Human Action Recognition by Principal Component Analysis of Motion Curves

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

by

Daniel S. Chivers
M.S., Wright State University, 2007

2012
Wright State University
I HEREBY RECOMMEND THAT THIS DISSERTATION PREPARED UNDER MY SUPERVISION BY Daniel S. Chivers ENTITLED Human Action Recognition by Principal Component Analysis of Motion Curves BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy.

Arthur A. Goshtasby, Ph.D.
Dissertation Director

Arthur A. Goshtasby, Ph.D.
Director, Computer Science and Engineering Ph.D. Program

Andrew Hsu, Ph.D.
Dean, School of Graduate Studies

Committee on Final Examination

Arthur A. Goshtasby, Ph.D.

Mateen M. Rizki, Ph.D.

Thomas Wischgoll, Ph.D.

David F. Miller, Ph.D.
Human action recognition is used to automatically detect and recognize actions performed by humans in a video. Applications include visual surveillance, human-computer interaction, and robot intelligence, to name a few. An example of a surveillance application is a system that monitors a large public area, such as an airport, for suspicious activity. In human-machine interaction, computers may be controlled by simple human actions. For example, the motion of an arm may instruct the computer to rotate a 3-D model that is being displayed. Human action recognition is also an important capability of intelligent robots that interact with humans.

General approaches to human action recognition fall under two categories: those that are based on tracking and those that do not use tracking. Approaches that do not use tracking often cannot recognize complex motions where movement of different parts of the body is important. Tracking-based approaches that use motion of different parts of the body are generally more powerful but are computationally more expensive, making them inappropriate for applications that require real-time responses.

We propose a new approach to human action recognition that is able to learn various human actions and later recognize them in an efficient manner. 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 into “basic motion” segments by detecting peak curvature points. 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. It is shown that the motion curves obtained by tracking a small number of points are sufficient to recognize various human actions with a high degree of accuracy.

Furthermore, the proposed approach can improve the recognition power of other approaches by recognizing detailed basic motions, such as foot steps, while introducing efficient tracking and recognition compared to previous approaches. Recognition of basic motions allows a high-level recognizer to recognize more complex or composite actions by using the proposed system as a low-level recognizer.

Contributions of this work include reducing each video frame to a few key points on the subject’s body, using curve fitting to smooth trajectory data and provide reliable segmentation of the motion, and efficient recognition of basic motions using PCA.
# List of Symbols

## Chapter 4

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\eta$</td>
<td>Gaussian probability density function</td>
</tr>
<tr>
<td>$w$</td>
<td>Mixture model weight</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>Gaussian distribution covariance matrix</td>
</tr>
<tr>
<td>$X$</td>
<td>Pixel color</td>
</tr>
<tr>
<td>$\mu$</td>
<td>Mean</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Variance</td>
</tr>
<tr>
<td>$k$</td>
<td>Distributions per pixel</td>
</tr>
<tr>
<td>$R$</td>
<td>Lightness threshold</td>
</tr>
<tr>
<td>$I_n$</td>
<td>Pixel color at specified location in image</td>
</tr>
</tbody>
</table>

## Chapter 5

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rho_i$</td>
<td>Radius of curvature</td>
</tr>
<tr>
<td>$p_i^u$</td>
<td>First derivative of curve at point $p_i$</td>
</tr>
<tr>
<td>$p_i^{2u}$</td>
<td>Second derivative of curve at point $p_i$</td>
</tr>
<tr>
<td>$V$</td>
<td>Feature Vector</td>
</tr>
<tr>
<td>$P_i$</td>
<td>Position sample from parametric curve</td>
</tr>
<tr>
<td>$V_i$</td>
<td>Velocity sample from parametric curve</td>
</tr>
</tbody>
</table>

## Chapter 6

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>Training matrix</td>
</tr>
<tr>
<td>$C$</td>
<td>Covariance matrix</td>
</tr>
<tr>
<td>$N$</td>
<td>Feature vector length</td>
</tr>
</tbody>
</table>
## Contents

1 Introduction 1

2 Related Work 5
   2.1 Surveys on Action Recognition 5
   2.2 Multi-level Approaches 9
   2.3 Commonly Used Tools 10
   2.4 Action Recognition Without Tracking 11
   2.5 Action Recognition Using Tracking 14

3 Proposed Approach 19
   3.1 Overview of Proposed Approach 19
   3.2 Strengths of Proposed Approach 22

4 Data Collection 24
   4.1 Motion Capture Data 24
   4.2 Video Data 25
   4.3 Motion Data Generation from Videos 26
      4.3.1 Background Subtraction 26
      4.3.2 Tracking of Feet 33
      4.3.3 Tracking of Hands 36
      4.3.4 Detecting the Bend-Over Action 37

5 Motion Analysis 39
   5.1 Curve Fitting and Motion Segmentation 39
   5.2 Feature Vector Generation 41

6 Training and Recognition Using PCA 45
   6.1 Preparing the Eigenspace 45
   6.2 Feature Vector Classification 46

7 Results 49
   7.1 Results from Motion Capture 50
      7.1.1 Action Recognition Using 3-D Data 68
List of Figures

3.1 Flow diagram of the proposed action recognition system. 1) In the data collection phase, trajectories are produced from video or motion capture data. 2) Curves are fitted to the trajectories in the motion analysis phase. Next, the curves are segmented at peaks in curvature forming basic motion curves. Feature vectors are then created for each basic motion curve using properties of the curve such as position, velocity, or curvature over time. 3) In the training phase, PCA is used to calculate an eigenspace matrix from a set of training feature vectors. 4) In the recognition phase, a feature vector created from an unknown basic motion is classified by first projecting it onto the eigenspace. Next, the Euclidean distance between the basic motion projection and each training data feature vector projection is used to classify the basic motion by using the k-nearest neighbor classifier.

4.1 Visualization of motion trajectories representing the motion of the left hand and the left foot of a runner in a video.

4.2 (a) A walking person. (b) Extracted foreground without shadow removal. Background pixels are shown in white. (c) Extracted foreground with shadow removal. Background pixels are shown in white.

4.3 The RGB color components of a single pixel are plotted for each sample frame. This pixel becomes part of a shadow from sample 310 to 350. As expected, the $R$, $G$, and $B$ values decrease when the pixel is in the shadow area indicating that the color is darker than the non-shadow areas.

4.4 The HSV color components of a single pixel are plotted for each sample frame. This pixel becomes part of a shadow from sample 310 to 350. Notice that the $V$ value decreases in the shadow area, as expected, but the $H$ value increases unexpectedly. This indicates that $HSV$ does not isolate the shadow in the $V$ component.

4.5 The $Lab$ color components of a single pixel are plotted for each sample frame. This pixel becomes part of a shadow from sample 310 to 350. Notice that the value of the $L$ component decreases in the shadow while the $a$ and $b$ components remain stable. This makes $Lab$ a good choice for shadow removal because the shadow is isolated in the $L$ component.
4.6 Tracking the feet in various actions. (a) Walking, (b) jumping, (c) running, and (d) side stepping. ................................. 35

4.7 (a), (b) Hand tracking in a 2-handed wave action. (c) Hand tracking in a right-handed wave action. (d) Hand and feet tracking in jumping-jack action. .................................................. 37

4.8 The bend-over action brings the hands in proximity to the feet, causing errors in feet tracking. We remove these errors by detecting when the person is bending over using changes in height of the silhouette. ....................... 38

5.1 (a) Visualization of a running person. The curve shown represents the motion of the left foot. (b) Curve fitting and segmentation of a foot motion curve during running. (c) The curvature plot of the motion curve. A large spike in curvature is observed each time the foot is placed on the ground. The curve is segmented at the points labelled “seg point.” There are two segmentation points for each curvature spike corresponding to where the motion of the foot starts and stops. ................................. 43

5.2 (a) Feature vector containing the y-position of both feet for the “run” action. Samples 0-99 are the y-values for the foot producing the basic motion. Samples 100-200 are the y-values for the other foot. (b) Feature vector containing the y-position of both feet for the “forward jump” action. Samples 0-99 are the y-values for the foot producing the basic motion. Samples 100-200 are the y-values for the other foot. ................................. 44

6.1 Projection of training feature vectors onto the 1st and 2nd eigenvectors created using the Weizmann video dataset. The unknown input feature vector corresponds to a “run” basic motion and the nearest neighbors are “run” training data projections. Notice that two clusters have formed for the “pjump” (vertical jump) action. This is because each vertical jump is segmented into two basic motions: one for the upward motion and one for the downward motion. ................................................. 47

6.2 Projection of training feature vectors onto the 1st and 3rd eigenvectors created using the Weizmann video dataset. The unknown input feature vector corresponds to a “run” basic motion and the nearest neighbors are “run” training data projections. ................................................. 48

7.1 “Forward jump,” “jump,” and “walk up the stairs” actions. The curves in each action show the trajectories of the right hand and the right foot. ....... 50

7.2 The “march” action. The curves show the trajectories of the right hand and the right foot. .................................................. 51

7.3 The “walk” action. The curves show the trajectories of the right hand and the right foot. .................................................. 51

7.4 The “run” action. The curves show the trajectories of the right hand and the right foot. .................................................. 52
7.5  Curve fitting and segmentation for the “walk” action. Two basic motions are identified using peak curvature. Notice that multiple curvature peaks occur when the foot is nearly stationary. This is due to small inaccuracies in tracking. The speed of the motion is used to locate where the foot starts and stops moving so that the motion curve is accurately segmented into foot steps. See Figure 7.6 for a plot showing the speed and velocity of this motion. The motion is segmented at the points labeled “seg point.”

7.6  Speed and velocity of the “walk” action. The speed of the motion is used to find the segmentation points after identifying peaks in curvature. The curve labeled “speed” is the speed calculated from the trajectory data. The speed calculated from the fitted curve is labelled “fit speed.” Notice that the “fit speed” curve is a smooth representation of the speed calculated from the trajectory data. This shows that our fitted curve accurately models the motion over time. The velocity of the motion calculated from the trajectory data is labelled “dx,” “dy,” and “dz” for the x, y, and z velocity components, respectively. The curves labelled “x vel,” “y vel,” and “z vel” are the velocity curves calculated using the fitted curve.

7.7  Positional data for the first “walk” basic motion from Figure 7.5. This data is used to create a feature vector for training or recognition.

7.8  Positional data for the second “walk” basic motion from Figure 7.5. Notice that this data appears to be very similar to the first basic motion data shown in Figure 7.7. This is promising for PCA recognition.

7.9  Curve fitting and segmentation for the “run” action. One complete basic motion is identified by the curve segmentation process. The partially complete basic motion is discarded.

7.10 Speed and velocity of the “run” action. One complete running step can be identified using speed in this example. Notice that the foot is not stationary for very long compared to the “walk” action. Like in the walk example, the fitted curve provides a smooth approximation of the speed calculated using the trajectory data.

7.11 Positional data for the “run” basic motion from Figure 7.9. The foot is lifted high in the first half of the running foot step and is put back on the ground at a slightly slower rate near the end of the motion.

7.12 Curve fitting and segmentation for the “jump” action. Since the jump action is a vertical direction, high curvature is found at the start, apex of the jump, and end of the jump. This causes the jump to be segmented into an upwards basic motion and a downwards basic motion.

7.13 Speed and velocity of the “jump” action. The dip in speed in the middle of the motion (around sample 220) is where the apex of the jump occurs. Notice that using speed alone to segment this motion may not work reliably because the observed speed is not zero at the apex. This is because the sample rate is not high enough to observe where the jump is perfectly at the apex. We easily detect the transition using curvature as shown in Figure 7.12.

7.14 Positional data for the “jump” upwards basic motion from Figure 7.12.

7.15 Positional data for the “jump” downwards basic motion from Figure 7.12.
7.16 Curve fitting and segmentation for the “forward jump” action. The segmentation process correctly identifies four distinct forward jump basic motions in this example. ........................................... 65

7.17 Speed and velocity of the “forward jump” action. Four distinct forward jump motions can be identified using the speed curve. ........................................... 66

7.18 Positional data for one of the “forward jump” basic motions from Figure 7.16. ........................................... 67

7.19 Curve fitting and segmentation for the “walking up stairs” action. Unlike the previous examples, this action was performed along the negative \( z \)-axis (from right to left in the plot). It is rotated along the positive \( z \)-axis before creating the feature vector positional data shown in Figure 7.21. The segmentation process correctly identifies two distinct “walking up stairs” basic motions in this example. ........................................... 69

7.20 Speed and velocity of the “walk up stairs” action. Two foot steps can be observed using the speed curve. The fitted curve provides a smooth representation of the speed and velocity curves (shown in red) in this example. ........................................... 70

7.21 Positional data for one of the “walk up stairs” basic motions from Figure 7.19. The basic motion is rotated so that the horizontal motion is along the positive \( z \)-axis. ........................................... 71

7.22 Curve fitting and segmentation for the “marching” action. The segmentation process correctly identifies two complete basic motions in this example. This example was performed in the direction of the negative \( z \)-axis. ........................................... 72

7.23 Speed and velocity of the “marching” action. Sharp changes in speed are observed in the speed calculated from the trajectory data. This may indicate that the motion capture system was not as accurate when capturing this motion compared to the other actions. ........................................... 73

7.24 Positional data for one of the “march” basic motions from Figure 7.22. A distinct shape is formed by the motion made by the foot. First it is lifted high in the air, then it is kept in the air as the body leans forward, and finally it is placed back on the ground. ........................................... 74

7.25 The result of the camera simulation on a “walk” basic motion at view directions -60, -40, -20, 0, 20, 40, and 60 degrees. Each feature vector is mean centered in the \( y \)-direction as a part of the feature vector generation. 78

7.26 Tracking for the “jack” action. ........................................... 81

7.27 Tracking for the “jump” action. ........................................... 81

7.28 Tracking for the “skip” action. The skip action is not included in our final results because each foot is overlapping with the other foot or the leg throughout much of the video. This illustrates a weakness in our tracking system. ........................................... 82

7.29 Tracking for the “run” action. ........................................... 82

7.30 Tracking the hands and feet for the “wave1” action. ........................................... 83

7.31 Tracking the hands and feet for the “wave2” action. ........................................... 83

7.32 Tracking for the “bend” action. Notice that the feet tracking algorithm detects the hand as a foot once it enters the foot region. This action is detected as “bend” by the bend over detection algorithm, so the results from the feet tracking are ignored by the recognition system. ........................................... 84
7.33 Tracking for the “walk” action. ........................................ 85
7.34 Tracking for the “side” action. ........................................ 85
7.35 Tracking for the “pjump” (vertical jumping) action. The feet may occlude each other in some frames, but the tracking system reacquires the location of each foot by either finding separation of the legs in a later frame or by assuming the feet are together. ........................................ 86
7.36 Curve fitting and segmentation for the “run” action from the Weizmann video dataset. One complete running step is observed. The fitted curve provides a continuous smooth representation of the trajectory. It also approximates the missing data where the tracker could not locate the feet because they were overlapping. ........................................ 88
7.37 Speed and velocity of the “run” action from the Weizmann video dataset. The speed calculated from the fitted curve, labelled “fit speed,” is much smoother than the speed calculated from the trajectory data. This is because the trajectory is sparse and noisy. ........................................ 89
7.38 Positional data for one of the “run” basic motions. ...................... 90
7.39 Curve fitting and segmentation for the “walk” action from the Weizmann video dataset. Again, the fitted curve is a smooth curve that approximates the motion made by the foot. Two complete foot steps are detected by the segmentation process. ........................................ 91
7.40 Speed and velocity of the “walk” action from the Weizmann video dataset. The speed of the motion is used to find the segmentation points after peaks in curvature are identified. ........................................ 92
7.41 Positional data for one of the “walk” basic motions. Because the videos are low resolution, detail in the motion is lost compared to the motion capture example in Figure 7.7. However, the overall motion of the foot is captured. ........................................ 93
7.42 Curve fitting and segmentation for the “side” action from the Weizmann video dataset. ........................................ 94
7.43 Speed and velocity of the “side” action from the Weizmann video dataset. Two side step motions can be identified using the speed curve. ........ 95
7.44 Positional data for one of the “side” basic motions. In this motion, the foot is lifted higher in the beginning of the motion and then is gradually placed on the ground. ........................................ 96
7.45 Curve fitting and segmentation for the “jack” action from the Weizmann video dataset. The curve is segmented every time the foot changes direction. Five complete “jack” basic motions are created using this curve. .... 97
7.46 Speed and velocity of the “jack” action from the Weizmann video dataset. The up and down motion of the leg can be seen in the $y$-velocity plot. .......... 98
7.47 Positional data for one of the “jack” basic motions. ..................... 99
7.48 Curve fitting and segmentation for the “jump” action from the Weizmann video dataset. This is a forward jumping motion. Five complete jumping motions are identified by the segmentation process. ..................... 100
7.49 Speed and velocity of the “jump” action from the Weizmann video dataset. ........................................ 101
7.50 Positional data for one of the “jump” basic motions. ..................... 102
7.51 Curve fitting and segmentation for the “pjump” action from the Weizmann video dataset. The “pjump” action is a vertical jumping action. Each jump is segmented into a basic motion for the upwards part of the motion and a basic motion for the downwards part of the motion.

7.52 Speed and velocity of the “pjump” action from the Weizmann video dataset. Five complete “pjump” basic motions can be identified using the speed. Using the $y$-velocity curve, we can tell which basic motion is an upwards motion and which are downwards motions depending on whether the velocity is positive (upwards) or negative (downwards).

7.53 Positional data for one of the “pjump” basic motions. The majority of the motion is in the vertical direction. A small motion in the horizontal ($z$-axis) is visible.

7.54 Curve fitting and segmentation for the “wave” action from the Weizmann video dataset. The motion curve is segmented each time the hand changes direction creating upwards and downwards waving basic motions. Two complete waving motions are detected. The first and last waving motions are not detected as complete motions because there are no spikes in curvature at the beginning and end of the motion curve.

7.55 Speed and velocity of the “wave” action from the Weizmann video dataset. A smoother curve fit than what is shown here may be desirable, but the overall motion is captured.

7.56 Positional data for one of the “wave” basic motions. This is an upwards waving motion.
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Surveys on human action recognition.</td>
<td>17</td>
</tr>
<tr>
<td>2.2</td>
<td>Action recognition papers summarized in this review.</td>
<td>18</td>
</tr>
<tr>
<td>7.1</td>
<td>Instances of the same action for each action in the database.</td>
<td>50</td>
</tr>
<tr>
<td>7.2</td>
<td>Index of figures generated using the CMU motion capture dataset.</td>
<td>53</td>
</tr>
<tr>
<td>7.3</td>
<td>This table shows the recognition rate of different feature vector types using 3-D data. The feature vector types are abbreviated as follows: $P = \text{feature vectors representing } x, y, z \text{ position samples}; PV = \text{feature vectors representing } x, y, z \text{ position and } x', y', z' \text{ velocity samples}; 1f = 1 \text{ foot}, 1f1h = 1 \text{ foot and } 1 \text{ hand}, 2f = 2 \text{ feet}, 2f2h = 2 \text{ feet and } 2 \text{ hands}. \text{ The best results were obtained when data from both feet were used in feature vector generation. Adding the motion of the hands to the feature vectors reduced recognition power compared to when hands were not included in “forward jump” and “vertical jump” actions for most test cases.}</td>
<td>76</td>
</tr>
<tr>
<td>7.4</td>
<td>Accuracy of classifying 2-D feature vectors composed of $(x, y)$ positions at various viewing angles when using only training data from the 0-degree view. The recognition rate of “vertical jump” doesn’t depend on viewing angle. This is because the motion is in the vertical direction. In contrast, the recognition rates of “run” and “walk” actions are greatly affected by viewing angle.</td>
<td>77</td>
</tr>
<tr>
<td>7.5</td>
<td>Accuracy of classifying 2-D feature vectors composed of $(x, y)$ positions captured from various viewing angles when using training data from view directions $-60, -40, -20, 0, 20, 40,$ and $60$ degrees. These results show that PCA is effective at learning actions observed from multiple viewing angles and can be used to create a view-invariant action recognition system.</td>
<td>77</td>
</tr>
<tr>
<td>7.6</td>
<td>Index of figures generated using the Weizmann action dataset.</td>
<td>87</td>
</tr>
<tr>
<td>7.7</td>
<td>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.</td>
<td>110</td>
</tr>
</tbody>
</table>
7.8 The recognition accuracy of our approach is compared to other approaches that were evaluated using the Weizmann dataset.

7.9 Execution speed of each component. The videos in the Weizmann dataset are short with an average length of 2.41 seconds. Feet and hand tracking is done by the tracker component. The tracker executes at a near real-time average speed of 2.76 seconds per video or 22 frames per second. The cfitv component performs curve fitting, segmentation, and feature vector generation at a rate of 13.6 seconds per video. Recognition of basic motions is performed by the recog program. It runs for 8.81 seconds per video.
I would like to take this opportunity to thank my advisor, Dr. Arthur Goshtasby. His insight, support, and guidance throughout my time in the Ph.D. program has been invaluable.

I would also like to thank my committee members, Dr. Matten Rizki, Dr. Thomas Wischgoll, and Dr. David Miller, for their support. Their helpful suggestions greatly enhanced this study.

I would like to sincerely thank the Department of Computer Science and Engineering at Wright State University for making my experience at Wright State both rewarding and positive.

I would also like to thank the Robotics Institute, Carnegie Mellon University, for making the Motion Capture Database and the motion capture visualization software available to this study.

I would also like to thank the Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, for making the Action Video Database available to this study.

Last but not least, I would like to thank my family. It would not have been possible to complete my degree without their support and understanding.
Dedicated to

Kai, Daniel, Darwin, Judy, and Joel Chivers
The goal of a human action recognition system is to automatically detect and recognize actions of humans in a video. Usually, the computer system monitors an environment using a video camera. The task of the recognition system is to make sense of the massive amount of data provided by the camera and recognize the actions that are important for the application under consideration. The large number of potential applications of such a system makes human action recognition a very important field of research. Applications range from visual surveillance scenarios to innovative human-computer interaction.

Examples of surveillance applications include systems that monitor very busy and large public places such as airport terminals or subway stations for suspicious behavior [31]. In these settings, human security personnel could use computer assistance to make monitoring of a large area easier. Another example of a surveillance application is in unmanned autonomous fighting vehicles that are quickly becoming important in military applications. Automated surveillance is one of the primary functions of unmanned aerial vehicles.

In human-machine interaction, a computer may respond to certain actions or gestures of an operator. Examples of human-machine interaction include visualization interfaces where actions can be interpreted to control a system that displays medical image data. An-
other example is video game interfaces where the player is immersed in the game play with the aid of a realistic action-based interface. Human-machine interaction is also useful for human-robot interaction where gesture-based interfaces are used to make the interaction between human and robots more natural [54]. Others have combined gesture recognition with speech recognition to create multimodal human-machine communication interfaces [37]. The capabilities provided by human-machine interaction, and human action recognition in general, are also important to researchers in robotics creating intelligent robots. One example is the ASIMO humanoid robot developed by Honda [12]. As robots become more intelligent, they will also have to be able to understand their environment. Human action recognition will be an important part of this understanding.

In this document, following Moeslund et al. [31], we will distinguish between actions and activities for recognition. A human action is defined as a motion or group of motions that can be identified as a primitive behavior having a beginning and an end. Examples of human actions include extending one’s arm, performing a certain gesture, or taking a step when walking. A human activity is a higher-level event that is formed by a particular set or sequence of human actions. For example, walking and running can be considered basic activities. Actions in walking or running include taking steps with the feet and swinging the arms in a certain way. We can also extend these into more advanced activities that consist of sub-activities. For example, the activity of playing soccer includes running and kicking sub-activities.

Approaches to human action recognition can be grouped into tracking and non-tracking approaches. Non-tracking approaches 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, although successful in many applications, have difficulty recognizing complex actions where movement of different parts of the body is
important. For this reason, tracking-based approaches that use motion of different body parts are often more powerful. However, approaches that use tracking are generally more computationally expensive, making them inappropriate for applications that require real-time responses. Many surveillance, human-machine interaction, and robotics applications require real-time responses.

We propose a new framework for human action recognition that can learn various human actions and is able to recognize them in a very efficient manner without loss of recognition power. Our action recognition system obtains the trajectories of one or more key points on a subject’s body. The trajectories are then analyzed to recognize the underlying action.

In this work, by tracking feet and hands in a video, the 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.

4. Although, in general, information gathered from multiple motion curves reinforces and improves action recognition, in certain situations, use of information from multiple motion curves can confuse the system and reduce recognition rate. This happens when, for example, different parts of the body are not synchronized to the same action.

5. A view-invariant action recognition system can be developed by allowing PCA to observe and learn the same action from multiple views.

We begin with a literature review in Chapter 2. In Chapter 3, an overview of the proposed action recognition framework is discussed. Details of the data collection, motion analysis, and training and recognition steps are given in Chapters 4-6. Next, experimental results are discussed in Chapter 7. Contributions of this work are discussed in Chapter 8. In Chapter 9, we discuss how to adapt the proposed approach to a new data set. Finally, the results of this work are summarized in Chapter 10 and future work is discussed in Chapter 11.
Chapter 2

Related Work

We begin our discussion of related work by examining surveys on action recognition in Section 2.1. Refer to Table 2.1 for a list of surveys. Next, we examine multi-level approaches to action recognition in Section 2.2. Hidden Markov Models, grammers, and other commonly used tools are discussed in Section 2.3. Next, we discuss approaches that do not use tracking in Section 2.4 and approaches that use tracking in Section 2.5. Refer to Table 2.2 for a list of approaches discussed in this chapter.

2.1 Surveys on Action Recognition

Several surveys on human action recognition have appeared in the literature, the most recent being by Poppe [38]. Poppe offers insights into the challenges of human action recognition, such as the need to generalize motion so that recognition is invariant to the speed of the motion and other variations in the performance of an action. We discuss the means by which our method overcomes these challenges in Section 3.2. Poppe categorizes the image processing and the representation of the data extracted from images into two groups: global representations and local representations. Global representation provides a top-down ap-
Local representation focuses on local features of an image or image sequence by extracting local points of interest, such as corner points with fast motion. Poppe also describes the classification phase as either direct, which does not pay special attention to the temporal domain, or a temporal state-space model, which uses techniques such as dynamic time warping to deal with changes in rates of execution of an action. Our proposed approach fits into global data extraction representation and direct classification.

Turaga et al. [51] offer a broad overview of the field of action recognition. The work identifies many of the common action and activity recognition techniques ranging from simple techniques such as template matching, to more advanced hidden Markov model (HMM) based recognition and context free grammar (CFG) techniques. The authors break down the modeling of actions into three categories: graphical models such as Bayesian network models, syntactic approaches like CFG models coupled with HMMs as a recognizer, and logic-based approaches, which include rule-based methods. By comparing Turaga’s survey to Poppe’s survey, one can get a sense of the depth of the research that has been done in this field.

Kruger et al. [26] covers action recognition techniques used in visual surveillance, the entertainment industry, and robot learning and control. While these application areas are closely related, the goals of a specific application and the methods used to achieve the goals can be different. For example, the focus of many visual surveillance applications is to detect actions of interest and possibly even predict the intentions of the person performing the action. In contrast, many robotics applications involve teaching a robot to perform a task or series of actions by human example. The entertainment industry is mostly concerned with tracking motion to map an actor’s movement to a digital animation or to map a player’s movement to events in a video game. Kruger presents methods in three major areas that correspond to the different application areas: interpretation and recognition of action, action learning and imitation, and intention recognition. Our approach fits into
the interpretation and recognition category. Recognition methods are further divided into methods that do not distinguish between body parts, and body-part–based approaches. Our approach is a body-part–based approach since we use trajectories of individual body parts for recognition.

Moeslund [31] provides a survey of vision-based tracking, motion capture, and recognition techniques and covers research in the area during 2000–2006. The methods are classified into 4 main groups: initialization, tracking, pose estimation, and recognition. Initialization is the problem of determining the initial state of a tracking algorithm given a video. This may include determining the initial orientation and position of the various parts of a human model that is used in tracking. Incremental tracking techniques can be applied once initialization is successful. Pose estimation is used to determine the orientation of the various parts of the body. Many approaches exist, but one common method is to use hidden Markov models to test a pose against a set of model poses. Moeslund first presents an overview of low-level techniques, such as background subtraction and other segmentation techniques. Pose estimation and recognition topics, such as action hierarchies, are then covered. Moeslund [30] covers methods that appeared during 1980–2000 using the same overall organization as his later work.

Aggarwal and Park [2] survey the topics of human body modeling, level of detail needed to recognize action, approaches to human action recognition, and high-level recognition schemes. Human body modeling approaches are grouped into model-based and appearance-based methods. In a model-based approach, a fitting process is used to calculate parameters of the human body model so that it best matches the image data. In contrast, an appearance-based approach detects features in an image and doesn’t use a model of the human body. The level of detail needed to perform recognition is based on the application and the types of action being recognized. Aggarwal categorizes the various levels of detail into three groups: gross, intermediate, and detailed. At the gross level, only bounding boxes or other boundaries are used for recognition. For example, Sato and Aggarwal [42]
recognize events such as people meeting, walking away from each other, or passing by another person. Another example is provided by Ivanov and Bobick [21] where events in a parking lot are recognized. At the intermediate level, parts of the body such as the head, torso, arms, and legs are identified. The detailed level includes approaches where recognition is detailed enough to identify action based on motion of a single body part. Hand gesture recognition is one example.

Wang et al. [56] provides a survey of work that appeared from 1997–2001. Basic tracking and image segmentation methods are discussed first, followed by more advanced techniques such as model-based and active contour-based tracking. Common action recognition techniques such as template matching, hidden Markov models, and dynamic time warping are discussed.

Buxton [8] offers a view of work with a focus on artificial intelligence combined with vision to create intelligent systems. Learning and training methods are discussed with an emphasis on generative models and other statistical learning techniques.

Other surveys covering work that appeared before 1999 include Gavrila [15], Aggarwal and Cai [1], Bobick [6], and Cedras and Shah [10].

Gavrila [15] covers the visual analysis of gestures and whole-body movement. Tracking in 2-D and 3-D is discussed along with 3-D pose recovery. Action recognition work is covered with a focus on hidden Markov models and dynamic time warping.

Aggarwal and Cai [1] provide an earlier survey of human motion analysis. Segmentation of human body parts in images is covered, including model-based and non-model-based techniques. Next, tracking techniques are covered including feature-based tracking with a single camera and multiple cameras. Finally, human activity recognition is covered. Approaches are grouped into template matching and methods using state-space models such as hidden Markov models.

Bobick [6] classifies types of human motion into three groups: movement – a motion that is executed consistently and is easily characterized; activity – a sequence of move-
ments; and actions – large scale events such as interaction with the environment, which often require a context (e.g., cooking).

Cedras and Shah [10] provide a survey of motion-based recognition. Topics include extraction of motion information (e.g., optical flow, trajectory parametrization), motion recognition (e.g., cyclic motion detection and gesture interpretation), and human motion tracking and recognition. Human motion tracking includes segmentation of body parts in images and human body modeling. Like the other earlier surveys, most of the work in recognition uses hidden Markov models.

2.2 Multi-level Approaches

Much of the research in action recognition has focused on the use of sequences of 2-D images from a camera. This is natural considering the large number of video cameras already in use by surveillance systems. In order to develop techniques for a high-level understanding of human motion, lower-level image processing steps are used to extract the necessary information from the image sequences. Techniques such as tracking, pose estimation, and gesture recognizers are examples of low-level techniques. Many authors approach the topic by using an example low-level processing technique and add high-level systems that are somewhat independent of the lower-level processes. The high-level system is used to classify actions and activities based on the input from the low-level system.

One example of a high-level system is presented by Ryoo and Aggarwal [40][41]. Their system uses the body-part extraction, pose estimation, and gesture recognition that was presented in Park and Aggarwal [36] as a low-level layer. The low-level gesture recognition is HMM-based and determines the gesture based on the sequence of poses extracted by the pose recognizer. A higher-level recognition layer is developed by Ryoo that can recognize composite actions and interactions by representing actions as a hierarchy of subactions in a CFG-based representation. Subactions include atomic actions that are recognized
Ivanov and Bobick [21] present a system where the high-level recognition layer uses stochastic context-free grammar representations. During string parsing, substitution errors and insertion errors are considered to handle uncertainty in the input string. To illustrate the power of their high-level layer, several example applications are presented. The first system recognizes shapes that are created by the movements of a person’s hands. The higher-level layer can recognize the hand making a shape of a square, for example. Another example system recognizes movements that are made by a music conductor. Both of these examples used hidden Markov models to recognize basic gestures. Finally, a surveillance example is presented that can recognize interaction between cars and people in a parking lot. The low-level layer detects events like when a car enters the parking lot or when a person is detected. The high-level layer can recognize a person being dropped off or picked up.

2.3 Commonly Used Tools

While there are many approaches to action recognition, one of the most common tools used is the hidden Markov model. The primary benefit of hidden Markov models is that they are able to compute the probability of a sequence given a certain model. This makes it easy to match a sequence of observed states in the action being classified to one of the known actions that have a hidden Markov model already trained. As previously mentioned, Park and Aggarwal [36] classify sequences of poses that make up a gesture using hidden Markov models. Another example is given by Bashir [5]. Bashir uses hidden Markov models to classify motions based on the principal component analysis (PCA) coefficients of submotions. Luhr et al. [29] use a hierarchical hidden Markov model that is able to capture the hierarchy often available with human action and activity recognition.

Other common tools used in action recognition are grammars, such as attribute grammars or context-free grammars. Since many actions can be decomposed into logical sub-
actions, it follows that a grammar is a good description tool. This is a form of syntactic pattern recognition where subactions are defined by terminal symbols. A string of terminal symbols is recognized as being a certain action if the string is in the language described by the grammar.

Joo and Chellappa [23] demonstrate the use of attribute grammars to recognize actions based on events generated by a vision system. As an example, the author presents a parking lot surveillance system that detects when a person examines a parked car more than once, indicating that they might have dishonest intentions.

Moore and Essa [32] use stochastic context-free grammars to describe actions, and they extend the previously mentioned work of Ivanov and Bobick [21]. Stochastic context-free grammars include a probability for every production rule in the grammar. This can be used to rank a parse by computing the probability of the parse. Moore’s contributions are in error detection of string operations and recovery from error in string parsing. Collaborative tasks such as actions in card games are the focus of the paper.

While it is useful to model actions of interest (as is done in the proposed approach), Kim and Grauman [25] use an unsupervised approach to recognize abnormal activities that may have unpredictable variations. Interestingly, no action is described. They use optical flow patterns from a typical video using a mixture of probabilistic principal component analyzers introduced by Tipping and Bishop [50]. A Markov random field model and Bayesian inference is used to detect abnormal patterns of optical flow, which may correspond to an abnormality. A training video clip is used so that the system can learn the “normal” patterns of optical flow.

## 2.4 Action Recognition Without Tracking

Because the human brain is a highly evolved action classifier, it makes sense that some researchers try to mimic the brain’s behavior using neural network–based approaches. Es-
cobar et al. [13] introduces a neural network–based system that attempts to model specific areas of the brain that are dedicated to motion. In their approach, motion maps are built by analyzing the activation of cells in the network. The motion maps are then used for classification by comparing them to motion maps from a training dataset.

Chaudhry et al. [11] introduce a technique that uses a histogram of optical flow to detect actions. The author illustrates that a characteristic optical flow profile is present for different actions. Because optical flow depends on the direction of movement and how close the subject is to the camera (zoom), the histogram is normalized, and the angle of the optical flow vectors compared to the horizontal axis is used to compute the bin that the vector is added to in the histogram. This makes the optical flow independent of the left or right direction. The magnitude of each optical flow vector determines how much it contributes to the histogram. This removes the effect of noise from the static background. The advantage of this approach is its simplicity and that little low-level processing is needed. For example, background subtraction is not necessary. The video data used to evaluate the technique are low-resolution videos from the Weizmann database [53] and include actions such as walking, running, jumping, bending, and hand waving.

Bregonzio et al. [7] present another technique where background subtraction is avoided, although they do use frame differencing, which can be seen as a weaker form of background subtraction. They represent actions as clouds of interest points taken over various time intervals. Regions of interest are detected using 2-D Gabor filters.

Liu et al. [28] focus on unconstrained video where background subtraction techniques typically fail. They use feature detectors such as Harris-Laplacian, Hessian-Laplacian, and MSER detectors on each frame to extract static features, and then a spatiotemporal interest point detector is used to extract motion features. Next, feature pruning is done to remove features created by unintended camera shake or moving cameras. Adaptive boosting is used for classification.

Liu and Yuen [27] compute an information saliency map from video to segment peri-
Periodic motion and to form action units to represent the period cycle. PCA is used to reduce the dimensionality of the periodic cycle and classification is done using a multi-class AdaBoost classifier.

Gorelick et al. [17] use a form of concatenation of silhouettes to build a space-time volume that contains information about the pose of the human over time. Properties of the silhouette are extracted by solving the Poisson equation and computing several local feature types that are then used to form a feature vector. The action is matched to actions in the database using a nearest neighbor classifier.

Wang et al. [55] evaluate the performance of various feature detectors and descriptors used to represent features. In particular, the Harris3D, Cubiod, Hessian, and dense sampling detectors are evaluated. The descriptors evaluated include the Cuboid, HOG, HOG3D, and extended SURF descriptors. A bag-of-features Support Vector Machines (SVM) approach is used for recognition. They show that the recognition performance of the system is effected by both the feature detector and the descriptor. Furthermore, certain feature detectors perform better with certain descriptors, while other feature detectors perform better when used with different descriptors.

Guha and Ward [19] use sparse representations to learn histograms of spatio-temporal motion features. First, features of the video are detected. Cubiod features and a new feature type called \textit{local motion patterns} are explored. Next, a dictionary-learning algorithm is used to create overcomplete dictionaries. Finally, sparse representations are computed over the overcomplete dictionary. Actions are classified using a Support Vector Machine classifier. The author explores the use of a single dictionary for all actions, class-specific dictionaries, and a concatenated dictionary.

Castrodad and Sapiro [9] propose a system that uses two levels of sparse coding with dictionary learning to recognize actions in videos. First, motion features are extracted from the video. A dictionary learning algorithm is then used to learn a dictionary for each action class. Next, features that are detected in the video are represented with sparse coding.
Finally, a second training phase is applied where inter-class dictionaries are learned. It is shown that the derivative dictionaries and inter-class sparse coding improves performance over a single learning step.

Wang and Mori [58] use a hidden conditional random field (HCRF) model, commonly used in object recognition, to represent action. Interest points are found using feature detectors. Then, the appearance of the local patch surrounding each interest point is modeled using a conditional random field. They modify the HCRF model to include large-scale global features rather than using only the appearance of local patches. It is shown that using both local patch appearance and global features results in a higher recognition rate compared to approaches using only one of these methods.

Sun et al. [46] represent low-level features, such as those provided by the HOG3D detector, using a sequence of Self-Similarity Matrices (SSM). Next, a 3-D structure, called a Joint Self-Similarity Volume, is created from the SSM Sequence. The dimensionality of this structure is reduced using an approximation algorithm.

Thi et al. [48] recognized that many approaches that use feature detectors organize features as orderless sets (e.g., bag-of-features). This removes detail from the action and makes recognition of more complex actions difficult. Thi et al. overcome this by introducing structure to groups of local features. They present Dynamic Conditional Random Field and Structural Support Vector Machine techniques for this task.

2.5 Action Recognition Using Tracking

Su et al. [45] present a non-vision–based technique to recognition. Using an ultrasonic 3-D tracking system, they recognize 10 typical arm movements made when using Taiwanese Sign Language. The position of the hand containing an ultrasonic transmitter is tracked based on input signals from three microphones. Smoothing and filtering are applied to the input to remove the effect of noise. A simple method is used to find turning points in the
motion based on the rate of direction change in the curve. Starting and ending points of the curve are simply the first and last point in the sequence. Curves are labeled based on either their straightness and direction (horizontal, vertical, positive, negative) or, if the curve has enough arcedness, a label based on the basic shape is given. A fuzzy labeling method is used for classification.

Yao et al. [60] combine recognition without tracking and recognition with tracking into a single approach. The authors noticed that action recognition and pose estimation are closely related tasks. They use 2-D appearance-based action recognition using multiple cameras to help simplify 3-D pose estimation and then use the resulting pose information to recognize actions. First, appearance features, such as color, optical flow, and spatio-temporal gradients, are extracted from the video frames for each view. Hough-transform voting is then used to distribute particles in a particle-based optimization scheme over action-specific manifolds. These manifolds are used to estimate a 3-D pose of the person using optimization. Finally, pose-based features are used to perform the final recognition step. Initializing the pose system with 2-D appearance-based action recognition is used to reduce the computation time needed to estimate each 3-D pose. The final recognition results are improved using 3-D pose-based methods compared to the initial 2-D recognition.

Because motion trajectories contain valuable information for action recognition, there has been recent work using motion trajectories. Likewise, PCA is a common tool used to reduce the dimensionality of data. Here we present recent work that uses either motion trajectories, PCA, or both.

Bashir et al. [5] represented subtrajectories using PCA coefficients. Like our approach, trajectories are segmented at points of high curvature. The subtrajectories are then represented by their PCA coefficients by training Gaussian mixture models. Finally, recognition is performed using hidden Markov models. One HMM per class to be recognized is used. Bashir demonstrated his approach on an American sign language dataset. Our approach differs in that we classify submotions using clustering in eigenspace. We also make
use of curve fitting to segment the motion, interpolate over gaps in the data, and generate feature vectors. Bashir uses filtering on the raw trajectory data to perform resampling.

Similar to our approach, Gritai et al. [18] present a system that takes trajectories of different anatomical landmarks from a tracking system. A dissimilarity measure that they develop and dynamic time warping are used to match 3-D trajectories to exemplar actions. They also consider body size and proportion differences between subjects and apply transformations to compensate for the differences.

Wu and Li [59] use properties of trajectory data to create motion signatures. Wu explores reducing the dimensionality of the signature using PCA to create optimized signatures. Recognition is done using various methods including dynamic time warping, Mahalanobis distance between optimized signatures, and Bayesian classification using Gaussian mixture models.

Han et al. [20] use a Gaussian process latent variable model to learn movement in a hierarchical manifold space. They use PCA and k-means clustering to group motions in manifold subspaces. The motions are recognized using a tree-based cascade conditional random field model. No motion segmentation is used. Demonstration of the system is done using motion capture files from the CMU motion capture database.

Gong and Medioni [16] perform view invariant action recognition using spatio-temporal manifolds. First, trajectories of points on the body are obtained using motion capture data. Actions are learned using structure learning to produce low-dimensional spatio-temporal manifolds. Actions are then compared by aligning two motion sequences and calculating a motion similarity score using a form of dynamic time warping called dynamic manifold warping. Unlike our approach, their approach requires that many points on the body are tracked to produce the manifold structures. As a result, they were unable to apply their method to the Weizmann dataset because accurate tracking of many points on the body in videos is challenging.

The proposed approach, in contrast to the majority of approaches in action recogni-
Table 2.1: Surveys on human action recognition.

<table>
<thead>
<tr>
<th>Year</th>
<th>First author</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>Poppe [38]</td>
</tr>
<tr>
<td>2008</td>
<td>Turaga [51]</td>
</tr>
<tr>
<td>2007</td>
<td>Kruger [26]</td>
</tr>
<tr>
<td>2006</td>
<td>Moeslund [31]</td>
</tr>
<tr>
<td>2004</td>
<td>Aggarwal [2]</td>
</tr>
<tr>
<td>2003</td>
<td>Wang [56]</td>
</tr>
<tr>
<td>2003</td>
<td>Buxton [8]</td>
</tr>
<tr>
<td>1999</td>
<td>Gavrila [15]</td>
</tr>
<tr>
<td>1999</td>
<td>Aggarwal [1]</td>
</tr>
<tr>
<td>1997</td>
<td>Bobick [6]</td>
</tr>
<tr>
<td>1995</td>
<td>Cedras [10]</td>
</tr>
</tbody>
</table>

tion, focuses on a balance between recognition rate and computational efficiency. We first simplify the tracking problem by requiring only a few key points on the body to be tracked. We show that the curves that are produced by the trajectory of each tracked point are sufficient to recognize various actions. While tracking in the proposed approach is simpler than tracking the pose of the entire body, tracking individual body parts allows for the possibility of recognizing subtle actions [26]. Next, we perform curve fitting and segmentation on the motion curves to form basic motions. Then we produce a feature vector for the basic motion and use PCA to reduce the dimensionality of the feature vector. Classification is done by clustering in the eigenspace computed using training data.
Table 2.2: Action recognition papers summarized in this review.

<table>
<thead>
<tr>
<th>Year</th>
<th>First author</th>
<th>Tracking</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999</td>
<td>Tipping [50]</td>
<td>Mixture of probabilistic PCA</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>Ivanov [21]</td>
<td>Multi-level, stochastic context-free grammars</td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>Su [45]</td>
<td>Fuzzy labeling</td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>Moore [32]</td>
<td>Stochastic context-free grammars</td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>Lhur [29]</td>
<td>Multi-level</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Joe [23]</td>
<td>Attribute grammars</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Park [36]</td>
<td>HMM</td>
<td></td>
</tr>
<tr>
<td>2006</td>
<td>Ryoo [40]</td>
<td>Multi-level, HMM</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Bashir [5]</td>
<td>PCA, hierarchical HMM</td>
<td></td>
</tr>
<tr>
<td>2007</td>
<td>Gorelick [17]</td>
<td>Silhouette volumes</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Bregonzio [7]</td>
<td>2-D Gabor filters</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Chaudhry [11]</td>
<td>Histograms, optical flow</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Escobar [13]</td>
<td>Neural network</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Gritai [18]</td>
<td>Trajectory matching</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Kim [25]</td>
<td>Mixture of probabilistic PCA</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Liu [28]</td>
<td>Adaptive boosting</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Ryoo [41]</td>
<td>Multi-level</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Wang [55]</td>
<td>SVM</td>
<td></td>
</tr>
<tr>
<td>2009</td>
<td>Wu [59]</td>
<td>PCA</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Han [20]</td>
<td>PCA, hierarchical manifold</td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td>Liu [27]</td>
<td>PCA</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Gong [16]</td>
<td>Spatio-temporal manifolds</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Guha [19]</td>
<td>Dictionary learning, SVM</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Sun [46]</td>
<td>Self-similarity matrices</td>
<td></td>
</tr>
<tr>
<td>2011</td>
<td>Wang [58]</td>
<td>Hidden conditional random field</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Castrodad [9]</td>
<td>Dictionary learning</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Thi [48]</td>
<td>Dynamic conditional random field, structural SVM</td>
<td></td>
</tr>
<tr>
<td>2012</td>
<td>Yao [60]</td>
<td>Pose estimation</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 3

Proposed Approach

In this chapter, we present an overview of the proposed approach and discuss the strengths of our approach.

3.1 Overview of Proposed Approach

The flow diagram of the proposed action recognition system is given in Figure 3.1. The proposed approach is composed of four phases: data collection, motion analysis, training, and recognition.

In the data collection phase, trajectory data are produced by tracking key points on a subject’s body. Two types of input data are considered in this work: motion capture data and video data. The location of a key point on a subject’s body is chosen by taking into consideration the set of actions to be recognized. We focus on the feet and hands of subjects because the trajectories formed by the movement of the feet and hands can be used to recognize a wide variety of actions. Trajectories are extracted from motion capture data using transforms described in the motion capture data file. This step is described in more detail in 4.1. In Section 4.3, a tracking system is introduced that is able to track the feet.
Figure 3.1: Flow diagram of the proposed action recognition system. 1) In the data collection phase, trajectories are produced from video or motion capture data. 2) Curves are fitted to the trajectories in the motion analysis phase. Next, the curves are segmented at peaks in curvature forming basic motion curves. Feature vectors are then created for each basic motion curve using properties of the curve such as position, velocity, or curvature over time. 3) In the training phase, PCA is used to calculate an eigenspace matrix from a set of training feature vectors. 4) In the recognition phase, a feature vector created from an unknown basic motion is classified by first projecting it onto the eigenspace. Next, the Euclidean distance between the basic motion projection and each training data feature vector projection is used to classify the basic motion by using the k-nearest neighbor classifier.
and hands of a person in a video. By tracking a few key points on the subject’s body, we reduce each video frame to just a few points. We will show that the trajectories created by tracking these points is sufficient for recognizing various actions.

Once a set of trajectories is obtained, the trajectories are used as input for the motion analysis phase. First, curve fitting is used to represent each trajectory by a smooth motion curve. A motion curve reduces the effect of noise in the trajectory and allows computation of curvature and other features of the trajectory. We use curvature to segment a motion curve into meaningful parts called basic motion. Details of this segmentation process is provided in Section 5.1.

After segmenting a motion curve into basic motions, a feature vector is generated for each basic motion using the fitted curve model. The curve model used allows for the representation of any basic motion with a feature vector of a fixed size. Details of this step are given in Section 5.2. The feature vectors are then used to either build a training data set or recognize an unknown motion once the system is trained.

In the training phase, a representative set of feature vectors for each action is chosen to form the training data set. Principal component analysis (PCA) is used to create an eigenspace from the training data set. Once the eigenspace is created, the training feature vectors are projected to the eigenspace to reduce the dimensionality of the training data while maintaining the separability of the training data for classification. Clusters are formed in the eigenspace for each action. Details are provided in Section 6.1.

The final phase is the recognition phase. To classify an unknown feature vector, first, the feature vector is projected onto the eigenspace created during the training phase. Next, the Euclidean distance between the unknown feature vector and each training data feature vector is calculated. Finally, the feature vector is classified using the k-nearest neighbor classifier. Because feature vector projections belonging to the same action type cluster near each other in the eigenspace, the nearest neighbors to the unknown feature vector projection should be of the same action type. See Section 6.2 for more details.
3.2 Strengths of Proposed Approach

In this section, we describe the strengths of our approach and show how our approach overcomes the challenges of action recognition.

One of the primary benefits of our approach is its balance between speed and recognition capability. We show that the approach achieves a high recognition rate, yet it is capable of performing in near real-time on modern computer hardware. The majority of the existing approaches use computationally expensive recognition techniques and do not focus on computation time. We begin by reducing the tracking problem to tracking a few key points on the subject’s body rather than requiring the full pose of the person to be computed. This process by itself compresses each frame of video input to only a few points being tracked for recognition. Next, the classification process is made more efficient by using PCA to reduce the size of the projections used for classification. Classification is also a simple process that uses the distance between the input projection and cluster mean or nearest neighbor. A discussion of the execution time of our programs is given in Section 7.4.

Our approach performs the recognition step on each basic motion and identifies the action using the basic motions that we have observed. Recognizing action at this level of detail has a number of advantages that allow the creation of powerful high-level activity recognizers that use our technique. First, it is possible to detect actions of different parts of the body separately. This makes it possible to recognize composite actions where the feet and the hands are each performing an action that should be recognized. A high-level recognizer may also analyze the sequence of basic motions that are recognized to determine an overall activity being performed. Furthermore, with the proposed method, it is possible to create a high-level recognizer that is robust to misclassification of basic motions. For example, misclassification of a single basic motion can be corrected or ignored by observing basic motions that precede, occur at the same time, or follow the basic motion.

The human action recognition problem poses a number of inherent challenges. One of the major challenges is that the recognition method must generalize over variations between
instances of the same action, and at the same time it must distinguish between the different actions.

Variation between trials of the same action occur due to differences in execution rate of the action, differences in motion between different trials of the action, and variation caused by anthropometric differences between people [38]. For example, a person with long legs will tend to have a longer stride length when walking than a person with short legs. Variation also occurs due to changes in the viewpoint of the camera and configuration. For example, the distance between the camera and the subject changes the size of the observed action in the input video.

Our method removes or reduces the effect of the variations in the action by considering them when generating the feature vector for the basic motion. For example, by scaling the values of the feature vector appropriately, the differences in relative size between the features can be removed. Our method is also invariant to the execution rate of the action because the feature vector does not have to consider the execution rate at all. This is possible because the curve fitting process provides a continuous representation of the motion curve. We also do not rule out the use of timing, however, as it may be important for some applications. In Chapter 7, we compare the performance of our system when velocity is included or excluded in the feature vectors.

Our method is also robust to variation in the data due to the classification scheme we employ. We choose the most likely classification by finding the nearest training data projections in eigenspace.
Chapter 4

Data Collection

In this chapter, we introduce the datasets that were used to evaluate the performance of our action recognition system and the techniques that were used to produce motion data from the input data.

4.1 Motion Capture Data

To evaluate our action recognition system using 3-D trajectory data, we use the motion capture data prepared at the Robotics Institute, Carnegie Mellon University [52]. To create this motion capture dataset, an advanced tracking system was used to track small white balls or markers attached to key locations on the body. The tracking system used multiple cameras to extract the 3-D location of each marker in each frame of a video. Figure 4.1 is a visualization of a motion capture dataset for a runner. The curves show the trajectories for the left hand and the left foot of the runner. A trajectory is formed by tracking the location of the marker in different frames of a video.

The motion capture files contain a skeletal representation of the person, and joint angles for each sample time that describe the pose of the skeleton. To calculate the position
Figure 4.1: Visualization of motion trajectories representing the motion of the left hand and the left foot of a runner in a video.

of the key interest points for each sample time, the joint angles are used to create transformations that are applied to the skeletal structure. Once the transformations are applied, the position of each interest point is available.

In Section 7.1, screen shots of each action in the motion capture dataset are presented along with the results of experiments performed using the motion capture dataset.

4.2 Video Data

To evaluate our action recognition system using video sequences, we use the Weizmann Human Action dataset [53]. The Weizmann dataset is a commonly used dataset to evaluate the performance of action recognition methods making it appropriate for this study. Videos in the dataset contain a full-body view of human subjects taken with a stationary camera. One disadvantage of the Weizmann dataset is that the videos are low resolution (180 x 144). This can make tracking more difficult because details in the video are lost compared to higher resolution videos. Our experimental results show, however, that our method works well with low resolution videos.

Example screen captures of the video sequences are available in Section 7.2.
4.3 Motion Data Generation from Videos

In this section, the techniques used by our approach to track the feet and hands of human subjects in videos are described. First, background subtraction is used to separate the silhouette of the person from the static background in each video frame. Details are given in Section 4.3.1. Next, the location of each foot is found using the silhouette boundary formed by the background subtraction step. Feet tracking is described in Section 4.3.2. The location of each hand is found using dense optical flow and this technique is discussed in Section 4.3.3. In Section 4.3.4, our bend-over action detector is described.

4.3.1 Background Subtraction

Background subtraction is widely used to separate a moving object from a stationary background. In our approach, background subtraction is used to extract the silhouette of the person in motion in each frame of a video. Figure 4.2 illustrates an example. We use the mixture of Gaussian (MoG) background subtraction technique [44] with our own enhancements to delineate a clean silhouette of a moving person in each frame of a video.

MoG Method

In the MoG background subtraction method, a set of Gaussian distributions is used to model the color of each pixel. The color of a pixel is typically represented by three color component values (e.g., $R$, $G$, and $B$ in the $RGB$ colorspace). The set of Gaussian distributions forms a mixture where the probability of observing a pixel color $X_t$ at time $t$ is given by:

\[
\sum_{i=1}^{k} w_{i,t} \ast \eta(X_t, \mu_{i,t}, \Sigma_{i,t}),
\]

where $k$ is the number of Gaussian distributions in the mixture, $w_{i,t}$ is the estimated weight for the $i^{th}$ Gaussian distribution at time $t$, $\mu_{i,t}$ is the mean of the $i^{th}$ distribution at time $t$, and $\Sigma_{i,t}$ is the covariance of the $i^{th}$ distribution at time $t$. The function $\eta$ represents the Gaussian density function.
\[ \Sigma_{i,t} \] is the covariance matrix of the \( i \)th Gaussian at time \( t \), and \( \eta \) is the Gaussian probability density function given in the following equation:

\[
\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2} (X_t - \mu)^T \Sigma^{-1} (X_t - \mu)}. \tag{4.2}
\]

To reduce the time needed to compute the parameters of each distribution during training, the MoG method avoids calculating the inverse of the covariance matrix, \( \Sigma \), by assuming that the variance of each color component is equal to that of the other color components. By making this assumption, the covariance matrix can be defined as:

\[ \Sigma_{i,t} = \sigma_i^2 I, \tag{4.3} \]

where \( \sigma_i \) is the variance of the color and \( I \) is the identity matrix.

To classify a pixel color as foreground or background, the color of the pixel is compared to the corresponding set of distributions. The pixel color \( X_t \) is said to match distribution \( i \) if:

\[ ||X_t - \mu_i|| < 2.5\sigma_i^2. \tag{4.4} \]

If the color of the pixel matches one of the distributions, and the distribution has enough historical evidence, then it is considered a background pixel. Otherwise, it is considered a foreground pixel. The matched distribution is then updated by adjusting its mean, variance, and weight. The weight is used to indicate how many times the color has been observed. If a new color is observed, the distribution with the least amount of historical evidence (smallest weight) is updated.

To increase performance, we train the system during the first 60 frames of the video with only background pixels. This allows us to obtain an accurate model of the background pixel colors. After the training period, only the classification step is performed, which is a fast operation. We use three distributions per pixel \( (k = 3) \).
Shadow Removal

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 4.2(b) and 4.2(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. To accomplish this, we modify equation 4.4 so that a pixel color matches distribution \( i \) in the mixture if:

\[
|X_{t,L} - \mu_{i,t,L}| < R \land \sqrt{(X_{t,a} - \mu_{i,t,a})^2 + (X_{t,b} - \mu_{i,t,b})^2} < 2.5\sigma_i^2, \tag{4.5}
\]

where \( X_{t,L} \), \( X_{t,a} \), and \( X_{t,b} \) are the pixel’s \( L \), \( a \), and \( b \) component value at time \( t \), respectively, \( \mu_{i,t,L} \), \( \mu_{i,t,a} \), and \( \mu_{i,t,b} \) are the \( L \), \( a \), and \( b \) mean values of distribution \( i \) at time \( t \), \( \sigma_i^2 \) is the standard deviation of distribution \( i \), and \( R \) is an experimentally chosen lightness threshold. The value of \( R \) depends on how dark the shadow is expected to be compared to the mean value. Using \( R = 50 \) produced good results for the videos that we tested.

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. This is demonstrated in the following figures. Figure 4.3 is a plot of a single pixel using the RGB colorspace. The pixel becomes part of a shadow between samples 310 and 350. The \( R \), \( G \), and \( B \) component values decrease in the shadow region because the pixel color becomes darker. In Figure 4.4, the
same pixel is plotted using the HSV colorspace. Notice that the value of the V component decreases in the shadow area. This is expected because as V decreases, the pixel becomes darker. Unexpectedly, however, the value of the H component increases when the value of V changes. This means that the shadow is not isolated in the V component. Another undesirable property of HSV is that the H component has a large variance. Finally, Figure 4.5 is a plot of the same pixel using the Lab colorspace. Notice that the a and b component values remain stable while the L component isolates the shadow. This indicates that Lab is a good choice for shadow removal.

Because of the importance of shadow removal in computer vision, other work in shadow detection and removal exists in the literature. For example, Shan et al. [43] compare the effectiveness of using different colorspace for shadow removal. Their work indicates that the HSV and normalized rgb colorspace may be better suited for removing dark shadows, while Lab is better for light shadows. Nghiem et al. [34] combine the use of the RGB and HSV colorspace to detect the chromaticity consistency in a region. They also use texture consistency because the texture of the region should not change due to light shadowing. We found that our Lab shadow removal technique is sufficient for the videos that we encountered, so more complex methods were not evaluated.
Figure 4.3: The RGB color components of a single pixel are plotted for each sample frame. This pixel becomes part of a shadow from sample 310 to 350. As expected, the $R$, $G$, and $B$ values decrease when the pixel is in the shadow area indicating that the color is darker than the non-shadow areas.
Figure 4.4: The *HSV* color components of a single pixel are plotted for each sample frame. This pixel becomes part of a shadow from sample 310 to 350. Notice that the *V* value decreases in the shadow area, as expected, but the *H* value increases unexpectedly. This indicates that *HSV* does not isolate the shadow in the *V* component.
Figure 4.5: The Lab color components of a single pixel are plotted for each sample frame. This pixel becomes part of a shadow from sample 310 to 350. Notice that the value of the $L$ component decreases in the shadow while the $a$ and $b$ components remain stable. This makes Lab a good choice for shadow removal because the shadow is isolated in the $L$ component.
4.3.2 Tracking of Feet

The location of each foot is found using a method similar to that described by Jean et al. [22]. 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 also in the way we estimate the size of the feet to find their position in a greater variety of poses. Figure 4.6 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. [22]. In Figure 4.6, the line of separation is shown.
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 4.6(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 of available data to fill in the missing data. Figure 5.1 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.

Figure 4.6: Tracking the feet in various actions. (a) Walking, (b) jumping, (c) running, and (d) side stepping.
4.3.3 Tracking of 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. See Figure 4.7 for example screen captures where the location of the hands are tracked.

We accomplish tracking of hands using optical flow estimation to find the motion made by the hands. In general, optical flow is used to estimate the change in location \((dx, dy)\) for a pixel (or pixel neighborhood) in two images such that

\[
I_n(x, y) = I_{n+1}(x + dx, y + dy),
\]

where \(I_n\) represents the pixel at location \((x, y)\) in image \(n\) and \(dx\) and \(dy\) represent the \(x\) and \(y\) displacement of the pixel, respectively, in image \(n + 1\).

We use the dense optical flow estimation method introduced by Farnebäck [14]. In this method, the neighborhood of each pixel is approximated using polynomial expansion. The change in location of a pixel neighborhood between two images can then be solved for by using the two polynomials (one for each image) that estimate the pixel neighborhood. This method provides a 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. To find the motion made by the hands, 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.

4.3.4 Detecting the Bend-Over Action

Certain actions, such as bending over to pick up something, bring the hands in proximity of the feet. This results in feet tracking errors. See Figure 4.8 for an example. We alleviate this
Figure 4.8: The bend-over action brings the hands in proximity to the feet, causing errors in feet tracking. We remove these errors by detecting when the person is bending over using changes in height of the silhouette. We reliably detected the bend-over action in all videos in the Weizmann dataset.
Chapter 5

Motion Analysis

In this chapter, the processes that are used to analyze the trajectory data and produce feature vectors are discussed. The first step is to fit a curve to each trajectory. Next, the curves are segmented into basic motions. Details are given in Section 5.1. Feature vectors are created using the curve segments corresponding to the basic motion segments. See Section 5.2 for details.

5.1 Curve Fitting and Motion Segmentation

Once a trajectory is created by tracking a point, curve-fitting is used to smooth the trajectory and produce a parameterized representation for the trajectory. We use cubic B-spline in our implementation, but other parametric curves may be used as well. A parametric curve is defined in terms of a parameter $u$ that varies from 0 to 1.

In order to include time information in a curve, we consider a 4-dimensional curve, 3 location coordinates, and 1 time coordinate. This allows computation of velocity at any point on a curve and makes it possible to maintain the time information in the curve model, which is essential when generating feature vectors for multiple tracks. More details of this
A motion curve is partitioned into basic motion segments. 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 locally peak curvatures of a motion curve provide a reliable means of segmenting the curve into basic motions.

Curvature is defined as:

$$\frac{1}{\rho_i} = \frac{|p_i^u \times p_i^{uu}|}{|p_i^u|^3},$$  \hspace{1cm} (5.1)$$

where $\rho_i$ is the radius of curvature, $p_i^u$ is the first derivative of the curve at point $p_i$, and $p_i^{uu}$ is the second derivative of the curve at point $p_i$ [33].

The segmentation algorithm begins by finding the set of locally peak curvature points along the curve where the curvature exceeds a minimum curvature threshold. Next, the number of points in this set is reduced if two or more consecutive peak curvature points are located close to one another in space. The point with largest curvature is kept and the others are removed from the set. Multiple spikes in curvature may occur when the trajectory is nearly stationary, such as when the foot is stationary during the walk action, due to inaccuracies in tracking. Finally, the curve is segmented by analyzing the speed of the motion before and after each curvature spike. A speed threshold is used to identify the parts of the curve that form basic motions, such as a foot step.

Figure 5.1 shows the result of curve fitting and curve segmentation using the motion curve obtained by tracking a foot of a runner. In this example, only one spike in curvature occurs each time the foot is placed on the ground, making segmentation simple. A more complex example involving multiple curvature peaks is given in Figure 7.5.

Using curvature and speed to identify where basic motions begin and end is more reliable than using speed alone. This is because changes in speed may not be easily detected
when the direction of motion changes rapidly. A good example of this is when a person is waving a hand. The hand’s trajectory may change direction too quickly for it to be detected using speed alone because the sample rate of the video is too low. Curvature identifies the sharp change in motion easily. However, using curvature alone does not always identify the exact points on the curve where motion starts or stops. For this reason, we use speed to improve the segmentation process. Consider the motion made by a foot of a walking person. When the foot is stationary, the curve may form one or more spikes in curvature, as was shown in Figure 7.5, due to inaccuracies of the tracker when finding the foot’s position in each frame. Using speed allows us to find where the foot starts and stops moving near the peak curvature points to accurately segment the curve into basic motions.

5.2 Feature Vector Generation

The goal of feature vector generation is to produce a feature vector of a fixed size for each basic motion. The feature vectors must produce similar feature vectors for similar motions and produce dissimilar feature vectors for different motions, allowing discrimination of various action classes with PCA. The size of a feature vector depends on the number of samples taken of a basic motion and the number of values saved per a sample.

A feature vector is computed by uniformly sampling a chosen property of a motion curve from start to end. A simple procedure outlined in Section 7.1 uses $x$-, $y$-, and $z$-coordinates of curve points after being transformed to become invariant of view angle. Such a feature vector is of form:

$$V = [x_1, y_1, z_1, \ldots, x_n, y_n, z_n], \quad (5.2)$$

where $n$ is the number of samples taken from a basic motion curve.

Another type of a feature vector contains both position and velocity measures and is
of the form:

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

(5.3)

where \( P_i = (x_i, y_i, z_i) \) is the position and \( V_i = (x'_i, y'_i, z'_i) \) is the velocity at \( P_i \).

In order to make a feature vector invariant of view, the coordinate values are translated such that the first coordinate is at the origin. Then, the motion is rotated so the first and last points fall along the \( x \)-axis. This aligns all motions for comparison when using positional values within feature vectors.

We have found that it is useful to include information from more than one track point into a feature vector. For example, when creating a feature vector describing the basic motion for the left foot, it is useful to include information about the motion (or lack of motion) of the right foot. This is accomplished by evaluating each curve in the range given by the start and end parameter 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:

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

(5.4)

where \( a_i \) is a sample from the left foot over the basic motion range and \( b_i \) 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. See Figure 5.2 for two example feature vectors containing the y-position of both feet. Notice that if you compare the left half of both plots, the feature vectors appear to be somewhat similar. Adding the other foot to the right half of both plots reduces the similarity of the two feature vectors.
Figure 5.1: (a) Visualization of a running person. The curve shown represents the motion of the left foot. (b) Curve fitting and segmentation of a foot motion curve during running. (c) The curvature plot of the motion curve. A large spike in curvature is observed each time the foot is placed on the ground. The curve is segmented at the points labelled “seg point.” There are two segmentation points for each curvature spike corresponding to where the motion of the foot starts and stops.
Figure 5.2: (a) Feature vector containing the y-position of both feet for the “run” action. Samples 0-99 are the y-values for the foot producing the basic motion. Samples 100-200 are the y-values for the other foot. (b) Feature vector containing the y-position of both feet for the “forward jump” action. Samples 0-99 are the y-values for the foot producing the basic motion. Samples 100-200 are the y-values for the other foot.
Chapter 6

Training and Recognition Using PCA

6.1 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 [47].

We compute the covariance matrix $C$ from $M$:

$$C = \frac{1}{N-1}(M \times M^t), \quad (6.1)$$
where $N$ is the number of elements in each feature vector, and $M^t$ 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 of eigenvectors to form the final eigenspace where $l$ is the desired number of 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 [47].

### 6.2 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 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.

Figures 6.1 and 6.2 show the projection of training data onto the 1st and 2nd, and 2nd and 3rd eigenvectors, respectively, of an eigenspace created using the Weizmann video dataset. Notice that the clusters are already visible. Other dimensions further separate the data into clusters. The unknown input feature vector, marked with an ‘x,’ corresponds to a “run” basic motion and will be classified as such because the projection’s nearest neighbors are “run” training data projections.
Figure 6.1: Projection of training feature vectors onto the 1st and 2nd eigenvectors created using the Weizmann video dataset. The unknown input feature vector corresponds to a “run” basic motion and the nearest neighbors are “run” training data projections. Notice that two clusters have formed for the “pjump” (vertical jump) action. This is because each vertical jump is segmented into two basic motions: one for the upward motion and one for the downward motion.
Figure 6.2: Projection of training feature vectors onto the 1st and 3rd eigenvectors created using the Weizmann video dataset. The unknown input feature vector corresponds to a "run" basic motion and the nearest neighbors are "run" training data projections.
In this chapter, we discuss the performance of our recognition system and the experiments we used to evaluate the proposed approach. The evaluation of our system was carried out in two phases. In the first phase, we used motion capture data from the CMU Motion Capture Database [52]. Using motion capture data allowed us to evaluate our approach early in the research process because trajectories can be obtained from the motion capture data files. This allowed us to evaluate the curve fitting, feature vector generation, and recognition portions of our system without needing a tracker. In the second phase, we evaluated our system using 2-D tracking of the feet and hands of subjects in video. We chose the Weizmann Human Action dataset [53] because it is a commonly used dataset for evaluating action recognition in the literature.

In the following sections, examples of each type of action in each dataset are illustrated, we demonstrate the curve fitting and segmentation process, and we discuss the recognition rate under various circumstances. We begin with a discussion of the results using the motion capture dataset in Section 7.1. Next, the results using the video dataset is described in Section 7.2. The recognition accuracy obtained using our approach is compared to the accuracy of other approaches that were evaluated using the Weizmann dataset in Section 7.3. Finally, the execution speed of our programs is discussed in Section 7.4.
Table 7.1: Instances of the same action for each action in the database.

<table>
<thead>
<tr>
<th>Action</th>
<th>Instances</th>
<th>Action</th>
<th>Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward Jump</td>
<td>98</td>
<td>Jump</td>
<td>23</td>
</tr>
<tr>
<td>March</td>
<td>55</td>
<td>Run</td>
<td>29</td>
</tr>
<tr>
<td>Stairs</td>
<td>24</td>
<td>Walk</td>
<td>180</td>
</tr>
</tbody>
</table>

Figure 7.1: “Forward jump,” “jump,” and “walk up the stairs” actions. The curves in each action show the trajectories of the right hand and the right foot.

7.1 Results from Motion Capture

The CMU database contains a broad range of motions, some with variations, which makes it ideal for this study. Forward jump, vertical jump, march, run, walk, and walk up the stairs were chosen because multiple instances of the same action were available. Table 7.1 shows the instances of each action type in the database. Five instances of an action were used to train the system and the remaining instances were used to evaluate the recognition ability of the system for each action. Figures 7.1-7.4 are screen shots of the visualization software, showing the path of the right foot and the right hand in an action.
Figure 7.2: The “march” action. The curves show the trajectories of the right hand and the right foot.

Figure 7.3: The “walk” action. The curves show the trajectories of the right hand and the right foot.
Figure 7.4: The “run” action. The curves show the trajectories of the right hand and the right foot.

The curve fitting and segmentation step is shown in Figure 7.5 for the walk action. First, a curve is fitted to the data. Then, peak curvature points are analyzed and the curve is segmented into basic motions. Notice that several peaks in the curvature may exist where the foot is nearly stationary. We handle this by analyzing the velocity of the curve around each peak in curvature. The curve is segmented only after the speed of the motion exceeds a threshold. The final points of segmentation are labeled “seg point” in the Figure. The speed and velocity of the motion is shown in Figure 7.6. The curve labeled “fit speed” is the speed calculated from the fitted curve model.

Once the curve is segmented into one or more basic motions, a feature vector is generated for each basic motion. The feature vector containing positional data for each basic motion in Figure 7.5 is shown in Figures 7.7 and 7.8. The positional data used in the feature vector is translated so that the basic motion begins at the origin and is then mean centered in the y-component. It is then rotated around the vertical axis (y-axis) so that the motion begins and ends on the z-axis. These transformations align all of the feature vectors along a common axis so that they can be compared. The data in the two feature vectors appear to be very similar, which is promising for PCA recognition.
Figure 7.5: Curve fitting and segmentation for the “walk” action. Two basic motions are identified using peak curvature. Notice that multiple curvature peaks occur when the foot is nearly stationary. This is due to small inaccuracies in tracking. The speed of the motion is used to locate where the foot starts and stops moving so that the motion curve is accurately segmented into foot steps. See Figure 7.6 for a plot showing the speed and velocity of this motion. The motion is segmented at the points labeled “seg point.”

Table 7.2: Index of figures generated using the CMU motion capture dataset.

<table>
<thead>
<tr>
<th>Action</th>
<th>Track</th>
<th>Curve Fit</th>
<th>Speed and Velocity</th>
<th>Basic Motion</th>
<th>Positional Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walk</td>
<td>7.3</td>
<td>7.5</td>
<td>7.6</td>
<td>7.7, 7.8</td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td>7.4</td>
<td>7.9</td>
<td>7.10</td>
<td>7.11</td>
<td></td>
</tr>
<tr>
<td>Jump</td>
<td>7.1</td>
<td>7.12</td>
<td>7.13</td>
<td>7.14, 7.15</td>
<td></td>
</tr>
<tr>
<td>Forward jump</td>
<td>7.1</td>
<td>7.16</td>
<td>7.17</td>
<td>7.18</td>
<td></td>
</tr>
<tr>
<td>Stairs</td>
<td>7.1</td>
<td>7.19</td>
<td>7.20</td>
<td>7.21</td>
<td></td>
</tr>
<tr>
<td>March</td>
<td>7.2</td>
<td>7.22</td>
<td>7.23</td>
<td>7.24</td>
<td></td>
</tr>
</tbody>
</table>
Figure 7.6: Speed and velocity of the “walk” action. The speed of the motion is used to find the segmentation points after identifying peaks in curvature. The curve labeled “speed” is the speed calculated from the trajectory data. The speed calculated from the fitted curve is labelled “fit speed.” Notice that the “fit speed” curve is a smooth representation of the speed calculated from the trajectory data. This shows that our fitted curve accurately models the motion over time. The velocity of the motion calculated from the trajectory data is labelled “dx,” “dy,” and “dz” for the x, y, and z velocity components, respectively. The curves labelled “x vel,” “y vel,” and “z vel” are the velocity curves calculated using the fitted curve.
Figure 7.7: Positional data for the first “walk” basic motion from Figure 7.5. This data is used to create a feature vector for training or recognition.
Figure 7.8: Positional data for the second “walk” basic motion from Figure 7.5. Notice that this data appears to be very similar to the first basic motion data shown in Figure 7.7. This is promising for PCA recognition.
Figure 7.9: Curve fitting and segmentation for the “run” action. One complete basic motion is identified by the curve segmentation process. The partially complete basic motion is discarded.

The curve fitting and segmentation example for the “run” action is shown in Figure 7.9. One complete running step basic motion appears in this example. The partial running step on the right side of the plot is discarded automatically by our system because a peak in curvature to signify that the basic motion is complete is missing in the trajectory. The speed and velocity for the “run” example is shown in Figure 7.10. Similar to the “walk” example, the speed calculated from the fitted curve, labelled “fit speed,” is a smooth approximation of the speed calculated using the trajectory data. Finally, the feature vector is generated from the positional data for the observed basic motion. This is shown in Figure 7.11. As expected, the majority of the motion is along the $z$-axis. The foot is lifted and lowered in the $y$-direction. A smaller motion is observed in the $x$-axis.
Figure 7.10: Speed and velocity of the “run” action. One complete running step can be identified using speed in this example. Notice that the foot is not stationary for very long compared to the “walk” action. Like in the walk example, the fitted curve provides a smooth approximation of the speed calculated using the trajectory data.
Figure 7.11: Positional data for the “run” basic motion from Figure 7.9. The foot is lifted high in the first half of the running foot step and is put back on the ground at a slightly slower rate near the end of the motion.
The curve fitting and segmentation of the “vertical jump” action is shown in Figure 7.12. One complete jump motion appears in this example. Notice that multiple spikes in curvature appear when the foot is stationary. Like in the “walk” action, this is handled by using the speed of the motion to find the final segmentation points. The speed and velocity of the motion is shown in Figure 7.13. The motion curve is also segmented at the top of the jump motion where the motion transitions from the upward to the downward direction. This creates two basic motions: one upward and one downward. The positional data that forms the feature vector for the upward basic motion is shown in Figure 7.14. As expected, the majority of the motion is in the upward direction (in the positive y-direction) although there is a smaller z-component to the motion. The positional data forming the downward basic motion is shown in Figure 7.15. Again, we see that the motion is mostly along the vertical axis, but this time in the downward direction.

Next, we examine the “forward jump” action. The curve fitting and segmentation example is shown in Figure 7.16. There are four consecutive forward jump motions in this example. All four motions are identified by our system because each has a curvature spike where the motion begins and ends. The speed and velocity curves for the motion are shown in Figure 7.17. Each jump can be clearly seen in the speed plot. The positional data for one of the basic motions is shown in Figure 7.18. The action is mostly down the z-axis, but forms an arch in the z-y plane. Unlike the vertical jump, the forward jump curve is not segmented at the apex because the motion does not form a sharp transition from the upward to downward motion.

The next action we examine is the “walking up stairs” action. Figure 7.19 shows the curve fitting and segmentation of an instance of this action. In this example, two basic motions are observed because two complete steps are taken. Like the walk and jump ac-
Figure 7.12: Curve fitting and segmentation for the “jump” action. Since the jump action is a vertical direction, high curvature is found at the start, apex of the jump, and end of the jump. This causes the jump to be segmented into an upwards basic motion and a downwards basic motion.
Figure 7.13: Speed and velocity of the “jump” action. The dip in speed in the middle of the motion (around sample 220) is where the apex of the jump occurs. Notice that using speed alone to segment this motion may not work reliably because the observed speed is not zero at the apex. This is because the sample rate is not high enough to observe where the jump is perfectly at the apex. We easily detect the transition using curvature as shown in Figure 7.12.
Figure 7.14: Positional data for the “jump” upwards basic motion from Figure 7.12.
Figure 7.15: Positional data for the “jump” downwards basic motion from Figure 7.12.
Figure 7.16: Curve fitting and segmentation for the “forward jump” action. The segmentation process correctly identifies four distinct forward jump basic motions in this example.
Figure 7.17: Speed and velocity of the “forward jump” action. Four distinct forward jump motions can be identified using the speed curve.
Figure 7.18: Positional data for one of the “forward jump” basic motions from Figure 7.16.
tions, multiple curvature spikes appear where the foot is stationary. We use the speed of the motion, shown in Figure 7.20, to aid in segmentation of the motion. The two foot steps are easy to spot in the speed plot. The positional data that forms the feature vector for one of the basic motions is shown in Figure 7.21. As expected, the position of the foot starts lower and ends higher along the $y$-axis because the foot is moved up to the next step in the staircase.

The final action in the motion capture dataset is the “march” action. The curve fitting and segmentation of a “march” action example is shown in Figure 7.22. Two complete marching foot steps are observed. The incomplete step is discarded by the system because there is no curvature spike to signify its starting point. The speed and velocity of the motion is shown in Figure 7.23. Interestingly, some sharp changes in speed are observed in the speed calculated from the trajectory data. This may indicate that the motion capture system was not as accurate when capturing this motion when compared to the other actions. The positional data for one of the “march” basic motions is shown in Figure 7.24. In this action, the foot is lifted, held in the air as the body leans forward, and then placed back on the ground for each step. This creates the distinct shape shown in the basic motion positional data plot.

### 7.1.1 Action Recognition Using 3-D Data

A number of experiments were carried out using 3-D track data with the goal of evaluating our recognition approach and examining the effectiveness of different feature vector types. The feature vectors tested consisted of the position alone, and the position and velocity for one foot, both feet, one foot and one hand, and both feet and both hands. In all cases the trajectory from a single foot was used to segment all motion curves that were captured
Figure 7.19: Curve fitting and segmentation for the “walking up stairs” action. Unlike the previous examples, this action was performed along the negative $z$-axis (from right to left in the plot). It is rotated along the positive $z$-axis before creating the feature vector positional data shown in Figure 7.21. The segmentation process correctly identifies two distinct “walking up stairs” basic motions in this example.
Figure 7.20: Speed and velocity of the “walk up stairs” action. Two foot steps can be observed using the speed curve. The fitted curve provides a smooth representation of the speed and velocity curves (shown in red) in this example.
Figure 7.21: Positional data for one of the “walk up stairs” basic motions from Figure 7.19. The basic motion is rotated so that the horizontal motion is along the positive $z$-axis.
Figure 7.22: Curve fitting and segmentation for the “marching” action. The segmentation process correctly identifies two complete basic motions in this example. This example was performed in the direction of the negative \( z \)-axis.
Figure 7.23: Speed and velocity of the “marching” action. Sharp changes in speed are observed in the speed calculated from the trajectory data. This may indicate that the motion capture system was not as accurate when capturing this motion compared to the other actions.
Figure 7.24: Positional data for one of the “march” basic motions from Figure 7.22. A distinct shape is formed by the motion made by the foot. First it is lifted high in the air, then it is kept in the air as the body leans forward, and finally it is placed back on the ground.
simultaneously.

To create a feature vector, the start point of the corresponding basic motion curve was translated to the origin, and the curve was rotated so that the start and end points of the curve align with the $z$-axis. This step generates the same motion vector when the same 3-D motion is observed from different views.

A training data set containing 5 feature vectors for each action type was created and used to construct the eigenspace. The training data set was chosen to contain representative samples of each action, but no effort was put into finding the optimal feature vectors for training.

Table 7.3 contains the recognition rate of our system for different feature vector types. The recognition rate is calculated as:

$$rate = \frac{c}{t} \times 100,$$

where $c$ is the number of basic motions that were classified correctly according to their labelled action class and $t$ is the total number of basic motions in the action class. For example, a “run” basic motion is classified correctly if the nearest neighbors of its projection are “run” training data projections.

The best results were obtained when data from both feet were used in feature vector generation. The data from both feet can be used to separate actions where the feet are moving together from actions where the feet are moving separately. This was important for classifying “forward jump” and “walk up the stairs.” The “forward jump” action contains considerable variation between trials. It includes short jumps, long jumps, and various degrees of hand movements.

Adding the motion of the hands to the feature vectors reduced recognition power compared to when hands were not included in “forward jump” and “vertical jump” actions. Hands add more variation between trials of the jump actions. Only in the “walk up the
Table 7.3: This table shows the recognition rate of different feature vector types using 3-D data. The feature vector types are abbreviated as follows: $P = \text{feature vectors representing } x, y, z \text{ position samples; } PV = \text{feature vectors representing } x, y, z \text{ position and } x', y', z' \text{ velocity samples; } 1f = 1 \text{ foot, } 1f1h = 1 \text{ foot and } 1 \text{ hand, } 2f = 2 \text{ feet, } 2f2h = 2 \text{ feet and } 2 \text{ hands.}$ The best results were obtained when data from both feet were used in feature vector generation. Adding the motion of the hands to the feature vectors reduced recognition power compared to when hands were not included in “forward jump” and “vertical jump” actions for most test cases.

<table>
<thead>
<tr>
<th>Action</th>
<th>$P \ 1f$</th>
<th>$P \ 1f1h$</th>
<th>$P \ 2f$</th>
<th>$P \ 2f2h$</th>
<th>$PV \ 1f$</th>
<th>$PV \ 1f1h$</th>
<th>$PV \ 2f$</th>
<th>$PV \ 2f2h$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward jump</td>
<td>65</td>
<td>51</td>
<td>98</td>
<td>100</td>
<td>66</td>
<td>48</td>
<td>98</td>
<td>100</td>
</tr>
<tr>
<td>Vertical jump</td>
<td>100</td>
<td>67</td>
<td>100</td>
<td>61</td>
<td>100</td>
<td>67</td>
<td>100</td>
<td>61</td>
</tr>
<tr>
<td>March</td>
<td>100</td>
<td>98</td>
<td>100</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>Run</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Walk up stairs</td>
<td>79</td>
<td>100</td>
<td>100</td>
<td>79</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Walk</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Avg. accuracy</td>
<td>91</td>
<td>86</td>
<td>100</td>
<td>93</td>
<td>90</td>
<td>86</td>
<td>99</td>
<td>93</td>
</tr>
</tbody>
</table>

7.1.2 Action Recognition Using Simulated 2-D Data

Use of 3-D data is possible when an advanced tracking system with a stereo camera setup is available. A general tracking system, however, generates 2-D data that is view-dependent. In order to evaluate our system with 2-D data, we designed an experiment that simulated 2-D camera-based tracking. Having 3-D trajectory data, we first rotate the trajectory with respect to a virtual camera to simulate different viewing angles. Next, we use perspective projection to project the trajectory onto the camera image plane. Except for positional information being 2-D, the remaining steps in our recognition system are similar to those for 3-D data.

Viewing angles were established by first aligning the trajectory parallel to the image plane. We refer to this as the 0-degree view. Then, other views were created by rotating the trajectory by 10-degree increments to the range from -60 to 60 degrees. Figure 7.25 shows
Table 7.4: Accuracy of classifying 2-D feature vectors composed of \((x, y)\) positions at various viewing angles when using only training data from the 0-degree view. The recognition rate of "vertical jump" doesn’t depend on viewing angle. This is because the motion is in the vertical direction. In contrast, the recognition rates of “run” and “walk” actions are greatly affected by viewing angle.

<table>
<thead>
<tr>
<th>Action</th>
<th>-60</th>
<th>-50</th>
<th>-40</th>
<th>-30</th>
<th>-20</th>
<th>-10</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward jump</td>
<td>81</td>
<td>88</td>
<td>94</td>
<td>100</td>
<td>100</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>92</td>
<td>86</td>
</tr>
<tr>
<td>Vertical jump</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>March</td>
<td>82</td>
<td>91</td>
<td>93</td>
<td>95</td>
<td>100</td>
<td>100</td>
<td>94</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>96</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Run</td>
<td>0</td>
<td>9</td>
<td>61</td>
<td>93</td>
<td>94</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>61</td>
<td>7</td>
</tr>
<tr>
<td>Walk up stairs</td>
<td>40</td>
<td>79</td>
<td>84</td>
<td>85</td>
<td>92</td>
<td>92</td>
<td>91</td>
<td>84</td>
<td>84</td>
<td>84</td>
<td>69</td>
<td>67</td>
<td>38</td>
</tr>
<tr>
<td>Walk</td>
<td>8</td>
<td>41</td>
<td>77</td>
<td>95</td>
<td>100</td>
<td>100</td>
<td>99</td>
<td>97</td>
<td>96</td>
<td>97</td>
<td>95</td>
<td>81</td>
<td>64</td>
</tr>
</tbody>
</table>

Table 7.5: Accuracy of classifying 2-D feature vectors composed of \((x, y)\) positions captured from various viewing angles when using training data from view directions -60, -40, -20, 0, 20, 40, and 60 degrees. These results show that PCA is effective at learning actions observed from multiple viewing angles and can be used to create a view-invariant action recognition system.

<table>
<thead>
<tr>
<th>Action</th>
<th>-60</th>
<th>-50</th>
<th>-40</th>
<th>-30</th>
<th>-20</th>
<th>-10</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward jump</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>Vertical jump</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>March</td>
<td>96</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>100</td>
<td>100</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>98</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Run</td>
<td>96</td>
<td>97</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Walk up stairs</td>
<td>90</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>88</td>
</tr>
<tr>
<td>Walk</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>85</td>
</tr>
<tr>
<td>Avg. accuracy</td>
<td>97</td>
<td>99</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>100</td>
<td>95</td>
</tr>
</tbody>
</table>

77
Figure 7.25: The result of the camera simulation on a “walk” basic motion at view directions -60, -40, -20, 0, 20, 40, and 60 degrees. Each feature vector is mean centered in the y-direction as a part of the feature vector generation.

Since our system performed well with feature vectors formed using the position of both feet in the 3-D experiments, feature vectors containing the positions of both feet were used in the 2-D experiments also.

In the first set of 2-D experiments, only data from the 0-degree view angle were used to train the system. Then the system was used to recognize various actions when observed from a wide range of views. The recognition results are summarized in Table 7.4. As can be observed, the recognition rate of “vertical jump” doesn’t depend on viewing angle. In contrast, the recognition rates of “run” and “walk” actions are greatly affected by viewing
angle. As the view angle of a motion changes, the trajectory becomes deformed by rotation and perspective projection when compared to the original 0-degree view trajectory. This can cause the basic motion to be misclassified because the feature vector projection may be closer to training vector projections belonging to other motions in the eigenspace.

In another experiment, we investigated the possibility of learning various view angles using PCA. Five feature vectors from view angles -60, -40, -20, 0, 20, 40, and 60 degrees were selected for each action and included in the training data set. The results from this experiment are shown in Table 7.5.

Note that almost perfect results are obtained for all viewing angles, including angles between the training views such as -50, -30, -10, etc. This is important because it implies that training is not required using data from every possible angle.

Classification rates for “walk” and “walk up the stairs” decreases rather sharply at 50- and 60-degree views. This is because the trajectory for “walk,” after being rotated 60 degrees, resembles the curve for “walk up the stairs” at 0 degrees. Depth information is lost due to projection to 2-D. This type of problem can be alleviated by considering the features of the environment (location of stairs) or the orientation of the person to add some depth information into the process.

### 7.2 Results from Weizmann Video Dataset

To evaluate our action recognition system using videos, we use the Weizmann Human Action dataset [53]. 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 4.3.2, our tracking system was unable to track the feet for the “skip” action so results for the skip video are not included.
After training the background subtraction system for 60 frames by looping through the background training videos in the Weizmann dataset, we observed that some 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 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.

7.2.1 Tracking Results

The first step in recognizing actions in videos is tracking the key points to form trajectories. We track the feet throughout the motion and track the hands when the body is stationary. Figures 7.26-7.35 illustrate tracking of the feet and hands (where applicable) for each action type in the dataset.

After tracking the feet and hands, curves are fitted to the trajectories and segmented into basic motions. This process is the same as the curve fitting and segmentation process used for previous experiments. Figures 7.36-7.56 illustrate this process for each action in the Weizmann dataset. See Table 7.6 for an index of Figures. Notice that the trajectory data
Figure 7.26: Tracking for the “jack” action.

Figure 7.27: Tracking for the “jump” action.
Figure 7.28: Tracking for the “skip” action. The skip action is not included in our final results because each foot is overlapping with the other foot or the leg throughout much of the video. This illustrates a weakness in our tracking system.

Figure 7.29: Tracking for the “run” action.
Figure 7.30: Tracking the hands and feet for the “wave1” action.

Figure 7.31: Tracking the hands and feet for the “wave2” action.
Figure 7.32: Tracking for the “bend” action. Notice that the feet tracking algorithm detects the hand as a foot once it enters the foot region. This action is detected as “bend” by the bend over detection algorithm, so the results from the feet tracking are ignored by the recognition system.
Figure 7.33: Tracking for the “walk” action.

Figure 7.34: Tracking for the “side” action.
Figure 7.35: Tracking for the “pjump” (vertical jumping) action. The feet may occlude each other in some frames, but the tracking system reacquires the location of each foot by either finding separation of the legs in a later frame or by assuming the feet are together.
Table 7.6: Index of figures generated using the Weizmann action dataset.

<table>
<thead>
<tr>
<th>Action</th>
<th>Track</th>
<th>Curve Fit</th>
<th>Speed and Velocity</th>
<th>Basic Motion</th>
<th>Positional Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>7.29</td>
<td>7.36</td>
<td>7.37</td>
<td>7.38</td>
<td></td>
</tr>
<tr>
<td>walk</td>
<td>7.33</td>
<td>7.39</td>
<td>7.40</td>
<td>7.41</td>
<td></td>
</tr>
<tr>
<td>side</td>
<td>7.34</td>
<td>7.42</td>
<td>7.43</td>
<td>7.44</td>
<td></td>
</tr>
<tr>
<td>jack</td>
<td>7.26</td>
<td>7.45</td>
<td>7.46</td>
<td>7.47</td>
<td></td>
</tr>
<tr>
<td>jump</td>
<td>7.27</td>
<td>7.48</td>
<td>7.49</td>
<td>7.50</td>
<td></td>
</tr>
<tr>
<td>pjump</td>
<td>7.35</td>
<td>7.51</td>
<td>7.52</td>
<td>7.53</td>
<td></td>
</tr>
<tr>
<td>wave</td>
<td>7.30, 7.31</td>
<td>7.54</td>
<td>7.55</td>
<td>7.56</td>
<td></td>
</tr>
</tbody>
</table>

is sparse compared to the trajectories taken from the motion capture dataset. Curve fitting smooths the data and approximates missing data when the feet are overlapping.

The curve fitting and segmentation for the “run” action is shown in Figure 7.36. The tracking algorithm has produced a fairly accurate trajectory, but the fitted curve removes noise and creates a continuous representation of the motion. In this example, one complete running step is delineated by two peaks in curvature. The speed and velocity curves are shown in Figure 7.37. The speed calculated from the trajectory data is given by the curve labelled “speed” and the speed calculated from the fitted curve is labelled “fitted speed.” The speed calculated from the trajectory is noisy compared to the smooth approximation provided by the fitted curve. The positional data for the observed basic motion is given in Figure 7.38.

The curve fitting and segmentation of the “walk” action appears in Figure 7.39. The trajectory is noticeably noisier than the “run” action. This is because the speed of the motion is slower and the overall size of the motion is smaller making the inaccuracy more obvious. Again, the fitted curve provides a smooth representation of the trajectory and approximates the gaps in the data where the feet were overlapping. Two complete foot steps are identified using segmentation. The speed, shown in Figure 7.40 is used to find the
Figure 7.36: Curve fitting and segmentation for the “run” action from the Weizmann video dataset. One complete running step is observed. The fitted curve provides a continuous smooth representation of the trajectory. It also approximates the missing data where the tracker could not locate the feet because they were overlapping.
Figure 7.37: Speed and velocity of the “run” action from the Weizmann video dataset. The speed calculated from the fitted curve, labelled “fit speed,” is much smoother than the speed calculated from the trajectory data. This is because the trajectory is sparse and noisy.
final segmentation points. The positional data for one of the basic motions is provided in Figure 7.41.

The “side” action is a side stepping action that is performed facing the camera, so there are no gaps in the data because the feet do not overlap. Figure 7.42 illustrates the curve fitting and segmentation of this action. Two complete side steps are detected in this trajectory. The speed of the motion is shown in Figure 7.43. The y-velocity does not appear to be smooth in the trajectory data, but the fitted curve smooths it nicely. Finally, one of the basic motions is provided in Figure 7.44. The foot is lifted higher in the y-direction at the beginning of the motion and is gradually lowered to the ground. This is consistent with what appears in the videos.

In Figure 7.45, the curve fitting and segmentation of the “jack” action is shown. In
Figure 7.39: Curve fitting and segmentation for the “walk” action from the Weizmann video dataset. Again, the fitted curve is a smooth curve that approximates the motion made by the foot. Two complete foot steps are detected by the segmentation process.
Figure 7.40: Speed and velocity of the “walk” action from the Weizmann video dataset. The speed of the motion is used to find the segmentation points after peaks in curvature are identified.
the jack action, the foot moves back and forth in an arcing motion. A peak in curvature is formed every time the foot changes direction. In this example, five basic motions are identified using the peak curvature values. The fitted curve doesn’t appear to closely match the trajectory, but the overall motion is captured. A tighter curve fit would cause the curve to fit more closely at the cost of smoothness. The speed of the motion, shown in Figure 7.46, indicates that two partial jack motions appear in the trajectory: one at the beginning and one at the end of the sequence. The partial motions are discarded automatically because a curvature peak is not included in the trajectory that marks the beginning of the first motion and end of the last motion. The positional data for one of the jack basic motions is shown in Figure 7.47.

The next action in the Weizmann dataset that we examine is the “jump” action. This is a forward jumping action. The curve fitting and segmentation example is given in Fig-
Figure 7.42: Curve fitting and segmentation for the “side” action from the Weizmann video dataset.
Figure 7.43: Speed and velocity of the “side” action from the Weizmann video dataset. Two side step motions can be identified using the speed curve.
Figure 7.44: Positional data for one of the “side” basic motions. In this motion, the foot is lifted higher in the beginning of the motion and then is gradually placed on the ground.

Figure 7.48. Five complete jump actions are detected in this trajectory using the curvature peaks. The five jumps can be seen in the speed of the action, shown in Figure 7.49. The positional data for one of the jump basic motions is provided in Figure 7.50.

The “pjump” action in the Weizmann dataset is a vertical jumping action. The curve fitting and segmentation of this action is shown in Figure 7.51. The motion curve is segmented at the bottom and top of each jump using curvature peaks and speed. Five complete basic motions are identified here. We also see that there were two partial motions using the speed plot shown in Figure 7.52. Again, the paritial motions are discarded automatically by the curve fitting and segmentation process. As expected, the positional data of the basic motion shown in Figure 7.53 illustrates that the motion is mostly in the vertical (y) direction with only small movement in the horizontal axis.
Figure 7.45: Curve fitting and segmentation for the “jack” action from the Weizmann video dataset. The curve is segmented every time the foot changes direction. Five complete “jack” basic motions are created using this curve.
Figure 7.46: Speed and velocity of the “jack” action from the Weizmann video dataset. The up and down motion of the leg can be seen in the $y$-velocity plot.
Figure 7.47: Positional data for one of the “jack” basic motions.
Figure 7.48: Curve fitting and segmentation for the “jump” action from the Weizmann video dataset. This is a forward jumping motion. Five complete jumping motions are identified by the segmentation process.
Figure 7.49: Speed and velocity of the “jump” action from the Weizmann video dataset.
Figure 7.50: Positional data for one of the “jump” basic motions.
Figure 7.51: Curve fitting and segmentation for the “pjump” action from the Weizmann video dataset. The “pjump” action is a vertical jumping action. Each jump is segmented into a basic motion for the upwards part of the motion and a basic motion for the downwards part of the motion.
Figure 7.52: Speed and velocity of the “pjump” action from the Weizmann video dataset. Five complete “pjump” basic motions can be identified using the speed. Using the $y$-velocity curve, we can tell which basic motion is an upwards motion and which are downwards motions depending on whether the velocity is positive (upwards) or negative (downwards).
The final action in the Weizmann dataset that we examine is the “wave” action. The curve fitting and segmentation of the wave action is given in Figure 7.54. In the wave action, the hand is waved back and forth. Each time the hand changes direction, a new basic motion is formed. This creates the upwards wave and a downwards wave basic motion types. In this example, two complete basic motions are identified. There are two waving motions that are not identified in this trajectory because no peak curvature points appear. Unlike the motion of the feet, it’s difficult to tell if these are complete motions or only partial motions. The positional data for an upwards waving basic motion is shown in Figure 7.56.
Figure 7.54: Curve fitting and segmentation for the “wave” action from the Weizmann video dataset. The motion curve is segmented each time the hand changes direction creating upwards and downwards waving basic motions. Two complete waving motions are detected. The first and last waving motions are not detected as complete motions because there are no spikes in curvature at the beginning and end of the motion curve.
Figure 7.55: Speed and velocity of the “wave” action from the Weizmann video dataset. A smoother curve fit than what is shown here may be desirable, but the overall motion is captured.
Figure 7.56: Positional data for one of the “wave” basic motions. This is an upwards waving motion.
7.2.2 Classifying Basic Motions and Videos

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 bend-over action is classified by detecting bend-over events as described in Section 4.3.4.

The training dataset used in our experiment consisted of basic motions from the videos of one of the human subjects with the exception of the 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 7.7 is a confusion matrix showing the accuracy of our system in recognizing basic motion types. The accuracy is calculated as:

\[
\text{rate} = \frac{c}{t},
\]

where \(c\) is the number of basic motions that were classified correctly according to their labelled action class (taken from the label of the video) and \(t\) is the total number of basic motions in the action class.

Using the most commonly observed basic motion type to classify each video, we obtain 100% recognition accuracy for the videos in the data set. That is, every video in the dataset is classified as the labelled type. We compare this result to other approaches in the next section.
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>pjump</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></td>
<td></td>
</tr>
<tr>
<td>jump</td>
<td></td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pjump</td>
<td></td>
<td></td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>run</td>
<td></td>
<td></td>
<td></td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>side</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.0</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>1.0</td>
<td></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>

### 7.3 Comparison to Other Approaches

The recognition accuracy of our approach for classifying the videos in the Weizmann dataset is compared to other approaches in Table 7.8. The accuracy of our approach compares favorably to other state-of-the-art approaches. Like our approach, other approaches have difficulty with the “skip” action and do not include it in their results. This is surprising, because all but one of the approaches do not use tracking. Instead, they detect features in the video using various feature detectors and classify actions based on the patterns of features that are detected. Our approach is the only approach that is able to perform tracking automatically for the Weizmann dataset. The approach introduced by Ali et al. [3] tracks trajectories from the hands, feet, head, and body center, but they labelled the points by hand when their tracker failed to detect these points correctly.
Table 7.8: The recognition accuracy of our approach is compared to other approaches that were evaluated using the Weizmann dataset.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Year</th>
<th>No. Actions</th>
<th>Accuracy (%)</th>
<th>Tracking-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed approach</td>
<td>2012</td>
<td>9 (no skip)</td>
<td>100</td>
<td>*</td>
</tr>
<tr>
<td>Sun et al. [46]</td>
<td>2011</td>
<td>10</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Yeffet and Wolf [61]</td>
<td>2009</td>
<td>9 (no skip)</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Wang and Mori [57]</td>
<td>2009</td>
<td>9 (no skip)</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>Guha and Ward [19]</td>
<td>2011</td>
<td>10</td>
<td>98.9</td>
<td></td>
</tr>
<tr>
<td>Thi et al. [48]</td>
<td>2012</td>
<td>10</td>
<td>98.2</td>
<td></td>
</tr>
<tr>
<td>Gorelick et al. [17]</td>
<td>2007</td>
<td>10</td>
<td>97.8</td>
<td></td>
</tr>
<tr>
<td>Wang and Mori [58]</td>
<td>2011</td>
<td>9 (no skip)</td>
<td>97.2</td>
<td></td>
</tr>
<tr>
<td>Riemenschneider and Bischof [39]</td>
<td>2009</td>
<td>10</td>
<td>96.7</td>
<td></td>
</tr>
<tr>
<td>Bregonzio et al. [7]</td>
<td>2009</td>
<td>10</td>
<td>96.7</td>
<td></td>
</tr>
<tr>
<td>Junejo et al. [24]</td>
<td>2010</td>
<td>9 (no wave2)</td>
<td>95.3</td>
<td></td>
</tr>
<tr>
<td>Thurau and Hlaváč [49]</td>
<td>2008</td>
<td>10</td>
<td>94.4</td>
<td></td>
</tr>
<tr>
<td>Ali et al. [3]</td>
<td>2007</td>
<td>9 (no skip)</td>
<td>92.6</td>
<td>*</td>
</tr>
<tr>
<td>Niebles et al. [35]</td>
<td>2008</td>
<td>10</td>
<td>90.0</td>
<td></td>
</tr>
</tbody>
</table>

7.4 Execution Speed

One of the benefits of our approach is that it is efficient when compared to other approaches. Our approach is implemented in three main programs:

1. tracker - takes a video and outputs trajectory data to files.

2. cfitv - takes trajectories and performs curve fitting, segmentation, and outputs feature vectors to files.

3. recog - computes the eigenspace of the training data and classifies a feature vector.

Table 7.9 summarizes the execution time needed for each component. The computer used to measure the speed of our approach was a MacBook Pro with a dual core 2.66 GHz Intel Core i7.

The average length of each Weizmann video is 2.41 seconds and the videos have a frame rate of 25 frames per second. Our tracker runs with an average execution time of
2.76 seconds per video, with a frame rate of 22 frames per second. This means that our tracker is executing at near real-time speed.

The cfitv program, perhaps surprisingly, is the slowest component in the system. It is implemented in python and outputs each feature vector to a file. The majority of the execution time is in the curve fitting process. Execution time can be reduced by implementing the program in C and changing the curve model to a more efficient implementation. The CPU load on the system is 66% idle when the cfitv program is running. Since there are two CPUs on the system and there is only one thread of execution in our implementation, this indicates that this process is roughly 16% bound by file I/O. A multithreaded approach would also increase execution speed.

The recognizer program, recog, is launched for each basic motion feature vector. It computes the eigenspace each time it initializes, but it is still able to achieve an average execution time of 8.81 seconds per video. A more efficient implementation would compute the eigenspace only once because it does not change between iterations.

While the execution speed of our system is not real-time, an implementation that focused on execution speed, rather than testing flexibility, would perform much closer to real-time. These are encouraging results, but also a topic for future work.

Very few papers include execution speed in their results. To our knowledge, no paper using a tracking based method has reported execution speed results. Yu et al. [62] focus on real-time recognition and report execution speeds between 10 and 20 frames per second. They do not test their method using the Weizmann dataset, making it difficult to compare their results to ours.
Table 7.9: Execution speed of each component. The videos in the Weizmann dataset are short with an average length of 2.41 seconds. Feet and hand tracking is done by the tracker component. The tracker executes at a near real-time average speed of 2.76 seconds per video or 22 frames per second. The cfitv component performs curve fitting, segmentation, and feature vector generation at a rate of 13.6 seconds per video. Recognition of basic motions is performed by the recog program. It runs for 8.81 seconds per video.

<table>
<thead>
<tr>
<th>Item</th>
<th>Average Time / Video (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>video</td>
<td>2.41 (25 FPS)</td>
</tr>
<tr>
<td>tracker</td>
<td>2.76 (22 FPS)</td>
</tr>
<tr>
<td>cfitv</td>
<td>13.6</td>
</tr>
<tr>
<td>recog</td>
<td>8.81</td>
</tr>
</tbody>
</table>
Chapter 8

Contributions

The primary contribution of this work to the field of computer vision is the design of a complete system that can learn various human activities and recognize them in a very efficient manner. In the development of this system, we offer a number of innovations.

1. We experimentally determined that it is sufficient to track only a few points on the human body in order to recognize various actions. The feet and hands are ideal points on the body to track over time because the motion curves that are formed from these points are different depending on the action performed. Our results show that tracking just the feet is enough to recognize actions where the primary motion is made by the feet. This was demonstrated using the walking, running, marching, vertical jumping, forward jumping, and walking up stairs actions using motion capture data. The video data set included jumping jacks, vertical jump, forward jump, run, walk, and side walk actions where only the feet were used for recognition. Similarly, our results show that tracking just the hands is enough to recognize actions involving the hands. Actions involving the hands include waving one arm and waving two arms in the video data set.

2. The use of curve fitting to smooth the trajectory data and to provide a continuous
curve over the entire motion is advantageous in that missing data, typical of a tracking system due to occlusion or low frame rate, will be approximated in such a way that a complete, natural motion is found. The curve model that results from fitting is also used to make segmentation of the curve into basic motions a simple and accurate process.

3. Recognition of basic motions, which are primitive motions that are logical for recognition of various actions, is another innovation of our system. Others have used sub-trajectories or primitive motions in recognition systems, but we are the first to realize that the shape of the basic motion can be used directly for recognition of an action. For example, the step a person makes while walking produces a curve with a certain shape in space. This shape can be used to recognize that the person is walking.

Recognition at the basic motion level has many advantages. First, the action performed by the feet and hands are decoupled. Information from the motion of the feet and hands can be used together to confirm the action being performed or they can be used alone to determine more complex action classes. Basic motion recognition allows a high-level recognizer to recognize more complex or composite actions by using the proposed system as a low-level recognizer.

4. We use the curve model to generate the basic motion feature vector. The curve model provides a continuous and smooth motion that is required to generate a feature vector of a fixed length, providing a repeatable representation of the motion.

5. Finally, PCA is used to reduce the dimensionality of the data and recognize actions by clustering in eigenspace. PCA is a commonly used tool in pattern recognition and human action recognition, but it has not been used to recognize human action. In our approach, PCA provides a compact representation for the training feature vectors and provides an efficient means to recognize actions.
Adapting to a New Data Set

When adapting our approach to work with a new data set, first, the tracker should be evaluated to ensure that accurate trajectories are produced. Variations in lighting conditions or subject body types may adversely affect the performance of the tracker in such a way that the tracker is unable to track the subject or produces incorrect locations for the parts of the body being tracked. The tracking approach may need to be enhanced when considering a new data set.

The motion analysis, training, and recognition steps are robust to changes in data sets. This was demonstrated by testing our approach using the motion capture and video data sets. The curve fitting process is sensitive to the size of the tracker’s coordinate system. The trajectories may be scaled to the appropriate size needed to achieve a good curve fit or the curve fitting parameters can be adjusted.

The composition of the training data should be considered when using a new data set. One of the benefits of our approach is that different variations of an action can be learned by including examples of each variation in the training data set. This was demonstrated by learning and recognizing actions at various viewing angles. Variations in the performance of an action may also occur due to terrain or other factors in the environment.

For some applications it is beneficial to recognize variation between actions in the
same action class. For example, the trajectories made by the feet of a running person may be altered if the person is carrying a heavy load. Depending on the accuracy of the tracker, it may be possible to distinguish between a person running with a heavy load and a person running without a load.

New actions or new variations of an action can be incorporated into an action recognition system by retraining the system over time. The training step may be performed repeatedly as new motions are added to the training data.
Chapter 10

Conclusions

Human action recognition is an important research topic due to its many applications. Research in this area will help shape the future of surveillance, human-machine interaction, and robot intelligence, to name a few.

We proposed a new action recognition framework that is able to learn various human activities 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, feature generation, and PCA classification.

The proposed approach begins by tracking key points on the body in the input data. We have shown that the motion of one or more key points like the feet can be used to recognize various actions. A curve is fitted to each trajectory sequence to produce a smooth, continuous motion curve. The resulting motion curve is then segmented into basic motions by finding points of peak curvature. An example of a typical basic motion is a walking step taken when a person is walking. In this case, the basic motion begins when the foot is lifted off the ground and ends when the foot is placed back on the ground. Next, a feature vector is generated for each basic motion. The feature vector contains a fixed number of dimensions and characterizes the basic motion so that the basic motion can be classified using PCA. We
begin the classification phase by projecting the feature vector into an eigenspace that was created using a training dataset. The basic motion is classified using the k-nearest neighbor classifier by computing the Euclidean distance between the basic motion’s feature vector projection and the training data feature vector projections.

Experimental results demonstrate that the proposed approach can recognize various human actions with a high degree of accuracy. This makes our approach an attractive solution and comparable in recognition power to other state-of-the-art approaches.

Two data sets were used to evaluate our approach: namely, the CMU motion capture data set and the Weizmann video data set.

Experimental results using the CMU motion capture data set demonstrate the following:

1. The performance of our recognition system depends on the data contained in the feature vectors. For example, recognition is improved if feature vectors containing positional data for both feet are used rather than feature vectors containing the positional data for only one foot. Furthermore, adding the motion of the hands when the feet are the primary actor in the action, such as in the “vertical jump” action, reduces recognition power due to variation in hand movement.

2. PCA can learn and later recognize actions from 3-D trajectories in cases where 3-D tracking is possible.

3. PCA can learn and later recognize actions from 2-D trajectories at various viewing angles. This is important for a recognition system that is invariant to viewpoint.

Experimental results using the Weizmann video data set demonstrate that our system is able to reduce each frame of video to a few key points by tracking the position of the feet and hands in each video frame. Furthermore, we demonstrated that our recognition system achieves a high degree of recognition accuracy even when low resolution videos, such as those provided in the Weizmann data set, are used.
Future Work

We have shown that the proposed approach is capable of recognizing actions with a sufficiently high accuracy, but our method could be improved with more robust tracking. Our tracking method is fast and accurate when the feet can be separated for enough frames to obtain a trajectory. However, the skip action in the Weizmann dataset revealed a weakness in our tracking approach because the feet are overlapping for much of the action. A solution to this problem might be to detect the pose of the person, and use the pose information to aid in locating the position of the feet.

We improved on the mixture of Gaussian method by incorporating shadow removal. Other improvements can be made in background subtraction. While we attempt to reduce the effect of errors in background subtraction during tracking, our tracking approach relies on fairly clean background subtraction results. Background subtraction fails to segment the person from the background when the color of the clothing and skin closely resembles the color of the background. The process might be improved by considering the position of the moving object in previous frames rather than only the per pixel color.

Another area of future work is in higher-level action and activity recognizers. We have shown that our approach can recognize basic actions, such as the foot steps of a walking person. This makes our approach an ideal tool for use in a higher-level recognition system.
that can recognize complex actions and activities where the action is detected by observing patterns in the basic motions that are detected.

A final area of future work is to improve the recognition speed of our approach. Real-time responses are important for many action recognition applications. Our implementation, so far, has not been optimized to generate real-time responses.
Bibliography


[52] Carnegie Mellon University. CMU graphics lab motion capture database, Dec 2002. 
http://mocap.cs.cmu.edu.


