

Action Recognition in Semi-synthetic Images using Motion Primitives

P. Fihl, M.B. Holte, T.B. Moeslund, L. Reng
Laboratory of Computer Vision and Media Technology
Aalborg University, Denmark
Email: tbm@cvmtdk

June 26, 2006

Abstract

This technical report describes an action recognition approach based on motion primitives. A few characteristic time instances are found in a sequence containing an action and the action is classified from these instances. The characteristic instances are defined solely on the human motion, hence motion primitives. The motion primitives are extracted by double difference images and represented by four features. 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 method is evaluated on five one-arm gestures. A test is performed with semi-synthetic input data achieving a recognition rate of 96.5%.

1 System overview

This technical report describes the details about an action recognition approach based on motion primitives. This approach does not rely on first reconstructing the human and the pose of his/her limbs and then do the recognition on the joint angle data, but rather on recognition directly on the image data. We find a few characteristic time instances in an image sequence of a person performing an action and the action is classified from these instances.

In this approach the characteristic instances are defined solely on the human motion, hence motion primitives. The presented system recognizes a set of five one arm gestures. The gestures are defined as *point right*, *point forward*, *move left*, *move right*, and *move closer*. A more specific definition with examples can be seen in figure 5.

This technical report will describe the system and provide details on some of the considerations made during the implementation of the system.

The system consists of four modules (see figure 1): Motion detection, motion representation, recognition of primitives, and action recognition.

Motion is detected by double difference images which are then filtered with morphological filters to obtain a more stable cloud of motion pixels. The detected motion is then represented with four features describing the position and shape of the motion cloud in a scale invariant manner. Recognition of primitives calculates the Mahalanobis distance between each set of four features and the primitives defined in the training phase. Each set of features is in this way classified as a primitive. A set of primitives forming a string is input to the action recognition where the string is pruned and weighted before calculating the edit distance between the new set of primitives and each of the defined actions.

The following will describe each module in greater detail. The basis of the work described in the technical report is mainly semi-synthetic images, but considerations regarding real images are made where appropriate. The

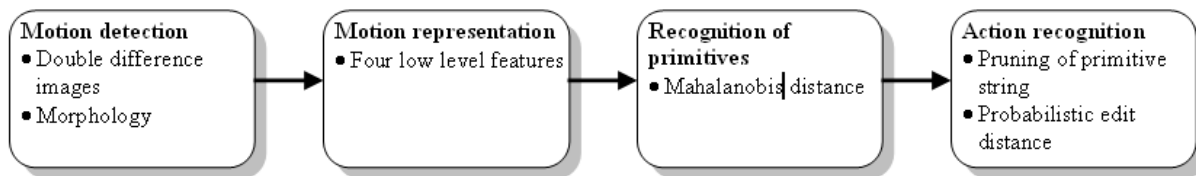


Figure 1: An overview of the modules in the action recognition method.

semi-synthetic images are based on real motion data from a magnetic tracking system which is visualized with commercial software. In this way we get a real movement and at the same time we are able to control camera positions, lights, clothes of the model etc.

A review of the related literature can be found in [1].

2 Motion detection

We calculate double difference images to detect motion in the images [7]. The double difference images are rather independent to illumination changes and clothing types and styles. Furthermore, no background model or person model is required and double difference images are more robust than difference images. The double difference images used to detect motion in frame number f are calculated from frames $f-2$, f , and $f+2$. This is done to ensure that the motion cloud of the double difference image represents the whole moving arm and not just a few pixels along the arm. Using frames $f-2$, f , and $f+2$ minimizes the average edit distance for the best matches, meaning that each action is classified with greater certainty.

Before calculating the double difference images we apply a threshold to the difference images to eliminate the weakest motion (which is assumed to be noise). When working on purely synthetic images the best results are obtained by setting this threshold to 0¹ because all motion in the image is originating from the arm movement and because no noise is present in the image. When working with real images this threshold will need to be adjusted and this will have a rather large effect on the shape of the motion cloud.

The morphological filters applied to the double difference images are designed to close holes in the motion

¹The motion threshold is measured in gray scale values.

cloud and to have the motion cloud represent the whole arm (including the shoulder). The current morphological filters are:

```

dilate( square, size 5 )
erode( square, size 7 )
dilate( square, size 3 )
  
```

These filters will need to be reanalyzed and possibly redesigned when the system is to work on real video instead of synthetic video.

3 Motion representation

To represent motion we calculate four low level features. To make features scale and translation invariant they are defined relative to a reference point. This reference point is currently defined as the center of gravity of the silhouette of the person².

The four features are calculated from the largest motion blob. The area with the most motion is assumed to be the area around the arm so this is done to help eliminate noise from eg. head movement³.

An ellipse is fitted to the motion cloud. The length and orientation of the axes of the ellipse are calculated from the Eigen-vectors and Eigen-values of the covariance matrix defined by the motion pixels. The four features are calculated as listed here (illustrated in figure 2):

1. The eccentricity of the motion cloud defined as the ration between the minor and major axes of the el-

²The reference point could probably be found in a more stable way by using the median. To avoid background subtraction altogether the reference point could also be defined as a point in the face which could be tracked.

³This is of cause not important in synthetic images but very important in real images.

lipse.

$$\text{Eccentricity} = \frac{\text{Minor axis length}}{\text{Major axis length}}$$

2. The orientation ϕ of the ellipse.
3. The ratio r between the length of the major axis and the distance d from the reference point to the center of the ellipse.

$$r = \frac{\text{Major axis length}}{d}$$

4. The angle θ between the reference point and the center of the ellipse.

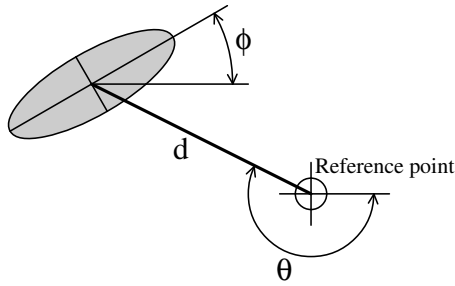


Figure 2: An illustration of the four features used to describe the primitives.

Feature two, the orientation of the ellipse θ , does require some additional calculations. Rotating the ellipse 180 degrees will result in the same angle for θ and since we want positive values we find θ in the range $[0;180]$ degrees. The problem arises when the angle is close to 0 and 180. The problem is depicted in figure 3 where the two ellipses have comparable orientation but very different θ (shown in red).

The problem is solved differently in training of the primitives and in the classification process.

In training we have several samples of feature two for each primitive so situations where the problem occurs can be detected and θ can be corrected if needed ($\theta + 180$). This is necessary to get the correct mean value and variance.

In the classification process there is no way of determining whether θ , $\theta + 180$, or $\theta - 180$ is the correct value

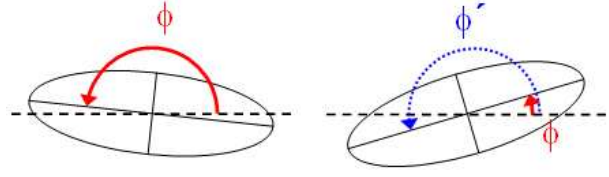


Figure 3: An illustration of the *wrap around* problem of feature two. See text for details.

so the Mahalanobis distance is calculated with all three values, and the value of feature two giving the smallest Mahalanobis distance is used. This can introduce errors but it is assumed that the influence of the other three features will minimize the errors.

A similar *wrap around* problem is present with feature four but because of the current placement of the reference point the problems never occurs.

4 Recognition of primitives

We define a set of primitives so that each primitive corresponds to a characteristic arm configuration. Each action consists of 5-8 primitives but each primitive can be a part of more than one action. In this way we can represent the five action with only 10 primitives.

Each primitive is represented by the four mentioned features. From 20 samples of each primitive we calculate the mean values and the covariance matrices of the feature vector.

First, the recognition of the primitives was carried out by evaluating the Euclidean distance between a set of features extracted from an image, and a set of features from representative primitive examples. Later, the recognition was improved by introducing a Mahalanobis classifier based on the covariance of the data.

The features are weighted in accordance with their importance, yielding the following classifier for recognizing a primitive at time, t :

$$\text{Primitive}(t) = \arg \min_i (\mathbf{W} \cdot (\mathbf{f}_t - \mathbf{p}_i))^T \Pi_i^{-1} (\mathbf{W} \cdot (\mathbf{f}_t - \mathbf{p}_i)) \quad (1)$$

where \mathbf{f}_t is the feature vector estimated at time t , \mathbf{p}_i is the mean vector of the i th primitive, Π_i is the covariance

matrix of the i th primitive, and W contains the weights and are included as an element-wise multiplication. Only the diagonal of Π is used. This gives a more stable result. The best results on synthetic data were obtained with $W = [1 \ 1 \ 2 \ 8]^T$ ⁴. However, all features are needed even though feature four is more important than the other features.

To reduce noise a minimum Mahalanobis distance is introduced. Thus, all primitive matches with a too large distance in respect to the trained primitive classes will not be considered. Additionally, a hysteresis threshold is applied to reduce the flickering effect, which occur when the primitive matches is located at a border region between multiple primitive classes. This hysteresis threshold favors the primitive recognized in the preceding frame.

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 \emptyset corresponds to time instances where no primitives are below the minimum required Mahalanobis distance. The string is pruned by first removing \emptyset 's, isolated instances, and then all repeated letters, see equation 3. A weight is generated to reflect the number of repeated letters.

$$\text{String} = \{\emptyset, \emptyset, B, B, B, B, B, E, A, A, F, \quad (2)$$

$$F, F, F, \emptyset, D, D, G, G, G, G, \emptyset\}$$

$$\text{String} = \{B, A, F, D, G\} \quad (3)$$

$$\text{Weights} = \{5, 2, 4, 2, 4\} \quad (4)$$

5 Action recognition

To recognize an action, the computed string of primitives is compared to a set of learned action-strings⁵. For this purpose a probabilistic Edit Distance is used, which can handle matching of strings with different lengths.

The edit distance[3] 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

⁴Many combinations of weights have been tested to find the optimal set of weights.

⁵It is assumed that each action can be described by a unique string.

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. The algorithm is illustrated in figure 4 where the strings *motions* and *octane* are compared.

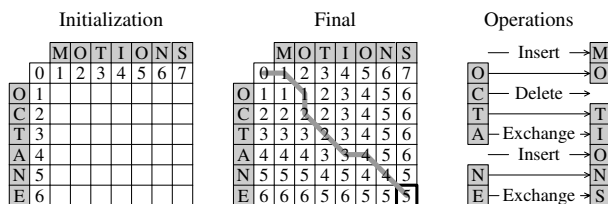


Figure 4: Measuring the distance between two strings using edit distance.

The first step is initialization. The two strings are placed along the sides of the matrix, and increasing numbers are placed along the borders beside the strings. Hereafter the matrix is filled cell by cell by traversing one column at a time. Each cell is given the smallest value of the following four operations:

Insert: The value of the cell above + 1

Delete: The value of the cell to the left + 1

Exchange: The value of the cell up-left + 1

No change: The value of the cell up-left + 0. This is the case when the letters in question in the two strings are the same.

Using these rules the matrix is filled and the value found at the bottom right corner is the edit distance required in order to map one string into the other, i.e., the distance between the two strings. The actual sequence of operations can be found by back-tracing the matrix. Note that often more paths are possible.

When the strings representing the actions are of different lengths, the method tends to favor the shorter strings. To handle this problem we normalize the edit distance by dividing it by the length of the action-string.

By applying the weights in equation 4 to represent the likelihoods, the method becomes probabilistic. We incorporate the weights into the edit distance method by in-

creasing the cost by β^6 multiplied by the weight whenever a primitive is *deleted* or *exchanged*. The cost of *inserting* remains 1. This probabilistic edit distance improves the action classification, but it still has its weak points. The cost of keeping a letter is zero so if noise fits with the string of an action it will result in a smaller edit distance. The weight of the noisy letter might indicate that it is in fact noise, but since no deletion, exchange, or insertion is needed the weight is never applied.

In a real scenario, 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 substring with the minimum distance.

Two approaches have been examined for this purpose:

1. Introducing a start/stop-primitive to determine the beginning and end of an action in the primitive-string.
2. Using variable start point and length of the substring.

The second approach is the most appropriate, since the first one restricts the actions which can be recognized and depends on that the start/stop-primitive always can be found.

Denoting the start point and length of the substring, s and l , respectively, we recognize the action present in the detected string as:

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

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

6 Test

6.1 Test Setup

To evaluate our approach we use five arm gestures inspired by [2, 4], see figure 5. The test is performed with

⁶Currently, β is set to $\frac{1}{8}$. This value has been found to be optimal through tests.

semi-synthetic data. Concretely we use a magnetic tracking system with four sensors to capture movements of the test subjects. 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 [5] 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 trajectory is used to generate a sequence of synthetic images using Poser 6’s FireFly Render Engine [6].

We use seven test subjects, who each perform each gesture 20 times. This leads to 840 synthetic sequences. We manually evaluate the sequences from three of the test subjects and found 10 primitives to describe the five different actions. The criteria for finding 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. Through an iterative evaluation of the primitives the number of primitives was reduced from initially 16 to the current 10. Two of the 10 primitives does however have *double representation*, meaning that a single primitive consists of two of the original primitives. In recognition of primitives the two original primitives are treated as separate primitives but because they bear strong resemblance to each other they are treated as one in the action recognition process.

Based on the manually selected primitives we randomly choose 20 sequences for each primitive. The sequences are aligned temporally and the double difference images are calculated and represented by the four features, yielding a 4x4 covariance matrix for each primitive. The maximum Mahalanobis distance for primitive recognition is set to 40, the weighting of the features are $\{1, 1, 2, 8\}$, 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 $\in [4, 16]$.

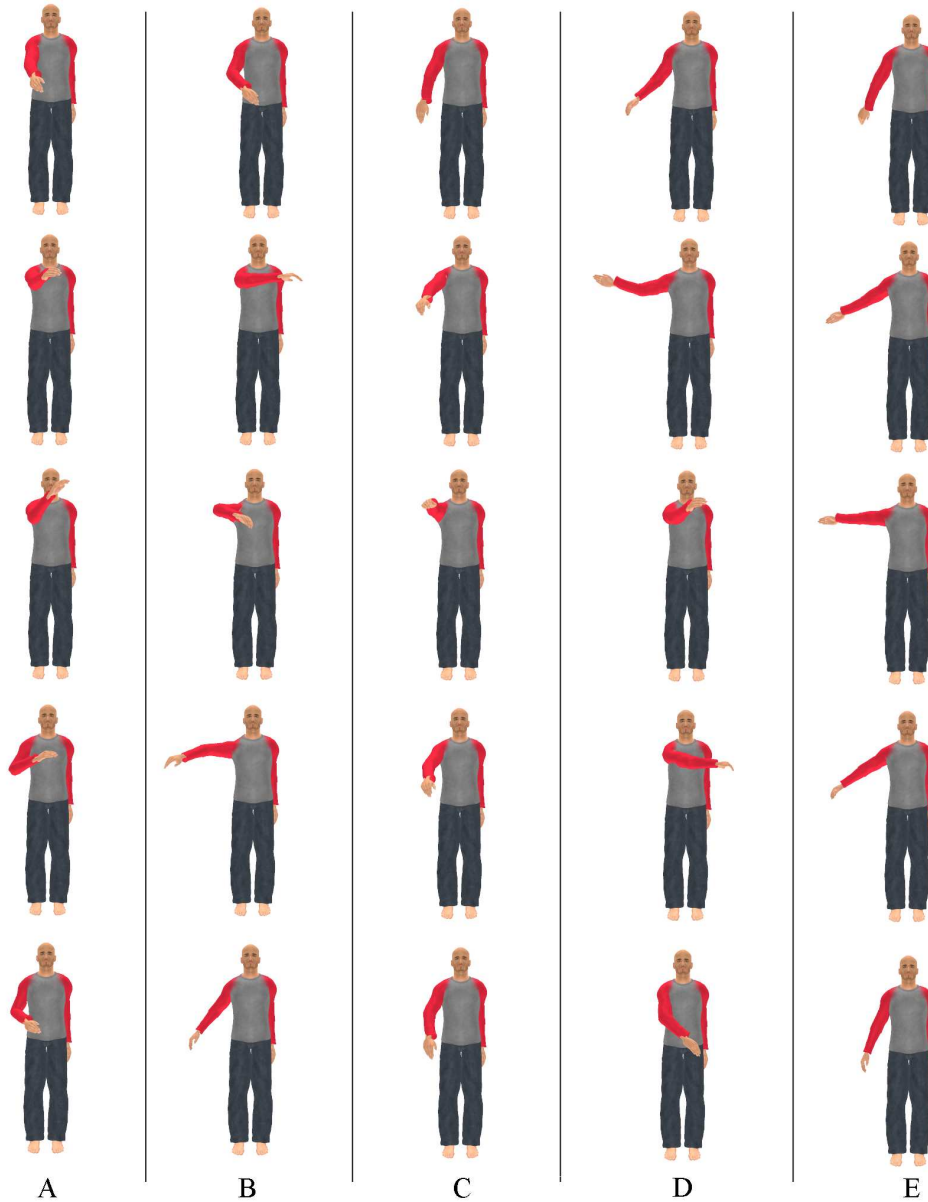


Figure 5: Examples of images generated by Poser using real motion captured data. Each column shows samples from the five gestures. **A - Move closer:** A stretched arm is raised to a horizontal position pointing forward while the palm is pointing upwards. The hand is then drawn to the chest, and lowered down. **B - 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. **C - Point forward:** A stretched arm is raised to a horizontal position pointing forward, and then lowered down. **D - 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. **E - Point right:** A stretched arm is raised to a horizontal position pointing right, and then lowered down.

6.2 Tests

The tests are performed on the four test subjects not included in the training data. We randomly choose 23 sequences of each gesture, yielding 115 test sequences. For each sequence we add "noise" in both the beginning and end of the sequence. The noise is in the form of approximately half a sequence of a different gesture. This introduces the realistically problem of having no clear idea when an action commence and terminates.

The overall recognition rate is 96,5%. The confusion matrix can be seen in figure 6.

	1.	2.	3.	4.	5.
1. Point right	23				
2. Point forward		23			
3. Move left		1	22		
4. Move right				23	
5. Move closer		3			20

Figure 6: The confusion matrix for the recognition of the different actions.

The confusion matrix shows that the action *move closer* is recognized as *point forward* in three cases. The person performing the move closer action in these three cases performs the action in a way very similar to the point forward action which makes them very hard to distinguish.

7 Future work

To show the strength of the action recognition approach presented in this technical report it is necessary to have non-synthetic input images. This will give some new challenges. To be capable of handling these challenges, the following assignments needs to be addressed:

- Record new real video sequences.
- Segmentation of reference point in real images
 - Fixed reference point.
 - Uniform white/black background.
 - Manual segmentation.
 - Using a head-tracking algorithm.

- Segmentation of motion pixels in double difference images.
 - Body/head movements.
 - Preprocessing (update morphological filters)
 - Multiple ellipse fitting.
- Primitives
 - New/updated features to represent primitives.
 - Define primitives on real or synthetic data.
 - Manuel vs. automatic primitive selection.
- Eventually optimize the current probabilistic edit distance.

References

- [1] P. Fihl, M.B. Holte, T.B. Moeslund, and L. Reng. Action Recognition using Motion Primitives and Probabilistic Edit Distance. In *AMDO*, July 2006.
- [2] 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.
- [3] V.I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965.
- [4] 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.
- [5] <http://polhemus.com/>, January 2006.
- [6] <http://www.poserworld.com/>, January 2006.
- [7] 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.