3D Human Action Recognition Using Spatio-temporal Motion Templates*

Fengjun Lv, Ramakant Nevatia, and Mun Wai Lee

University of Southern California
Institute for Robotics and Intelligent Systems
Los Angeles, CA 90089-0273
{flv,nevatia,munlee}@usc.edu

Abstract. Our goal is automatic recognition of basic human actions, such as stand, sit and wave hands, to aid in natural communication between a human and a computer. Human actions are inferred from human body joint motions, but such data has high dimensionality and large spatial and temporal variations may occur in executing the same action. We present a learning-based approach for the representation and recognition of 3D human action. Each action is represented by a template consisting of a set of channels with weights. Each channel corresponds to the evolution of one 3D joint coordinate and its weight is learned according to the Neyman-Pearson criterion. We use the learned templates to recognize actions based on χ^2 error measurement. Results of recognizing 22 actions on a large set of motion capture sequences as well as several annotated and automatically tracked sequences show the effectiveness of the proposed algorithm.

1 Introduction and Related Work

Visual input is an important component of human computer interaction. In many applications, the objective is for human to *control* the responses of a computer without using keyboard and mouse [15]. In these examples, the actions of the human are deliberate and designed for easier understanding by the computer. Furthermore, the visual sensing is arranged to simplify the problems. The sensed images are of high resolution where hands and head motions are clearly visible and 2D analysis suffices [13] [4] [5]. In contrast are applications where the human is engaged in normal life activities and it is the computer's task to understand the human behavior and react according to the design goals of the system. Recognition of human actions, such as standing up, sitting down, pointing, waving arms etc. is essential to obtaining such capabilities and is the focus of this paper.

Human action recognition has been of interest to researchers in computer vision [2] [3] [9] [11] [13] [8] for many years. The problem can be defined as follows: given a motion sequence, the computer should identify the actions being

* This research was supported, in part, by the Advanced Research and Development Activity of the U.S. Government under contract No. MDA904-03-C1786

N. Sebe, M.S. Lew, and T.S. Huang (Eds.): HCI/ICCV 2005, LNCS 3766, pp. 120–130, 2005.

performed by the human subject. First difficulty is in estimating the positions of body parts; we do not address this problem but assume that such data is available, either from a Motion Capture (*MoCap*) system or from an automatic pose tracking system. Even if accurate 3D positions are available, action recognition is difficult due to the large dimensionality of pose space which not only increases computational complexity but may also hide key features of the action and due to significant spatial and temporal variations in an action for different, or even the same, subject.

Proposed methods can be divided into two categories with respect to the data types that they use: those based on 2D image sequences, e.g. [3] [11] [13] [8] and those based on 3D motion streams, e.g. [2] [9]. A 3D approach has many advantages over a 2D approach as the dependence on viewpoint and illumination has been removed. Nonetheless, most algorithms use a 2D approach because of the easy availability of video inputs and difficulty of recovering 3D information.

Our approach is based on 3D joint position trajectories. In this paper, our emphasis is on action recognition, given the 3D joint trajectories; we explore the effects of noise in automatically tracked data but robust tracking under complex conditions is a separate topic of research by itself. The actions we consider are *primitive* components that may be composed to form more complex actions. Examples of actions are: walk, sit, stand, wave hand, nod etc. One type of action is represented by one spatio-temporal motion template. Each template consists of a set of motion channels where each channel corresponds to the evolution of one joint coordinate. Channels with strong discriminative power are highlighted according to *Neyman-Pearson* criterion [14]. The recognition is based on the minimum weighted sum of χ^2 distance to each motion channel.

The proposed system achieves a recognition rate of 90.7% on 848 motion capture sequences consisting of 22 actions, showing the effectiveness of the proposed algorithms. Good results on annotated and automatically tracked videos show the potential of using real data.

2 Dataset and Pre-processing

We collected 848 MoCap sequences consisting of 22 Actions from Internet[1]. We also generated some sequences of 3D joint position from a 3D annotation software [7] and from an automatic 3D tracking software [7]. The generated data are much less accurate compared with MoCap. They are used in testing only to show that our algorithm can work on real data and that training on MoCap data transfers to video sequences; we do not claim to have solved the tracking problem as well.

Actions in these videos can be grouped into 3 categories according to the involved primary body parts: *leg+torso, arm, head*. The categorization is illustrated in Fig.1. Actions in the same group are mutually exclusive to each other, but actions from different groups can be recognized simultaneously. This allows us to execute logical queries such as *find the sequence in which the subject is walking while his head is nodding.*

[1] Part of data come from mocap.cs.cmu.edu, which was created with funding from NSF EIA-0196217

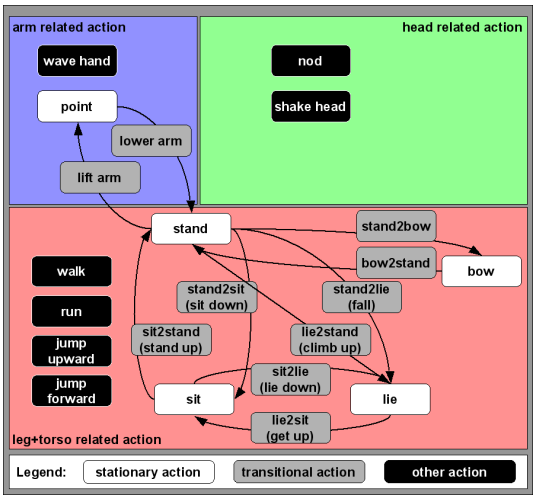

Fig. 1. Categorization of actions that need to be recognized

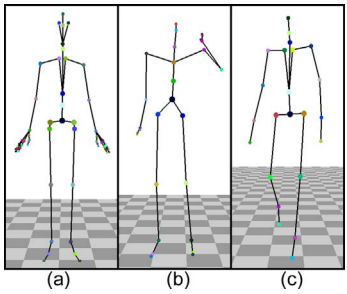

Fig. 2. Three different joint configurations. Notice the difference in the number of joints and joint hierarchy

Actions can also be classified based on the overall motion of human body. We view a *stationary* pose, *e.g. stand or sit* (*white blocks* in Fig.1) as a special type of action, with the constraint on the minimum duration. The *transitional* actions (*gray blocks*) transit from one stationary pose to another. The remainder (*black blocks*) consist of *periodic* actions (*e.g. walk, run*) and other actions.

MoCap data provides three rotation angles of each joint at each frame, however, we can not use them directly primarily because the data are heterogeneous in terms of different joint configurations (i.e. number of joints/bones, joint names and joint hierarchy). One example of this difference is shown in Fig.2. Using rotation representation, it is difficult to find correspondence between two sequences with different joint configurations. Instead, we compute 3D joint positions from rotation by kinematics. To find correspondence between two different joint configurations, we simply specify the corresponding joints the same name and ignore unwanted joints. We unify all joint configurations to the one shown in Fig.2(c) that consists of 23 joints. The joint positions are normalized so that the motion is invariant to the absolute body position, the initial body orientation and the body size.

3 Action Representation Using Spatio-temporal Motion Templates

As there are 23 joints and each has 3 coordinates (only y coordinate is used for hip), the whole body pose at each frame can be represented by a vector (called a *pose vector*) in a 67D space (called *pose space*). Consequently, any instance

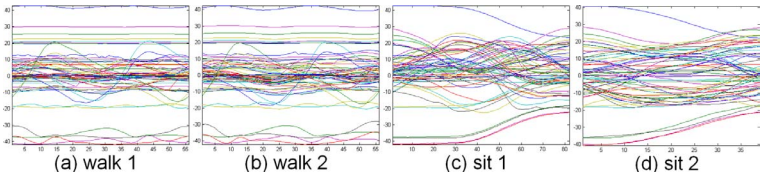

Fig. 3. Examples of action *walk* and *sit*. The horizontal axis is the frame number. The difference between the two *sit* sequences shown in (c) and (d) is significant. Note that the length of (c) and (d) is also different

of action can be viewed as a trajectory in the pose space. Fig.3 shows two instances of action *walk* and *sit*. Here for purpose of visualization, we overlay 67 1D trajectories of each single coordinate together. For example, the pose shown in Fig.2(c) corresponds to all points at the first frame of Fig.3(a).

An action recognition algorithm has to take dynamic information into consideration. Classification techniques based on static poses, such as Bayesian Network or *Support Vector Machine* (*SVM*), clearly won't work because different actions can share the same static poses. We tried a Bayesian network to classify static pose and found the classification accuracy to be less than 50%. Johansson's seminal study on *Moving Light Displays* (*MLDs*) [6] also confirms this result.

Hidden Markov Models (*HMM*) [10] is a powerful tool to capture dynamics of a time series. However, training a continuous *HMM* in such a high dimensional space requires large amount of training samples, which is not always available, especially for those infrequent actions. Other widely-used techniques include *Dynamic Time Warping* and template matching. Template matching is appealing to us because of its simplicity and less demanding requirement for training samples. Action recognition can be intuitively formulated as a template matching problem: If we are given one motion template (trajectory of pose vectors) of each action type, we can recognize an unknown sequence by comparing the trajectory of the unknown sequence with each template and find the best match using some distance measurement, e.g. *Sum of Squared Differences* (*SSD*).

However, this idea over-simplifies the problem because it does not consider noise and the spatial variation among the different instances of the same action type. For example, the idea may be able to recognize the *walk* sequence shown in Fig.3(b), given Fig.3(a) as the template, because these two sequences appear to be very similar. But there is apparent difference between the two *sit* sequences shown in Fig.3(c) and (d) and thus template matching based on simple distance measurement may misclassify this example.

Another problem is that trajectories have different discriminative power. For instance, the sine wave-like trajectories in the *walk* examples are salient and suitable for distinguishing *walk* from *sit*. The trajectories shown at the top and bottom of the *sit* examples look also unique. However, the difference between most of corresponding trajectories (those in the middle of each figure) in *walk* and *sit* can not be easily seen. Thus those trajectories are not good features for distinguishing *walk* and *sit*. Using all trajectories evenly for template match-

ing is undesirable because the discriminative power of those good features is compromised when the distance between two whole pose vectors is computed.

We address these two problems in next two sub-sections and we describe the recognition algorithm in section 4.

3.1 Learning Motion Channels

We define a motion channel as the information that encodes the evolution of a single joint coordinate for a specific action class. If we are given several training samples, the simplest form of a motion channel is the mean vector of the samples and *SSD* can be applied as the matching function. But as stated above, this can not accommodate noise and the spatial variation among real sequences. So we also include the variance information in the motion channel and use the following χ^2 function to measure the closeness, or fitting error in other words, of an unknown trajectory X to the motion channel C. μ and σ^2 are mean and variance vector of C. f is the frame index.

$$err(X,C) = \chi^2(X,C) = \frac{\sum_{f=1}^{L} \left(\frac{X_f - \mu_f}{\sigma_f}\right)^2}{L} \tag{1}$$

Unlike *SSD*, the χ^2 function considers the prior distribution of the training samples and consequently allows large deviation from the mean when the variance is also large. Note that Eq.1 is normalized by L, the length of the channel, to eliminate the discrepancy caused by different channels lengths. Fig.4(a) shows some training samples of one channel and (b) shows the corresponding mean vector μ and error bar $(\mu_f - \sigma_f, \mu_f + \sigma_f)$ at each frame.

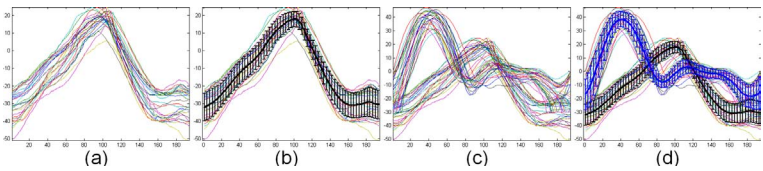

Fig. 4. (a) and (b) show some training samples of a channel with single cluster and the corresponding mean vector μ and error bar $(\mu_f - \sigma_f, \mu_f + \sigma_f)$ at each frame. (c) and (d) show some examples of a channel with multiple clusters and the clustering result using *k-means* algorithm

Eq.1 works fine when there is only one cluster. However, as shown in Fig.4(c), a channel may have multiple clusters (The training samples shown here contain two types of *sit*: *sit* on a high stool and *sit* on a step stool). So we use *k-means* algorithm to cluster the training data first and then find the closest match, as shown in Eq.2. μ_c and ${\sigma_c}^2$ are mean and variance of the c-th cluster of C.

$$err(X,C) = \min_c \frac{\sum_{f=1}^{L} \left(\frac{X_f - \mu_{c,f}}{\sigma_{c,f}}\right)^2}{L} \tag{2}$$

The number of clusters is set manually. Note that *k-means* algorithm takes vectors with length L as the input so the clustering result is not affected when two cluster centers are crossing at some frames, as shown in Fig.4(c) and (d).

We also tried other *soft* clustering algorithm such as Gaussian Mixture Model. However, in many cases, because we don't have enough training data (i.e. the number of training samples is less than 67), the covariance matrix of each Gaussian is singular and thus each training sample determinately belongs to one of clusters. Therefore the result is almost the same as *k-means*.

Temporal variation also exists, i.e. training samples contain different number of frames. In such cases, we first resample these trajectories using *Spline* interpolation so that they all have L frames. For *periodic* actions, we manually align training samples such that starting and ending phases are approximately the same.

3.2 Learning Motion Templates

We define a motion template as a set of N motion channels $\{C_j\}$ with weights $\{w_j\}$, j=1,2,...,N, which as a whole can capture the uniqueness of the motion of one specific action type. The error of fitting an unknown sequence Y (with length L) to template T is the weighted sum of fitting error of component Y_j to the corresponding channel C_j, as shown in Eq.3.

$$err(Y,T) = \sum_{j=1}^{N} w_j \cdot err(Y_j, C_j) \tag{3}$$

Weight w_j should reflect the discriminative power of C_j. We learn w_j according to the *Neyman-Pearson* criterion [14]: We impose constraint on classification rate and try to minimize false alarm rate. Suppose we have P training samples of T and Q training samples of all other templates, which are considered as P positive and Q negative training samples of T. Suppose $(e^+_{j,1}, e^+_{j,2}, ..., e^+_{j,P}, e^-_{j,1}, e^-_{j,2}, \ ... \ , e^-_{j,Q})$ are errors of fitting P positive and Q negative samples to channel C_j. We sort these P+Q values in ascending order and the sorted list is $(e_{j,1}, e_{j,2}, ..., e_{j,P+Q})$. If the classification rate is γ, we have to accept γP positive samples. Suppose the shortest sub-list that contains γP positive samples is $(e_{j,1}, e_{j,2}, ..., e_{j,R_j})$. It also contains $R_j - \gamma P$ negative samples. $\frac{R_j - \gamma P}{Q}$ is thus the minimum possible *false alarm* rate. Let w_j be the *true rejection* rate and it is normalized as follows:

$$w_j = \left(1 - \frac{R_j - \gamma P}{Q}\right) \Big/ \sum_{n=1}^{N} \left(1 - \frac{R_n - \gamma P}{Q}\right) \tag{4}$$

Here e_{j,R_j} defines an error threshold for channel C_j in that samples with a larger error are rejected. We also define an error threshold for template T: $\varepsilon = max\{e_{j,R_j}\}$, j=1, 2, ..., N, which will be used in the recognition algorithm.

It is important to note that although this algorithm does not consider the joint distribution of channels, the correlation between channels is actually preserved in that all channels are in nature synchronized by time. However, a counterexample would be: Suppose we have two channels A and B, each contains

two clusters $(A1, A2)$ and $(B1, B2)$. If the only possible combination of A and B is $(A1, B1)$ or $(A2, B2)$, an unknown sequence with combination $(A1, B2)$ or $(A2, B1)$ will generate a false alarm by the algorithm. Fortunately, in practice, such counterexample rarely exists (It never occurs in our dataset actually).

For *stationary* actions such as *stand*, the process is somewhat different in that there is no dynamic information. Therefore, the processing unit is a scalar value instead of a vector. This is actually a special case of the above algorithm: We can still use above equations except that L=1. Also when given training sequences, each frame is considered as one occurrence of the action. So P, the number of training samples, equals the total number of frames.

4 The Recognition Algorithm

Suppose we have trained a set of motion templates $\{T_i\}$ (with length L_i), the recognition becomes straightforward: Given an input sequence Y, at each frame t, we compare the fitting errors of Y within the sliding window with each of templates and the one with the minimum fitting error is considered to be the ongoing action, as shown in Eq.5.

$$action(Y) = \underset{i:err(Y,T_i)\leq\varepsilon_i}{\arg\min} \; (err(Y, T_i)) \tag{5}$$

Because there are three action groups *leg+torso, arm, head* and we only compare templates in the same group, there may be up to three actions recognized at one time. But if $err(Y, T_i) > \varepsilon_i$ for all T_i in one group, none of the actions in that group is chosen.

The correct recognition can not be done until the whole course of the action has been discovered. This means the previous L_i-1 frames may have already been recognized as something else. We keep track of the minimum error and the best action so far of each frame. At frame t, suppose T_i is the best match using the predictor in Eq.5 and the fitting error is $err_{T_i t}$. The program looks back and compares $err_{T_i t}$ with error stored in each frame of $[t - L_i + 1, t]$. If in one frame $err_{T_i t}$ is smaller, the minimum error in that frame is updated to $err_{T_i t}$ and the best action is changed to T_i. This method can correct wrong recognition of the previous frames but there is still L_i frames of delay.

As T_i has a fixed length L_i, if Y and T_i have different time scale (*e.g.* someone walks faster than others), the fitting error will be large even if Y and T_i represent the same action. So we resample T_i to form a series of templates $\{T_{i,L_{i_l}}\}_{l=1,2,\ldots,d_i}$. In our experiment, d_i=10.

For *stationary* actions, to make recognition results more stable, we impose a constraint such that the duration should be longer than some threshold L_δ. To clarify, we compute errors of fitting each frame in $[t - L_\delta + 1, t]$ to T_i using Eq.3. The overall error at current frame t is the mean of these L_δ fitting errors. We chose L_δ=60, which is equivalent to 2 seconds video in NTSC format.

5 Experimental Results

5.1 Results on MoCap Data

The 848 MoCap sequences in our dataset contain 159,564 frames in total. We manually segmented these sequences such that each segment contains a whole course of one action. In total we have 2343 action segments. The distribution of these segments in each action class is not uniform. *Walk* has 204 segments while *lie2stand* has only 19 segments. The average number is 107. The length of these segments are also different, ranging from 47 to 91 frames. The average is 68 frames.

In **Experiment 1**, we randomly selected half of segments of each action class for training and the remainder for classification. In **Experiment 2**, we reduced the amount of training data to 1/3. We repeated these experiments five times and the average classification rate of each class is shown in Fig.5(a). Here we chose γ=95% as the classification rate in training. The overall classification rate of *Exp.1* and 2 is 90.7% and 86.9%, respectively. As expected, the performance of the latter decreased, but not too much, which indicates the robustness in terms of the amount of available training data. Compared with *Exp.1*, most of the first 3 best channels (not shown due to limited space) for each action did not change (although the order may be different). This shows consistency of our algorithm in selecting good features.

Fig.5(b) shows the confusion matrix of the action group *leg+torso* in *Exp.1*. As indicated in those black grids on the diagonal, most of actions have been

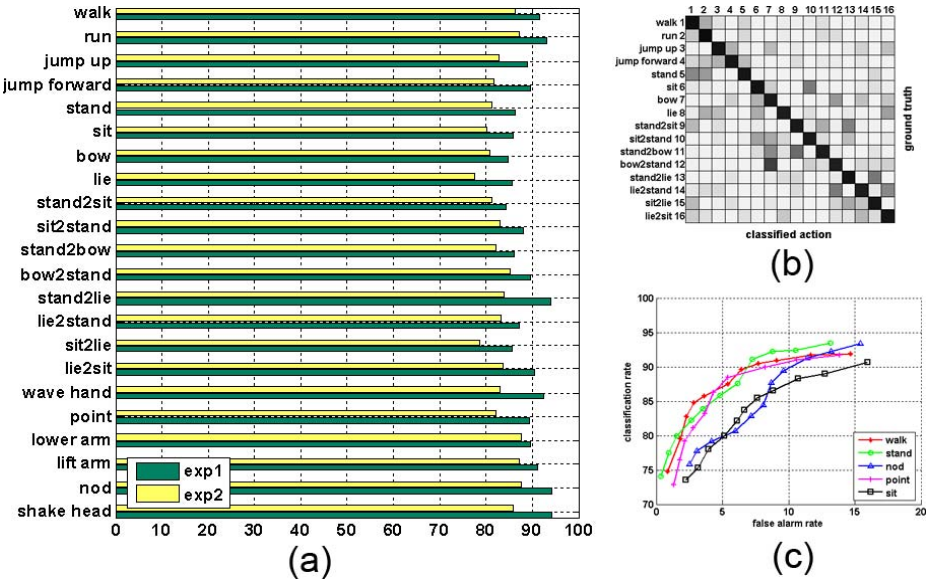

Fig. 5. (a)Classification result of *Exp.1* and 2; (b)Confusion matrix of the action group *leg+torso* in *Exp.1*. A dark grid indicates strong confusion. (c)ROC curves shows the influence of γ on the classification performance

correctly classified. Strong confusion (dark grids) only occurs between similar actions such as *bow* and *bow2stand*.

The classification rate γ plays an important role in our algorithm in that it affects not only the channel weights in training but also the error threshold ε_i in recognition. In **Experiment 3**, we used the same setup as in *Exp.1* except we chose different γ to see its influence. Although only in training phase can γ have a direct influence, it can end up affecting the classification rate and false alarm rate in testing. The ROC curves are shown in Fig.5(c). (Only some action types are shown here.) As expected, there is a tradeoff between a high classification rate and a low false alarm rate.

In **Experiment 4**, we tested our recognition algorithm on 73 unsegmented long sequences (from testing set) with average length of 914 frames. We use the templates learned in *Exp.1*. The algorithm achieves a recognition rate of 90.7% (in terms of frames).

Our algorithm is fast. Training about 75,000 frames takes about 28 minutes (computation time only, not counting the time of loading files) on a PC with a P4 2.4GHz CPU. Recognizing about 75,000 frames takes only 9.3 minutes.

5.2 Results on Annotated and Tracked Data

We tested our algorithm on two annotated (994 frames in total) and one automatically tracked (159 frames) sequence. Fig.6(a) shows some key frames of one annotated sequence. The rendered annotation results using POSER are displayed on the right. The text in top-right corner is the frame number with ground truth label and the text in bottom-right corner is the recognition result. The complete ground truth and the recognition result are shown in Fig.6(b).

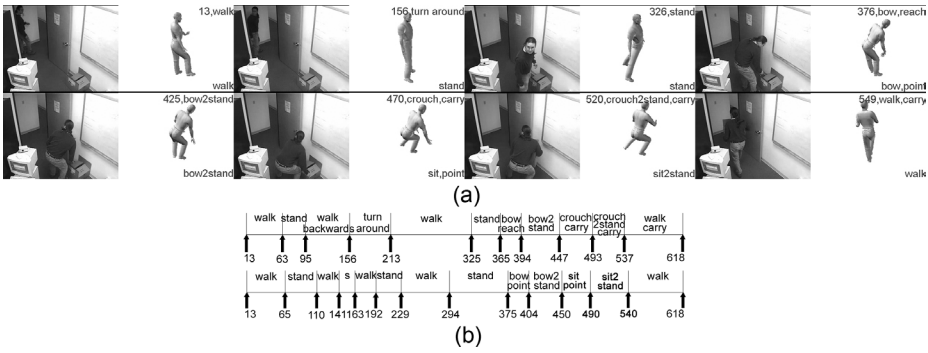

Fig. 6. (a)Key frames of one annotated video; (b)The ground truth (top) and recognition result (bottom)

Fig.6(b) shows that most of actions have been correctly recognized although the segmentation is not perfect. Errors occur when the subject turns around because we don't model such actions in our action set. "Carry" was not recognized

for the same reason. "Reach" and "crouch", however, were recognized as "point" and "sit", which are reasonable substitutions for "reach" and "crouch".

The recognition rate on the annotated and tracked data is 82.3% and 80.6%, respectively. This is satisfactory considering a substantial amount of jittery noises contained in the data (root mean square position error rates of about 10 pixels (∼10 cms) and joint angle errors of about 20°). The proposed algorithm in general is robust to this type of errors with short duration because the algorithm eliminates their effects on the overall fitting error by averaging all frames within the sliding window.

6 Conclusions

We have presented a learning-base algorithm to represent and recognize 3D human actions using spatio-temporal motion templates. Each template consists of a set of motion channels where each channel corresponds to the evolution of one 3D joint coordinate. Channels with strong discriminative power are highlighted according to *Neyman-Pearson* criterion. The recognition is based on the minimum weighted sum of χ^2 distance to each motion channel. This simple representation and recognition algorithm performs satisfactorily well in terms of fast speed and high recognition rate on a large set of human actions.

We separate action recognition from the motion recovery problem (though, the two may not be entirely independent) by using 3D data. This imposes one limitation on the proposed approach. However, good results on the noisy annotated and automatically tracked videos show the potential of using real data.

Our future work includes adding more arm-related actions and extending the algorithm to recognize actions at the semantical level.

References

1. A. Agarwal and B. Triggs. 3D Human Pose from Silhouettes by Relevance Vector Regression. In *Proc. of CVPR*, pp. 882-888, 2004.
2. L. Campbell and A. Bobick. Recognition of human body motion using phase space constraints. In *Proc. of ICCV*, pp. 624-630, 1995.
3. J. Davis and A. Bobick. The Representation and Recognition of Action Using Temporal Templates. In *Proc. of CVPR*, pp. 928-934, 1997.
4. K. Derpanis, R. Wildes and J. Tsotsos. Hand Gesture Recognition within a Linguistics-Based Framework. In *Proc. of ECCV*, pp. 282-296, 2004.
5. S. Gong, M. Walter and A. Psarrou. Recognition of temporal structures: Learning prior and propagating observation augmented densities via hidden Markov states. In *Proc. of ICCV*, pp. 157-162, 1999.
6. G. Johansson. Visual perception of biological motion and a model for its analysis. *Perception and Psychophysics* 14 (2) (1973) 201-211.
7. M.W. Lee and R. Nevatia. Dynamic Human Pose Estimation using Markov chain Monte Carlo Approach. In *Proc. of the IEEE Workshop on Motion and Video Computing* (WACV/MOTION'05), 2005.

8. A. Oikonomopoulos, I. Patras and M. Pantic. Spatiotemporal saliency for human action recognition. In *Proc. of IEEE Int'l Conf. on Multimedia and Expo* (ICME '05), 2005.
9. V. Parameswaran and R. Chellappa. View invariants for human action recognition. In *Proc. of CVPR*, pp. 613-619, 2003.
10. L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proc. of the IEEE*, 77(2):257-286, 1989.
11. C. Rao, A. Yilmaz and M. Shah. View-Invariant Representation and Recognition of Actions. In *Int'l Journal of Computer Vision* 50(2), Nov. 2002, pp. 203-226.
12. E. Shechtman and M. Irani. Space-Time Behavior Based Correlation. In *Proc. of CVPR*, pp. 405-412, 2005.
13. A. Shokoufandeh, S.J. Dickinson, C. Jonsson, L. Bretzner and T. Lindeberg. On the representation and matching of qualitative shape at multiple scales. In *Proc. of ECCV*, pp. 759-775, 2002.
14. V. Trees and H. L. Detection, Estimation and Modulation Theory, Part I. *John Wiley and Sons*, New York, 1968. ISBN 0-47109517-6.
15. Z. Zhang, Y. Wu, Y. Shan and S. Shafer. Visual panel: Virtual mouse keyboard and 3d controller with an ordinary piece of paper. In *Workshop on Perceptive User Interfaces*. ACM Digital Library, November 2001. ISBN 1-58113448-7.