Human Action Recognition in Videos via Principal Component Analysis of Motion Curves

Daniel S. Chivers and A. Ardeshir Goshtasby
CSE Dept., Wright State University, Dayton, OH, USA

Abstract—A new approach to human action recognition in videos is presented and evaluated using the Weizmann action dataset. In this approach, motion trajectories are formed by tracking one or more key points on the human body. In particular, points on the hands and feet are tracked. A curve is fitted to each motion trajectory to smooth noise and to form a continuous and differentiable curve. A motion curve is then segmented at peak curvature points, representing each segment by a “basic motion.” To recognize an observed basic motion, a vector of curve features describing the motion is created, the vector is projected to the eigenspace created during PCA training, and the action most similar to a learned action is identified using the $k$-nearest neighbor decision rule.

The proposed approach simplifies action recognition by requiring that only a small number of points on a subject’s body be tracked.

Keywords: Human Action Recognition, PCA, Motion Analysis

1. Introduction

The goal of a human action recognition system is to automatically detect and recognize actions of humans in a video. An action recognition system has to analyze a massive amount of data in a video to detect and recognize actions of interest for use in applications such as visual surveillance or human-computer interaction.

Approaches to human action recognition can be grouped into tracking and non-tracking approaches. Non-tracking approaches do not track parts of the human body when recognizing an action. Instead, they detect feature points in each video frame and from the pattern of the points recognize the action. Alternatively, they use silhouettes of a human in different frames to build a structure in space and time and then by analyzing the structure, recognize the action.

Approaches that do not use tracking can recognize simple actions quickly, but they have difficulty recognizing complex actions that involve motions of different parts of the body. Tracking-based approaches that use information about the motion of different parts of the body can recognize complex actions. Tracking-based approaches are, however, computationally more demanding than non-tracking based approaches.

In the following, a human action recognition system is described that efficiently learns various actions and recognizes them when observed later. This action recognition system tracks key points on a subject’s body. The trajectories obtained by this tracking are then used to learn or to recognize an action.

In this work, by tracking feet and hands in a video, a video is reduced to a small number of trajectories. A curve is fitted to each trajectory to smooth noise and produce a continuous and smooth motion curve. A motion curve is segmented into basic motions at locally peak curvature points. An example of a basic motion is a step, which appears repeatedly in a walk action. A feature vector is generated for each basic motion. The feature vector created from a basic motion is then projected to an eigenspace formed by the principal component analysis (PCA) of a training data set. A basic motion is recognized by measuring the distances between its feature-vector projection to the eigenspace and the feature-vector projections of training data to the eigenspace, and taking the action most frequently encountered within the $k$th nearest neighbor of the observed motion as the observed motion.

The following new conclusions were reached in this study:

1) The motion curves obtained by tracking a few key points on a subject’s body are sufficient to recognize various actions.
2) A basic motion obtained by segmenting a motion curve and its repetition in a motion curve are sufficient to recognize an action.
3) The tracking of feet is sufficient to recognize various actions involving the motion of feet. Likewise, tracking only the hands is sufficient to recognize various actions involving the hands. These conclusions are reached and verified using the Weizmann action data set.

In the following sections, first, past work in action recognition is reviewed. Then, the details of the proposed action recognition system are provided. Next, experimental results are reported and analyzed, and finally, concluding remarks are made.
2. Related Work

2.1 Multi-level Approaches

To understand human actions in a video, a number of methods rely on motion tracking, pose estimation, and gesture recognition. A number of action recognizers use low-level processes to generate the information needed by a recognizer. Such methods have been described by Ryoo and Aggarwal [1][2] and Ivanov and Bobick [3].

In our approach we recognize actions by recognizing basic motions. A more elaborate method may use information from our method to recognize composite actions or activities in a video.

2.2 Tracking-based Approaches

Because a motion trajectory contains information about the nature of an action, motion trajectories have become increasingly important during recent years in action recognition. Similarly, PCA has become a popular tool in dimensionality reduction of high-dimensional trajectory data for various applications.

Bashir et al. [4] segmented trajectories into subtrajectories at points of high curvature. The subtrajectories were then represented by their PCA coefficients via a Gaussian mixture training model. Finally, recognition was achieved using a hidden Markov model (HMM). One HMM per class was considered. Bashir et al. demonstrated their approach on an American sign language data set. Our approach differs from that of Bashir et al. in that we use clustering in an eigenspace to classify basic motions obtained through a training process. This makes our approach computationally efficient when compared to HMM classifiers. We also use curve fitting to segment a motion curve, fill gaps in data, and generate feature vectors. Curve fitting makes it possible to track imperfections in a video caused by noise and occlusion.

Gritai et al. [5] developed a system that uses trajectories of different landmarks on a subject’s body. Unlike our approach, a dissimilarity measure and a dynamic time warping method were used to match 3-D trajectories for many points on the body directly to exemplar actions. Motion segmentation was not considered, but they normalized the trajectories for body size. We normalize trajectory data with respect to the distance between the starting and ending points of the basic motion. This is discussed in Section 3.3.

Wu and Li [6] created a motion signature from a trajectory. They explored the dimensionality reduction of the motion signatures by PCA. Recognition was achieved using various methods including dynamic time warping, Mahalanobis distance between optimized signatures, and Bayesian classification using Gaussian mixture models. Trajectory smoothing was performed but curve fitting and curve segmentation were not explored.

Han et al. [7] used a Gaussian process latent variable model to learn various actions in a hierarchical manifold space. They used PCA and k-means clustering to group motions in manifold subspaces. The motions were recognized using a tree-based cascade conditional random field model. No motion segmentation was used and the system was evaluated using motion capture files from the CMU motion capture database.

2.3 Other Approaches

Various other approaches to action recognition in videos have appeared in the literature. An excellent review of action recognizers is provided by Poppe [8]. Poppe offers insights into the challenges in human action recognition and proposes methods to make the recognition process invariant to the speed of an action.

Turaga et al. [9] offer a broad overview of action recognizers. They identify many common action and activity recognition methods, ranging from simple template matching based recognizers to advanced hidden Markov model based recognizers.

To achieve a high computational efficiency while maintaining a high recognition rate, we identify and track a number of key points on a subject’s body and use the obtained trajectories to recognize actions. Feature vectors that can distinguish various actions from each other are created from the trajectories and PCA is employed to reduce dimensionality of data and reach a fast recognition rate. We demonstrate a feet and hand tracking system on the Weizmann data set and show that trajectories produced through tracking key points on a subject’s body can be used to first learn and then recognize various actions.

3. Proposed Approach

The flow diagram of the proposed action recognition system is depicted in Figure 1. First, trajectory data are produced by tracking key points on a subject’s body. The location of a key point on a subject’s body, which is determined automatically, depends on the action to be recognized.

Once a set of key points are selected and the associating trajectories are obtained, a curve is fitted to each trajectory to represent the trajectory by a smooth motion curve. A motion curve reduces the effect of noise in a trajectory and allows the computation of curvature and other features of the motion curve. We use the curvature feature to partition a motion curve into segments called basic motion. Details of the segmentation step are provided in Section 3.2.

Once a motion curve is segmented into basic motions, a feature vector is generated for each basic motion using the obtained motion curve. The curve model we use allows for the representation of any basic motion with a feature vector of a fixed size. Details of this step are provided in Section 3.3. The feature vectors are then used to either build a training dataset or recognize an unknown motion after the system is trained.
In Section 3.4, steps to create an eigenspace from the feature vectors of a training data set are given, and in Section 3.5 recognition of an observed motion using the feature vectors of its motion is detailed.

3.1 Tracking Key Points

To evaluate the proposed action recognition system, key points on the feet and the hands of a subject are tracked using the Weizmann dataset [10]. First, background subtraction is performed to isolate and delineate the subject in each frame. Next, the delineated boundary is used to find the location of each foot. Details of this step are given in Section 3.1.2. Dense optical flow is used to determine the motion of the hands in a frame. Details of this step are provided in Section 3.1.3.

3.1.1 Background Subtraction

Background subtraction is widely used to separate a moving object from a stationary background. Background subtraction is used to extract the silhouette of the person in motion in each frame of a video. Figure 2 illustrates an example. We use the mixture of Guassian (MoG) background subtraction technique [11] with our own enhancements to delineate a clean silhouette of a moving person in each frame of a video.

While experimenting with various videos we have found that shadows often present a problem in extracting clean silhouettes in videos. Even though shadows in the Weizmann dataset are subtle, we have found that by adding a shadow removal step to the process, cleaner silhouettes are extracted from the videos. The background subtraction results obtained without and with the shadow removal step are illustrated in Figures 2(b) and 2(c), respectively.

We accomplish shadow removal by performing the background subtraction in the Lab color space. When a pixel is classified as foreground or background, the lightness component, L, of the pixel’s color is allowed a wider range of deviation from the background model’s mean while the a and b components of the color are fit tightly to the mean of the model. We found that Lab is a better choice than HSL or HSV because the a and b components remain stable under variation of lightness, while H and S components in HSL and HSV models change with variation in lightness.

After training the mixture models for 60 frames by looping through the background training videos in the Weizmann dataset, we observed that a few of the background training data did not perfectly match the background in respective action videos. In a few cases, the background training video was darker than the corresponding action video. For action videos where background subtraction could be improved with a better background training data, we created new background training data ourselves. Because there are no frames in an action video that contain only the background, we used portions of the frames to create a new frame that contains only the background. For example, if the person was on the right side of a frame, we replaced the right side of that frame with that of another frame from the same video where the person was on the left side of the frame. This removed the person from the background, creating a background training data more closely matching the corresponding action video.

3.1.2 Tracking Feet

The location of each foot is found using a method similar to that described by Jean et al. [12]. To locate the feet, Jean et al. first fit a bounding box around the foreground pixels for the lower 33% of the extracted silhouette. Next, they separated the legs by scanning horizontally in the bounding box for a foreground-background-foreground pattern. Overlapping areas of background pixels are used to find a path...
between the legs. The tallest path defines the vertical line position. The location of each foot is found by calculating the mass center of foreground pixels for each leg region until 25% of the foreground pixels in the leg region is reached.

Our method differs from that of Jean et al. in the way we find the separation of the legs and that we estimate the size of the feet to find their position in a greater variety of poses. Figure 3 illustrates feet tracking by our method for various actions in the Weizmann dataset.

Like Jean et al., we fit a bounding box around the lower 33% of the silhouette. We found that this captures the area where the feet are located in the most common poses. Next, we find a vertical line of separation for the legs by scanning the bottom of the foot bounding box from left to right and counting the number of background pixels in the vertical direction before reaching a foreground pixel. A line of separation is considered if it is at least 30% of the feet box in height and it separates a certain number of foreground pixels on each side. This foreground threshold is necessary so that a line is not chosen to the left or right of the foot in the case of a bent-leg pose.

The tallest line segment that satisfies the minimum height and minimum number of foreground pixels on both sides is chosen as the line of separation. If no line segment satisfies the two conditions, the legs are considered to be together and one foot is considered to be occluded by the other. This method of vertical line separation was found to be more reliable in the presence of noise than scanning horizontally for a foreground-background-foreground pattern as done by Jean et al. [12]. In Figure 3 the line of separation is shown in red.

Once the line of separation is found, the location of the foot to the left and right of the separation is found by fitting a bounding box to the foreground pixels that are to the left and right of the separation line, respectively. The bounding box height is limited to 15% of the height of the silhouette, which is the approximate height of a foot. The position of each foot is calculated using the average position of foreground pixels in a foot’s bounding box.

While performing some actions, such as jumping, a subject’s feet remain together or in close proximity to each other, as are the legs. See Figure 3(b) for an example. We make a simple provision for this case to approximate the location of each foot. If the legs cannot be separated for more than 6 frames, the feet are assumed to be together. The location of each foot is then found by first approximating the size of each foot using the height of the silhouette. The approximate size is used to capture each foot by placing each foot at the bottom of the bounding box containing both feet. One bounding box is placed so that the left side of the box aligns with the leftmost foreground pixel in the foot area. The other bounding box is placed in such a way that the right side aligns with the rightmost foreground pixel. The two boxes will likely overlap, but the degree of overlap will depend on the degree of overlap of the feet. The center of mass calculation for each foot is used to find an approximate position. This method is sufficient to capture the overall motion of the feet for jumping motions and when the person is not in motion.

The final capability needed in feet tracking is to determine which foot is on which side of the silhouette. We accomplish this by examining the velocity of both feet over time. If one foot is stationary while the other is moving, we assume the feet have swapped after observing an occlusion.

The advantages of this feet tracking approach are simplicity of implementation and high accuracy when the legs are separated. A disadvantage of the approach is the absence of points when the feet are occluded. We compensate for this deficiency by using curve fitting the available data to fill in the missing data. Figure 6 shows such an example. We could not find enough points to reliably track the skipping action in the Weizmann dataset. One foot is occluded by the opposite leg, making it impossible to track the foot.

We had some success in tracking the foot under occlusion by using rotation invariant template matching. The template is built from the last frame where the foot was located and includes background-foreground classification information so that only the foot is matched and the background information is excluded. We found that the Weizmann videos have resolution too low for this type of template-matching process to be reliable.

3.1.3 Tracking Hands

We have experimentally determined that information about the motion of feet alone is sufficient to classify actions where feet have a primary role. If feet are stationary, we
switch to using the hands if hand motion is present. This means we only need to track the hands when the rest of the body is stationary.

We accomplish tracking of hands using dense optical flow Farnebäck [13]. This method of optical flow estimation provides an accurate dense flow field, enabling isolation of the hands from the body and the background by looking for peaks in the flow magnitude.

Dense optical flow provides a vector representing the optical flow for every pixel. We first calculate the magnitude of each flow vector. If it is below an experimentally determined threshold value, we set the pixel’s flow magnitude to 0. This removes noise and flow created by small movements. We then find connected regions of non-zero flow magnitude using a connected components algorithm. Next, we examine the two largest connected regions. If both regions are larger in size than a size threshold, we consider them to be two arms in motion. If only one region meets the size criterion, only one arm is considered to be moving. Finally, the locations of the hands are determined by finding the location where average flow magnitude in a $5 \times 5$ window is the greatest for each connected flow region. This identifies each hand when the arms are waving because the hands are the fastest part of the motion and have the largest flow magnitude.

3.1.4 Detecting Bend Over

Certain actions, such as bending over to pick up something, bring the hands in proximity of the feet. This results in feet tracking errors. We alleviate this problem by detecting a bend over action using height of the silhouette. If the silhouette’s height is observed to shrink by a sufficient amount and the feet are stationary, then a bend over action is detected. We reliably detected the bend over action in all videos in the Weizmann dataset.

3.2 Curve Fitting and Motion Segmentation

Once a trajectory is created for a track point, we use curve fitting to smooth the data and produce a parameterized representation for the trajectory. In our implementation we use cubic spline, but other parametric curves may be used as well. A parametric curve is normally defined with parameter $u$ varying from 0 to 1.

In order to include time information in a curve, we consider a 3-dimensional curve, 2 trajectory coordinates and 1 time coordinate. This allows us to compute velocity at any point on a curve and allows the curve model to maintain timing information, which is essential when generating feature vectors for multiple tracks. This is discussed in more detail in Section 3.3.

A motion curve is partitioned into basic motion segments that can be used in recognition. We require that a basic motion segment represent a periodic segment that is repeated over a trajectory. For example, a walking motion curve is segmented in such a way that each foot step becomes a basic motion. We have found that peak curvatures of a motion curve provide a reliable means to segment a motion curve.

Figure 6 shows the result of curve fitting and curve segmentation using a running motion curve obtained by tracking a foot.

3.3 Generating Feature Vectors

The goal of feature vector generation is to produce a feature vector of a fixed size for each basic motion so it can be used to recognize the basic motion. The feature vectors must have the property that similar motions produce similar feature vectors and different motions produce dissimilar feature vectors, allowing discrimination of various action classes from each other by PCA. The size of a feature vector depends on the number of samples taken of a basic motion and the number of values per a sample.

A feature vector is computed by uniformly sampling a chosen property of a motion curve from start to end. The feature vector used to obtain the recognition results in Section 4 consists of the position and velocity information as follows:
process, each

Since time information is incorporated into the curve fitting

by the start and end

accomplished by evaluating each curve in the range given

the motion from the other foot in the feature vector. This is

vectors for basic motions from each foot, while including

vector. For actions involving the feet, we generate feature

information for more than one track point in each feature

Finally, motion is normalized with respect to scale:

left foot basic motion, the feature vector has the following

for all motion curves. In the case of the feature vector for the

basic motion is from right to left, the trajectory is reflected

motion moves to the origin. Next, if the trajectory of the

are translated such that the first coordinate of the basic

motion curve. (b) The curvature plot of the motion curve

Fig. 6: (a) Curve fitting and segmentation of a running

motion curve. (b) The curvature plot of the motion curve

to a foot.

\[ V = [P_1, V_1, ..., P_n, V_n], \]  

where \( P_i = (x_i, y_i) \) is the position and \( V_i = (x'_i, y'_i) \) is the

velocity at \( P_i \).

We found that recognition is improved by including

information for more than one track point in each feature

vector. For actions involving the feet, we generate feature

vectors for basic motions from each foot, while including

the motion from the other foot in the feature vector. This is

accomplished by evaluating each curve in the range given

by the start and end \( u \)-values of the basic motion curve.

Since time information is incorporated into the curve fitting

process, each \( u \)-value corresponds to the same time instant

for all motion curves. In the case of the feature vector for the

left foot basic motion, the feature vector has the following

form:

\[ V = [a_1, ..., a_n, b_1, ..., b_n], \]  

where \( a_i \) is a sample (containing both position and velocity)

from the left foot over the basic motion range and \( b_j \) is

information from the right foot over the same \( u \)-value range.

This method effectively captures the information for both

feet and remains repeatable since the information is taken

over the \( u \) range of the basic motion. The same concept is

applied to hands for actions where no motions of the feet

are detected.

In order to align all basic motions for comparison when

using positional values in the feature vectors, the coordinates

are translated such that the first coordinate of the basic

motion moves to the origin. Next, if the trajectory of the

basic motion is from right to left, the trajectory is reflected

across the \( y \)-axis so that the motion will be from left to right.

Finally, motion is normalized with respect to scale:

\[ P'_i = \frac{P_i}{|P_n - P_1|} \]  

where \( P_1 \) and \( P_n \) are the first and last positions of the basic

motion, respectively. This removes the effect of zoom and

also ensures that motions aren’t classified solely based on the

motion along a dominant direction. For example, the motion

of the foot for walking and running tends to be much larger

in the \( x \)-direction than in the \( y \)-direction. If a long walking

step is taken, the motion may be classified as a running step

if it is nearest in length to other running motions. By scaling

the motion by its overall length, other features such as the

overall shape of the motion become more important.

3.4 Preparing the Eigenspace

A training data set is created using a set of feature vectors

of known actions. For each action to be recognized, several

examples of the action are used so that a cluster can be

formed for the action in the eigenspace. Each feature vector

in the training data set is labeled with an action or class

name that identifies the action producing the feature vector.

Two feature vectors from the same action category are given

the same action name.

PCA is used to create the eigenspace from the training

data set. First, the training data set is used to create a matrix

\( M \) where each column in the matrix represents a different

feature vector in the training data set. Next, each row of \( M \)

is mean-centered by calculating the mean across each row

and subtracting the mean from each item in the row. This is

a requirement of PCA [14].

We compute the covariance matrix \( C \) from \( M \):

\[ C = \frac{1}{N-1}(M \times M'), \]  

where \( N \) is the number of elements in each feature vector,

and \( M' \) is the transpose of \( M \).

Next, the eigenvectors and eigenvalues of the covariance

matrix are computed. The eigenspace is formed by placing

the eigenvectors in columns of a matrix and sorting them

from largest to smallest according to the corresponding

eigenvalues. \( l \leq N \) columns may be chosen from the set

eigenvectors to form the final eigenspace where \( l \) is the

desired number dimensions. \( l \) may be chosen in such a

way that the sum of squared errors caused by discarding

eigenvectors corresponding to \( N-l \) smallest eigenvalues is

smaller than a required error tolerance [14].

3.5 Feature Vector Classification

Now that the eigenspace has been created using a training

data set, we project each training feature vector to the

eigenspace. Each projection is labeled by remembering the

class that each training vector belongs to. Each action class

will form a cluster in the eigenspace as similar feature

vectors project to nearby locations in the eigenspace. This

enables classifying an unknown basic motion by projecting

its feature vector to the eigenspace and finding the Euclidean

distance between that projection and the training feature
Table 1: Confusion matrix showing the recognition rate of basic motions for each action. Even though recognition for a basic motion is not 100% for all actions, 100% accuracy is achieved for video recognition by classifying each video using the most commonly observed basic motion. For example, a video of a walking person is classified as a video containing the walk action because the walk basic motion was the most commonly observed basic motion.

<table>
<thead>
<tr>
<th></th>
<th>jack</th>
<th>jump</th>
<th>pump</th>
<th>run</th>
<th>side</th>
<th>walk</th>
<th>wave1</th>
<th>wave2</th>
</tr>
</thead>
<tbody>
<tr>
<td>jack</td>
<td>.92</td>
<td>.07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.01</td>
<td></td>
</tr>
<tr>
<td>jump</td>
<td></td>
<td>.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pump</td>
<td></td>
<td></td>
<td>.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>run</td>
<td></td>
<td></td>
<td></td>
<td>.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>side</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>walk</td>
<td>.06</td>
<td>.03</td>
<td>.06</td>
<td>.86</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>wave1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.10</td>
<td></td>
</tr>
<tr>
<td>wave2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.02</td>
<td>.98</td>
</tr>
</tbody>
</table>

vector projections. We use the $k$-nearest neighbor classifier to recognize various basic motions and their associating actions. Videos containing a single action are classified using the most commonly observed basic motion classification.

### 4. Results

To evaluate our action recognition system, we use videos from the Weizmann Human Action dataset [10]. The Weizmann dataset is a commonly used dataset to evaluate the performance of action recognition methods. It contains low resolution videos of 10 different individuals, each performing the same 9 actions. Even though the Weizmann dataset is low resolution, we achieve good tracking results and recognition accuracy for the basic motions. As mentioned in Section 3.1.2, our tracking system was unable to track the feet for the skip action so results for the skip video are not included.

We used two separate classifiers, one for the feet and one for the hands. When basic motions are produced for the feet, the system classifies these motions using the Eigenspace for the feet. Actions involving the feet include jack, jump, pjump, run, side, and walk. Actions involving the hands, such as wave1 and wave2, are classified using the Eigenspace trained for the hands. The bendover action is classified by detecting bend over events as described in Section 3.1.4.

The training dataset used in our experiment consisted of basic motions from the videos of one of the human subjects with the exception of run action. We used two videos from the set of run videos to train the system because there is a small number of basic motions included in each run video.

Table 1 is a confusion matrix showing the accuracy of our system in recognizing basic motion types. Using the most commonly observed basic motion type to classify each video, we obtain 100% recognition accuracy for the videos in the data set.

### 5. Conclusions

Human action recognition is an important research area due to its various applications. Progress in this area will help surveillance, human-machine interaction, and robot vision, to name a few.

We introduced a new action recognition approach that can learn various human actions and recognize them in an efficient manner. We make a number of contributions, including recognizing actions using a small number of tracked points on the human body and providing an efficient recognition system using curve fitting, curve segmentation, feature generation, and PCA classification. Finally, we demonstrated how our system works by tracking the feet and hands of a subject in a video using the Weizmann data set.

### References


