Aalborg Universitet

Action Recognition using Motion Primitives

Moeslund, Thomas B.; Fihl, Preben; Holte, Michael Boelstoft

Publication date:
2006

Document Version
Publisher's PDF, also known as Version of record

[Link to publication from Aalborg University](#)

Citation for published version (APA):
Moeslund, T. B., Fihl, P., & Holte, M. B. (2006). *Action Recognition using Motion Primitives*. Paper presented at The 15th Danish conference on pattern recognition and image analysis, Copenhagen, Denmark.

Action Recognition using Motion Primitives

T.B. Moeslund, P. Fihl, M.B. Holte, T.B. Moeslund

Laboratory of Computer Vision and Media Technology
Aalborg University, Denmark
Email: tbm@cvmt.dk

Abstract. The number of potential applications has made automatic recognition of human actions a very active research area. Different approaches have been followed based on trajectories through some state space. In this paper we also model an action as a trajectory through a state space, but we represent the actions as a sequence of temporal isolated instances, denoted primitives. These primitives are each defined by four features extracted from motion images. The primitives are recognized in each frame based on a trained classifier resulting in a sequence of primitives. From this sequence we recognize different temporal actions using a probabilistic Edit Distance method. The method is tested on different actions with and without noise and the results show recognizing rates of 88.7% and 85.5%, respectively.

1 Introduction

Automatic recognition of human actions is a very active research area due to its numerous applications. As opposed to earlier the current trend is not as much on first reconstructing the human and the pose of his/her limbs and *then* do the recognition on the joint angle data, but rather to do the recognition directly on the image data, e.g., silhouette data [18] [17] or temporal templates[4] [1].

Common for these approaches is that they represent an action by image data from all frames constituting the action, e.g., by a trajectory through some state-space or a spatio-temporal volume. This means that the methods in general require that the applied image information can be extracted reliably in every single frame. In some situations this will not be possible and therefore a different type of approach has been suggested. Here an action is divided into a number of smaller temporal sequences, for example movemes [6], atomic movements [7], states [5], dynamic instants [13], examplars [11], behaviour units [9], and key-frames [8]. The general idea is that approaches based on finding smaller units will be less sensitive compared to approaches based on an entire sequence of information.

For some approaches the union of the units represents the entire temporal sequence, whereas for other approaches the units represent only a subset of the original sequence. In Rao *et al.* [13] dynamic hand gestures are recognized by searching a trajectory in 3D space (x and y-position of the hand, and time) for certain dynamic instants. Gonzalez *et al.* [8] look for key-frames for recognizing actions, like walking and running. Approaches where the entire trajectory (one action) is represented by a number of sub-sequences, are Barbic *et al.* [2] for full body motion, where probabilistic PCA is used

for finding transitions between different behaviors, and Bettinger *et al.* [3] where like-lihoods are used to separate a trajectory into sub-trajectories. These sub-trajectories are modeled by Gaussian distributions each corresponding to a temporal primitive.

2 Paper Content and System Design

In this paper we address action recognition using temporal instances (denoted primitives) that only represent a subset of the original sequence. That is, our aim is to recognize an action by recognizing only a few primitives as opposed to recognition based on the entire sequence (possibly divided into sub-trajectories).

Our approach is based on the fact that an action will always be associated with a movement, which will manifest itself as temporal changes in the image. So by measuring the temporal changes in the image the action can be inferred. We define primitives as temporal instances with a significant change and an action is defined as a set of primitives. This approach allows for handling corrupted input sequences and as we shall see, does not require the lengths, the start point, nor the end point to be known, which is the case in many other systems.

Measuring the temporal changes can be done in a number of ways. We aim at primitives that are as independent on the environment as possible. Therefore, we do not rely on figure-ground segmentation using methods like background subtraction or personalized models etc. Instead we define our primitives based on image subtracting. Image subtraction has the benefit that it measures the change in the image over time and can handle very large changes in the environment.

Concretely we represent our primitives by four features extracted from a motion-image (found by image subtraction). In each frame the primitive, if any, that best explains the observed data is identified. This leads to a discrete recognition problem since a video sequence will be converted into a string containing a sequence of symbols, each representing a primitive. After pruning the string a probabilistic Edit Distance classifier is applied to identify which action best describes the pruned string. The system is illustrated in figure 1. The actions that we focus on in this work are five one-arm gestures,
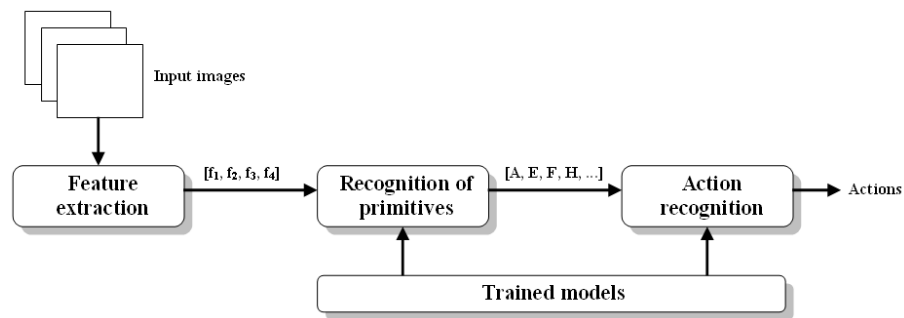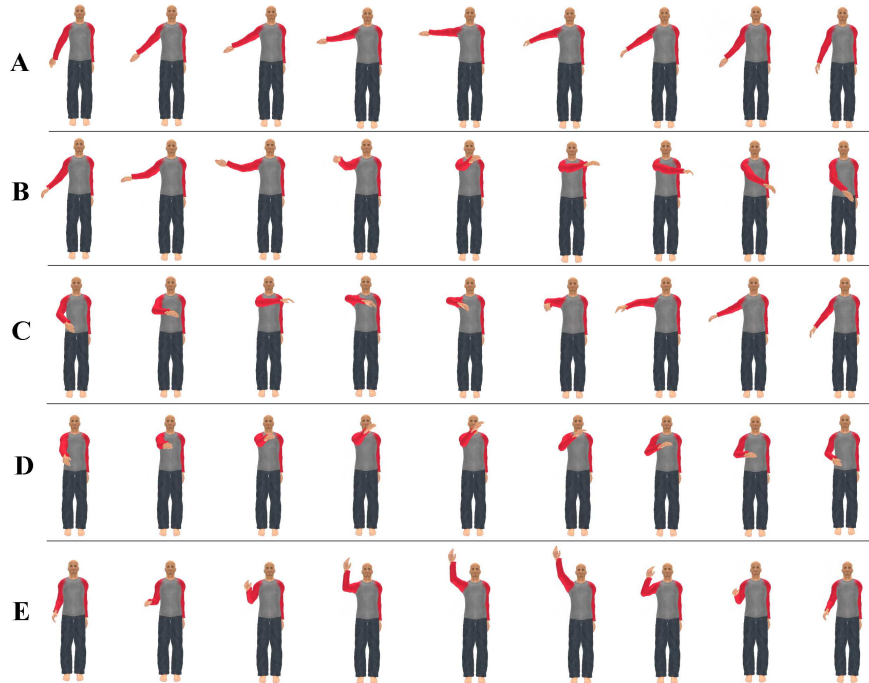
Fig. 1. System overview.

but the approach can with some modifications be generalized to body actions. The actions are inspired by [10] and can be seen in figure 2. The paper is structured as follows.

Fig. 2. Samples from the five actions. **A - Point right:** A stretched arm is raised to a horizontal position pointing right, and then lowered down. **B - Move left:** A stretched arm is raised to a horizontal position pointing right. The arm is then moved in front of the body ending at the right shoulder, and then lowered down. **C - Move right:** Right hand is moved up in front of the left shoulder. The arm is then stretched while moved all the way to the right, and then lowered down. **D - Move closer:** A stretched arm is raised to a horizontal position pointing forward while the palm is pointing upwards. The hand is then drawn to the chest, and lowered down. **E - Raise arm:** The arm is moved along the side of the person and stretched above the head, and then lowered again.

In section 3 we describe how our features are extracted. In section 4 we describe how we recognize the primitives, and in section 5 we describe how we recognize the actions. In section 6 the approach is evaluated on a number of actions and in section 7 the approach is discussed.

3   Feature Extraction

Even though image subtraction only provides crude information it has the benefit of being rather independent to illumination changes and clothing types and styles. Further-

more, no background model or person model is required. However, difference images suffer from "shadow effects" and we therefore apply double difference images, which are known to be more robust [19]. The idea is to use three successive images in order to create two difference images. These are thresholded and ANDed together. This ensures that only pixels that have changed in both difference images are included in the final output. Multiple steps between the three successive images used to generate the double difference image have been tried out (frame 1-2-3, frame 1-3-5, and frame 1-4-7, etc.). The approach is rather invariant to this choice, i.e., invariant to the frame-rate and the execution speed of the actions. Frame 1-3-5 are used in this work.

When doing arm gestures the double difference image will roughly speaking contain a "motion-cloud". However, noise will also be present. Either from other movements, e.g., the clothes on the upper body when lifting the arm (false positives), or the motion-cloud will be split into a number of separate blobs, e.g., due to the shirt having a uniform color (false negatives). Since the two noise sources "work against each other", it is difficult to binarize the difference image. We therefore apply a hysteresis principle consisting of two thresholds T_1 and T_2 with $T_1 > T_2$. For all difference pixels above T_1 we initiate a region growing, which continues to grow until the pixel values falls below T_2, see figure 3. The resulting connected motion components are further sorted in respect to their size to obtain robustness towards noise. This hysteresis threshold helps to ensure that noisy motion-clouds are not broken up into multiple fragments and at the same time eliminates small noisy motion blobs. The result is one connected motion-cloud. We model the motion-cloud compactly by an ellipse. The length and orientation
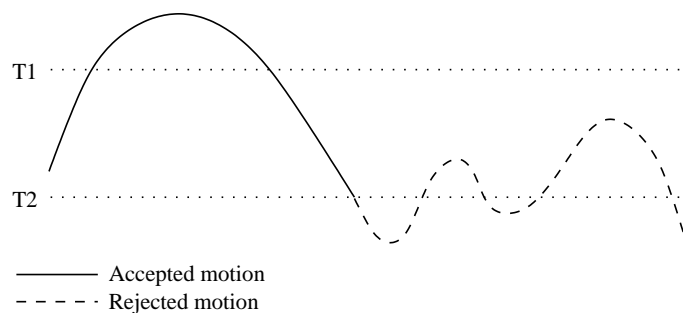
Fig. 3. An illustration of the hysteresis with an upper threshold T_1 and a lower threshold T_2. The figure illustrates the advantage of the hysteresis, where most of the "motion-blob" of interest is accepted while the smaller "noise-blobs" are rejected.

of the axes of the ellipse are calculated from the Eigen-vectors and Eigen-values of the covariance matrix defined by the motion pixels.

We use four features to represent this cloud. They are independent of image size and the person's position in the image. Furthermore, two are defined with respect to a reference point currently defined manually as the center of gravity of the person. The features are: the eccentricity of the ellipse, the orientation of the ellipse, the size of the

ellipse with respect to the distance from the reference point to the ellipse, and the angle between the reference point and the ellipse.

4   Recognition of Primitives

Each incoming frame is represented by the four extracted features described above. In this block the feature vector is classified as a particular primitive or noise. A Mahalanobis classifier is build by forming the covariance matrix for each primitive based on a set of representative training examples, see below. The four features are not equally important and therefore weighted in accordance with their importance. This yields the following classifier for recognizing a primitive at time, t:

$$\text{Primitive}(t) = \arg\min_{i} \left[(\boldsymbol{W} \cdot (\boldsymbol{f}_t - \boldsymbol{p}_i))^T \Pi_i^{-1} (\boldsymbol{W} \cdot (\boldsymbol{f}_t - \boldsymbol{p}_i)) \right] \tag{1}$$

where $\boldsymbol{f}_t$ is the feature vector estimated at time t, $\boldsymbol{p}_i$ is the mean vector of the ith primitive, Π_i is the covariance matrix of the ith primitive, and $\boldsymbol{W}$ contains the weights and are included as an element-wise multiplication.

The classification of a sequence can be viewed as a trajectory through the 4D feature space where, at each time-step, the closest primitive (in terms of Mahalanobis distance) is found. To reduce noise in this process we introduce a minimum Mahalanobis distance in order for a primitive to be considered in the first place. Furthermore, to reduce the flickering observed when the trajectory passes through a border region between two primitives we introduce a hysteresis threshold. It favors the primitive recognized in the preceding frame over all other primitives by modifying the individual distances. The classifier hereby obtains a "sticky" effect, which handles a large part of the flickering.

After processing a sequence the output will be a string with the same length as the sequence. An example is illustrated in equation 2. Each letter corresponds to a recognized primitive and $\varnothing$ corresponds to time instances where no primitives are below the minimum required Mahalanobis distance. The string is pruned by first removing '$\varnothing$'s, isolated instances, and then all repeated letters, see equation 3. A weight is generated to reflect the number of repeated letters (this is used below).

$$\text{String} = \{\varnothing, \varnothing, B, B, B, B, B, E, A, A, F, F, F, F, \varnothing, D, D, G, G, G, G, \varnothing\} \tag{2}$$
$$\text{String} = \{B, A, F, D, G\} \tag{3}$$
$$\text{Weights} = \{5, 2, 4, 2, 4\} \tag{4}$$

4.1   Learning Models for the Primitives

In order to recognize the primitives we need to have a prototypical representation of each primitive, i.e., a mean and covariance in the 4D feature space. As can be seen in figure 2 the actions are all fronto-parallel.

Ongoing work aims at generalizes this work by allow for multiple viewpoints. One problem with this is how to train the system - it will require a very large number of test sequences. Therefore we have captured all training data using a magnetic tracking system with four sensors. The sensor placements are: one at the wrist, one at the elbow, one at the shoulder, and one at the upper torso (for reference). The hardware used is the

Polhemus FastTrac [15] which gives a maximum sampling rate of 25Hz when using all four sensors. The data is converted into four Euler angles: three at the shoulder and one at the elbow in order to make the data invariant to body size. An action corresponds to a trajectory through a 4D space spanned by the Euler angles.

The data is input to a commercial computer graphics human model, Poser [16], which then animates all captured data. This allows us to generate training data for any view point and to generate additional training data by varying the Euler angles (based on the training data) and varying the clothing of the model. Figure 4 shows a person with magnetic trackers mounted on the arm, two different visualizations of the 3D tracker data from Poser, and an example of the test data. Based on this synthetic training data we build a classifier for each primitive.

Fig. 4. An illustration of the different types of data used in the system. From left to right: 1) 3D tracker data is acquired from magnetic trackers mounted on persons who perform the five actions. 2) The tracker data is animated in Poser from a fronto-parallel view. 3) The tracker data can be animated from any view point with different clothings and models. 4) After training the primitives on semi-sythetic data we recognize actions in real video.

### 4.2 Defining the Primitives

Defining the number of primitives and their characteristics ("human movement") is quite a significant optimization problem. We are aiming at automating this process [14], but in this work it was done manually.

The primitives are defined based on an evaluation of video sequences showing three different people performing the five actions. The criteria for defining the primitives are 1) that they represent characteristic and representative 3D configurations, 2) that their projected 2D configurations contain a certain amount of fronto-parallel motion, and 3) that the primitives are used in the description of as many actions as possible, i.e., fewer primitives are required. In this way we find 10 primitives that can represent the five actions. Each primitive is appearing in several actions resulting in five to eight primitives for each action.

To obtain the prototypical representation we randomly select 20 samples of each primitive from the training video sequences. The double difference images of these samples are calculated and the motion-clouds are each represented by the four features.

The 20 samples then yields a mean vector and a 4x4 covariance matrix for each primitive.

In figure 4 the 10 primitives and their representations are visualized together with the letter denoting the primitive.
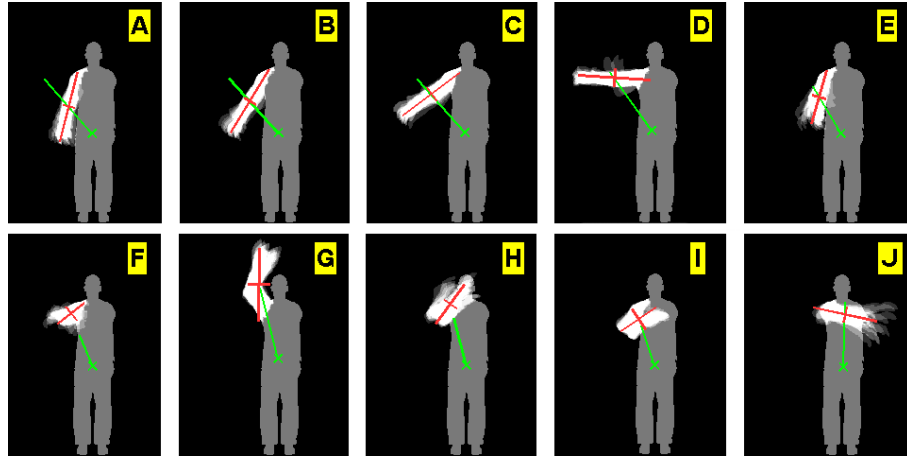
Fig. 5. The figure of each primitive contains the silhouettes of the 20 samples added together which gives the gray silhouette. The 20 motion clouds from the double difference images of the samples are added on top of the silhouette as the white cloud. The figures furthermore illustrates the mean of the four features for each primitive by depicting the axes of the fitted ellipse and the distance and direction from the reference point to the motion cloud.

5 Recognition of Actions

The result of recognizing the primitives is a string of letters referring to the known primitives. During a training phase a string representation of each action to be recognized is learned. The task is now to compare each of the learned actions (strings) with the detected string. Since the learned strings and the detected strings (possibly including errors!) will in general not have the same length, the standard pattern recognition methods will not suffice. We therefore apply the Edit Distance method [12], which can handle matching of strings of different lengths.

The edit distance is a well known method for comparing words or text strings, e.g., for spell-checking and plagiarism detection. It operates by measuring the distance between two strings in terms of the number of operations needed in order to transform one to the other. There are three possible operations: *insert* a letter from the other string, *delete* a letter, and *exchange* a letter by one from the other string. Whenever one of these operations is required in order to make the strings more similar, the score or distance is increased by one.

When the strings representing the actions are of different lengths, the method tends to favor the shorter strings. Say we have detected the string $\{B, C, D\}$ and want to

classify it as being one of the two actions: $\#1 = \{J, C, G\}$ and $\#2 = \{A, B, C, D, H\}$. The edit distance from the detected string to the action-strings will be two in both cases. However, it seems more likely that the correct interpretation is that the detected string comes from action #2 in a situation where the start and end has been corrupted by noise. In fact, 2 out of 3 of the primitives have to be changed for action #1 whereas only 2 out of 5 have to be changed for action #2. We therefore normalize the edit distance by dividing the output by the length of the action-string, yielding 0.67 for action #1 and 0.2 for action #2, i.e., action #2 is recognized.

The edit distance is a deterministic method but by changing the cost of each of the three operations with respect to likelihoods it becomes a probabilistic method[1]. Concretely we apply the weights described above, see equation 4. These to some extent represent the likelihood of a certain primitive being correct. The higher the weight the more likely a primitive will be. We incorporate the weights into the edit distance method by increasing the score by the weight multiplied by β (a scaling factor) whenever a primitive is *deleted* or *exchanged*. The cost of *inserting* remains 1.

The above principle works for situations where the input sequence only contains one action (possibly corrupted by noise). In a real scenario, however, we will have sequences which are potentially much longer than an action and which might include more actions after each other. The action recognition problem is therefore formulated as for each action to find the substring in the detected string, which has the minimum edit distance. The recognized action will then be the one of the substrings with the minimum distance. Denoting the start point and length of the substring, s and l, respectively, we recognize the action present in the detected string as:

$$\text{Action} = \arg\min_{k,s,l} PED(\Lambda, k, s, l) \tag{5}$$

where k index the different actions, Λ is the detected string, and $PED(\cdot)$ is the probabilistic edit distance.

6 Results

6.1 Test Setup

Two kind of tests are conducted: one with known start and stop time of action execution, and another with "noise" added in the beginning and end of the sequences (unknown start time). By adding noise to the sequence we introduce the realistic problem of having no clear idea about when an action commence and terminates which would be the case in a real situation. To achieve a test scenario that resembles this situation we split the five actions into halves and add one of these half actions randomly to the beginning and one to the end of each action to be processed by the system. In this way we get an unknown start and end point of the real action.

We use eleven test subjects, whom each performs each gesture 10 times. This leads to 550 sequences. The weighting of the features $\boldsymbol{W}$ are set to $\{1, 4, 2, 4\}$, and $\beta = 1/8$. A string representation of each action is found and since the shortest string contains five primitives and the longest eight primitives, we only perform the probabilistic edit distance calculation for substrings having the lengths $\in [3, 15]$.

---

[1] This is related to the Weighted Edit Distance method, which however has fixed weights.

6.2 Tests

The overall recognition rate for the test with known start time is 88.7%. In figure 6(a) the confusion matrix for the results is shown. As can be seen in the figure, most of the errors occur by miss-classification between the two actions: *move closer* and *raise arm*. The main reasons for this confusion are different performances by the test subjects (some do not raise their arm very much when preforming the *raise arm* action), the similarity of the actions, and the similarity of the primitives in these actions. As can be seen in figure 2 both actions are performed along the side of the person when seen from the fronto-parallel view and differs mainly in how high the arm is raised. From figure 5 it can be seen that primitives 'F', 'G', 'H', and 'I' have similar angles between the reference point and the motion cloud and 'F', 'H' and 'I' also have similar orientation of the ellipse. These two features, which are the ones with highest weights, make these four primitives harder to distinguish.

Figure 6(b) shows the confusion matrix for the test results with noise. The overall recognition rate for this test is 85.5%, which is 3.2% lower than without noise. The errors are the same as before but with some few additional errors caused by the unknown start and end time of the actions.

	1.	2.	3.	4.	5.
1. Point right	110				
2. Move left	7	100		3	
3. Move right	6		102	1	1
4. Move closer		3	2	78	26
5. Raise arm				12	98

(a) Known start and stop time.

	1.	2.	3.	4.	5.
1. Point right	109		1		
2. Move left	10	99		1	
3. Move right	8		99	3	
4. Move closer	1	5	2	69	33
5. Raise arm	2	2		12	94

(b) Unknown start and stop time.

Fig. 6. The confusion matrix for the recognition of the different actions with and without noise.

7 Conclusion

In this paper we have presented an action recognition approach based on motion primitives as opposed to trajectories. Furthermore, we extract features from temporally local motion as opposed to background subtraction or another segmentation method relying on learned models and a relatively controlled environment. We hope this makes our approach less sensitive, but have still to prove so in a more comprehensive test.

The results are promising due to two facts. First, the models are generated from synthetic data (generated based on test subjects) while the test data are real data. In fact, the test data and training data are recorded several months apart, hence this is a real test of the generalization capabilities of the action recognition process. This means that we can expect to use the same scheme when learning models for the next incarnation of the system, which is aimed at view-invariant action recognition. Secondly, the system does not break down when exposed to realistic noise. This suggests that the approach taking has potential to be expanded into a real system setup, as opposed to a lab setup which is virtually always used when testing action recognition systems.

The primitives used in this work are found manually. This turned out to be quite an effort due to the massive amount of data and possibilities. Currently we are therefore working to automate this process [14]. Another ongoing activity is to avoid manually defining the reference point, see section 3, by using the face (found by an Adaboost trained face detector) as a reference for the features.

References

1. R.V. Babu and K.R. Ramakrishnan. Compressed domain human motion recognition using motion history information. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, Hong Kong, April 6-10, 2003.
2. J. Barbic, N.S. Pollard, J.K. Hodgins, C. Faloutsos, J-Y. Pan, and A. Safonova. Segmenting Motion Capture Data into Distinct Behaviors. In *Graphics Interface*, London, Ontario, Canada, May 17-19 2004.
3. F. Bettinger and T.F. Cootes. A Model of Facial Behaviour. In *IEEE International Conference on Automatic Face and Gesture Recognition*, Seoul, Korea, May 17 - 19 2004.
4. A. Bobick and J. Davis. The Recognition of Human Movement Using Temporal Templates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(3), 2001.
5. A.F. Bobick and J. Davis. A Statebased Approach to the Representation and Recognition of Gestures. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(12), 1997.
6. C. Bregler. Learning and Recognizing Human Dynamics in Video Sequences. In *Conference on Computer Vision and Pattern Recognition*, pages 568 – 574, San Juan, Puerto Rico, 1997.
7. L. Campbell and A. Bobick. Recognition of Human Body Motion Using Phase Space Constraints. In *International Conference on Computer Vision*, Cambridge, Massachusetts, 1995.
8. J. Gonzalez, J. Varona, F.X. Roca, and J.J. Villanueva. *aSpaces*: Action spaces for recognition and synthesis of human actions. In *AMDO*, pages 189–200, AMDO02, 2002.
9. O.C. Jenkins and M.J. Mataric. Deriving Action and Behavior Primitives from Human Motion Data. In *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, pages 2551–2556, Lausanne, Switzerland, Sept.30 – Oct.4, 2002.
10. A. Just and S. Marcel. HMM and IOHMM for the Recognition of Mono- and Bi-Manual 3D Hand Gestures. In *ICPR workshop on Visual Observation of Deictic Gestures (POINTING04)*, Cambridge, UK, August 2004.
11. A. Kale, N. Cuntoor, and R. Chellappa. A Framework for Activity-Specific Human Recognition. In *International Conference on Acoustics, Speech and Signal Processing*, Orlando, Florida, May 2002.
12. V.I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965.
13. C. Rao, A. Yilmaz, and M. Shah. View-Invariant Representation and Recognition of Actions. *Journal of Computer Vision*, 50(2):55 – 63, 2002.
14. L. Reng, T.B. Moeslund, and E. Granum. Finding Motion Primitives in Human Body Gestures. In S. Gibet, N. Courty, and J.-F. Kamps, editors, *GW 2005*, number 3881 in LNAI, pages 133–144. Springer Berlin Heidelberg, 2006.
15. *http://polhemus.com/*, January 2006.
16. *http://www.poserworld.com/*, January 2006.
17. D. Weinberg, R. Ronfard, and E. Boyer. Motion History Volumes for Free Viewpoint Action Recognition. In *IEEE Int. Workshop on Modeling People and Human Interaction*, 2005.
18. A. Yilmaz and M. Shah. Actions Sketch: A Novel Action Representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, San Diego, CA, June 20-25, 2005.
19. K. Yoshinari and M. Michihito. A Human Motion Estimation Method using 3-Successive Video Frames. In *Int. Conf. on Virtual Systems and Multimedia*, Gifu, Japan, 1996.