An Unsupervised Framework for Action Recognition Using *Actemes*

Kaustubh Kulkarni[1], Edmond Boyer[1], Radu Horaud[1], and Amit Kale[2]

[1]INRIA, Grenoble, Rhone Alpes
[2]Siemens Corporate Technology, Bangalore
{firstname.lastname}@inria.fr, kale.amit@siemens.com

Abstract. In speech recognition, phonemes have demonstrated their efficacy to model the words of a language. While they are well defined for languages, their extension to human actions is not straightforward. In this paper, we study such an extension and propose an unsupervised framework to find phoneme-like units for actions, which we call *actemes*, using 3D data and without any prior assumptions. To this purpose, build on an earlier proposed framework in speech literature to automatically find actemes in the training data. We experimentally show that actions defined in terms of actemes and actions defined by whole units give similar recognition results. We define actions out of the training set in terms of these actemes to see whether the actemes generalize to unseen actions. The results show that although the acteme definitions of the actions are not always semantically meaningful, they yield optimal recognition accuracy and constitute a promising direction of research for action modeling.

1 Introduction

Recognition of human actions is an important part of research in dynamic scene understanding. The applications of classifying human actions in a video extend from video indexing and retrieval, video surveillance, human-robot and human-computer interactions. There are several challenges which arise while tackling the problem of human action recognition. One such fundamental problem is the temporal representation of actions. Phonemes in spoken language are the smallest or distinct segmental unit of sound which can be combined or concatenated to form words. This fact is exploited in speech recognition where Hidden Markov Models (HMMs) are learned on these phonemes. These models are combined to define the words of a vocabulary. Motivated from speech recognition, we investigate whether such a hierarchical definition is possible for human actions using sub-action units which we call *actemes*.

Intuitively, there must exist a restricted set of generic motions of a human body which can define all actions. This set, if it exists, can be likened to a set of phonemes which can define every word in the dictionary from a given language. There are certain advantages if such actemes can be learned. Firstly, the actemes would allow us to define a large number of actions in a compact representation. Secondly, the advantage of having a hierarchy i.e. where an action is described

as sequence of actemes is that when a new action is added to the list of actions to be recognized, this new action can be described in terms of actemes thus obviating the need for learning a new model every time an action is added.

The concept of phoneme definition for words exist from linguists. No such widely accepted definitions of actemes exist for human actions. Several researchers have tried to come up with such definitions. Green et. al.[1] proposes the use of 35 *Dynemes* which form the basic units of human actions or skills. The *dynemes* are defined in terms joint angles. An HMM model is used for action recognition. Another work by [2] defines *kinetemes* on the joint angle space of human motion. These *kinetemes* form the basic unit of a human activity language. Using these *kinetemes* and language grammar like rules the authors propose to construct any complex human action. Bregler [3] defines *Movemes* as linear dynamical systems over which an HMM model is learned for recognition. Since the space of all possible human motions is very large and since no widely acceptable definition exist it is better to automatically come up with these definitions for actemes as opposed to [1]. Also, we assume no rules while labelling the actions in terms of the learned actemes, as done in [2] instead we use the recognition algorithm it self to provide the labelling. In [3], the author proposes a method to automatically learn the *Movemes* from the training set. The results shown in this paper are evaluated on actions consisting on repeated segments such as walking, running and skipping. In such a scenario the basic blocks constructing the actions are obvious and eliminate the need for labelling the actions in terms of *Movemes*. In this paper, for the experiments we evaluate the efficacy of our proposed method exclusively on actions which do not consist of repeated segments.

In this paper, we build on a speech recognition formalism [4, 5], which proposes to design a recognizer terms of acoustic subword units (ASWU).This method assumes no prior information while learning the ASWUs from words. It learns these definitions in an unsupervised data-driven manner. We apply this method from speech recognition for obtaining actemes because it is completely data-driven and makes no prior assumptions on the definitions of actemes. This is a completely different way to approach the problem of human action representation and recognition than the earlier proposed methods. Secondly, the number of actemes per action is also known so a data-driven approach is best suited to come up with acteme units. To summarize, the main contributions of the paper are the following:

1. We use a speech recognition formalism to learn the actemes and the representation of actions in terms of the actemes in an unsupervised framework.
2. We show that actions from outside the training set can be represented in terms of these learned actemes and recognized without explicitly learning a new model for the actions.

2 Related Work

Automatic annotation of actions in videos is a challenging task and various action recognition methods can be grouped together depending on the types

of features used and the method employed to model the temporal and spatial representations of actions. A brief survey of temporal representations similar to ours has been discussed in Sec. 1. We restrict our survey to 2D silhouettes, 3D visual hulls and key frames as features. The recognition methods discussed are HMMs and dynamic time warping (DTW) based methods. For a detailed survey of action recognition methods see [6].

The earliest features used were silhouettes extracted from each frame over time and an HMM was learned from them [7]. These were used to recognize tennis strokes from single views. A later paper [8] describes temporal templates for human action recognition. [9] extends the 2D temporal templates to 3D volumes. [10] also describes a view invariant recognition method where they learn parametric HMMs from 3D data and use the HMMs as a generative model to synthesize 2D action sequences closest to an unknown 2D test action sequence. Another way of classifying actions is by using dynamic time warping (DTW). [11] learns the warping bounds for the actions from the training data. [12] proposes to use distance between linear dynamical systems for action classification. [13],[14] perform action recognition by defining actions as trajectories on the Grasmann Steifel manifold. [15] extends the DTW framework using average templates with multiple features to model intra-class variances and perform simultaneous recognition and localization of actions in a video sequence. All these methods learn the model on entire actions.

Another popular method is to define actions as a set of poses or key frames or exemplars [16]. They use single key frames to recognize backhand and forehand in tennis. There also has been work which uses short snippets of frames [17] to recognize actions instead of a single frame. In [18], the authors use the forward selection algorithm to find the most discriminative set of exemplars to describe an action vocabulary. [19] model actions as a sequence of atomic body poses where the authors consider the order in which the poses appeared. In this paper, we express action interms of sequence of short segments or actemes instead of sequence of key poses.

3 The Method

To learn the actemes we employ a method proposed in speech recognition [4, 5]. Actemes are equivalent to phonemes or ASWUs and the whole actions are equivalent to a word. In this method, the authors propose to 1) optimal cut the words into piecewise stationary segments, 2) get a reduced set of ASWUs by applying K-means on the means of each optimal segment , 3) learn HMMs on these ASWU, 4) apply the connected word Viterbi algorithm to label the training data in terms these ASWU to generate a lexicon or a phonetic definition for each word in terms of the ASWUs, and 5) then use this definition in the Viterbi framework to perform recognition. Instead of using HMMs to model the actemes we use the earlier proposed average template models [15] and the one-pass dynamic programming algorithm [20] for labeling and the modified one-pass dynamic programming algorithm [21] for recognition. The average template

model is shown to outperform the HMMs in [15] . Figure 1 and Figure 2, explains the building blocks of the algorithm. These building blocks are explained in the following sub-sections from 3.1 to 3.5.

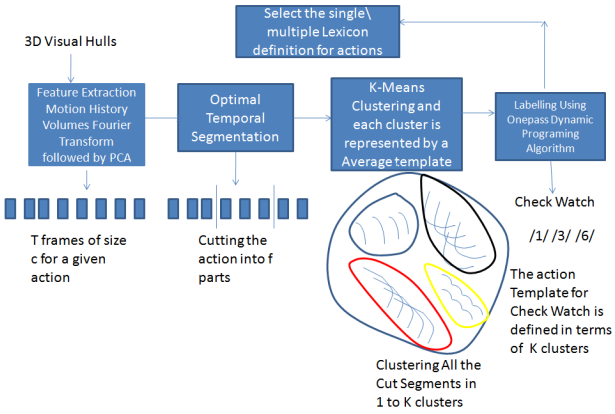

Fig. 1. In this figure, we have the block diagram of the acteme training steps

.

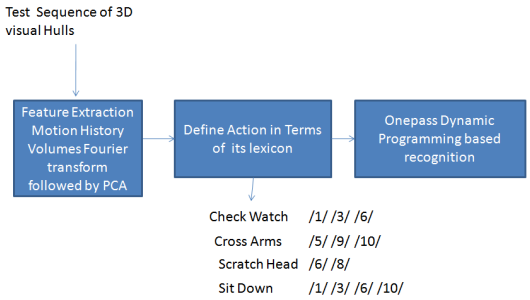

Fig. 2. In this figure, we have the block diagram of the steps for recognition

.

3.1 Feature Computation

The features needed to interface with a time synchronous onepass-DP algorithm should be a set of feature vectors over time given by $E = \{e_1, e_2, \ldots, e_b, \ldots, e_T\}$ where e_b is a vector of dimension c at a given time instant b. In this paper, we use 3D visual hulls as our features. We then compute motion history volumes (MHVs) [9] to recognize action using DTW. The MHVs store the motion history on a 3D occupancy grid in a given window. In this paper, we use a short window of size 5. The occupancy grid of the MHVs is of the size $64 \times 64 \times 64$ for each frame. Since the actors are allowed to change their view point freely we convert the Cartesian coordinates to cylindrical coordinates followed by Fourier transform on this occupancy grid. This Fourier magnitudes ,of size $16 \times 16 \times 16$, are invariant to the rotation around the z-axis. We perform further dimensionality reduction using principal component analysis to reduce the feature vector to a size of 100. Therefore, in this paper we have $c = 100$.

3.2 Temporal Segmentation

Several approaches using HMMs have been used for action, gesture and sign language recognition. The implicit assumption of using a left to right HMMs for recognition is that the action is composed of piecewise stationary regions. These regions are modelled by the states of the HMM. Hence the number of states is an important parameter to correctly estimate for action recognition. The piecewise stationary regions in a word are the phonemes and if we fit a left to right HMM model with the number of states equal to the number of phonemes we get a good recognition accuracy [5]. Also the steady state regions are most likely to lie between abrupt motion changes or discontinuities which can be used for temporal segmentation [22] using MHVs. We do not take the approach of using velocity discontinuities because actions like "'stand"' or "'sit down"' do not have abrupt changes in the direction of velocity. We motivate our strategy to cut actions into relevant regions by assuming that the actions can be decomposed into piecewise stationary regions. The method cuts the actions into segments such that the global distortion of these segments w.r.t their means is minimized. This can be formulated as a dynamic programming problem [23].

Consider an action template defined as set of features over time by E where e_b is a feature vector corresponding to the bth frame of size $c = 100$ and the action is performed over T time instances. The task here is to segment E into f homogeneous segments by minimizing the sum of the distances between frames of the segments to their respective means. Let the segment boundaries for a given action template be $G = \{g_1, g_2, g_i, \ldots, g_f\}$ where g_i are integers indicating the frame numbers of the boundaries. The i^{th} segment starts at $g_{i-1} + 1$ and ends at g_i; $g_1 = 0$ and $g_f = T$. The optimal boundaries G^* can be found by minimizing the following function over all possible segmentations:

$$D_1(f, T) = \sum_{i=1}^{i=f} \sum_{b=g_{i-1}+1}^{g_i} d_1(e_b, \bar{e}_i) \tag{1}$$

where $D_1(f, T)$ is the total accumulated distance for segmenting E into f segments. The mean of the ith segment is given by $\bar{e}_i$ which is the average of the frames of the ith segment given by $H = \left\{e_{g_{i-1}+1}, \ldots, e_{g_i}\right\}$. The distance metric used is euclidean; $d_1(e_b, \mu_i) = \|e_b - \bar{e}_i\|$.

The problem of solving for optimal boundaries can be efficiently solved using a treillis realization. This can be achieved by solving the following dynamic programming recursions as given in [23],[24]:

$$D_1(i, g_i) = \min_{g_{i-1}} \left[D_1(i-1, g_{i-1}) + d_1(e_b, \bar{e}_i)\right] \text{where } b = g_{i-1} + 1 \text{ to } g_i \tag{2}$$

where $D_1(i, g_i)$ is the cost of dividing the template E into i segments till the frame g_i where $i < f$. This cost is given by the minimum over cost accumulated by dividing E into $i - 1$ segments till frame g_{i-1} plus the distance of the ith

segment with its mean. The optimal segmentation can be found by backtracking through the trellis starting from $\min D(f, T)$.

If the number of segments f for a given word is equal to the number of phonemes in that particular word then ASWUs are equivalent to phonemes of that language otherwise the ASWUs are not semantically meaningful. The number of phonemes in a given word is not always known because of the pronunciation. In [4, 5], it is shown that even if ASWUs are not semantically meaningful the algorithm still provides a good recognition accuracy. Since the number of actemes in an actions are unknown the method given in [4, 5] is more suited to be applied to the problem of action recognition using actemes as opposed to other approaches [1–3] motivated from speech recognition systems.

3.3 Clustering and Computing average-template Model

This procedure to segment each action template into f segments is repeated for all actions in the training set. Therefore, if there are N training instances of all actions then we will have a set of $f \times N$ variable length temporal segments. To get a compact representation we apply K-means on this set of temporal segments to get the K actemes. Since, we assumed that each segments is piecewise stationary we represent each segment in this set by its mean and apply the K-means on the these segment means.

To represent the cluster corresponding to each of the acteme we compute a temporal average or nominal template [25] over all the instances of a a given acteme. In this section, we describe a method to represent each acteme as an average of the templates in that cluster of actemes. The average pattern or average-template R^k is computed by mapping the segments, $H = \{H_1, H_2, \ldots, H_l, \ldots\}$, in the cluster corresponding to the acteme k using DTW. We use Euclidean distance as the local distance $d_2(i, j)$ between the frame i of R^k and frame j of H. If I is the length of R^k and J is the length of H, the path is forced to begin at the point $D_2(1, 1)$ and end at $D_2(I, J)$ on the trellis to compute the accumulated distance $D_2(i, j)$. This accumulated distance is defined as:

$$\begin{aligned} D_2(i,j) = \min[&D_2(i-2, j-1) + 3d_2(i,j), \\ &D_2(i-1, j-1) + 2d_2(i,j), \\ &D_2(i-1, j-2) + 3d_2(i,j)] \end{aligned} \tag{3}$$

where i is the frame index of the average reference pattern R^k and j is the frame index of the train pattern H.

Backtracking from the point $D_2(I, J)$ on the treillis yields the optimal path $p = [i_m, j_m]$ and the corresponding mapped set of feature vectors $[R^k(i_m), H(j_m)]$. Here m, is the index of a point on the optimal path p. The average reference pattern R^k_l for an activity is computed by the successive weighted averaging of l instances as follows:

$$R^k_l(m) = \left(1 - \frac{1}{l}\right) R^k_{l-1}(i_m) + \frac{1}{l} H_l(j_m), m = 1 \ldots M \tag{4}$$

where M is the number of points on the optimal path p and $R^k_{l-1}(i_m)$ is the average of the previous $l-1$ templates. The new time axis for the instance R^k_l is computed as:

$$p_1(m) = \left(1 - \frac{1}{l}\right) i_m + \frac{1}{l} j_m, m = 1 \dots M \tag{5}$$

We linearly transform this new time axis to a constant length P where P is the average length of all segments in the cluster of acteme k. The transformation is done as follows:

$$p_2(m) = \frac{P}{M} p_1(m) \tag{6}$$

as $p_2(m)$ would have non-integer values we define a time axis $p_3(m')$ where $m' = 1, 2, 3 \dots P$. The feature values of the average pattern $R^k_l(m)$ are interpolated to get the new average pattern representing the cluster corresponding to acteme k $R^k_l(m')$.

3.4 Labelling

In this section, we discuss the method to label each of the training sequences in terms of the learned K learned actemes. We use a 'connected word recognition' algorithm based on the one-pass DP, well known in speech recognition [20]. Continuous labelling of action templates in terms of actemes is a difficult task to do on line, primarily because this involves the problem of jointly determining the optimal number of actemes M^* in the train sequence $\mathbf{O}$, their boundaries $S^* = \{s^*_0, s^*_1, s^*_{m-1}, s^*_m, \dots, s^*_{M^*}\}$ and associated optimal acteme indices $I^* = \{i^*_1, i^*_2, \dots, i^*_m, \dots, i^*_{M^*}\}$ (where $v_{i^*_m} \in V$), by minimizing a measure of distance $D(\mathbf{O}, \mathbf{R})$ between the train sequence $\mathbf{O}$ and a typical reference acteme template sequence $\mathbf{R} = \{R_{v_{i_1}}, R_{v_{i_2}}, \dots, R_{v_{i_m}}, \dots, R_{v_{i_M}}\}$ each drawn from V. The decoding problem of determining (M^*, S^*, I^*) is solved by minimizing $D(\mathbf{O}, \mathbf{R})$ over the variables (M, B, I) using the time-synchronous one-pass DP decoding algorithm.

To compute the optimal cumulative distance, we use two types of transition rules (a) for acteme interior i.e. Within Acteme Recursion (b) for acteme boundary i.e Cross-Acteme Recursion. These recursions are computed for all frames of the train action template w.r.t the all frames of all average template acteme models in a left to right time synchronous manner. These recursions would then result in many possible paths. The optimal action sequence or path will be the one which corresponds to the minimum cumulative distance (Termination and Backtracking).

We now provide the mathematical details pertaining to the above intuitive explanation of the algorithm. The acteme vocabulary of size K is given by $V = \{v_1, v_2, \dots, v_K\}$. Each acteme corresponds to a reference pattern $R_{v_k}(k')$, where $k' = 1, 2, 3 \dots P_{v_k}$; P_{v_k} is the number of frames of the average template v^{th}_k acteme where $k = 1, 2, 3 \dots K$. The train action template frame index is given by q and Q is the length of the train action template $\mathbf{O}$. During the labelling pass

the sequence of warping is given by the average-templates. The local distance between one frame of the average template of a given acteme and a frame of the training action sequence is computed in the following way:

$$d(q, k', v) = \|R_v(k') - O(q)\| \tag{7}$$

Let D denote the global accumulated distance between the train action frame and the reference pattern frame. The one-pass DP decoding would look to minimize the global accumulated distance over all the frames of the train action pattern. The following steps give a method to accumulate the global distance between a given train action frame and a frame of the reference pattern to find a globally optimal path:

1. **Within acteme recursion**: This recursion is computed for all frames Q of the train action pattern and all frames k' of all reference patterns except for $k' = 1$ i.e. the recursions are applied to to all frames except at the acteme beginning. This recursion can be denoted as:

$$D(q, k', v) = d(q, k', v) + \min_{k'-2 \leq r \leq k'}(D(q-1, r, v)) \tag{8}$$

2. **Cross-acteme Recursions**: This recursion is computed for all Q test frames and for $k' = 1$ frames of all reference patterns. This recursion allows a transition into the first frame of a given reference pattern from the last frame of all other reference pattern including the given reference pattern or it allows the path to be in the last frame of that given reference pattern i.e. the algorithm either stays in the particular acteme or transits into the first frame on any other acteme depending on which of the two paths yields a minimum score. It can be denoted as:

$$D(q, 1, v) = d(q, 1, v) + \min[\min_{1 \leq v \leq K}[D(q-1, P_v, v)], D(q-1, 1, v)] \tag{9}$$

3. **Termination and Backtracking**: To find the best acteme sequence the algorithm uses the following termination condition at the train action frame Q:

$$D^* = \min_{1 \leq v \leq K}[D(Q, P_v, v)] \tag{10}$$

The algorithm checks for the minimum accumulated distance for the best path at the last frame of every reference pattern at the train action frame Q. The best path is backtracked from that point through back-pointers stored during the Within Acteme and Cross Acteme recursions.

The output of running the onepass-DP algorithm will be a sequence of optimal acteme indices I* for every training action template. For eg. the a given sequence **O** could be labelled as $\{v_3, v_2, v_7\}$. This is a completely unsupervised labelling by an onepass DP algorithm which is also the same algorithm we use for recognition. We choose the acteme representation or lexicon which repeats itself the most number of times as the model for a given action while recognition. Since, there is intra class variance in the manner in which different actors

perform the action we find that upto 4 lexicons have to be used to get results close to our baseline. This is true in the case of speech recognition where a given word can be pronounced by different by different speakers one phonetic representation is not enough to obtain good recognition results. The lexicons chosen are in descending order of their occurrence while labelling the training data.

3.5 Recognition

While recognizing the actions we assume that the action boundaries in the video sequence are known. Therefore, we only recognize the action and do not localize it in the video sequence. This assumption is necessary as isolated action recognition is the true test of the efficacy of this approach as it gives only substitution errors i.e. the an action can be recognized as itself or confused as some other action. Simultaneous recognition and localization causes insertion and deletion errors.

We use the method proposed in [21] to perform action recognition when each action is defined in terms of actemes. The proposed algorithm can be used for simultaneous recognition and localization of action in a video sequence. We switch off the *Cross-Word transitions* [21] since we are only recognizing the actions and assume that the boundaries in the video sequence are known.

Let the number of action to be recognized be $W = \{w_1, w_2, \ldots, w_m, \ldots, w_M\}$ where $m = 1 \ldots M$ is the total number of actions in the recognition vocabulary. The number of lexicons per actions is defined as l which is the constant and same for every action to be recognized in the vocabulary. The l^{th} representation of each action w_m in terms of the actemes given by $\{a^l_{1m}, a^l_{2m}, \ldots, a^l_{jm}, \ldots, a^l_{N_{am}}\}$ where $j = 1 \ldots N_{am}$ is the number of actemes representing the action w_m and $l = 1 \ldots L$. The local distance between the test sequence and the average templates be $d(m, a^l_{jm}, l, k', q)$ where $k' = 1, 2, 3 \ldots P_{a^l_{jm}}$; $P_{a^l_{jm}}$ is the number of frames of the average template a^l_{jm} acteme. The test action sequence consist of $q = 1 \ldots q \ldots Q$ frames. The local distance is given by euclidean distance between the k'^{th} frame of the average template representing each acteme and the q^{th} frame of the test data. The dynamic time warping time synchronously calculates the minimum global accumalted distance $D(m, a^l_{jm}, l, k', q)$ to reach the k'^{th} frame of the word w_m represented by lexicon l till the q^{th} of the test action sequence. Since, the cross action recursions are switched off there are only with action recursions which can be divided into two types:

1. **Within Acteme recursions**: These recursions are applied for all frames of the average template of each of the acteme except for the template beginning i.e. $k' = 1$

$$D(m, a^l_{jm}, l, k', q) = d(m, a^l_{jm}, l, k', q) + \min_{k'-2 \leq r \leq k'} [D(m, a^l_{jm}, l, r, q-1)] \quad (11)$$

2. **Cross Acteme recursions**: The actions are represented by variable number of actemes in a given order. This order is given in the labelling step. Therefore, in this recursion the transition occurs into the first frame of the average template of each acteme from the last frame of the average template of the

previous acteme. This is known as forced alignment of the concatenated action model to the test sequence. This recursion are applied at the first frame $k' = 1$ of every acteme except the first. This step can be mathematically denoted as:

$$\begin{aligned} D(m, a^l_{jm}, l, k' = 1, q) = min[&D(m, a^l_{jm}, l, k' = 1, q-1), \\ &D(m, a^l_{(j-1)m}, l, P_{a^l_{(j-1)m}}, q-1)] \\ &\text{where } j = 2 \text{ to } N_{am} \end{aligned} \tag{12}$$

3. **Termination and Backtracking**: The action is assumed to begin at the first frame of the average template of the first acteme of the given action and end at the last frame of the last acteme of the same action. Therefore, the optimal accumulated distance D^* can be obtained by checking the last frames of the last actemes of all actions with all representative lexicons at the last frame Q of the test sequence:

$$D^* = \min_{m=1\ldots M} \min_{l=1\ldots L} D(m, a^l_{N_{am}}, l, P_{a^l_{N_{am}}}, Q) \tag{13}$$

The test sequence is classified as the action index m for which the D is minimum.

4 Experimental Results

In this section, we show that the actions described as actemes give equivalent performance to the actions when modelled as whole units themselves. We add actions which are not included in the training set to check whether the learned actemes generalize to unseen data. We evaluate our method on the INRIA XMAS dataset. In our experiments, we assume that the boundaries of the actions in the video sequences are known.

For the first set of experiments,we use a reduced vocabulary of *check watch, sit down, get up, punch and kick*. The recognition experiments are performed on the set of 10 actors and are validated by the standard leave-one-out testing procedure.In these experiments, we show that the actions described by actemes give equivalent recognition performance to actions described as whole units themselves. We first obtain the 100 dimensional feature vectors in time from the 3D visual hulls using the procedure described in Sec. 3.1. We apply the temporal segmentation procedure on all available training instances of each actions to get 2 and 3 segments respectively. If the method tries to cut the actions into more than three segments we observe that the actions start breaking into segments which are only 1-2 frames long. These segments cannot be averaged with longer segments to learn an average template model because the warping of very short with long segments is meaningless [5].

We apply K-means on these cut segments and plot the recognition results with K varying from 10 to 50 in the steps of 10. Due to the intra class variance observed in the performance of the actions we increase the number of lexicons per actions from 1 to 4 in the descending order of their occurrence. We find that

both help in increasing the recognition accuracy. The recognition results of the first set of experiments are given in Fig. 3. We observe two facts from this result:

1. For the case where training actions are cut into 2 segments the recognition accuracy increases till two lexicons and then it starts to decrease. This is because there is trade off between the number of lexicons per actions and the recognition accuracy as increasing the number of lexicons per actions also increases the possibility of confusions.
2. Recognition accuracy is better when the actions are segmented into 3 parts than 2 parts because if we observe the reduced vocabulary of actions apart for *stand up and sit down* in the XMAS dataset they consist of three parts an initial movement, the main action and the relaxation part. In *sit down*, there is an initial movement of bending the back, crossing the legs sitting down and coming to a relaxed pose after sitting down. *Stand up* is exact opposite of sitting down.

In the second set of experiments in the paper, we add the following set of actions one at a time to the *cross arms, scratch head, pick up* to the list of actions to be recognized. Thus, forming a vocabulary of 6 actions every time one of the 3 actions is added. The training instances of these actions are not used to train the actemes. We only use the training instances from these actions to get the lexical representation of the added action in terms of the actemes learned from the earlier 5 actions.

We observe that the recognition accuracy is the best for *pick up* because the initial part of *pick up* is very similar to the *sit down* and the latter part of pick is similar to the *stand up* action. Therefore, the actemes for *pick up* are present in the reduced vocabulary of 5 actions. The next best performance is achieved by *cross arms* again because the action *check watch* is similar to it. The actions *scratch head* action when added to the vocabulary of 5 action gives the worst recognition result because there are no actions in the reduced vocabulary of 5 actions similar to the *scratch head* action. The recognition results for the second set of experiments are given in Fig. 4 to Fig. 6.

We compare our results with other methods which were applied to the INRIA XMAS database. We model these whole units of actions using an average template model as discussed in 3.3 and perform isolated action recognition [15] to get the recognition baseline. This baseline is shown as the blue baseline in Fig. 3 to Fig. 6. We also compare our results with the recognition method proposed in [9]. The size of the FFT features used to obtain recognition results $16 \times 16 \times 16$. We compare our method with the approach in [11] which proposes another method to learn average or nominal trajectories.The average trajectories are computed using the 100 dimensional features described in Sec. 3.1 . We find that the proposed acteme based representation performs slightly better than the method proposed in [11] and comes close to the recognition performance of [15, 9] for the first set of experiments using 5 actions. For the second set of experiments we find the for added actions, *cross arms, pick up*, which have a similar action in the 5 action training set the results are comparable to all the baselines. The

recognition accuracy in the *actemes* column is the recognition accuracy achieved for 3 cuts and $K = 50$. All the methods use leave-one-out testing strategy. [9] uses a best segment representative of the action. While actemes,[15] and [11] use the boundaries extracted from the ground truth.

Table 1. *Comparison of actemes representation with other Recognition Methods*

Added Action	LDA [9]	PCA [9]	Mahala-nobis [9]	Average Template [15]	*Actemes* This Paper	Average Trajectory[11]
None	94.67	86.67	95.33	95.33	94.00	92.00
CrossArms	97.78	81.67	97.79	97.79	87.29	86.74
ScratchHead	92.22	77.22	93.33	97.78	81.67	91.11
PickUp	96.67	83.89	94.44	97.24	91.71	92.82

In the experimental section, we have discussed the efficacy of the actemes w.r.t. the recognition accuracy. The representation of *check watch and sit down* actions in terms of the actemes is shown in the video uploaded with the paper.

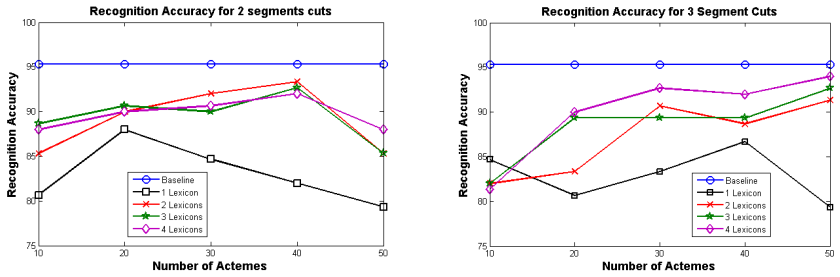

Fig. 3. Recognition accuracy plots for 2 and 3 segments cuts. We can see that the recognition accuracy of acteme based representation is close to the baseline of actions when modelled as whole units themselves.

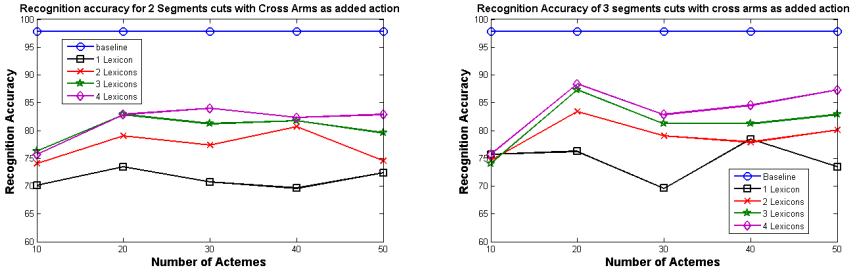

Fig. 4. Recognition accuracy plots for 2 and 3 segments cuts with *Cross Arms* as the added action. We can see that for the 3 cut case this 6 word vocabulary gives satisfactory results

.

5 Conclusion and Future Work

To conclude, we have demonstrated an unsupervised framework to learn a set of actemes from a given training database to represent actions. We experimentally

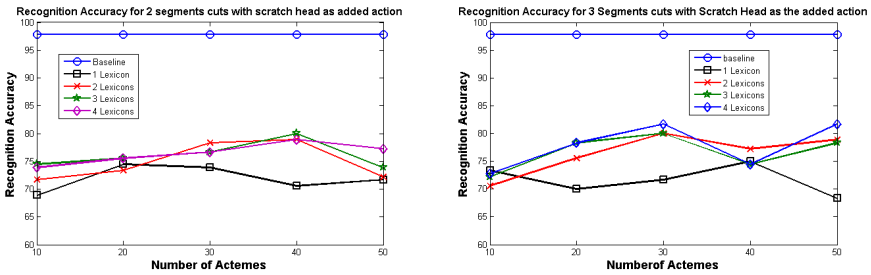

Fig. 5. Recognition accuracy plots for 2 and 3 segments cuts with *Scratch Head* as the added action. We observe that the recognition results are poor because the *scratch head* does not have a similar action in reduced vocabulary of 5 actions used to learn the actemes

.

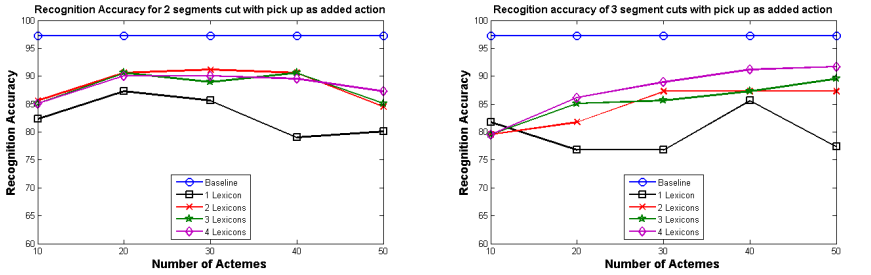

Fig. 6. Recognition accuracy plots for 2 and 3 segments cuts with *Pick Up* as the added action. We observe that the recognition results are good because the actemes for pick up are present in the *Stand Up* and *Sit Down* action

.

show that actions defined in terms of these actemes can give the similar recognition accuracy as compared to the whole unit themselves. We also showed that satisfactory recognition results can be achieved even with action which are not included in the training set for learning the actemes. For future work we would like to explore techniques which can be semi-supervised to learn semantically meaningful actemes. We would also like to extend this framework to bag-of-words like approaches. The next obvious step would be to do simultaneous recognition and localization of actions in a video sequence.

Acknowledgements: This work was done under European project *HUMAVIPS* (FP-ICT 2009 247525). The authors would like to thank Pavan Kumar Turaga, Center for Automation Research, Univ. of Maryland for providing the code to compute the FFT features. We thank Dr. V. Ramasubramanian for his time and helpful insights on speech recognition algorithms. Our sincere thanks to Ashok Veeraraghvan for sharing with us his code from [11].

References

1. Green, R.D., Guan, L.: Quantifying and recognizing human movement patterns from monocular video images-part i: a new framework for modeling human motion. IEEE Trans. Circuits Syst. Video Techn. **14** (2004) 179–190

2. Guerra-Filho, G., Aloimonos, Y.: A language for human action. Computer **40** (2007) 42–51
3. Bregler, C.: Learning and recognizing human dynamics in video sequences. In: CVPR. (1997)
4. Lee, C.H., Soong, F., Juang, B.H.: A segment model based approach to speech recognition. In: ICASSP. (1988)
5. Rabiner, L., Juang, B.: Fundamentals of speech recognition. Prentice Hall, New Jersey, USA (1993)
6. Poppe, R.: A survey on vision-based human action recognition. Image and Vision Computing **28** (2010) 976 – 990
7. Yamato, J., Ohya, J., Ishii, K.: Recognizing human action in time-sequential images using hidden markov models. CVPR (1992) 379–385
8. Bobick, A.F., Davis, J.W.: The recognition of human movement using temporal templates. PAMI (2001)
9. Weinland, D., Ronfard, R., Boyer, E.: Free viewpoint action recognition using motion history volumes. CVIU **104** (2006) 249–257
10. Weinland, D., Boyer, E., Ronfard, R.: Action recognition from arbitrary views using 3d exemplars. ICCV (2007)
11. Veeraraghavan, A., Chellappa, R., Roy-Chowdhury, A.K.: The function space of an activity. CVPR (2006)
12. Turaga, P.K., Veeraraghavan, A., Chellappa, R.: From videos to verbs: Mining videos for events using a cascade of dynamical systems. CVPR (2007)
13. Turaga, P.K., Veeraraghavan, A., Chellappa, R.: Statistical analysis on stiefel and grassmann manifolds with applications in computer vision. CVPR (2008)
14. Turaga, P.K., Chellappa, R.: Locally time-invariant models of human activities using trajectories on the grassmanian. CVPR (2009)
15. Kulkarni, K., Cherla, S., Kale, A., Ramasubramanian, V.: A framework for indexing human actions in video. ECCV Workshops (2008)
16. Carlsson, S., Sullivan, J.: Action recognition by shape matching to key frames. CVPR Workshops (2001)
17. Schindler, K., Gool, L.V.: Action snippets: How many frames does human action recognition require? In: CVPR. (2008)
18. Weinland, D., Boyer, E.: Action recognition using exemplar-based embedding. CVPR (2008)
19. Ogale, A.S., Karapurkar, A., Aloimonos, Y.: View-invariant modeling and recognition of human actions using grammars. In ICCV Workshops (2005)
20. Ney, H.: The use of one-stage dynamic programming algorithm for connected word recognition. IEEE Trans. on Acoustic Speech and Signal Processing **32(2)** (1984) 263–270
21. Ramasubramanian, V., Kulkarni, K., Kaemmerer, B.: Acoustic modeling by phoneme templates and modified one-pass dp decoding for continuous speech recognition. ICASSP (2008)
22. Weinland, D., Ronfard, R., Boyer, E.: Automatic discovery of action taxonomies from multiple views. In: CVPR. (2006)
23. Svendsen, T., Soong, F.: On the automatic segmentation of speech signals. (1987)
24. Ramasubramanian, V., Sreenivas, T.: Automatically derived units for segment vocoders. In: ICASSP. Volume 1. (2004) I – 473–6 vol.1
25. Zelinski, R., Class, F.: A learning procedure for speaker-dependent word recognition systems based on sequential processing of input tokens. ICASSP (1983)