Human Activity Encoding and Recognition Using Low-level Visual Features

Zheshen Wang and Baoxin Li
Department of Computer Science and Engineering
Arizona State University
{zheshen.wang, baoxin.li}@asu.edu

Abstract

Automatic recognition of human activities is among the key capabilities of many intelligent systems with vision/perception. Most existing approaches to this problem require sophisticated feature extraction before classification can be performed. This paper presents a novel approach for human action recognition using only simple low-level visual features: motion captured from direct frame differencing. A codebook of key poses is first created from the training data through unsupervised clustering. Videos of actions are then coded as sequences of super-frames, defined as the key poses augmented with discriminative attributes. A weighted-sequence distance is proposed for comparing two super-frame sequences, which is further wrapped as a kernel embedded in a SVM classifier for the final classification. Compared with conventional methods, our approach provides a flexible non-parametric sequential structure with a corresponding distance measure for human action representation and classification without requiring complex feature extraction. The effectiveness of our approach is demonstrated with the widely-used KTH human activity dataset, for which the proposed method outperforms the existing state-of-the-art.

1 Introduction

Being able to recognize human activities in video is a critical capability of an intelligent vision system in applications such as video surveillance, content-based video retrieval, and human-robot interaction. On this task, human vision still outperforms any existing automatic techniques. Humans tend to describe, remember, perform and compare an action as a sequence of key poses of the body. For example, in both choreography and sports, body movements are usually described as an ordered sequence of key poses. Most of the existing vision techniques attempt to mimic this to certain degree, explicitly or implicitly, by modeling and classifying human actions based on key poses and their orders. Unfortunately, such techniques typically reply on sophisticated feature extraction (e.g., explicit detection and tracking of body parts, or doing so implicitly by complex representation and detection of the body motion through high-dimensional spatial-temporal features), which are a challenging task on its own especially considering the wide variability of acquisition condition.

One interesting observation is that, humans can casually take a glimpse of a video clip (even at a critically-downsampled resolution) and recognize the underlying action correctly without careful thinking. The recognition can also be done accurately with optical flow only without the original video clip. This may suggest that only very rudimentary visual features are necessary for action recognition. In this work, we propose a novel approach to video-based recognition of human actions using simple visual features: motion captured in direct frame differencing. In the proposed approach, a codebook of atom poses is first formed using unsupervised clustering of difference frames from videos of various actions. A video is then coded based on its corresponding sequence of atom poses augmented with other discriminative attributes such as the duration of the poses, resulting in a stream of (*pose*, *attributes*) couples. Further, we introduce a novel *weighted-sequence distance* (WSD) measure for comparing the similarity between two sequences, based on not only the constituent poses but also their attributes plus the global structure of the pose sequences. The compact metric WSD is further embedded into a Support Vector Machine (SVM) classifier as a kernel for classifying the coded video clips. As the result, the proposed approach does not rely on sophisticated feature extraction but rather direct frame differencing and thus it is more universally applicable for various acquisition conditions such as different imaging sensors or illumination conditions.

The rest of this paper is organized as follows. We first review related literature in Section 2. The action coding scheme and the WSD are proposed in Section 3 and 4 respectively. Classification strategies are described in Section 5. In Section 6, experiments with a real-world dataset are presented and analyzed. We conclude in Section 7 with a brief discussion on future directions.

2 Related Work

Existing approaches for human action recognition can be generally classified into two categories: graphical-model-based approaches and bag-of-words approaches. The former utilizes a graphical structure consisting of states and models the activities as sequential transitions between the states. Hidden Markov Model (HMM) [Niu and Abdel-Mottaleb, 2005] is one of the most commonly-used examples in this category. Other sophisticated extensions include Abstract HMM [Bui and Venkatesh, 2002], Hidden Permutation Model [Bui et al., 2008], Relational Markov Networks [Liao, 2005], etc. While being capable of modeling temporal transitions in a sequence, these methods usually suffer from an excess of parameters and overfitting of the data with insufficient training samples. In addition, the rigid structure also constrains its flexibility of dealing with structural variations among different sequences. Bag-of-words approaches have become popular in recent years. Methods in this category view the activity recognition problem as a standard classification problem which involves two stages: feature extraction and classification. Typically, the feature extraction step is very complex (such as spatial-temporal word feature proposed in [Dollar et al., 2005]), and additional pre-processing steps or extra information are often required (e.g., single action circle segmentation [Schindler and Gool, 2008], accurate alignment [Wong et al, 2007; Kim et al., 2007], accurate scale information [Liu and Shah, 2008], etc.). Often, the feature vectors are treated as independent data points in the classification stage (e.g., [Niebles et al., 2008]). Therefore, in general, these approaches simplify temporal sequences into a set of independent features and thus while *local* temporal information is encapsulated in the extracted spatial-temporal descriptors, *global* temporal information is not utilized. Nowozin et al. proposed a discriminative subsequence mining approach [Nowozin et al., 2007] which employs spatial-temporal features provided by [Dollar et al., 2005] and further formed a sequential representation of the original videos. However, it does not achieve noticeable improvements over previous approaches.

The proposed approach attempts to incorporate both local spatial-temporal information (through unsupervised clustering of motion-capturing difference frames in forming the key poses) and global temporal structure (through a novel WSD metric for comparing sequences of poses and an SVM classifier based on it) into a unified action coding and classification scheme.

3 Action Coding

In this section, we present the action coding scheme based on the (*pose*, *attributes*) couples, which is termed as super-frames. Formally, a super-frame f is defined as a 2-tuple (c, w) which consists of an atom pose c and its attributes w. The set of atom poses form a codebook. Each frame of a video is first assigned a codeword c and then adjacent frames with the same codeword are merged with the attribute w denoting the duration of the same codeword. With this strategy, the coded sequence reflects the local spatial-temporal information through the codewords (which are based on frame differencing) while retaining the global temporal order of the original sequence. This results in a compact yet descriptive representation of the original video clip. Details of codebook learning and sequence coding are elaborated in the following sub-sections.

3.1 Codebook Learning

We create a codebook from the training data through clustering difference frames, computed as the difference between two nearby frames (not necessarily consecutive ones if the video frame rate is high and thus the motion magnitude is not large between consecutive frames). The distance measure for clustering is based on Euclidean distance between two difference frames. For simplicity, we will still call the difference frames as "frames". The centroids from the clustering stage are kept as atom poses in the codebook. It is worth emphasizing that, although we use the term "atom pose" for the codeword, the centroids are not original video frames but rather a representation of the difference frames, which capture the local spatial-temporal variation of the underlying actions. (Another reason we use difference frames instead of the original frames is that they may naturally be robust to cluttered background since the difference is taken between nearby frames, which presumably have similar clutter signatures, assuming a decent frame-rate.) A learning approach automatically covers variations present in the training set, such as scale changes. Figure 1 illustrates some sample atom poses and their corresponding video frames.

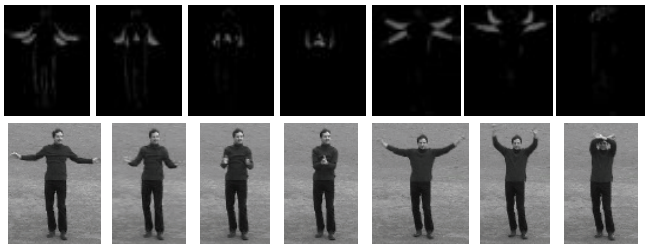

Figure 1: An illustration of atom poses (the top row) and their corresponding video frames (the bottom row).

Like any unsupervised clustering scheme, the choice of the number of clusters is an issue. To address this problem, we introduce a distance matrix which records pair-wise Euclidean distances (normalized to [0, 1]) among all codewords in the codebook. The distance matrix is then taken into consideration in computing the weighted-sequence distance between two super-frame sequences. More details are discussed in Section 4.

3.2 Video Coding Based on Super-frames

With an established codebook, we code a given video clip by the following three steps: (1) Assign a codeword to each frame based on similarities between the frame and each codeword; (2) Group adjacent frames with the same codeword

and store the duration of the codeword in the attribute, resulting in a super-frame; (3) Store all the super-frames in their original temporal order in the video.

Figure 2 illustrates two coded sequences using key poses in Figure 1.

$(c_i^{(1)}, w_i^{(1)})$: (A, 4) (B, 5) (C, 4) (D, 5) (C, 6) (B, 6)

(a) Super-frame sequence $s^{(1)}$ for a "hand clapping" clip

$(c_j^{(2)}, w_j^{(2)})$: (A, 6) (F, 7) (G, 8) (F, 5) (E, 4)

(b) Super-frame sequence $s^{(2)}$ for a "hand waving" clip

Figure 2: Two examples of coded sequences (Weights here are shown as the number of frames each group includes. In the experiments, we use normalized frame numbers as weights.)

4 Weighted-sequence Distance

After a video clip is coded as a super-frame sequence, action recognition boils down to classification of the super-frame sequences, for which a measure of the distance between two data points (super-frame sequences) is needed. In this section, we propose a generalized version of the classical Levenshtein Distance (also known as String Edit Distance [Levenshtein, 1966]), which takes both the attribute and the character (codeword) distances into consideration.

String distance/similarity problems widely appear in many areas of computer science. For example, in Web search, string matching/text comparison is one of the basic problems for text-based retrieval (e.g., [Crochemore and Rytter, 1994]; [Cancedda et al., 2003]); in computational biology, string matching techniques are often used for comparing biological patterns (e.g., [Leslie et al., 2004]). Edit distance, a basic string similarity metric, is defined as the minimum number of operations (including *Copy/Substitution*, *Insertion* and *Deletion*) required for turning one string to the other [Levenshtein, 1966]. Typically, fixed costs are assigned to each operation respectively in computing the overall cost of a series of editing operations. This formulation assumes that characters are of equal importance and that distance between two characters is binary (either "same" or "different"). However, in our problem of comparing super-frame sequences, these assumptions are no longer reasonable.

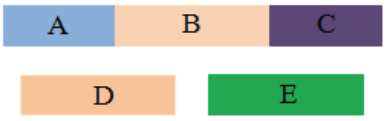

Figure 3: An illustration of distances between two characters.

Firstly, "characters" in a super-frame sequence are atom poses with corresponding weights (which may reflect the significance of the atom pose). Intuitively, operations on a crucial atom pose (e.g., one lasting for a longer period) should cost more than on a less important one. For example, deleting a super-frame of significant length during comparison should result in a large cost for a relative short video.

Secondly, the similarity between the atom poses varies and thus in operations, such as *Substitution,* the cost of the operation relies on what to use for the replacement. For example, in Figure 3, we assume that the distance between two characters equals the color difference between the corresponding bars. Obviously, to substitute "B" in the "ABC" sequence, "D" would cost less than using "E", since the color of "D" is much closer to "B" than "E". As mentioned earlier, such information is kept in a distance matrix in the codebook creation step and thus we should be able to systematically address such issues.

In the following sub-sections, we propose a weighted-sequence distance (WSD), which is able to address both of the above practical issues.

4.1 WSD for Super-frames

We define a weighted character a (e.g., the super-frame f in our problem) as a 2-tuple (c, w) which consists of the label c (e.g., the codeword in our super-frame formulation) and its weight w. Then a weighted string can be written as

$$s = \{a_1, a_2, ..., a_n\},\ a_i = (c_i, w_i),\ i = 1, ..., n \quad (1)$$

where n is the number of characters in s. Assume that we have two weighted strings $s^{(1)}$ and $s^{(2)}$:

$$s^{(1)} = \{a_1^{(1)}, a_2^{(1)}, ..., a_{n_1}^{(1)}\},\ a_i^{(1)} = (c_i^{(1)}, w_i^{(1)}),\ i = 1, ..., n_1 \quad (2)$$

$$s^{(2)} = \{a_1^{(2)}, a_2^{(2)}, ..., a_{n_2}^{(2)}\},\ a_j^{(2)} = (c_j^{(2)}, w_j^{(2)}),\ j = 1, ..., n_2 \quad (3)$$

A $(n_1 + n_2) \times (n_1 + n_2)$ symmetric matrix D_c (with zero elements on the diagonal) records pair-wise distances of the vocabulary (range of values in D_c is [0, 1]):

$$\{c_1^{(1)}, c_2^{(1)}, ..., c_{n_1}^{(1)}, c_1^{(2)}, c_2^{(2)}, ..., c_{n_2}^{(2)}\} \quad (4)$$

Then the weighted-sequence distance between $s^{(1)}$ and $s^{(2)}$ is defined as the sum of costs caused by operations for turning $s^{(1)}$ to $s^{(2)}$:

$$D^{WSD}(s^{(1)}, s^{(2)}) = \sum_{l=1,...,L} Cost_l \quad (5)$$

in which L is the number of operations involved; $Cost_l$ denotes the required cost for the l^{th} operation. Three types of editing operations: *Substitution*, *Insertion* and *Deletion* and corresponding costs are defined as follows:

$$Cost^S(a^{(1)}, a^{(2)}) = \alpha_1 \cdot |w^{(1)} - w^{(2)}| + \min\{w^{(1)}, w^{(2)}\} \cdot D_c(c^{(1)}, c^{(2)}) \quad (6)$$

$$Cost^I(a^{(1)}, a^{(2)}) = \alpha_2 \cdot w^{(2)} \quad (7)$$

$$Cost^D(a^{(1)}, a^{(2)}) = \alpha_3 \cdot w^{(1)} \quad (8)$$

where $Cost^S(a^{(1)}, a^{(2)})$ denotes the cost of substituting $a^{(1)}$ in $s^{(1)}$ by $a^{(2)}$ from $s^{(2)}$. $Cost^I(a^{(1)}, a^{(2)})$ indicates inserting $a^{(2)}$ to

$s^{(1)}$. $Cost^D(a^{(1)}, a^{(2)})$ means deleting $a^{(1)}$ from $s^{(1)}$ when comparing $a^{(1)}$ and $a^{(2)}$. α_1, α_2 and α_3 are weights for balancing the components. We use $\alpha_1 = \alpha_2 = \alpha_3 = 1/k$ in our experiments, where k is the codebook size.

With the above definitions, we propose an algorithm as shown in Figure 4 for computing the WSD by extending the conventional Edit Distance algorithm based on dynamic programming. Landau and Vishkin [Landau and Vishkin, 1989] have shown that classical edit distance problem can be solved in $O(mn)$ time using dynamic programming. Since our generalized version does not change the structure of the original algorithm, it still maintains the same computational complexity.

Algorithm: *Weighted-Sequence Distance (WSD)* computes the weighed-sequence distance between two weighted-sequences $s^{(1)}$ and $s^{(2)}$ with given distance matrix D_c.

Input: Weighed-sequence $s^{(1)}$ and $s^{(2)}$ and distance matrix D_c.

1 **if** $n_1 = 0$
2 | **return** $\sum_{j=1,...,n_2} w_j^{(2)}$;
3 **end**
4 **if** $n_2 = 0$
5 | **return** $\sum_{i=1,...,n_1} w_i^{(1)}$;
6 **end**
7 Construct an empty matrix M ;
8 Initial the first row of M as $w_1^1, (w_1^1 + w_2^1), ..., \sum_{i=1,...,n_1} w_i^1$;
9 Initial the first column of M as $w_1^2, (w_1^2 + w_2^2), ..., \sum_{j=1,...,n_2} w_j^2$;
10 **while** $i \le n_1$ **do**
11 | **while** $j \le n_2$ **do**
12 | | Compute the following costs respectively:
13 | | $Cost^{copy/substitution} = Cost^S(a_i^{(1)}, a_j^{(2)})$;
14 | | $Cost^{insertion} = Cost^I(a_i^{(1)}, a_j^{(2)})$;
15 | | $Cost^{deletion} = Cost^D(a_i^{(1)}, a_j^{(2)})$;
16 | | Let $M(i+1, j+1) =$
17 | | $\min\{M(i, j+1) + Cost^{insertion},$
18 | | $M(i+1, j) + Cost^{deletion},$
19 | | $M(i, j) + Cost^{copy/substitution}\}$;
20 | **end**
21 **end**
22 $D^{WSD} = M(n_1 + 1, n_2 + 1)$;
23 **return** D^{WSD}.

Figure 4: Weighed-sequence distance (WSD) algorithm.

4.2 Properties of WSD

In this sub-section, we analyze the properties of the WSD defined in the previous subsection, which serve to verify that WSD is a desired distance measure for comparing two weighted-sequences.

Property 1 (Zero Copy Cost): Assume that $a^{(1)} = (c^{(1)}, w^{(1)})$ and $a^{(2)} = (c^{(2)}, w^{(2)})$ are two weighted characters from sequence $s^{(1)}$ and $s^{(2)}$, respectively. If $D_c(c^{(1)}, c^{(2)}) = 0$ and $w^{(1)} = w^{(2)}$, then the substitution cost for $a^{(1)}$ and $a^{(2)}$ in turning $s^{(1)}$ and $s^{(2)}$ equals zeros:

$$Cost^S(a^{(1)}, a^{(2)}) = 0 \tag{9}$$

This property can be easily proved by inserting $D_c(c^{(1)}, c^{(2)}) = 0$ and $w^{(1)} = w^{(2)}$ into Eq. (6). It means that when the two weighted characters are equal, *Substitution* is reduced to *Copy* and its cost becomes zero.

Property 2 (Commutative Law): Assume that $s^{(1)}$ and $s^{(2)}$ are two weighted-sequences and the weights of *Insertion* and *Deletion* costs (α_2 and α_3 in Eq. (7) and Eq. (8)) are equal. The cost of turning $s^{(1)}$ to $s^{(2)}$ equals the cost of turning $s^{(2)}$ to $s^{(1)}$:

$$D^{WSD}(s^{(1)}, s^{(2)}) = D^{WSD}(s^{(2)}, s^{(1)}) \tag{10}$$

Proof: If we can prove that the costs of any operations for turning $s^{(1)}$ to $s^{(2)}$ equal the cost of those for turning $s^{(2)}$ to $s^{(1)}$, Property 2 is proved. *Substitution* cost defined in Eq. (6) obeys Commutative Law. For *Insertion* and *Deletion*, inserting $a^{(2)}$ to $s^{(1)}$ in turning $s^{(1)}$ to $s^{(2)}$ is essentially the same as deleting $a^{(2)}$ from $s^{(2)}$ in its inversed process—turning $s^{(2)}$ to $s^{(1)}$. According to Eq. (7) and (8), we have

$$Cost^I(a^{(1)}, a^{(2)}) = Cost^D(a^{(2)}, a^{(1)})$$

Similarly, we can obtain

$$Cost^D(a^{(1)}, a^{(2)}) = Cost^I(a^{(2)}, a^{(1)})$$

Thus Eq. (10) is always held.

For properties 3-6, we assume there are three weighted-sequences $s^{(1)}$, $s^{(2)}$ and $s^{(3)}$ which are of the same length.

Property 3 (Character Variation): If all corresponding weights of three sequences are equal

$$w_i^{(1)} = w_i^{(2)} = w_i^{(3)}, i = 1, .., n$$

while there are p characters in $s^{(2)}$ and q characters in $s^{(3)}$ which are different from the corresponding characters in $s^{(1)}$ and the corresponding character distances are all equal to 1:

$$D_c(c_{x_j}^{(1)}, c_{x_j}^{(2)}) = 1,\ j = 1, ..., p,\ x_j \in [1, n]$$

$$D_c(c_{x_k}^{(1)}, c_{x_k}^{(3)}) = 1,\ k = 1, ..., q,\ x_k \in [1, n]$$

and $p > q$, then $D^{WSD}(s^{(1)}, s^{(2)}) > D^{WSD}(s^{(1)}, s^{(3)})$.

Property 4 (Order Variation): Assume that $s^{(1)}$, $s^{(2)}$ and $s^{(3)}$ consist of the same set of elements (tuples of characters and corresponding weights), but with different order. If there are p and q elements in $s^{(2)}$ and $s^{(3)}$ respectively, which are different from the corresponding elements in $s^{(1)}$:

$$a_{x_j}^{(1)} \neq a_{x_j}^{(2)}, j = 1,...,p, x_j \in [1,n]$$

$$a_{x_k}^{(1)} \neq a_{x_k}^{(3)}, k = 1,...,q, x_k \in [1,n]$$

and

$$p > q,\ a_{x_r}^{(2)} = a_{x_r}^{(3)}, r = 1,...,q$$

then $D^{WSD}(s^{(1)}, s^{(2)}) > D^{WSD}(s^{(1)}, s^{(3)})$.

Property 5 (Weight Variation): If all corresponding characters of three sequences are equal

$$c_i^{(1)} = c_i^{(2)} = c_i^{(3)}, i = 1,..,n$$

while there are p weights in $s^{(2)}$ and $s^{(3)}$ respectively, which are different from the corresponding ones in $s^{(1)}$:

$$w_{x_j}^{(1)} \neq w_{x_j}^{(2)}, w_{x_j}^{(1)} \neq w_{x_j}^{(3)}, j = 1,...,p,\ x_j \in [1,n]$$

and

$$| w_{x_j}^{(1)} - w_{x_j}^{(2)} | > | w_{x_j}^{(1)} - w_{x_j}^{(3)} |, j = 1,...,p, x_j \in [1,n]$$

then $D^{WSD}(s^{(1)}, s^{(2)}) \geq D^{WSD}(s^{(1)}, s^{(3)})$.

Property 6 (Character Distance Variation): If all corresponding weights of three sequences are equal

$$w_i^{(1)} = w_i^{(2)} = w_i^{(3)}, i = 1,..,n$$

while there are p characters in $s^{(2)}$ and $s^{(3)}$ respectively, which are different from the corresponding characters in $s^{(3)}$ and

$$D_c(c_{x_j}^{(1)}, c_{x_j}^{(2)}) > D_c(c_{x_j}^{(1)}, c_{x_j}^{(3)}), j = 1,...,p, x_j \in [1,n]$$

then $D^{WSD}(s^{(1)}, s^{(2)}) \geq D^{WSD}(s^{(1)}, s^{(3)})$.

Properties 3-6 can all be proved formally but we skip the proofs due to the space limitation. These properties show that when variations involved are increased, WSD between the reference sequence and the new created sequence raises monotonically. They coincide with our heuristic requirements in terms of characters, weights of characters, distances of characters and sequential order of characters for measuring the distance between two weighed-sequences.

5 Weighed-sequence Classification

With the distance measure WSD proposed in Section 4, classification algorithms can be designed for the weighed-sequences (the super-frame sequences). In our experiment, we use both 1-Nearest Neighbor (1-NN) strategy based on WSD and an SVM classifier [Cristianini and Shawe-Taylor, 2000] with our WSD kernel.

For non-numerical data, one possible solution of using SVM is revising non-numerical data to numerical (e.g., using binary digits to represent a categorical attribute [Chang and Lin, 2001]). In addition, it can be solved by using sequence based kernels, such as the string kernel used for text documents [Cancedda et al., 2003], sequence kernels for speaker recognition [Campbell, 2001] or protein classification [Leslie et al., 2004]. However, to the best of our knowledge, none of existing string/sequence kernels is able to deal with weighted-sequence. In this paper, based on our proposed weighted-sequence distance, we define a WSD kernel function as

$$\exp(-\gamma \cdot D^{WSD}(s^{(1)}, s^{(2)}, D_c)),\ \gamma > 0 \qquad (11)$$

in which γ is a model parameter. In our experiments, we use LibSVM [Chang and Lin, 2001] with kernels computed from our kernel function defined in Eq. (11). Model parameter γ is selected from an n-fold cross validation on the training set. Since the kernel matrix is not always positive semi-definite, to guarantee a global optimum in SVM, we revise the kernel matrix through shifting all the eigen values by a positive constant [Roth et al., 2003]. The constant is set as the absolute value of the minimum eigen value in our experiments.

6 Experiments

In order to verify the effectiveness of our proposed algorithm, we performed experiments with a real-world human activity recognition dataset.

The KTH human activity dataset [Schuldt et al., 2004] contains 6 human actions (walking, jogging, running, boxing, hand waving and hand clapping) performed by 25 subjects in 4 different scenarios: outdoors, outdoors with scale variation, outdoors with different clothes and indoors (599 video clips available in total with a 25 fps frame rate). Figure 5 shows some sample frames from the 6 actions under 4 scenarios. We can see that this dataset covers subject variation, appearance variation, scale variation, illumination variation, and action execution variation, and thus it is deemed as very challenging.

Figure 5: Sample frames from KTH dataset: row—action, column—scenario.

In our experiments, we first compute frame differences for the first 204 frames in each sequence with a step size 4. (200 difference frames obtained for each sequence.) In order to reduce dimension, we crop out the human body regions by using an 80 pixel $\times$ 100 pixel bounding box. For hand motion sequences without scale variations, we manually set a fixed cropping region for the first frame of each sequence and use it for the entire sequence; for the remaining sequences, we estimate the center of the cropping region by using the center of the foreground pixels in the difference frames. We further sub-sample the sequences spatially to $1/4$ of the original size and temporally by a factor of 3, resulting in 67 difference frames for each video clip to be used as input for classification. Compared to the strong assumptions required in some other work (discussed in the related work and the following experiment analysis part), our preprocessing steps are by design very simple and the obtained results are still quite noisy in reality.

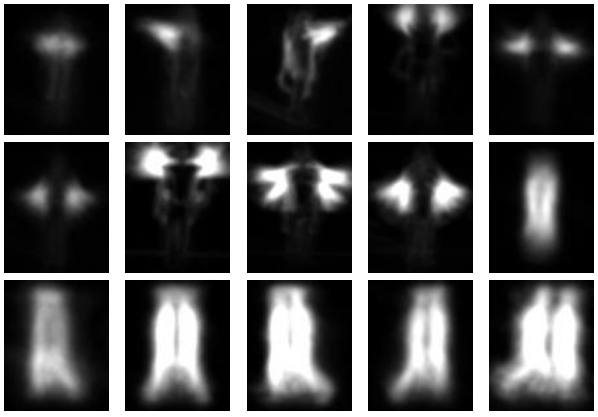

Figure 6: Samples of obtained atom poses.

%	Box	Clap	Wave	Jog	Run	Walk
Box	**88**	9	3	0	0	0
Clap	14	**81**	3	0	0	2
Wave	6	11	**83**	0	0	0
Jog	0	0	0	**70**	22	8
Run	0	0	0	27	**73**	0
Walk	0	0	0	8	2	**91**

Table 1: Results using 1-NN classifier with WSD measurement.

%	Box	Clap	Wave	Jog	Run	Walk
Box	**94**	6	0	0	0	0
Clap	9	**91**	0	0	0	0
Wave	6	8	**86**	0	0	0
Jog	0	0	0	**80**	8	13
Run	0	0	0	16	**84**	0
Walk	0	0	0	8	0	**92**

Table 2: Results using SVM classifier with WSD kernel.

In the codebook creation step, we set the cluster number as 30 for our experiments. As mentioned in the previous part, we take the distances among codewords into consideration in computing the weighted-sequence distance, thus the choice of the number of clusters in creating the codebook is not critical. Figure 6 shows some sample codewords of the atom actions (cluster centroids) from our unsupervised classification. As we can expect, results of such simple clustering strategy applied on the inaccurately-cropped regions are very noisy. However, our weighted-sequence distance naturally incorporates the global temporal structure of the sequences in computing the final measure. The noisy, individual results are actually implicitly filtered during the computation. And the 87.8% average classification accuracy (see Table 3) further verified that our proposed approach is robust to noise and works well for data obtained from low-level visual features without using sophisticated feature extraction.

For algorithm training and testing, we use sequences from 9 random selected subjects for creating the codebook and adopt the leave-one-out (LOO) testing paradigm, which is employed by all but one of the reference algorithms chosen for comparison. Each LOO round consists of sequences from 24 subjects for training and those from the remaining one for testing. Excluding sequences from those 9 subjects which are used for creating the codebook, 16 LOO rounds were run. Results are shown in Table 1 and Table 2. It was found that the best performance was obtained by the proposed WSD kernel SVM classifier (which is better than the 1-NN classifier with WSD by 6.8% on average).

Several most recent state-of-the-art approaches are chosen for comparison. These include Niebles et al.'s work [Niebles et al., 2008], Dollar et al.'s work [Dollar et al., 2005], and Nowozin et al.'s result [Nowozin et al., 2007]. (Although some other methods also reported results on the same dataset, they were not selected for comparisons here since they rely on excessive manual segmentation or strong assumption of the availability of accurate preprocessing steps for feature extraction, which may significantly boost the performance. For example, [Schindler and Gool, 2008; Fathi and Mori, 2008; Mikolajczyk and Uemura, 2008] require accurate bounding box for each frame and [Liu and Shah, 2008] assumes accurate scale information for each frame. Both of the assumptions explicitly avoid the scale variation problem, which occurs in 1/4 of the sequences in KTH dataset.)

%	Box	Clap	Wave	Jog	Run	Walk	Ave
Our (1-NN)	88	81	83	70	73	91	81.0
Our (SVM)	94	91	86	80	84	92	**87.8**
Niebles et al '08	98	86	93	53	88	82	83.3
Dollar et al. (1-NN)	80	82	84	63	73	89	78.5
Dollar et al. (SVM)	85	77	93	57	85	90	81.2
Nowozin et al '07	86	89	92	69	86	86	84.7

Table 3: Comparisons to state-of-the-art results on KTH dataset

It was found that the WSD kernel SVM algorithm is able to outperform each of the reference methods in terms of overall accuracy, as shown in Table 3. Further, we can also observe from Table 3 that our algorithm is able to discriminate well "jogging", "running" and "walking" classes, which are good examples where the global temporal structure of the se-

quences are key to correct classification. This suggests that the proposed super-frame coding strategy is able to retain the global temporal information as desired. It is worth pointing out that both versions of the proposed methods (1-NN-based and SVM-based) outperform the corresponding algorithms from Dollar et al.

7 Conclusion and Future Work

We proposed an approach for human action recognition based on a novel super-frame-based coding scheme and a novel weighted-sequence distance for comparing super-frame sequences. Our approach takes into consideration both local and global spatial-temporal structures of an action that are deemed as critical for classification. The approach has been evaluated on a challenging real database and was found to be able to outperform many existing state-of-the-art approaches by a non-trivial margin, even if it uses very simple low-level visual features as the input.

There is still much room for further improving our work. For example, in our current work, we only use the temporal length of the atom poses as the attributes. Other information or features that may contribute to better discrimination of the actions can also be added to the attributes for each atom pose. Also, in our experiments, while the chosen dataset is challenging and has been widely-used, it does not include multiple actions/subjects and appearance variations due to view angle changes. Our future work includes evaluating and extending the proposed method to cover more complex cases. In addition, we also plan to explore the application of the proposed WSD on other problems where comparison between sequences of symbols with attributes is involved.

References

[Bui and Venkatesh, 2002] H. H. Bui and S. Venkatesh. Policy Recognition in the Abstract Hidden Markov Model. *Journal of Artificial Intelligence Research*, vol. 17, pp. 451-499, 2002.

[Bui et al., 2008] H. H. Bui, D. Phung, S. Venkatesh, and H. Phan. The Hidden Permutation Model and Location-Based Activity Recognition. *National Conference on Artificial Intelligence (AAAI)*, 2008.

[Campbell, 2001] W. M. Campbell. A Sequence Kernel and its Application to Speaker Recognition. *Neural Information Processing Systems*, vol. 14, pp. 1157-1163, 2001.

[Cancedda et al., 2003] N. Cancedda, E. Gaussier, C. Goutte, and J. M. Renders. Word Sequence Kernels. *The Journal of Machine Learning Research*, vol. 3, pp. 1059-1082, 2003.

[Chang and Lin, 2001] C.-C. Chang and C.-J. Lin. LIBSVM : a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[Cristianini and Shawe-Taylor, 2000] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*: Cambridge University Press, 2000.

[Crochemore and Rytter, 1994] M. Crochemore and W. Rytter. *Text Algorithms*: Oxford University Press, 1994.

[Dollar et al., 2005] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior Recognition Via Sparse Spatio-temporal Features. *2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005.

[Fathi and Mori, 2008] A. Fathi and G. Mori. Action Recognition by Learning Mid-level Motion Features. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[Kim et al., 2007] T. Kim, S. Wong and R. Cipolla. Tensor Canonical Correlation Analysis for Action Classification. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[Landau and Vishkin, 1989] G. Landau, and U. Vishkin. Fast parallel and serial approximate string matching. *Journal of Algorithms, vol. 10*, 157–169, 1989.

[Leslie et al., 2004] C. Leslie, E. Eskin, A. Cohen, J. Weston, and a. W. S. Nobley. Mismatch String Kernels for Discriminative Protein Classification. *Bioinformatics Advance Access*, vol. 20, pp. 467-476, 2004.

[Levenshtein, 1966] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physic Doklady*, vol. 10, pp. 707-710, 1966.

[Liao et al., 2005] L. Liao, D. Fox, and H. Kautz. Location-based Activity Recognition Using Relational Markov Networks. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[Liu and Shah, 2008] J. Liu and M. Shah. Learning Human Actions via Information Maximization. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[Mikolajczyk and Uemura, 2008] K. Mikolajczyk and H. Uemura. Action Recognition with Motion-Appearance Vocabulary Forest. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[Niebles et al., 2008] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words. *International Journal of Computer Vision*, vol. 79, pp. 299-318, 2008.

[Niu and Abdel-Mottaleb, 2005] F. Niu and M. Abdel-Mottaleb. HMM-Based Segmentation and Recognition of Human Activities from Video Sequences. *IEEE International Conference on Multimedia and Expo (ICME)*, 2005.

[Nowozin et al., 2007] S. Nowozin, G. Bakir, and K. Tsuda. Discriminative Subsequence Mining for Action Classification. *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[Roth et al., 2003] V. Roth, J. Laub, M. Kawanabe, and J. M. Buhmann. Optimal cluster preserving embedding of nonmetric proximity data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25, 1540–1551.

[Schindler and Gool, 2008] K. Schindler and L. v. Gool. Action snippets: How many frames does human action recognition require? *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[Schuldt et al., 2004] C. Schuldt, I. Laptev, and B. Caputo. Recognizing Human Actions: A Local SVM Approach. *International Conference on Pattern Recognition (ICPR)*, 2004.

[Wong et al., 2007] S. Wong, T. Kim and R. Cipolla. Learning Motion Categories using both Semantic and Structural Information. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.