each frame based on the center of mass (CoM), bounding box, and grid-based shape descriptors. Then an input sequence, represented by a sequence of these feature vectors, is classified as one of the several actions learned in a training phase. This paper proposes a TDIRBF for this classification task, and it introduces a new growing-based algorithm to train the network. A TDIRBF, in addition to preserving the desirable features of TDRBF, has a simpler structure and an easier learning process. Furthermore, it can be trained to meet different objectives (high speed and high accuracy), and hence applicable to a wider range of applications: flexibility.

The rest of this paper is organized in the following way. Section II briefly reviews the relevant literature to action representation and classification. Section III presents the proposed method for action representation. Section IV discusses the TDIRBF as a new architecture for action classification. Section V reports the experimental results and assesses the performance of the proposed method. Eventually, Section VI concludes the paper with some remarks and suggested directions for future work.
II. RELATED WORK

Due to the increasing need of autonomous systems to recognize human actions in various applications such as surveillance and HCI, a considerable amount of literature has been published on this topic within the last two decades. Since two important aspects of action recognition are action representation and classification, a summary of the associated works based on these two aspects is presented.

A. Action Representation

Using an appropriate approach to represent human actions plays a decisive role in the performance of action recognition systems. Therefore, a great deal of attention has been devoted to the development of methods representing human actions. These works can be categorized into a few groups such as appearance-based representation, shape-based representation, motion-based representation, and volume-based representation. In this paper, appearance-based and motion-based approaches are briefly discussed, which are more related to proposed work. For other categories, the interested readers are referred to a few good survey papers [24] [2] [8] [25] [26] [27], and the references cited therein.

1) Appearance-based Representation: It was the most common approach in the 1990s. Example features used in this group are gray-scale values of image pixels [28], statistical descriptor such as mean and variance [29], position and angles of important points and axes [30], grid-based descriptors of silhouettes [16]. Appearance-based descriptors are still in use, but usually along with another type of representation. For example, recently, Jiang, et al. [18] used an appearance-based representation in which edge features were computed using a distance transform [31] and log-polar bins. In another work, Tran et al. [15] computed the appearance features of silhouettes using a grid-based descriptor combined of rectangle and radial bins. Our method has two similarities with their work: we also take an appearance- and motion-based approach to represent human actions, and use the same grid-based shape features to capture the appearance of the silhouettes. However, our method has a few essential differences with theirs as will be described in the next section.

2) Motion-based Representation: Motion provides very good cues about moving objects, and therefore, has been widely used as a main source or complementary for action description. Image gradient and optical flow have been frequently used by researchers as the main tools to compute motion information, and thus sometimes these methods are referred as optical-based approaches. The main problem of this group is the large processing time required to compute optical flow despite using Lucas-Kanade algorithm [32], which is known as the fastest algorithm [33] for this purpose.

Two well-known examples of motion-based approaches are motion energy image (MEI) [34] and motion history image (MHI) [20]. MEIs are binary cumulative images indicating moving parts in the image sequences, but MHIs are functions to describe how recently movements have occurred. Venkatesh Babu and Ramakrishnan [35] has also made use of MHI. They constructed several features from MHI and Motion Flow History. To overcome the problem of view dependency, the MHI has been extended by Weinland et al. [36] and, more recently, by Roh et al. [10]. The motion-based methods have exploited optical flow via two different approaches. The first group has made direct use of optical flow [37], while the second group has used an extra representation layer in which flow images are represented by other sets of descriptors such as Zernike polynomials [38] and affine features [35]. Two other examples in this group are works by Efros et al. [39], and Wang and Mori [40], in which the motion features from optical flow were extracted in a few steps, including dividing the flow vector into its horizontal and vertical components and making four nonnegative channels of these components by half-wave rectification. Similarly, Tran, et al. [15], separated horizontal and vertical components of optical flow, but they integrated them over each bin of a radial grid. In a recent work, to represent human action in videos, Ali and Shah [19] proposed a set of kinematic features including divergence, vorticity, symmetric and asymmetric fields derived from optical flow. More recently, Seo and Milanfar [14] used shape-and-motion-based descriptors derived from space-time local steering kernels for action recognition. An essential benefit of this method is the ability to recognize actions using only one example. Furthermore, it does not require preprocessing steps such as foreground/background extraction and tracking.

We also use motion features for action description, but in contrast with the common approach in the literature, without using time-consuming techniques like optical flow. Instead, our method relies on the changes in the CoM and bounding box that can be extracted considerably fast.

B. Action Classification

Over the last few decades, a large number of classification algorithms have been developed. The most common classifiers for vision-based action recognition can be categorized into four groups: Dynamic Time Warping (DTW) [41] [42], Hidden Markov Models (HMMs) [43] [44], k-Nearest Neighbor (kNN) [20] [12] [15] [14], and Artificial Neural Networks (ANNs) [23] [45] [46] [47] [48]. Since the discussion about all these approaches is beyond the
scope of this paper, we only provide a general overview of the neural-based classifiers for action recognition. The interested reader can find a summary about the other three approaches in two good review papers [2], [25].

ANNs are biologically-inspired methods, which have been applied to a lot of real-world problems in various areas to classify and recognize patterns of data or to approximate linear and nonlinear functions. Many different types of ANNs have been proposed such as MLP, RBF, and SOM, to name a few.

Although MLPs are well suited to classification of static data, they cannot learn the temporal relations existing in time series such as speech and human actions. In a seminal work by Waibel et al. [49], this issue was addressed by introducing a TDNN in which each neuron has a tapped delay line (TDL) at its input. The presence of the TDL enables the net to simultaneously observe inputs applied at different times.

Two major characteristics of TDNNs, insensitivity to small variation in duration and also to small temporal misalignment of the input sequence, make them well-suited for human action recognition. In this context, Yang et al. [47] proposed a method to recognize hand gestures of ASL using a TDNN. As it is common, the network was trained by the BP algorithm. In [45], a slightly different network was proposed to interpret the human emotion in a dance performance. A TDNN has two major problems: (1) requirement of too many experiments to find optimal parameters of the network such as the size of sliding windows, number of neurons in the hidden layers, and the length of input sequences [47], and (2) “long training times” [22].

To overcome the difficulties in the TDNN, Berthold [22] took the advantage of RBF networks and introduced a TDRBF in three layers, the first layer is similar to TDNNs, but the second layer is composed of RBF units. A TDRBF can be trained using several training algorithms in two steps. In [23], a growing cell structure was used to train the RBF layer. Then the weights between RBF layer and the output layer were trained using the BP algorithm. Howell and Buxton [23] employed a simpler approach which is common when RBF networks are used for function approximation. They constructed the RBF layer by storing all training examples; a prototype was assigned to each example. Then the radius of each unit was obtained based on an iterative clustering technique proposed in [50]. The weights were calculated using a pseudo inverse solution. The main drawback of this algorithm is the huge memory requirement.

Despite reporting excellent performance for the TDRBF [22], the ability of this network for action recognition has not been examined except “...to distinguish the presence and direction of movement [detecting head movement from left to right and vice versa] in simple fixed sequences” [23].

There are few examples of using SOM for action recognition. For example, in a recent work, Huang and Wu [51] used SOM for clustering the feature space, and for each sequence created a trajectory based on the best matching unit clustered by the SOM. Then they employed dynamic programming to match the two sequence trajectories.

III. PROPOSED ACTION DESCRIPTOR

Representing actions with feature vectors is an important step in action recognition and usually involves several processes depending on the type and complexity of the features. The proposed action descriptor is extracted in a few steps. Each step is discussed in a separate section below.

A. Bounding Box

A human in an image can be limited to the region surrounded by a rectangle. This box can be obtained by projecting silhouettes onto the image plane axes as shown in Fig. 2.

B. Grid-based Shape Feature

This work adopted a type of grid descriptors [16] [52] from [15]. The process of extracting shape features is depicted in Fig. 3. The outcome is a grid with 72 (2x2x18) bins. At each time a feature is assigned to each bin by accumulating human body's pixels at the corresponding bin:

\[ f_i(t) = \sum_{(x,y) \in b_i} I(x,y,t), \]  

where \( f_i(t) \) is the assigned feature to bin \( b_i \) at time \( t \), and \( I(x,y,t) \) is the image value at a point \( (x,y) \) at time \( t \). Then
all these features are combined into a shape feature vector $f_c(t) = [f_1, f_2, ..., f_{22}]$. Preliminary results demonstrated that this descriptor is not sufficiently discriminative. Therefore, the motion features based on the mass center and the bounding box are also employed, as will be discussed in the next two sections.

C. Mass Center

The center of mass (CoM) for an object with uniform density is the centroid of the object. For rigid objects, having motion information of this point is sufficient to describe the movement of the object. Although the human body is non-rigid, this point still carries important cues about human motion. Therefore, this paper suggests the changes in the CoM within consecutive frames as another information channel to describe actions.

From classical physics [53], CoM for an object with continuous mass distribution can be expressed by

$$r_{cm} = \left( \int \int \int \rho(r)dV \right)^{-1} \int \rho(r)dV,$$  \hspace{1cm} (2)

where $r$ is a vector representing a point in 3D space, $\rho(r)$ denotes the mass density, and $V$ is the volume of the object. In this work, the concept of the CoM has been adapted for image representation. As the result, Eq. 1 for image $I$ with width $w$ and height $h$ in 2D Cartesian coordinates can be rewritten as follows

$$x_{cm} = \frac{1}{S} \int_{0}^{h} \int_{0}^{w} x \cdot I(x,y) dx dy,$$

$$y_{cm} = \frac{1}{S} \int_{0}^{h} \int_{0}^{w} y \cdot I(x,y) dx dy,$$  \hspace{1cm} (3)

where $S = \int_{0}^{h} \int_{0}^{w} I(x,y) dx dy$, $x_{cm}$ and $y_{cm}$ are the $x$ and $y$ components of the CoM. Finally, the mass center descriptor is defined as follows:

$$f_{ma}(t) = [dx_{cm}(t)/dt, dy_{cm}(t)/dt].$$  \hspace{1cm} (4)

D. Box Corners

Large scale human actions such as walking, running, and sitting up result in changing the bounding box over time. These changes follow patterns that can be used to differentiate actions. There have been studies that demonstrated the usefulness of bounding box as a complementary feature for object classification, e.g., the aspect ratio of the human for object classification [54]. However, here a different use of the bounding box is presented. Since the corners provide more information than the width, length, and aspect ratio, they are used for creating an extra channel representing motion. These points can be described by a 4D vector composed of $x_1, y_1, x_2$, and $y_2$ as shown in Fig. 4. Using these components, the box corner feature is formed as

$$f_c(t) = [dx_1(t)/dt, dy_1(t)/dt, dx_2(t)/dt, dy_2(t)/dt].$$  \hspace{1cm} (5)
E. Action Description

In the previous sections, the process of extracting three feature vectors from images was explained. The computations are simple and hence appropriate for real-time applications. To the best of our knowledge, the last two descriptors have not been used for action representation before this work. After normalization, these three feature vectors are stacked into a vector called the frame descriptor, i.e., $f = [f_1, f_2, f_3]$.

There are three different strategies to describe an action: 1) using all consecutive frames; 2) subsampling; and 3) using key frames or key points [55]. Furthermore, one may use whole sequence or a part of it. To obtain an efficient descriptor, this work follows a few guidelines. First, using all consecutive frames is too dense and often leads to redundancy in representation. Second, a representation with key frames can be too sparse and results in missing some important data. Third, a reliable decision can often be made using only some parts of an action instead of the entire sequence. Fourth, in practice, access to whole sequence of an action is often impossible. Consequently, the action descriptor is defined as follows:

$$F(t) = [f(t - t), f(t - 2t), ..., f(t)], \quad (6)$$

where $t$ is the length of the sequence, and $t_i$ is the sampling interval for frames of an action. A recent study [56] used almost similar approach, called snippets.

IV. ACTION RECOGNITION USING TDIRBF

To classify action, this paper introduces a new TDNN based on RBFs, called TDIRBF. This model was motivated by previous works such as [47] [22] [23]. Although the new model has some similarities to traditional time delay RBFs [22], there are clear distinctions between the two networks in both structure and features. The proposed network, in addition to keeping the desirable features of TDIRBFs, has a simpler structure and an easier learning process.

A. Architecture of the TDIRBF

The proposed TDIRBF consists of an input layer, a hidden layer, and an output layer (see Fig. 5). The input layer distributes the input data among the hidden layer units. Its hidden layer, similar to TDIRBFs, is composed of RBF units with a time delay (TD) line at its input. Using the TD units at this layer allows RBF units look at the input sequence over a time window ($w_i$). In contrast to TDIRBFs, this is the only use of TD units in the TDIRBF. The structure of this network has two other major distinctions from TDIRBFs: there is neither sliding window nor integration layer in the TDIRBF. In TDIRBFs, a sliding window scans the input sequence and enables RBF units to be fed by different portions of the input in time. This sliding window makes TDIRBFs shift-invariant in time, but it also imposes a costly training to find the optimal window. In the TDIRBF, it is removed to eliminate the problem of parameter selection. However, this property (shift-invariant) is maintained by using an appropriate training algorithm. Finally, the output layer is composed of competitive neurons. This layer does not need any training and this is another benefit of the TDIRBF over TDIRBFs.

A RBF is a radial kernel which maps an input to the output layer based on distance between the input and the kernel center. Between possible choices for RBFs, this work makes use of the Gaussian kernel. Thus the output of RBF unit $j$ is

$$\phi_j(t) = \frac{1}{(2\pi)^{D/2}} \exp\left(-\frac{1}{2}(x(t) - \mu_j)^T \Sigma_j^{-1} (x(t) - \mu_j)\right), \quad (7)$$

where $\mu_j$ and $\Sigma_j$ are the mean vector and covariance matrix of kernel $j$, respectively. In the RBF layer, several units represent each class. As it can be clearly seen from this equation, closer inputs to the mean result in higher outputs and vice versa.
B. Recognition in the TDIRBF

In the TDIRBF network, recognition is achieved through a simple and fast process. At each time instant \( t \), each RBF unit has access to a portion (of length \( w_t \)) of the input sequence. Each unit separately compares its input with its stored prototype (learned in a training phase) then generates an output depending on how close they are to each other, as described by Eq. 7. In the last layer, there exist \( Q \) competitive neurons, each of which represents a class. They receive their inputs only from units of the corresponding classes and choose the biggest output from each class as the representative of the respective class, i.e.,

\[
y_q = \max\{y_j | j \in \text{class}(q), q = 1,2,\ldots,Q\}.
\]  
(8)

Now, the classifier predicts the input sequence as the class with the maximum output value:

\[
q^* = \arg\max_q y_q.
\]  
(9)

C. Training

Training of the TDIRBF is considerably simpler than TDRBFs, because TDIRBFs have an extra parameter (the size of the integration window). Moreover, TDIRBFs involves two steps of learning, but the TDIRBF is trained in only one step since the output layer does not need any training.

Several options for training of the TDIRBF are possible. One can employ an approach similar to the one in [23] or [22]. Algorithm in [23] is simple, but impractical for large datasets and too slow for real time application. A more practical algorithm to train the RBF layer is “the growing cell structure” [22]. In this approach, new prototypes are added to the network when necessary, and the radii of all stored units are adjusted to avoid clashes between distinct classes. Application of this method for the proposed network here is limited due to two drawbacks. First, suggested algorithm in [22] is susceptible to poor performance for unseen data (i.e., generalization problem) because of using “signum” function. Second, there is no control over the process of storing prototypes in the network; it is merely imposed by the distribution of data.

In order to overcome these problems, this paper suggests a new algorithm by extending the original growing cell structure in [22]. First, a Gaussian function is used instead of “signum” to remove crisp boundaries and improve generalization. The second problem is addressed by introducing a parameter \( \sigma \) into training algorithm to control adding new prototypes to the network. This parameter is a real number between zero and one. Another desirable feature of \( \sigma \) is that it allows controlling the trade-off between speed and accuracy, and hence choosing an optimal training. This parameter is empirically obtained to meet the required objective (see the next section).

As already mentioned, the TDIRBF is shift-invariant in time despite this fact that there is no sliding window in its architecture. To achieve this property, it stores an adequate number of prototypes for each action in training phase. For a detailed description of training procedure, please see Algorithms 1-2.

---

**Algorithm 1. Human Action Recognition (HAR) algorithm**

1. procedure HAR(i,ax,a)
   
   // \( N_f(i, j) \): number of frames for example \( j \) of action \( i \)
   // \( N_{av}(i) \): number of available examples of action \( i \)

2. net \( \leftarrow \) initialize-TDIRBF(\( Q, t, \tau, \alpha \))
3. \( t \leftarrow 1, j \leftarrow 1, t \leftarrow 1 \)
4. \( f(t) \leftarrow \text{read-image}(i, t) \)
5. \( s(t) \leftarrow \text{extract-silhouette}(f(t)) \)
6. \( x(t) \leftarrow \text{feature-extraction}(s(t)) \)
7. \( net \leftarrow \text{TRAIN-TDIRBF}(net, x, t) \)
8. \( t \leftarrow t + 1 \) // increment time \( t \) to read a new frame
9. if \( t \leq N_f(i, j) \) then
10. \( \text{go to 4} \)
11. end if
12. \( j \leftarrow j + 1 \) // increment \( j \) to select a new example
13. if \( j \leq N_{av}(i) \) then
14. \( t \leftarrow 1, \text{and go to 4} \)
15. end if
16. \( i \leftarrow i + 1 \) // increment \( i \) to select a new action
17. if \( i \leq Q \) then
18. \( j \leftarrow 1, t \leftarrow 1 \text{and go to 4} \)
19. end if
20. save net
21. end procedure
Figure 6. Examples of video sequences and extracted silhouettes from UIUC dataset. (a) Action running. (b) Action jumping jack.

Algorithm 2. TDIRBF Training algorithm

1. procedure TRAIN_TDIRBF(net, x, t)
   // L(c): number of prototypes for action c
2. for k = 1; k ≤ L(c) do
3.   X(k) ← X(k - 1)
4. end for
5. X(1) ← x
6. j* ← argmax_j = 1 to c φ_j(X(1)) // compute φ_j using Eq. 7
7. if j* ∈ class c then
8.   if φ_j(X(1)) ≤ α then
9.     go to 22
10. else // add a new prototype to the network
11.   new Prototype ← add-prototype(X(1))
12.   L(c) ← L(c) + 1
13.   net.class(c).node(L(c)) ← new Prototype
14. end if
15. else
16.   for all j ≠ c do
17.     if φ_j(X(1)) > α then
18.       adjust prototype’s radii such that φ_j(X(1)) < α
19.     end if
20. end for
21. end if
22. end procedure

V. RESULTS AND DISCUSSION

A. Experimental Setup

For experiments, the UIUC human action dataset [15] was used. It contains 14 different actions (walking, running, jumping, waving, jumping jacks, clapping, jump from sit up, raise one hand, stretching out, turning, sitting to standing, sitting to standing, crawling, pushing up, standing to sitting) performed by eight actors; for simplicity in reporting results, each one is assigned a number (1-14), respectively. This dataset provides each action as an isolated sequence of high-resolution (1024×768 pixels) frames. In addition to color frames, their silhouettes are given. For a fair comparison, the provided silhouettes by the dataset without any enhancement were used. Fig. 6 shows example frames and the extracted silhouettes.

This work, similar to [15], was evaluated using leave-one-actor-out (L1AO) protocol, whose concept is depicted in Fig. 7. In the L1AO procedure, data are divided into two subsets: training and query (test) sets. Training set is obtained by excluding examples, of all actions, performed by
query actor from the whole data. The query set consists of examples of query actor performing query action. The two specified sets are used for one training-test experiment in two phases, in each of which the corresponding set is used. This process should be repeated for all other possible sets which can be obtained in a similar manner. The final prediction of the performance (or error) is calculated as the mean of separate predictions of the entire tests.

B. Experimental Results

In the first step of tests, the effects of the model parameters \((t_x, \tau, \alpha)\) on the performance (in terms of both speed and accuracy) were studied by performing many tests using different sets of parameters. The performance average for each set of parameters is summarized in Fig. 8. The tests were implemented in Matlab 7 and run on an Intel dual core 2.53 GHz PC with 4GB of RAM. The results demonstrated several issues as follows.

First, the average processing time for extracting the proposed features for each frame was 20 msec, hence suitable for real-time applications with a frame rate slower than 50 fps. Second, an optimal training according to required objective is achievable by choosing an adequate value for \(\alpha\): increase in \(\alpha\) improves accuracy while degrading speed, and vice versa. However, a logarithmic relation between accuracy and \(\alpha\) was observed for \(\alpha > 0.5\). This explains the fact that in the beginning of training when the network does not have sufficient information about actions, adding prototypes to the network significantly improves accuracy, but after this step it is not effective anymore. Consequently, the existence of parameter \(\alpha\) provides flexibility since it allows users to control the trade-off between speed and accuracy, and hence choosing an optimal training based on their objectives. To measure the performance of the classifier for different applications, two objectives were set:

- Objective 1: to achieve the highest possible speed.
- Objective 2: to achieve the best possible accuracy.

To meet each of these objectives, a different set of parameters was empirically set:
  - To meet objective 1: \(t_x = 4, \tau = 17, \alpha = 0.1\).
  - To meet objective 2: \(t_x = 2, \tau = 17, \alpha = 0.75\).

To measure a more realistic prediction of the model performance and avoiding the risk of biasing, further experiments were carried out using unseen data and the two mentioned parameter sets. The results are given in Table 1.

When choosing the first objective, the execution time for the classifier is approximately 0.064 second. This speed is almost enough for real-time applications with frame rates less than or equal to 15 fps. For objective 1, the average of recognition error is 9%. However, when the speed is not the main concern, the accuracy can be improved by choosing objective 2: the error rate is reduced to less than 6%. For distribution of error for each class please see the corresponding confusion matrix (Fig. 9). The main reasons for these confusions are the high degree of similarity for some actions and high level of self-occlusion for single-view
TABLE I. COMPARISON OF AVERAGE PERFORMANCE IN TERMS OF EXECUTION TIME, ACCURACY, AND OVERALL PERFORMANCE

<table>
<thead>
<tr>
<th>Method</th>
<th>Dataset</th>
<th>Execution time (second/frame)</th>
<th>Accuracy (%)</th>
<th>Overall performance (% × fps × pix)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2 = 1</td>
</tr>
<tr>
<td>Gorelick et al. [12]</td>
<td>Weizmann</td>
<td>0.600</td>
<td>N/A</td>
<td>&gt; 0.600</td>
</tr>
<tr>
<td>Tran et al. [15]</td>
<td>UIUC</td>
<td>170.600</td>
<td>N/A</td>
<td>&gt; 170.600</td>
</tr>
<tr>
<td>Our method (Obj. 1)</td>
<td>UIUC</td>
<td>0.020</td>
<td>0.0465</td>
<td>0.8545</td>
</tr>
<tr>
<td>Our method (Obj. 2)</td>
<td>UIUC</td>
<td>0.020</td>
<td>0.4219</td>
<td>0.4419</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>act 1</th>
<th>act 2</th>
<th>act 3</th>
<th>act 4</th>
<th>act 5</th>
<th>act 6</th>
<th>act 7</th>
<th>act 8</th>
<th>act 9</th>
<th>act 10</th>
<th>act 11</th>
<th>act 12</th>
<th>act 13</th>
<th>act 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>99.54</td>
<td>81.06</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>7.97</td>
<td>33.95</td>
<td>81.06</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>95.28</td>
<td>94.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>3.77</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>81.97</td>
<td>94.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>81.97</td>
<td>94.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>40.00</td>
<td>7.38</td>
<td>2.68</td>
<td>0.00</td>
<td>0.00</td>
<td>85.23</td>
<td>67.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>40.00</td>
<td>7.38</td>
<td>2.68</td>
<td>0.00</td>
<td>0.00</td>
<td>85.23</td>
<td>67.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 9. Confusion matrix with model parameters $t = 2, \tau = 17, \alpha = 0.75$.

silhouettes. For instance, in actions 'raise one hand' and 'waving', the movement of the hand for several frames is mainly covered with torso and head.

C. Comparative analysis

To assess the significance of this work, it has been compared with the two state-of-the-art methods: Space-Time Shapes [12] and Metric Learning [15]. The results of our method and the other two works are summarized in Table 1. Comparisons have been drawn based on execution time (or speed), accuracy, and overall performance.

Performance evaluation using either speed, without considering accuracy, or accuracy without speed, is not rational. Furthermore, comparing performance of different methods using the two factors, separately, is not tangible enough. Therefore, this paper proposes a new metric for performance measurement by combining accuracy and speed into a new measure, named overall performance, as follows:

$$0. Prf. = acc \times (sp \times w \times h)^4,$$  (10)

where $acc$ and $sp$ denote accuracy (in %) and speed (in fps), $w$ and $h$ are width and height of images in pixels, and $\lambda$ is a binary parameter ($\lambda = 0$ or $\lambda = 1$) such that for applications in which speed is not important it is zero, otherwise one; for $\lambda = 0$, the overall performance is identical to accuracy. Since the speed is a subjective issue influenced by the size of images, this factor has been also added to the formula. In this way, comparing methods in which different image sizes have been used for test becomes possible. The results given in Table 1 under the column of overall performance have been calculated using Eq. 10.

The results show that the proposed method outperforms "Metric Learning" [15], one of state-of-the-art methods, in terms of execution time and overall performance: 1) the proposed action descriptor is 8500 = 170/0.02 times faster,
and 2) a significant improvement, approximately $1800 \times 843762427/456432$ times increase, in overall performance is observed. Similarly, the method presented in this paper shows much better performance than Space-Time Shapes [12] in terms of speed and overall performance. For instance, the proposed action descriptor in this paper is about 3000 $(0.6/0.02)\times(1024\times678)/(110\times70)$ times faster and overall performance is improved more than 600 times $(600 < 843.762.427/1255.485)$. In addition, the performance of our method based on accuracy is almost comparable to these state-of-the-art methods, particularly when the network is trained based on objective 2.

-VI. CONCLUSION AND FUTURE WORK

This paper presented a fast method to understand human actions from video sequences. To achieve this goal, a new action descriptor, based on appearance and motion, and an action classifier (TDIRBF) were proposed. The TDIRBF has a few advantages over TDRBFs, including simplicity (in structure and training) and flexibility (ability to train for different objectives). Furthermore, the application of the proposed method is not limited to a specific type of actions like cyclic and acyclic; instead, it can be used for the both groups. Based on empirical evidence, as summarized in Table I and discussed in Section 5, the suggested descriptor is appropriate for real-time applications with a frame rate slower than 50 fps. Comparison with the two state-of-the-art methods in the literature demonstrated that the proposed method improves very significantly upon those works in terms of speed and overall performance while preserving an average accuracy above 90%. The best obtained accuracy for the UIUC dataset was 94.5%. For this setting, the overall system was more than one hundred times faster than both Metric Learning and Space-Time Shapes.

As discussed in Section 5, high degree of self-occlusion, which is a major drawback of single-view silhouettes, causes a significant increase in the similarity between some actions. This was the most serious obstacle to the perfect performance of the suggested approach by this paper in terms of accuracy (i.e., 100%). One possible avenue for future work is to employ an algorithm to choose a proper view (from multiple cameras) of actions before being applied to the proposed method in this paper.

REFERENCES
