HULL CONVEXITY DEFECT FEATURES FOR HUMAN ACTION RECOGNITION

Dissertation
Submitted to
The School of Engineering of the UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for
The Degree of
Doctor of Philosophy in Electrical Engineering

By
Menatoallah M. Youssef
Dayton, Ohio
August, 2011
HULL CONVEXITY DEFECT FEATURES
FOR HUMAN ACTION RECOGNITION

Name: Youssef, Menatoallah M.

APPROVED BY:

Vijayan K. Asari, Ph.D.
Committee Chair
Professor, Dept. Electrical & Computer Engineering & Ohio Research Scholars
Chair Wide-Area Surveillance

Keigo Hirakawa, Ph.D.
Committee Member
Assistant Professor
Dept. Electrical & Computer Engineering

Eric J. Balster, Ph.D.
Committee Member
Assistant Professor
Dept. Electrical & Computer Engineering

Donald J. Kessler, Ph.D.
Committee Member
Research Scientist
University of Dayton Research Institute

John G. Weber, Ph.D.
Associate Dean
School of Engineering

Tony E. Saliba, Ph.D.
Dean, School of Engineering
& Wilke Distinguished Professor
ABSTRACT

HULL CONVEXITY DEFECT FEATURES FOR HUMAN ACTION RECOGNITION

Name: Youssef, Menatoallah M.  
University of Dayton

Advisor: Dr. Vijayan Asari

Human action recognition is a rapidly developing field in computer vision. Accurate algorithmic modeling of action recognition must contend with a multitude of challenges. Machine vision and pattern recognition algorithms can be used to aid in the identification of these actions. In recent years research has focused on recognizing complex actions using simple features. Simple cases of action recognition, wherein one individual is captured performing a single action, form the foundation for developing more complex scenarios in real environments. This can be especially useful for surveillance of public locations such as subways, shopping centers, or parking lots in order to reduce crime, monitor traffic flow, and offer security in general. An effective action recognition algorithm must address the following challenges that affect feature extraction for accurate representation: non-rigidity, spatial-variance, temporal-variance, camera perspective. Where face detection seeks to identify the location of an individual’s face, activity recognition seeks to recognize the motion or action of an individual. There is generally a commonality of features in the true positive set with face recognition; certain rigid features are present on every human face. Action recognition, on the other hand, must deal with the non-rigidity of the
human body. The arms and legs can be at a number of positions relative to one another, and at varying distances and angles. These relative positions describe actions or intermediary poses.

We consider developing a taxonomic shape driven algorithm to solve the problem of human action recognition and develop a new feature extraction technique using hull convexity defects. To test and validate this approach, we use silhouettes of subjects performing ten actions from a commonly used video database by action recognition researchers. A morphological algorithm is used to filter noise from the silhouette. A convex hull is then created around the silhouette frame, from which convex defects will be used as the features for analysis. A complete feature consists of thirty individual values which represent the five largest convex hull defects areas. A consecutive sequence of these features form a complete action. Action frame sequences are pre-processed to separate the data into two sets based on perspective planes and bilateral symmetry. Features are then normalized to create a final set of action sequences. We then formulate and investigate three methods to classify ten actions from the database. Testing and training of the nine test subjects is performed using a leave one out methodology. Classification utilizes both PCA and minimally encoded neural networks. Performance evaluation results show that the Hull Convexity Defect Algorithm provides comparable results with less computational complexity. This research can lead to a real time performance application that can be incorporated to include distinguishing more complex actions and multiple person interaction.
DEDICATION

I would like to dedicate this dissertation to my parents, Hanaa and Mohamed Youssef and brothers Louay and Eyad, sister Hebbat Youssef, aunt Yumna Khalil and my family in Egypt.
ACKNOWLEDGEMENTS

I would like to thank Dr. Vijayan Asari; his support, guidance, and mentorship provided me with invaluable motivation and support.

I start with R. Cortland Tompkins (Figure 2.1c) and Jake Foytik (Figure 2.7a), who helped break down cubicle walls and build friendships. Working with you all has been a once in a lifetime experience which I hope can be repeated. Yakov Diskin (Figure 2a), your enthusiasm for all things Vision Lab is contagious. Praveen Sankaran, thanks for blazing the trail for the Vision Lab. Fatemah, Binu, and the rest of Vision Lab, your support and friendship meant a great deal to me.

I would also like to thank Linda Marshall, Hero Barker, and Romina Samson, originals from the start who always pushed me to get in, get done, and get out. Thanks to Dr. Zahorian for all your support. Thanks to friends foreign and domestic whose have always had faith in me.

Finally thanks, to my parents Hanaa and Mohamed Youssef, my siblings, Louay, Eyad, and Hebba, I could not have gone half as far in life without your love and support.
TABLE OF CONTENTS

ABSTRACT ................................................. iii
DEDICATION ................................................. v
ACKNOWLEDGMENTS ................................. vi
LIST OF ILLUSTRATIONS ......................... x
LIST OF TABLES ........................................... xii
LIST OF ALGORITHMS ............................... xiii
NOMENCLATURE ......................................... xiv
INTRODUCTION ........................................... 1

I  BACKGROUND .......................................... 9
1.1 Detection ............................................. 10
    1.1.1 Motion Segmentation ......................... 10
        1.1.1.1 Background Subtraction ............. 10
        1.1.1.2 Temporal Differencing ............. 11
        1.1.1.3 Optical Flow ......................... 11
    1.1.2 Object Classification ..................... 14
1.2 Tracking ............................................. 15

vii
1.2.1 Region Based Tracking ........................................ 15
1.2.2 Feature Based Tracking .................................... 16
1.3 Recognition ...................................................... 16
  1.3.1 Single Layered Approach .................................. 17
    1.3.1.1 Space time .......................................... 17
    1.3.1.2 Sequential Approach ................................ 21
  1.3.2 Hierarchical Approach .................................. 23
1.4 State Space Approaches ........................................ 26
  1.4.1 Principal Component Analysis ......................... 26
  1.4.2 Fisher Linear Discriminant - Linear Discriminant Analysis .. 29
1.5 Neural Networks ................................................ 31
  1.5.1 Feed Forward Network Functions ...................... 31
  1.5.2 Network Training ....................................... 33
    1.5.2.1 Backpropagation ................................... 33

II HULL CONVEXITY DEFECT FEATURES FOR HUMAN AC-
TION RECOGNITION .............................................. 39

2.1 Segmentation of Silhouettes ................................. 39
  2.1.1 Motion Detection ....................................... 42
  2.1.2 Silhouette Extraction .................................. 42

2.2 Feature Extraction ............................................ 44
  2.2.1 Convex Hull Creation - Sklansky’s Algorithm .......... 44
  2.2.2 Convex Hull Defect .................................... 48
  2.2.3 Zero Buffer/Normalization .............................. 50
  2.2.4 Triangle Registration .................................. 50
  2.2.5 Feature Vector Creation ............................... 53

viii
III ACTION VECTOR CLASSIFICATION .......................... 54
  3.1 Testing and Training Sets ................................. 54
  3.2 Full Activity Cluster ..................................... 55
  3.3 Separate Activity Cluster ................................. 56
    3.3.1 K-Nearest Neighbor Classifications ..................... 56
  3.4 Minimally Encoded Neural Networks for Action Recognition .... 57
    3.4.1 Bilateral Symmetry Metric BSM ......................... 57

IV EXPERIMENTAL RESULTS AND DISCUSSION ............... 62
  4.1 Weizman Database ......................................... 63
  4.2 Experimental Setup ....................................... 64
  4.3 Frame-by-Frame Analysis .................................. 64
    4.3.1 Base Results .......................................... 64
  4.4 Multiple Action Subspace ................................. 67
  4.5 Neural Networks for Human Activity Recognition ............ 69
    4.5.1 Neural Network Modalities ............................. 73
    4.5.2 Inclusion of Skip ...................................... 82
  4.6 Discussion ................................................. 82

V CONCLUSION AND FUTURE WORK ......................... 84

LIST OF PUBLICATIONS ................................. 86

BIBLIOGRAPHY ........................................... 88
# LIST OF ILLUSTRATIONS

1. Run and Skip sub-states ........................................... 3
2. Features for face detection and iris recognition ............... 4
3. Flowchart of action recognition process .......................... 8

1.1 Action recognition .................................................. 9
1.2 Optical flow diagram .............................................. 14
1.3 Features tracked in progressive frames [33] ..................... 16
1.4 Human action recognition chart ................................. 17
1.5 Space-Time volumes [6] ........................................... 18
1.6 MEI and MHI [8, 9] ................................................ 20
1.7 HMM ............................................................... 22
1.8 HMMs for Action Recognition .................................... 24
1.9 Data projected on a line in a PCA space ......................... 27
1.10 Neural Network Decision Boundaries ............................ 32
1.11 Neural Network .................................................... 34
1.12 Learning curve ................................................... 37
1.13 Sigmoid activation function ..................................... 37

2.1 Examples of Actions for Recognition ............................. 40
2.2 Gait cycle .......................................................... 42
2.3 Detection ........................................................... 42
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4 Background Subtraction</td>
<td>43</td>
</tr>
<tr>
<td>2.5 Generalized convex hull logic</td>
<td>45</td>
</tr>
<tr>
<td>2.6 Sklansky’s square</td>
<td>48</td>
</tr>
<tr>
<td>2.7 Convex hull defects</td>
<td>49</td>
</tr>
<tr>
<td>2.8 Triangle Registration</td>
<td>52</td>
</tr>
<tr>
<td>3.1 Training and testing sets</td>
<td>54</td>
</tr>
<tr>
<td>3.2 Full set of actions</td>
<td>55</td>
</tr>
<tr>
<td>3.3 K-nearest neighbor</td>
<td>57</td>
</tr>
<tr>
<td>3.4 Clusters</td>
<td>58</td>
</tr>
<tr>
<td>3.5 Bilateral symmetry of bend and jumping jack</td>
<td>59</td>
</tr>
<tr>
<td>4.1 Actions in subspace</td>
<td>68</td>
</tr>
<tr>
<td>4.2 Bilateral actions in PCA subspace</td>
<td>69</td>
</tr>
<tr>
<td>4.3 Bend projections</td>
<td>70</td>
</tr>
<tr>
<td>4.4 Similarity matrix</td>
<td>70</td>
</tr>
<tr>
<td>4.5 Hybrid system</td>
<td>73</td>
</tr>
<tr>
<td>4.6 Hybrid system for 9 clusters</td>
<td>77</td>
</tr>
<tr>
<td>4.7 Cluster of n=25 for neural network</td>
<td>81</td>
</tr>
</tbody>
</table>
**LIST OF TABLES**

<table>
<thead>
<tr>
<th>Table Number</th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Summary Table of Recognition Techniques</td>
<td>25</td>
</tr>
<tr>
<td>2.1</td>
<td>Actions</td>
<td>41</td>
</tr>
<tr>
<td>3.1</td>
<td>Action clusters</td>
<td>57</td>
</tr>
<tr>
<td>3.2</td>
<td>Bilateral symmetry</td>
<td>61</td>
</tr>
<tr>
<td>4.1</td>
<td>Weizman</td>
<td>64</td>
</tr>
<tr>
<td>4.2</td>
<td>Confusion matrices</td>
<td>66</td>
</tr>
<tr>
<td>4.3</td>
<td>Confusion matrix training set one</td>
<td>71</td>
</tr>
<tr>
<td>4.4</td>
<td>Similarity matrix</td>
<td>71</td>
</tr>
<tr>
<td>4.5</td>
<td>Neural network setup</td>
<td>72</td>
</tr>
<tr>
<td>4.6</td>
<td>Confusion matrix 1st modality</td>
<td>74</td>
</tr>
<tr>
<td>4.7</td>
<td>Confusion matrix 2nd modality</td>
<td>76</td>
</tr>
<tr>
<td>4.8</td>
<td>Hybrid data cluster of N=9</td>
<td>78</td>
</tr>
<tr>
<td>4.9</td>
<td>Confusion Matrix 4th Modality</td>
<td>79</td>
</tr>
<tr>
<td>4.10</td>
<td>Confusion matrix 5th modality</td>
<td>80</td>
</tr>
<tr>
<td>4.11</td>
<td>Hybrid data N=25 clusters</td>
<td>81</td>
</tr>
<tr>
<td>4.12</td>
<td>Addition of skip</td>
<td>82</td>
</tr>
<tr>
<td>4.13</td>
<td>Performance comparisons [40]</td>
<td>83</td>
</tr>
</tbody>
</table>
LIST OF ALGORITHMS

2.1 Contour Detection and Cropping ........................................... 44
2.2 Silhouette Normalization .................................................... 44
2.3 Sklansky’s Algorithm ’72 ...................................................... 46
2.4 Sklansky’s Algorithm ’82 ...................................................... 47
2.5 Defect Normalization .......................................................... 51
2.6 Triangle Registration .......................................................... 52
3.1 Bilateral Symmetry ............................................................. 60
**NOMENCLATURE**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BSM</strong></td>
<td>Bilateral Symmetry Metric</td>
</tr>
<tr>
<td><strong>CFG</strong></td>
<td>Context-free Grammars</td>
</tr>
<tr>
<td><strong>FLD</strong></td>
<td>Fisher Linear Discriminant</td>
</tr>
<tr>
<td><strong>HAR</strong></td>
<td>Human Activity Recognition</td>
</tr>
<tr>
<td><strong>HMM</strong></td>
<td>Hidden Markov Model</td>
</tr>
<tr>
<td><strong>LDA</strong></td>
<td>Linear Discriminant Analysis</td>
</tr>
<tr>
<td><strong>LHMM</strong></td>
<td>Layered Hidden Markov Model</td>
</tr>
<tr>
<td><strong>LM</strong></td>
<td>Levenberg-Marquardt</td>
</tr>
<tr>
<td><strong>PCA</strong></td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td><strong>ROI</strong></td>
<td>Region of Interest</td>
</tr>
</tbody>
</table>
INTRODUCTION

Human action recognition is a rapidly developing field in computer vision. Accurate algorithmic modeling of action recognition can introduce a multitude of challenges. Computer vision and pattern recognition algorithms can aid in action identification. In recent years research has been emphasized in recognizing actions employing features extracted from more complex actions. Simple cases of action recognition of one individual executing an action are the foundation for developing these complex scenarios in real environments. This can be especially useful for surveillance of public locations such as subways, shopping centers, or parking lots to reduce crime, monitor traffic flow, and offer security in general.

Actions and Activities

Human activities can be categorized into four levels according to Aggarwal et al. [1, 2]. They indicate that activities can be sub-defined as gestures, actions, interactions, and group activities.

*Gestures: Elementary movements of a person’s body part (waving an arm or raising a leg).*

*Actions: Single person activities. These can be composed of multiple gestures (running, walking, waving etc.).*
Interactions: Human activities that involve two or more people (hugging, two people fighting).

Group Activities: Activities performed by groups of multiple people and/or objects (two groups fighting, a party with people dancing).

For the purpose of clarity, we define actions and activities for the remainder of the paper as follows:

**Actions:** a set of intermediate poses/sub-states that together describe the motion of the human body \{i.e., run, skip, jumping jacks\}

**Activities:** a set of sequential actions \{run, jump, walk\}.

A set of actions can also share sub-states or poses. For example a sub-state in run may be similar to a sub-state in skip as shown Figure 1. To this end we can further describe actions based on their complexity as either a simple action or a compound action:

**Simple Action:** a set of intermediate unique poses/sub-states that are not shared by other actions.

**Compound Action:** a set of intermediate poses/sub-states that are shared by other actions or combination of actions.

Detecting the action based solely on single sub-state is not accurate. This potentially leads to the use of temporal information to distinguish each action, or a state machine being used to identify separate actions by linking several sub-states into complete actions. Sequential analysis can play a role in determining actions based on sub-states. *A priori* knowledge of actions can help distinguish the number of sub-states in compound and single actions.
Figure 1: Run and Skip sub-states
Figure 2: Features for face detection and iris recognition

Challenges

An effective action recognition algorithm must address the following challenges that affect feature extraction for accurate representation: non-rigidity, spatial-variance, temporal-variance, camera perspective.

Where face detection seeks to identify the location of an individual’s face, action recognition seeks to recognize the motion or action of an individual. There is generally a commonality of features in the true positive set with face recognition; each individual has two eyes, a nose, and a mouth. These are rigid features that remain on a face. Action recognition must deal with the non-rigidity of the human body. The arms and legs can be at a number of positions relative to one another that can vary in distance and degree. It is these relative positions that describe actions or intermediary poses. The intermediate states introduce a multitude of action sub-states as well as difficulties in classification.

Another problem that emerges is that of spatial variance. In a given space, each individual can perform a standard set of actions in visibly different manners. Where
one individual can walk and have their arms swing along side them, another may choose to walk with their arms crossed. The physical size (length of arms, legs, and torso shape) of the individual can also pose a problem for action comparison.

With iris recognition, unique features in the iris can be extracted and encoded to represent identity. The biometric features used are calculated without respect to any underlying structural information. On the other hand, feature extraction for human activity recognition is dependent on individual body shape and must be normalized for the spatial variance issues mentioned above. Figure 2 shows the sample features selection methods for face and iris recognition.

In addition, silhouette extraction is subjected to varying amounts of noise. Robust background segmentation can be used to isolate the foreground of interest. The main difficulty with action recognition when using silhouettes is self-occlusion. The limbs of the body are often obstructed by the torso or other limbs; this is especially true with translational cyclic motion such as walking or running. As an individual walks across, the silhouette cannot discern the left and right arms as they swing, as well as the left and right legs.

Individual orientation to the camera can also cause classification difficulties. A person walking toward a camera is different to a person walking across the frame. The walk action will appear distinct in the sagittal and coronal perspectives. Likewise, a person walking across the frame at an angle yields yet another presentation of the same action, which can be further obstructed through self occlusion.

**Applications**

Human activity recognition can be used for a multitude of applications ranging from medical rehabilitation to security and surveillance. An activity is defined as a sequence of multiple actions. The applications can be used to enhance or used in combination
with already existing systems.

Turaga et al. [43] identify the following five application areas that can benefit from Human Activity Recognition:

- **Behavioral Biometrics**: This is a relatively new field where behavior can be used to recognize humans via their “physical attributes.” This is advantageous since it does not require subject cooperation as is often the case in face, iris, and fingerprint recognition systems. Observing a subject does not interrupt the subject’s activity. This is similar to gait analysis for identification.

- **Video Analysis**: Activity recognition can be used to index and flag sections of video. The video can then be organized and summarized based on content. One commercial application might be action recognition applied to sports videos or news footage.

- **Security and Surveillance**: Activity recognition may be useful in restricted area such as airports, borders, and bank lobbies. Suspicious activity can be flagged, and the system can notify human operators who usually are responsible for multiple cameras and areas.

- **Interactive Applications and Environments**: In smart rooms, computers and humans can interact based on visual cues a computer vision system recognizes from a user. Although these technologies are not readily available, this continues to be an area of interest in commercial development.

- **Animation and Synthesis**: Activity analysis can be used in the gaming and movie industry, where there is interest in creating realistic augmented realities. This is a particular useful application for simulated environments such as CAVEs (Cave Automatic Virtual Environment), which are used to train medi-
cal professionals or military personnel with simulated subjects and interactions.

In this case gesture recognition would be used to interact with the system.

**Focus and Contribution**

We consider developing a taxonomic shape driven algorithm to solve the problem of human action recognition. To test and validate this approach, we use silhouettes of subjects performing ten actions from a video sequence shown in Figure 2.1 on page 40. A morphological algorithm is used to filter noise from the silhouette. A convex hull is created around the silhouette frame, and convex defects are used as the features for analysis (Figure 2.7a on page 49). Individually these features are the start and end defect points and defect locations. An intermediate feature consists of these six values which form a triangle between the convex hull and the original silhouette. A complete feature set consists of thirty individual values which represent the five largest convex hull defects based on area of the segmented region.

Action frame sequences are then preprocessed to separate the data into two data sets based on perspective planes and bilateral symmetry. Features are normalized for a final set creation. Three methods are used to classify ten actions (walk, run, skip, jump, side, jumping jack, pogo jump, one hand wave, two hand wave and bend as shown in Figure 2.1 on page 40) from the Weizman database. The Weizman database consists of nine individuals performing ten actions. Testing and training of the nine test subjects is performed using a leave one out methodology. The average of each testing set is taken to assess a final result. Final classification utilizes PCA and minimally encoded neural network method.

The process thus consists of segmentation, feature extraction and classification of actions. These steps are outlined in detail in Figure 3. Finally we test the algorithm and evaluate its ability to handle and address the following issues in action recog-
Dissertation Outline

This dissertation is organized as follows: Chapter I consists of a literature survey with a generalization of certain popular methods for human action recognition. It also addresses objectives and algorithms that are used and considered. Chapter II addresses the objectives and specific methodology used to build the novel algorithm that extracts features from Convexity Hull Defects. Chapter III discusses the classification measures that are used for the algorithm. The experimental results and discussions on the findings are presented in Chapter IV as well as comparative results for each method. Chapter V presents findings of the dissertation research as well as suggestions for future work and applications.
Human action recognition can generally be divided into three basic steps as shown in Figure 1.1. Low level processing (video acquisition or background segmentation) is used for detection; mid level processing (feature extraction or contour extraction) is used for tracking; higher level processing (Hidden Markov Models, Linear Discriminate Analysis) is used for action recognition. The bulk of research is concentrated on these high level features. A summary of steps is outlined: Multiple algorithms and approaches are used for solving the problem of accurate action and activity recognition. They consider shape, color, and unique features as characteristics of human actions and activities that can be used for accurate classification.

Figure 1.1: Action recognition
1.1 Detection

Detection is the first step for human action recognition. Motion segmentation serves to detect those regions where objects are moving and then assigns the moving object as part of the foreground. This is used to highlight the regions of interest in the frame. From there the extracted regions of interest are considered as objects in motion, which in turn require further classification. To track humans, these regions need to be distinguished from other objects. This can be done by determining object shape, object motion, or a hybrid of shape and motion.

1.1.1 Motion Segmentation

Three basic motion segmentation algorithms can used as a basis for detection. They will be described in the subsequent sections.

1.1.1.1 Background Subtraction

A background subtraction method for separating the foreground and background regions is described below. It relies on separating the background and foreground based on motion in the frame. The Weizman database consists of one individual performing an action in a static environment. By default, the only object moving will be the test subject.

Videos with static backgrounds can be segmented with a background subtraction algorithm. A reference background \((p_r)\) image and the current image \((p_c)\) are differenced to produce an absolute difference \((p_{ab})\) image as in Equation 1.1. A threshold \((\tau)\) level for the image is used to separate foreground and background, and for noise reduction. If the pixel is designated as a background it is assigned a pixel value of 0 (black), else it is assigned a pixel value of 255 (white). This creates a mask of the
foreground regions.

\[ p_{ab} = |p_r - p_c| \]  

\[ p_{ab} = |p_r - p_c| \Leftrightarrow p_{ab}(x, y) = \begin{cases} 
0 & p_{ab}(x, y) < \tau \\
255 & p_{ab}(x, y) \geq \tau 
\end{cases} \]  

1.1.1.2 Temporal Differencing

Videos with dynamic backgrounds can utilize the temporal differencing algorithm. Temporal differencing takes an average of pixel values over a set of preceding frames and uses this as a dynamic background. The background images generate a statistical background model. The mean, \( \mu(x, y) \), and variance of the intensities of each pixel are computed over a set of initial frames (N), as in Equation 1.3. Each pixel, \( p(x, y) \), for the current frame, \( k \), is thresholded, \( \tau \) (defined in Equation 1.4), against the corresponding picture of the background as in Equation 1.5. Temporal differencing is useful in identifying a region of interest (ROI) of motion.

\[ \mu(x, y) = \frac{1}{N} \sum_{i=1}^{N} p(x, y|i) \]  

\[ \tau(x, y) = \max \{ |\mu(x, y) - p(x, y|i)| \} , \text{ for } 1 \leq i \leq N \]  

\[ p(x, y|k) = \begin{cases} 
0 & |\mu(x, y) - p(x, y|i)| < \tau \\
255 & |\mu(x, y) - p(x, y|i)| \geq \tau 
\end{cases} \]  

1.1.1.3 Optical Flow

Optical flow is used to describe the 2D motion of points or pixels between frames over a time and to detect moving objects. The pixel motion can be used to approximate
the velocities of an object. Optical flow can use different metrics to register pixels between frames.

Shafie et al. [34] indicate that optical flow is based on the following assumptions:

- Translation in a frame can be represented by a set of parallel motion vectors
- Translation in a frame of depth will form a set of vectors that have a focus point of expansion, where frame of depth is a movement parallel to the camera
- Rotation can be represented by a set of concentric motion vectors
- Rotation in a frame of depth can be represented as vectors emerging from straight line segments

The Lucas-Kanade method [4] is a two-frame method for flow estimation. For a pixel location, \((x, y, t)\), with intensity, \(I(x, y)\), it moves by \((\delta x, \delta y, \delta t)\) between frames. Thus the constraint is given as:

\[
I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{1.6}
\]

The constraint can be expanded to:

\[
I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t + \varepsilon \tag{1.7}
\]

From equation (1.7):
\[
\frac{\partial I}{\partial x} \delta x + \frac{\partial I}{\partial y} \delta y + \frac{\partial I}{\partial t} \delta t = 0
\]

\[
\Rightarrow \frac{\partial I}{\partial x} \cdot \delta x \delta t + \frac{\partial I}{\partial y} \cdot \delta y \delta t + \frac{\partial I}{\partial t} \cdot \delta t \delta t = 0
\]

\[
\Rightarrow \frac{\partial I}{\partial x} V_x + \frac{\partial I}{\partial y} V_y + \frac{\partial I}{\partial t} = 0
\]

Where \( V_x \) and \( V_y \) are the velocity or optical flow of \( I(x,y,t) \). \( I_x \) corresponds to the derivative \( \frac{\partial I}{\partial x} \), therefore:

\[
\begin{bmatrix}
I_{x1} & I_{y1} \\
I_{x2} & I_{y1} \\
\vdots & \vdots \\
I_{xn} & I_{yn}
\end{bmatrix}
\begin{bmatrix}
V_x \\
V_y
\end{bmatrix}
= \begin{bmatrix}
-I_{t1} \\
-I_{t1} \\
\vdots \\
-I_{tn}
\end{bmatrix}
\]

(1.9)

Since Equation 1.9 is overdetermined, it can be solved by using the least squares method as:

\[
A^T A \nu = A^T (-b)
\]

\[
\Rightarrow \nu = (A^T A)^{-1} A^T (-b)
\]

\[
\Rightarrow \begin{bmatrix}
V_x \\
V_y
\end{bmatrix}
= \begin{bmatrix}
\sum I_{x1}^2 & \sum I_{x1} I_{y1} \\
\sum I_{x1} & \sum I_{y1}^2
\end{bmatrix}^{-1} \begin{bmatrix}
-\sum I_{x1} I_{t1} \\
-\sum I_{y1} I_{t1}
\end{bmatrix}
\]

(1.10)

The calculations are performed for multiple scales with a weighting function to emphasize the central pixel of a window.

Whereas the first two methodologies are ideal for stationary cameras, optical flow can be used with a moving camera frame. The main disadvantage of using optical flow vectors is that flow computations are computationally expensive and are semi-sensitive to noise and illumination changes. This makes a real time implementation
without specialized hardware difficult.

1.1.2 Object Classification

After an object is detected, the object can be classified from the extracted regions of the foreground mask. To accurately track people we need to be able to distinguish them from other moving objects in the frame. The objects can be classified based on shape, motion, or a hybrid of both methods. Classifying an object based on shapes requires the shapes to be represented by points, boxes, silhouettes, or blobs. The relationship of these shapes can be unique to objects in the frame. Other features such as area and pixel color can also be used to further identify an object[27].

Motion classification is useful when attempting to extract human shapes, or blobs from a set of frames. Human motions tend to be cyclic in nature.

A motion $C(t), C : \mathbb{R} \to \mathbb{R}$ is considered cyclic if [10]:

$$C(t + T) = C(t) \quad (1.11)$$
The optical flow algorithm described in subsection 1.1.1.3 can help identify the periodicity of motion of objects. Rigid objects have little residual flow whereas nonrigid objects have higher residual flow and a periodic component associated with them. The optical flow of the motion of limbs in an action is shown in Figure 1.2. The arrows indicate the direction of the motion and color indicates the rate of change of the motion between frames.

Both of these basic methodologies are covered more in detail in Section 1.3 and can be used for action recognition.

1.2 Tracking

1.2.1 Region Based Tracking

Region based tracking serves to detect and discriminate moving targets. Based on the motions of those targets it is able to extract the region where an object is moving. Tracking over time can match objects over frames using features such as points, lines, or blobs and the relationships between these features, such as velocity, color, and position. With region based tracking, the goal is to identify a region associated with the moving object, label the features, and finally track them. It works well in non-cluttered environments.

In action recognition, this method has been used to successfully identify moving body parts. Wren, et al. [46] use small blob features to track human motion. They consider the human body as a composite of small blobs that represent the head, torso, arms and legs. The pixels are assigned to these body part blobs through a region growing algorithm. Tracking each individual blob leads to tracking the individual as a whole.

Tsukiyama et al. [1, 27] use a similar procedure. In an image frame, all moving
objects are detected. They then find objects that are considered humans based on size and shape, and finally track the motion of the humans over time.

1.2.2 Feature Based Tracking

As opposed to tracking whole objects, feature based tracking uses the sub-features of an object. Feature-based tracking consists of feature extraction of an object and feature matching between frames. For example, in a stick figure model the joint locations can be the extracted features, and their respective locations between frames can be matched using a distance metric. Feature extraction is useful when objects are partially occluded since some features are still available for matching. Segen [33] use the corner points of human silhouettes as the features to track and match the features based on position and the curvature of the points in successive frames. Figure 1.3 shows the path of features tracked in progressive frames. The path taken by the features can be used to distinguish an action.

1.3 Recognition

Aggarwal et al [2] indicate that recognition can be divided into two groups: single-layered approaches and hierarchical approaches. Single-layered approaches recognize human action based on a sequence of images whereas hierarchical approaches represent complex actions through an accumulation of simple actions. The latter consists of a multi-layered network used to analyze the system. These two groups can be further
divided into other techniques used in the actions recognition field as shown in Figure 1.4.

1.3.1 Single Layered Approach

The single-layered approach can be further classified according to how human action is modeled, either as a space-time approach or a sequential approach. In general, action is considered a class of image sequences. Single layered approaches benefit when a sequential pattern can be found from a training sequence. The training sequence is used as a template to recognize other actions. Algorithms can be used to make a decision on the sequence class. Single layered approaches seek to analyze simple sequential actions (running, walking, jumping, etc) of humans, but they differ on how they process the input videos. The space-time approach considers the input video as a 3D-XYT volume, whereas the sequential approach considers the input video a sequence of observations that define an action.

1.3.1.1 Space time

A video sequence can be represented as a 3D-XYT space time volume by essentially “stacking” the 2D-XY frames across time, and T to create a volume. The volumes created can then be compared to measure the similarity of it to other space-time volumes. Generally using a space-time approach consists of using training video to
construct a 3D XYT model for specific actions and using testing video to construct a space time volume of unknown actions. The testing volume is compared to the template volumes of the training videos of the actions. Shape similarity and appearances are measured between the training and testing sets. The action is chosen when the testing model has the highest similarity with one of the training templates. In essence, a 3D volume is used to represent the action and a template matching algorithm is used for recognition.

Blank et al [7] at the Weizman Institute used a technique called the space-time shape shown in Figure 1.5. Human silhouettes are considered to be three-dimensional shapes in a space-time volume. Space-time features are extracted as solutions to the Poisson equation and used for action recognition. Zelnik-Manor et al [50] consider events as long-term temporal objects.

Instead of space time volumes, space-time trajectories can also be used to track features (joints, limbs) in the space-time dimension. A recognition system can track a set of feature points through a time dimension to estimate an action. The subfeatures of the trajectories or volumes can be used in coordination with higher level pattern recognition algorithms to identify actions.

**Template Matching**

Template matching uses the static shape of a frame to compare it to stored shapes.
This method is computationally inexpensive and has a relatively simple implementation. For a silhouette, shown in Figure 1.6, the shapes are easy to extract from an image. It also is less sensitive to noise since it is a region based matching method [27].

Li et al. [23] match a silhouette against the silhouette of the model and the similarity measure is simply given as the area difference. They use an operator $S(I_1, I_2)$ to measure the shape similarity between two binary images $I_1$ and $I_2$. They consider the area difference between two shapes to be the ratio of positive error, $p$, and negative error $n$ as:

$$p = \frac{I_1 \cap I_2'}{I_1 \cup I_2}$$  \hspace{1cm} (1.12)$$

$$n = \frac{I_2 \cap I_1'}{I_1 \cup I_2}$$  \hspace{1cm} (1.13)$$

Similarity is defined as:

$$S(I_1, I_2) = e^{-(\alpha_1 \cdot p + \beta_1 \cdot n)} \cdot (1 - \alpha_2 \cdot p - \beta_2 \cdot n)$$  \hspace{1cm} (1.14)$$

where $\alpha_1, \alpha_2, \beta_1,$ and $\beta_2$ are degrees of emphasis on the error used in silhouette recognition. The actions are unique to the shape template created.

Bobick and Davis [8, 9] developed “temporal templates” to model different actions using two methods. The object of interest is extracted through background subtraction and then the background subtracted binary images are combined into a static image. These two methods together make up a representation of actions.

The “motion energy image” (MEI) representation gives equal weight to all images in the sequence and shows where the motion is occurring. Consider the following, if $I(x, y, t)$ is the image sequence and $D(x, y, t)$ is the binary image sequence which
Figure 1.6: MEI and MHI [8, 9]

indicate the regions of motion. The “motion-history image” (MHI) represents how the image is moving. The pixel intensity is a function of the temporal history of the motion at that point in time. Then, binary MEI, $E_{\tau}(x,y,t)$, is defined as:

$$E_{\tau}(x,y,t) = \bigcup_{i=0}^{\tau-1} D(x,y,t-i)$$  \hspace{1cm} (1.15)

and MHI, $H_{\tau}(x,y,t)$ is a replacement and decay operator defined as:

$$H_{\tau}(x,y,t) = \begin{cases} \tau & \text{if } D(x,y,t) = 1 \\ \max(0, H_{\tau}(x,y,t-1) - 1) & \text{otherwise.} \end{cases}$$  \hspace{1cm} (1.16)

The combination of a MEI and MHI values are used to identify actions.
1.3.1.2 Sequential Approach

The sequential approach to action recognition analyzes a sequence of features and looks to observe a particular sequence to identify it as an action. The sequence of images in this method becomes a sequence of feature vectors that describe an image. These feature vectors are analyzed to determine the likelihood of the sequence to action identifier. If the probability is high that the sequence matches the sequence identity of an action, then that action has occurred.

Exemplar based sequence approaches describe action membership based on the training data. Action membership is determined based on whether or not a representative sequence matches a new incoming sequence. The data is not transformed to a different subspace or assigned a class identifier. For this approach the template sequence or a set of sample sequences of known actions are compared to a new input sequence. The feature vectors are extracted from the video and compared to the known action set. Again if the similarity is high enough, the system will identify the given input as specific action.

State-based sequence approaches generate a sequence based on the training of a model for a system. The probability of all sequences is calculated by the training model. When a new sequence is introduced, a probability is calculated to determine if that sequence is likely. The model is statistically trained. A Hidden Markov Model is a prime example of this type system.

Hidden Markov Models

A Hidden Markov model is a state space model where the latent variables are discrete. We can describe these states as directed graphs as illustrated in Figure 1.7.

There are three issues we have to consider when dealing with HMMs. They are as follows:
Figure 1.7: HMM

- **The Evaluation Problem** - If we have transition probabilities $a_{ij}$ (hidden transition probability) and $b_{jk}$ (output probability) we would like to determine the probability that a particular sequence of visible states $\mathbf{V}^T$ was generated by the HMM.

- **The Decoding Problem** - If we have a HMM and a set of observations $\mathbf{V}^T$, we would like to determine the most likely sequence of hidden states $\omega^T$ that created $\mathbf{V}^T$.

- **The Learning Problem** - If we have a structure of a model but not the transition probabilities $a_{ij}$ and $b_{jk}$, and are given a training set of observations, we would like to determine $a_{ij}$ and $b_{jk}$.

The main issue in considering HMMs is the **The Learning Problem**. We seek to determine the model parameters (transition probabilities) from the training samples. We consider the Baum-Welch Algorithm, which is a generalized expectation maximization algorithm. The weights for transitioning will be iteratively updated to explain the training sequence. We offer a generalized explanation in terms of the goals of the project.

We can define the following parameters for discussion:
• $\alpha_i(t)$ - probability that the model is in state $\omega_i(t)$

• $\beta_i(t)$ - probability that the model is in state $\omega_i(t)$ and will generate the remainder of a given target sequence, from step $t$ to $T$

$$
\beta_i(t) = \begin{cases} 
0 & \omega_i(t) = \omega_0 \text{ and } t = T \\
1 & \omega_i(t) \neq \omega_0 \text{ and } t = T \\
\sum_j \beta_j (t+1) a_{ij} b_{jk} v(t+1) & \text{otherwise}
\end{cases}
$$

(1.17)

For human action recognition, states in the HMM will be the sub-states of the simple and compound actions. Simple actions have less sub-states and they will progress through the HMM quickly as opposed to compound actions which will progress through more states for a final action classification.

The classification of the silhouettes through a subspace method creates the states for our sequential actions. Those class assignments will be the multiple sub-states of an action. Complex actions also share certain sub-states in which actions get confused for one another in a frame by frame analysis. A sequential representation would be ideal when actions share classes and optimal for translational actions. The overall goal of our HMM would be to be able to move through states that multiple actions share. Examining Figure 1.8, the correct sequential path for the HMM is shown for run and skip, where the probability from transitioning from state to state is $\{a_{ij}, b_{ij}\}$ and the probability of transitioning to a visible (final) state is $\{ac_{56}, ac_{57}, bc_{56}, bc_{57}\}$. In an accurate system, $ac_{57} \gg ac_{56}$ and $bc_{56} \gg bc_{57}$.

### 1.3.2 Hierarchical Approach

Hierarchical approaches are used to identify more complex combination actions. They will be briefly covered in this section. The techniques used in Chapter II will utilize some variations of the methods described above. Hierarchical approaches are, in
general, multilayer systems that first recognize simple actions and use them as a basis to recognize higher level actions or combination of actions.

Statistical hierarchical approaches use a probabilistic state model with multiple layers of state models such as HMMs, now called Layered HMMs (LHMMs) to recognize a compound action. A single layered sequence approach can be used as the foundation layer for these models. Syntactic hierarchical approaches tend to model human activities as strings of symbols. These symbols represent a lower level action, which are recognized using the techniques mentioned above. The human activities are string of actions that are recognized by parsing techniques or using Context-Free Grammars or CFGs. The description of hierarchical approach stores the spatio-temporal structure of the activities. It describes the relationship between simple actions by describing the temporal, spatial and logical relationship between the simple actions. Recognition of an activity is performed by searching through the simple actions that satisfy a certain requisite enforced by the system or action. The advantages and disadvantages of each recognition technique is shown in Table 1.1.
### Table 1.1: Summary Table of Recognition Techniques

<table>
<thead>
<tr>
<th>Approach</th>
<th>ADVANTAGES</th>
<th>DISADVANTAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>SINGLE LAYER</td>
<td><strong>SPACE TIME</strong></td>
<td>• Good for recognition of periodic actions</td>
</tr>
<tr>
<td>APPROACH</td>
<td>• View invariant in most cases</td>
<td>• Difficulties handling speed and motion</td>
</tr>
<tr>
<td></td>
<td>• Good under noise and illumination</td>
<td>• Not good for complex actions</td>
</tr>
<tr>
<td></td>
<td>• Uses sequential relationship between features</td>
<td>• Feature relationship are dependent especially for non-periodic activity</td>
</tr>
<tr>
<td></td>
<td>• Detect complex actions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Uses a nonlinear matching methodology</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Less training data (Exemplar)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Probabilistic analysis (state-based)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Requires preprocessing for feature extraction</td>
<td>• Large amount of training data(state-based)</td>
</tr>
<tr>
<td>HIERARCHICAL</td>
<td><strong>STATISTICAL</strong></td>
<td>• Actions cannot have complex temporal information</td>
</tr>
<tr>
<td>APPROACH</td>
<td>• Good for recognition of sequential actions</td>
<td>• Features not easily observed and are more complex</td>
</tr>
<tr>
<td></td>
<td>• Recognize actions with noisy input</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Probabilistic analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Uses subevents</td>
<td>• Can’t recognize concurrent actions</td>
</tr>
<tr>
<td></td>
<td>• Can’t account for low level feature detection failures</td>
<td>• Limited to actions with subevents</td>
</tr>
<tr>
<td></td>
<td>• Features not easily observed and are more complex</td>
<td>• Features not easily observed and are more complex</td>
</tr>
<tr>
<td></td>
<td><strong>SYNTACTIC</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Recognize actions with complex temporal information</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Recognize actions sequentially and concurrently</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Can’t account for low level feature detection failures</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Features not easily observed and are more complex</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>DESCRIPTION</strong></td>
<td></td>
</tr>
</tbody>
</table>

25
1.4 State Space Approaches

1.4.1 Principal Component Analysis

Principal Component Analysis (PCA) is a statistical technique used for finding a pattern with high dimensionality. An orthogonal transformation is used to transform the number of features, which may or may not be correlated, into principal components or uncorrelated variables. The transform is defined such that the first principal component has as much variance as possible. This accounts and emphasizes the similarities and differences between data sets. Each consecutive component also tries to maintain the same relationship in addition to being orthogonal to the preceding component. The advantage of PCA is that it reduces dimensionality without significant loss of information.

The matrix set of vectors \([M \times n]\) can then be defined as a single vector, \(V_n = \langle 1 \times (z) \rangle\). Where \(z = M \cdot n\). \(V_n\) needs to be transformed into a vector \(V_0\) so that the \(V_0\) and various \(V_k\) have a minimum sum of the squared distance between them, i.e. the squared-error criterion, \(J_0(V_0)\) is minimized as:

\[
J_0(V_0) = \sum_{k=1}^{n} \|V_0 - V_k\|^2
\]  

(1.18)

A value of \(V_0\) is needed to minimize \(J_0\). The sample mean, \(m\), of the data set is found to be the value that minimizes \(J_0\), where \(m\) is defined as follows:

\[
m = \frac{1}{n} \sum_{k=1}^{n} V_k
\]  

(1.19)

and can be supported by the following proof:
\[ J_0 (V_0) = \sum_{k=1}^{n} \| V_0 - V_k \|^2 \]
\[ = \sum_{k=1}^{n} \| (V_0 - m) - (V_k - m) \|^2 \]
\[ = \sum_{k=1}^{n} \| V_0 - m \|^2 - 2 \sum_{k=1}^{n} (V_0 - m)^T (V_k - m) + \sum_{k=1}^{n} \| V_k - m \|^2 \]  
\[ = \sum_{k=1}^{n} \| V_0 - m \|^2 - 2 (V_0 - m)^T \sum_{k=1}^{n} (V_k - m) + \sum_{k=1}^{n} \| V_k - m \|^2 \]
\[ = \sum_{k=1}^{n} \| V_0 - m \|^2 + \sum_{k=1}^{n} \| V_k - m \|^2 \]  

\( J_0 \) is minimized by setting \( V_0 = m \). Therefore \( V'_k = \sum_{k=1}^{n} V_k - m \), which now has a zero mean and allows the first principal component to have a maximum variance.

The data can also be projected onto a line running through the sample mean. The unit vector \( e \) is the vector in the direction of the line. The equation of this line is \( x = m + a e \), where \( a \) is the the distance of point \( x \) from \( m \). Where the data projected along two components can be seen in Figure 1.9. Therefore \( x_k \) can be defined as:

\[ m + a_k e \]  

An optimum set of coefficients \( a_k \) can be obtained by minimizing the squared error criterion function given as:
\[ J_1 (a_1, \cdots, a_n, e) = \sum_{k=1}^{n} \| (m + a_k e) - x_k \|^2 = \sum_{k=1}^{n} \| a_k e - (x_k - m) \|^2 \]

\[ = \sum_{k=1}^{n} a_k^2 \| e \|^2 - 2 \sum_{k=1}^{n} a_k e^T (x_k - m) + \sum_{k=1}^{n} \| (x_k - m) \|^2 \]  

(1.22)

Taking the partial derivative of equation 1.22 with respect to \( a_k \) and setting it to 0, yields (\( \| e \| = 1 \)):

\[ a_k = e^T (x_k - m) \]  

(1.23)

This is the least-squares solution by projecting \( x_k \) onto a line in the direction of \( e \) that passes through \( m \). The next step is to find the direction of \( e \) for the line. This can be found by using the scatter matrix (characterize scatter of data) \( S \), defined as:

\[ S = \sum_{k=1}^{n} (x_k - m) (x_k - m)^T \]  

(1.24)

When \( a_k \) defined in equation 1.23 is inserted in Equation 1.22 we get:

\[ J_1 (e) = -e^T Se + \sum_{k=1}^{n} \| x_k - m \|^2 \]  

(1.25)

Using Lagrange multipliers to maximize \( e^T Se \), with the constraint \( \| e \| = 1 \) and \( \lambda \) is the undetermined multiplier thus \( u = e^T Se - \lambda (e^T e - 1) \), differentiation yields this gradient vector:

\[ \frac{\partial u}{\partial e} = 2Se - 2\lambda e \]  

(1.26)

Setting the gradient vector to zero, \( e \) is the eigenvector of the scatter matrix:

\[ Se = \lambda e \]  

(1.27)
The largest \( e \) corresponds to the largest eigenvalue of the scatter matrix. This result can be altered to be used for a \( d' \)-dimensional projection as:

\[
x = m + \sum_{i=1}^{d'} a_i e_i
\]  

(1.28)

The coefficients \( a_i \) are the principal components of \( x \). Principal components are plotted against one another for separation of variables in a scatter plot.

1.4.2 Fisher Linear Discriminant - Linear Discriminant Analysis

PCA uses components for representing data and allows data to be segmented out based on these components. Although this may segment the data, this does not show that components can be useful for discriminating data between different classes. PCA finds components that are efficient for representation whereas LDA looks to find components that are efficient for discrimination.

A decision surface is a linear function of the input vector \( x \) and can be defined by \((D-1)\) dimensions within a \( D \)-dimensional input space. We wish to take the input vector \( x \) and assign it to one of the \( K \) discrete classes \( C_k \), where \( k = 1, \ldots, K \). More specifically, we wish to use Fisher Linear Discriminant (FLD), which projects a linear classification model by dimensionality reduction. We cannot use a simple Fisher Linear Discriminate which usually distinguishes between two classes, we later assume that the action sets will be separated into multiple substate sets.

So in terms of action classes, Fisher Linear Discriminate will be defined as follows for multiple class. If \( K > 2 \) classes, the dimensionally \( D \) of the input state is greater than \( K \) number of classes. Also if \( D' > 1 \), the linear feature \( y_k = w_k^T x \) where \( k = 1, \ldots, D' \). The feature value can form the vector \( y \) and the weight vectors \( \{w_k\} \) can be the column matrix \( W \) so that:
y = W^T x \quad (1.29)

The within-class covariance can be defined as:

\[ S_W = \sum_{k=1}^{K} S_k \quad (1.30) \]

with

\[ S_k = \sum_{n \in C_k} (x_n - m_k)(x_n - m_k)^T \quad (1.31) \]

\[ m_k = \frac{1}{N_k} \sum_{n \in C_k} x_n \quad (1.32) \]

where \( N_k \) is the number of patterns in class \( C_k \).

The total covariance matrix is the sum of the within-class covariance and the between class covariance, \( S_B \):

\[ S_T = S_w + S_B \quad (1.33) \]

\[ S_B = \sum_{k=1}^{K} N_k (m_k - m)(m_k - m)^T \quad (1.34) \]

Defining the matrices in the projected \( D' \)-dimensional \( y \)-space we get:

\[ s_w = \sum_{k=1}^{K} \sum_{n \in C_k} (y_n - \mu_k)(y_n - \mu_k)^T \quad (1.35) \]

and
\[ \mathbf{s}_B = \sum_{k=1}^{K} N_k (\mu_k - \mu) (\mu_k - \mu)^T \]  

(1.36)

where,

\[ \mu_k = \frac{1}{N_k} \sum_{n \in C_k} y_n \quad \mu = \frac{1}{N} \sum_{k=1}^{K} N_k \mu_k \]  

(1.37)

The \( K \) classes can then be used to distinguish between actions for recognition [15, 5].

FLD calculates components from a data set such that they will be maximally useful for discriminating data between different classes. As stated in earlier, PCA finds components that are efficient for representation whereas FLD looks to find components that are efficient for discrimination. Using a linear model for classification may be advantageous for action recognition because the actions are represented in a fixed number of distinct classes whose identity within the training set are known \( a \ priori \). The testing set is constrained by the fixed number of classes, so the actions will be separated by their substates. These classes may then be used for states in a Hidden Markov Model.

1.5 Neural Networks

Neural networks are adaptive systems that change parameters based on input/output data. The network seeks to 'learn' by mapping connections between input/output layers. It used hidden nodes that are nonlinear to identify a state.

1.5.1 Feed Forward Network Functions

Figure 1.11 shows a multilayer feed forward neural network. Each hidden node calculates the weighted sum of its inputs which is called the net activation or simply \( net \).
The net activation is the product of the input and the weights at that hidden node. The net can be defined as follows:

$$net_j = \sum_{i=1}^{d} x_i w_{ji} + w_{j0} = \sum_{i=0}^{d} x_i w_{ji} \equiv w_j^T x$$  \hspace{1cm} (1.38)$$

where $i$ is the index unit in the input, $j$ is the index unit into the hidden node, and $w_{ji}$ is the input-to-hidden node weights at the hidden node $j$. The hidden nodes emit an output that is a nonlinear function, $f(net)$, also called the activation function defined as:

$$y_j = f(net_j)$$  \hspace{1cm} (1.39)$$

The threshold can be defined below, as in Equation 1.40, other complex functions can also be used and will be discussed in subsequent sections:

$$f(net) = Sgn(net) \equiv \begin{cases} 
1 & \text{if } net \geq 0 \\
-1 & \text{if } net \leq 0 
\end{cases}$$  \hspace{1cm} (1.40)$$

Each output computes its individual net activation based on the hidden unit nodes:
$$\text{net}_k = \sum_{j=1}^{n_H} y_i w_{kj} + w_{k0} = \sum_{j=0}^{n_H} y_j w_{kj} = w_k^T y$$  \hspace{1cm} (1.41)$$

where, $k$ is the index unit in the output, and $n_H$ is the number of hidden nodes.

The output node, $z_k$, that calculates the nonlinear function of its net is:

$$z_k = f(\text{net}_k)$$  \hspace{1cm} (1.42)$$

The output node can be generalized to $g_k(x)$ for multiple inputs as:

$$z_k = g_k(x)$$  \hspace{1cm} (1.43)$$

where,

$$g_k(x) \equiv z_k = f \left( \sum_{j=1}^{n_H} w_{kj} f \left( \sum_{i=1}^{d} x_i w_{ji} + w_{j0} \right) + w_{k0} \right)$$  \hspace{1cm} (1.44)$$

To classify $c$ output nodes, the signal from each output node is the discriminate function $g_k(x)$. This function can allow the activation function in the hidden node to be different from the activation function for each node [15, 5].

1.5.2 Network Training

1.5.2.1 Backpropagation

Backpropagation is the most common method for supervised training of a feed forward neural network. Where, feed forward networks take patterns from the inputs and pass those signals to yield outputs in the output nodes, backpropagation methods take the input and changes the network parameters so that the outputs are close to the target values that are known. It is used to create and train a network. The outputs are compared to the target values, $t$, where the difference in value is the training error of
the system. The criterion function is a scalar function of the weights and is minimized through output matching where the weights are dynamically changing to reduce the error. In a transient system, inputs of an unknown identity can then be sent through the system and their outputs estimated.

The training error on a specific pattern is defined as the sum of error over the output nodes of the squared difference between the targets $t_k$ and the actual output $z_k$ as:

$$ J(w) \equiv \frac{1}{2} \sum_{k=1}^{c} (t_k - z_k)^2 = \frac{1}{2} \| t - z \|^2 $$  \hspace{1cm} (1.45)

where $t$ is target, $z$ is the output vector, $c$ is the output vector length, and $w$ is all the weights in the network. Backpropagation weights are initialized with random values and then they are adjusted so that the error is reduced. The change in the weights with the learning rate is $\eta$ is:

$$ \triangle w = -\eta \frac{\partial J}{\partial w} $$  \hspace{1cm} (1.46)

The updated iterative weight algorithm of iteration $m$, can be defined as follows:
\[ \mathbf{w}(m + 1) = \mathbf{w}(m) + \Delta \mathbf{w}(m) \]  

(1.47)

Error is not specifically dependent on \( w_{jk} \), the error is differentiated:

\[
\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial w_{kj}} = -\delta_k \frac{\partial \text{net}_k}{\partial w_{kj}}
\]  

(1.48)

where the sensitivity of \( k \) which describes the overall error change of the net activation is

\[ \delta_k = -\frac{\partial J}{\partial \text{net}_k}, \]

Therefore the sensitively of the hidden node can be defined as:

\[ \delta_j \equiv f'(\text{net}_j) \sum_{k=1}^{c} w_{kj} \delta_k \]  

(1.49)

Equation 1.49 is the the sum of the individual sensitivities of the output nodes weighted by \( w_{kj} \) multiplied by \( f'(\text{net}_j) \). Hence the learning rule for the input-to-hidden weights is:

\[ \Delta w_{ji} = \eta x_i \delta_j \]  

(1.50)

Another method used to calculate the weights and the associated errors is the Levenberg-Marquardt (LM) algorithm. It uses the Jacobian matrix to measure the sensitivity of the output to changes in the input variables. The Jacobian matrix is the first order partial derivate of a vector with respect to another vector. Therefore known errors in the inputs can be propagated through the system to estimate how they contribute to the output error as:
\( \mathbf{w}(m + 1) = \mathbf{w}(m) - \Delta \mathbf{w}(m) \tag{1.51} \)

The LM algorithm is an iterative technique that uses the minimum of a multivariate function to find a numerical solution to minimize the error. It can be summarized as:

- Compute network outputs and errors after propagating inputs through a system
- Compute Jacobian matrix \( J(\rho) \), where \( \rho \) are the weights and biases of the network
- Solve Equation 1.52 to get \( \Delta w(m) \)
- Recompute error using \( w + \Delta w(m) \), where

\[
\Delta \mathbf{w}(m) = \left( J(\rho)^T J(\rho) + \lambda I \right)^{-1} J(\rho)^T \varepsilon \tag{1.52}
\]

\( \lambda \) is modified based on the error, and \( \varepsilon \) is the error vector and the Jacobian is taken of the error with respect to the weights [15, 5].

**Learning Curve**

Before a system is trained, the weights are randomized, so the error is higher. The learning curve shows that the error decreases as the number of iterations of learning increases shown in Figure 1.12. The criterion function is shown as a function of the number of iterations. A validation set, usually a subset of the training data, is used to find the end point of training. When a minimum error of the validation data is reached, the training is halted. This prevents overfitting of the data.
Sigmoid function

The activation function $f(\cdot)$ should have the following desirable properties to be used for a neural network:

- $f(\cdot)$ must be nonlinear—this allows computational power of a three-layer network to be higher than a two-layer network
- $f(\cdot)$ saturates—it has a maximum and minimum output value weights thus the activation will be bounded
- $f(\cdot)$ is continuous and has smoothness — $f(\cdot)$ and $f'(\cdot)$ are defined in a specified range
- $f(\cdot)$ should be monotonic

The sigmoid function shown in Figure 1.13 has these properties. In addition, it is centered on zero and is asymmetric; asymmetric sigmoids allow the system to
be trained faster [15, 5]. We choose the sigmoid function for system training for
classification.
CHAPTER II

HULL CONVEXITY DEFECT FEATURES FOR HUMAN ACTION RECOGNITION

This Chapter presents our methodology for Human Action Recognition. We begin by describing a silhouette extraction technique and move through the stages of the algorithm until a feature vector is created for use in the classification step.

2.1 Segmentation of Silhouettes

The actions described previously can be further subdivided into two sets based on their trajectory motion. This classification will be useful for action identification. The first subset will consists of actions, shown in Figure 2.1 that are translational cyclic actions and the second are stationary actions. We define each as follows:

**Cyclic actions** have stages of actions that are repetitive in nature, where at a time $t$ in a frame the cyclic motion $C$ fulfills the requirement at $C(t) = C(t + \Delta t)$.

**Stationary actions** are actions in which the torso is not moving translationally across the frame.

Distinguishing between stationary and cyclic actions is essential for classification. The cyclic action of walk is the gait cycle. The gait cycle shown in Figure 2.2 can be subdivided into two phases: stance (60% of gait) and swing (40% of gait).
Figure 2.1: Examples of Actions for Recognition

(a) bend  (b) jump  (c) one handed wave
(d) run  (e) side
(f) walk  (g) two handed wave
(h) skip  (i) pogo jump (pjump)  (j) jumping jack
Stance has four subphases:

- 'Heel strike to foot flat’ to 'foot flat to midstance’ to 'midstance to heel off’ to 'heel off to toe off’

Swing has two sub-phase :

- 'Acceleration to mid-swing’ to 'mid-swing to deceleration’

Therefore we have six distinct sub-states that together define the action of walk. In a subspace analysis, the action of walk would be represented by six regions. Unfortunately, the stances are similar when looking at the silhouette. The left and right appendages cannot be distinguished from one another. This is due to the self occlusion that occurs when performing an action. The remaining cyclic actions mentioned in Table 2.1 can also be further subdivided into sub-states for a frame by frame analysis. Multiple complete cycles are observed. When grouping actions for analysis, identification and separability of each phase is important, but can eventually conflict when actions share phases. The sequential frame information will be used to examine frames that are similar in shape.
2.1.1 Motion Detection

Temporal Differencing

Videos with dynamic backgrounds can utilize the temporal differencing algorithm described in Section 1.1.1.2. The goal is to capture the temporal change of the object and successive motions. This is especially useful with the translational actions described above. For the stationary actions, it can detect the motion of the limbs if the torso is motionless. We use this algorithm to detect a region of interest (ROI) as shown in Figure 2.3. The ROI is useful to crop the image and obtain a silhouette.

2.1.2 Silhouette Extraction

Background-Foreground Separation and Contour Detection

The videos are sampled at 12 frames/sec to get the action frame stills. The images are upsampled from their original size of 140×144 to 640×512.

After obtaining an ROI we can use background subtraction to separate the back-
ground from the foreground as shown in Figure 2.4. Additional morphological operations including erosion, dilation, opening, and closing are performed to reduce noise and to smooth the silhouette contour. We use these operations to try to correct for any background-foreground misclassification.

Contour detection and identification is a vital step in noise reduction. If the contour area, $c$, is below a certain area threshold, $\tau_{\text{area}}$, it is removed from the image. Since we know that databases have an individual in the frame, we can assume that the largest contour is in fact an individual, Algorithm 2.1.

To accurately classify actions, the silhouette of each individual needs to be normalized to be able to make a valid comparison between individuals. The silhouette is tightly cropped so that the feature locations are in the same relative space. The cropped image will be used for template matching the phase of an action. The silhouettes are now centered in the image, $I'$ and a uniform padded border is added. For translational actions this will remove the motion content of the frame. The silhouette will now be isolated independently of its motion within the frame. The padded border is important for creating the convex hull. If the filled contour lies on the edge of the cropped image a hull cannot be created. The average resolution of the final images are $110 \times 250$ for the ten actions. This is still relatively low resolution for action recognition analysis.

To attempt to correct for size invariance, Algorithm 2.2 normalizes the image $I'$
Algorithm 2.1 Contour Detection and Cropping

1. Search image for all contours, \( c_N \), in image \( I \), where \( N \) is the number of contours

2. Then

\[
c_N = \begin{cases} 
0 & c_N < \tau_{\text{area}} \\
255 & c_N \geq \tau_{\text{area}} 
\end{cases}
\]

3. Sort remaining contours, keep contour with largest area, \( C_L \)

4. Draw \( C_L \) to new image, \( I' \)

5. Fill contour \( C_L \) to obtain silhouette \( S_L \)

6. Crop \( I' \) to fit \( S_L \)

7. Add a uniform four pixel wide border

8. The image \( I' \) with \( S_L \) will be used for convex hull creation

Algorithm 2.2 Silhouette Normalization

1. Let

\[
I = \{ I'_1 \in \mathbb{N}^{m_1 \times n_1}, I'_2 \in \mathbb{N}^{m_2 \times n_2}, \ldots, I'_N \in \mathbb{N}^{m_N \times n_N} \}
\]

2. Let \( h = \max m_i \).

3. \( \forall I'_i \in I \), resize \( I'_i \) such that \( I'_i \in \mathbb{N}^{h \times n_i} \)

such that all frames in the set \( I \) have a uniform height, \( h \). The defect locations are then be somewhat aligned with respect to one another.

2.2 Feature Extraction

2.2.1 Convex Hull Creation - Sklansky’s Algorithm

To obtain our feature set, we propose the use of convex hulls. The convex hull of the silhouette, \( P \), is defined as the smallest area convex polygon which can enclose \( P \). Therefore, for a set of \( N \) points \( \{p_0, p_1, p_2, \ldots, p_N\} \in P \), it can be stated that a hull,
$H$, can be created with $M$ points from the set $N$ to create a minimum area convex polygon. Referring to Figure 2.5, convex hulls are created by taking the interior angle, $\theta$, of three adjacent points $\langle p_1, p_0, p_9 \rangle$. If $\theta > \pi$ then $p_0$ is considered a reflex point and $p_0 \notin M$. The final set of $H$ is \{$p_1, p_9, p_7, p_5, p_3$\}.

The Sklansky '82 algorithm [38] is a sequential linear time algorithm used to compute the convex hull for a set of points/simple polygon. Sklansky’s algorithm was one of the first algorithms used to solve for a convex hull. It was a precursor to the first generation of hull algorithms. Sklansky '82 algorithm is an update to the original Sklansky '72 [37] algorithm which failed for intersecting complex polygons. Although the update corrected some failures, the algorithm still fails with certain complex polygons. It has been shown to be robust with star-shaped polygons, as which categorically a majority of the silhouettes are classified. Since we are not concerned with complex polygons in silhouettes, this is ideal for silhouette analysis and offers a computational complexity of $O(N)$. Sklansky’s algorithm scans counterclockwise to remove the concave vertex points, $p$, of the polygon, $P$. Sklansky[38, 37] defines $P$ as:

- the sequence of vertices of a polygon, or
- a bounded closed polygonal region.
Algorithm 2.3 Sklansky’s Algorithm ’72

1. Let \( P = \{p_0, p_1, p_2, \ldots, p_N\} \) be the counterclockwise set of vertices in an \( N \)-vertex polygon.

2. Define \( p_L, p_R, p_T, \) and, \( p_B \) as the leftmost, rightmost, top, and bottom vertices in \( P \). If \( p_L \) is used and supposing \( p_L = p'_k \) then:
   (a) let \( p_1 = p'_k \),
   (b) \( p_2 = p'_{k+1}, \ldots, p_N = p'_{m+K-1} \mod m \)*

3. Let \((x_i, y_i)\) be defined as the Cartesian coordinate of \( p_i \), \( x_i \) as the horizontal coordinate and \( y_i \) as the vertical coordinate.

4. Compute \( S_i = (x_{i+1} - x_{i-1})(y_{i-1} - y_i) + (y_{i+1} - y_{i-1})(x_i - x_{i-1}) \)
   (a) \( p_i \) is a convex vertex is \( S_i < 0 \).
   (b) If \( S_i \geq 0 \), \( p_i \) is removed from set \( \{P\} \), Progress to \( p_{i+1} \). When vertex is removed, regress back one vertex to check for a newly formed non convex vertex.

5. Process is halted when \( p_i \) is reentered.

6. Let \( M \) be defined as the final set of points after removal that form the hull, \( H \).
Algorithm 2.4 Sklansky’s Algorithm ’82

1. Let \( P = \{p_0, p_1, p_2, \ldots, p_N\} \) be the counterclockwise set of vertices in an \( N \)-vertex polygon.

2. Find \( p_L, p_R, p_T, \) and, \( p_B \) as the leftmost, rightmost, top, and bottom vertices in \( P \) and order them in a counterclockwise set.

3. Create a square (Figure 2.6) that circumscribes the four points. This creates four triangular regions \( R_1, R_2, R_3, \) and \( R_4 \) and a quadrilateral, \( Q \), where, \( p_L p_B, p_R p_B, p_R p_T, \) and \( p_T p_L \) are the hypotenuses of the triangles formed.

4. Let \( p_1 \) be defined as \( p_L \) and let \( p_2, p_3, \ldots, p_N \) be defined as the remaining vertices of \( P \), ordered in a counterclockwise set of \( P = \{p_1, p_2, \ldots, p_N\} \).

5. Let \( H \) be defined as convex hull of \( P \) and Let \((x_i, y_i)\) be defined as the Cartesian coordinate of \( p_i \).

6. Referring to Figure 2.6,

   (a) Start with subspace \( R_1 \), keep \( p_1 \) in \( P \), since \( p_1 \in H \)
   (b) If \( p_2 \notin R_1 \), delete \( p_2 \) from \( P \), otherwise keep \( p_2 \)
   (c) If \( p_3 \notin R_1 \), delete \( p_3 \) from \( P \)
   (d) If \( p_2 \in H \), form two point subregions seen in Figure 2.6
      i. Compute tangent \( t_{12} \) of the angle of the vector displacement \( \langle p_1, p_2 \rangle \)
      ii. Compute tangent \( t_{13} \) of the angle of the vector displacement \( \langle p_1, p_3 \rangle \)
      iii. If \( x_3 - x_2 \geq 0 \) and \( y_3 - y_2 \leq 0 \), retain \( p_3 \) in \( P \), otherwise remove \( p_3 \) from \( P \) if \( t_{12} < t_{13} \) else delete \( p_2 \) since \( t_{12} > t_{13} \)

7. Supposing \( p_i \) and \( p_{i+1} \) have been accepted in \( P \)

   (a) Check if \( p_{i+2} \) lies in \( R_1 \)
      i. If \( p_{i+2} \notin R_1 \), remove \( p_{i+2} \) from \( P \)
      ii. Else, compute tangent \( t_{i,i+1} \) of the angle of the vector displacement \( \langle p_i, p_{i+1} \rangle \) and tangent \( t_{i,i+2} \) of the angle of the vector displacement \( \langle p_i, p_{i+2} \rangle \)
      iii. If \( x_{i+2} - x_{i+1} \geq 0 \) and \( y_{i+2} - y_{i+1} < 0 \), retain \( p_{i+1} \) in \( P \), otherwise:
         A. If \( t_{i,i+1} < t_{i,i+2} \) or if \( t_{i,i+1} > t_{i,i+2}, y_{i+2} > y_{i+1} \), and vertex was removed on the preceding iteration of this step, then remove \( p_{i+2} \)
         B. If A does not apply and \( V_{i+2} \in \text{subregion } A_2 \)

8. Repeat steps 6 and 7 until \( p_B \) is reached. Steps 6 and 7 will remove the reflex points. Repeat for remaining regions of \( R_2, R_3, \) and \( R_4 \).

9. Polygon \( P^* \) is obtained.

10. Apply Algorithm 2.3 to \( P^* \).

11. Final polygon \( P_{F}^* \) is the convex hull of \( P \) and \( P^* \).
Therefore, \( \partial P \) is the boundary of the polygonal region, \( P \). The detailed algorithms are described in Algorithm 2.3.

Algorithm 2.3 fails on certain occasions when a self-intersecting polygon is created by removing vertices. This led to the reimplementation of the Algorithm 2.3 to Algorithm 2.4. Figure 2.6 shows Sklansky’s square:

![Figure 2.6: Sklansky’s square](image)

**2.2.2 Convex Hull Defect**

The silhouette hull defects are used to derive feature vectors for classification. The defect in a hull is the space between the contour line and the actual object. If an individual is facing the camera we can assume that the human body can be described by five defect triangles \((A, B, C, D, E)\). Therefore each frame is represented by five triangles with three coordinates called the defect start \((x_{ds}, y_{ds})\), defect end \((x_{de}, y_{de})\), and defect position points \((x_{dp}, y_{dp})\), labeled as 1, 2, and 3 respectively in Figure 2.7. Therefore defect triangle \( A \) can be described as the following vector, called \( V_{id}^A \):

\[
\left\langle A \, (x_{ds}) \ A \, (y_{ds}) \ A \, (x_{dp}) \ A \, (y_{dp}) \ A \, (x_{de}) \ A \, (y_{de}) \right\rangle
\]

In reality, since the human body is piecewise convex, there are more than five defects seen in Figure 2.7. We assume the human body can be described by the
Figure 2.7: Convex hull defects
five largest defect areas in a given frame. In general, the five defect triangles should correspond to actions the subject is performing.

To register the triangles between frame sequences we sort based on the defect position points \((x_{dp}, y_{dp})\) of the top five triangles and assign them labels \((A, B, C, D, E)\). Therefore the triangles are organized in a clockwise fashion. Accurate registration is important between the frames for further analysis.

### 2.2.3 Zero Buffer/Normalization

The convex hull produces more than five defect triangles in most silhouettes. Some of these triangles can be deemed insignificant, whereas with some actions (such as bend or jump) there are four or less triangles that represent an action. Both these problems can be remedied by using a zero buffer. So when the triangle registration algorithm is run incorrect matches are made and thus across a sequence the error become cumulative. An area threshold of pixels can be applied to remove these insignificant triangles from the defect triangle data. If an action has a triangle whose area is less than the threshold, \(\tau\), the defect points in the triangle is zeroed out. All values are replaced with a null set. The null set is also used as a buffer where fewer than four triangles are detected and the null set is added to the frame set. This ensures that each frame’s feature vector for a frame has thirty components.

Before the zero buffer is applied, the image is normalized so that the hull defects are in approximately same position in each frame. The defect points are arranged and remapped in another subspace set, \(D_N\) as described in Algorithm 2.5.

### 2.2.4 Triangle Registration

To improve triangle registration, an alternative registration method is proposed and implemented in Algorithm 2.6 and depicted in Figure 2.8. An initial frame is described
Algorithm 2.5 Defect Normalization

1. For a given image, let

\[
D = \{d_1, d_2, \ldots, d_{15}\}, \quad d_i \in \mathbb{N} \times \mathbb{N}
\]

be the set of defect point locations.

2. Let

\[
d_{\text{max}} = \left( \max_i d_{i,1}, \max_i d_{i,2} \right)
\]

for \(i = \{1, 2, \ldots, 15\}\).

3. Then for each \(d_i\), let

\[
f : \mathbb{N} \times \mathbb{N} \to [0, 1] \times [0, 1]
\]

such that

\[
f(x, y) = \left( \frac{x}{d_{\text{max},1}}, \frac{y}{d_{\text{max},2}} \right)
\]

4. Let

\[
D_N = \{f(d_1), f(d_2), \ldots, f(d_{15})\}
\]

be the set of normalized defect locations.

As \(F_i = \{Z_1, Z_2, Z_3, Z_4, Z_5\}\) where \(Z_i\) is the location of the \(i^{\text{th}}\) largest defect triangle. Defect triangles in the initial frame are labeled as \(L_i = \{A_i, B_i, C_i, D_i, E_i\}\) based on the preliminary sorting process described in Section 2.2.2. The frame descriptor \(F'_{i+1}\) is the set of the top seven defect triangle locations sorted by area, therefore \(F'_{i+1} = \{Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, Z_7\}\). The seven defect triangles in set \(F'_{i+1}\) are matched to the five triangles to \(L_i\) based on the distance measure. The five triangles in \(F'_{i+1}\) with the smallest distance measure from the points in \(L_i\) now make up the set \(F_{i+1}\) and are assigned as \(L_{i+1} = \{A_{i+1}, B_{i+1}, C_{i+1}, D_{i+1}, E_{i+1}\}\). This cycles through all the frames. This offers an equivalent measure of arranging solely based on defect location.
Algorithm 2.6 Triangle Registration
1. For a given initial frame, let
   \[ F_0 = \{ z_1, z_2, \ldots, z_5 \}, \quad z_i \in \mathbb{N}^6 \]
   be the set of defect triangle locations, where \( z_k = (x_1, y_1, x_2, y_2, x_3, y_3) \) describes the vertices of the triangle.

2. Let the defect triangles in the initial frame be labeled, based on their areas, as
   \[ L_0 = \{ A_i, B_i, C_i, D_i, E_i \} \]

3. Then for a new frame descriptor
   \[ F_i = \{ z'_1, z'_2, \ldots, z'_7 \} \]

4. For each \( z'_k \in F_i \) assign a label \( \ell \in L_{i-1} \) such that
   \[
   \ell = \arg \min_v v \in L_{i-1} \| z'_k - v \|
   \]
   where \( v_k \neq v_{k-1}, \ldots, v_1 \)

5. Repeat steps 3 to 4 for all \( i \)

---

Figure 2.8: Triangle Registration
2.2.5 Feature Vector Creation

The features we extract are composed of the defect start \((x_{ds}, y_{ds})\), defect end \((x_{de}, y_{de})\), and defect position points \((x_{dp}, y_{dp})\). These three locations for each of the five triangles in a frame of video are represented as a data distribution. Each frame of video capturing an action can thus be defined as a \(5 \times 6\) element array \((A \in \mathbb{R}^{5x6})\). The matrix is then concatenated into a 30 element vector, which captures the shape information. This vector encapsulates the locations of all the hull convexity triangles and can be used for classification.

\[
\begin{bmatrix}
A(x_{ds}) & A(y_{ds}) & A(x_{dp}) & A(y_{dp}) & A(x_{de}) & A(y_{de}) \\
B(x_{ds}) & B(y_{ds}) & B(x_{dp}) & B(y_{dp}) & B(x_{de}) & B(y_{de}) \\
C(x_{ds}) & C(y_{ds}) & C(x_{dp}) & C(y_{dp}) & C(x_{de}) & C(y_{de}) \\
D(x_{ds}) & D(y_{ds}) & D(x_{dp}) & D(y_{dp}) & D(x_{de}) & D(y_{de}) \\
E(x_{ds}) & E(y_{ds}) & E(x_{dp}) & E(y_{dp}) & E(x_{de}) & E(y_{de})
\end{bmatrix}
\Rightarrow
\langle A(x_{ds}), A(y_{ds}), A(x_{dp}), A(y_{dp}), A(x_{de}), A(y_{de}), \ldots \\
B(x_{ds}) \ldots C(x_{ds}) \ldots D(x_{ds}) \ldots E(x_{de}), E(y_{de}) \rangle
CHAPTER III

ACTION VECTOR CLASSIFICATION

3.1 Testing and Training Sets

The Weizman Database consists of nine individuals performing ten actions. Nine training and testing sets are used to evaluate our feature extraction method. We use a leave one out strategy where there will be nine training and testing sets in which each individual is left out in a given set to test the accuracy to the algorithm. The training sets are separated by action and given identity labels.

Suppose we have the following set of vectors for $M$ (where $M$ is the number of frames) observations for $n$-dimensional (where $n = 30$ features) samples, our training and testing sets are shown in Figure 3.1:

Three experimental setups are used to determine which is optimal for activity recognition. The testing and training sets will differ in each setup.

$$S_{testing} = \begin{bmatrix}
A_1(x_{ds}) & A_1(y_{ds}) & \ldots & B_1(x_{ds}) & \ldots & C_1(x_{ds}) & \ldots & D_1(x_{ds}) & \ldots & E_1(x_{de}) & E_1(y_{de}) \\
A_2(x_{ds}) & A_2(y_{ds}) & \ldots & B_2(x_{ds}) & \ldots & C_2(x_{ds}) & \ldots & D_2(x_{ds}) & \ldots & E_2(x_{de}) & E_2(y_{de}) \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
A_N(x_{ds}) & A_N(y_{ds}) & \ldots & B_N(x_{ds}) & \ldots & C_N(x_{ds}) & \ldots & D_N(x_{ds}) & \ldots & E_N(x_{de}) & E_N(y_{de})
\end{bmatrix}$$

$$S_{training} = \begin{bmatrix}
A_1(x_{ds}) & A_1(y_{ds}) & \ldots & B_1(x_{ds}) & \ldots & C_1(x_{ds}) & \ldots & D_1(x_{ds}) & \ldots & E_1(x_{de}) & E_1(y_{de}) \\
A_2(x_{ds}) & A_2(y_{ds}) & \ldots & B_2(x_{ds}) & \ldots & C_2(x_{ds}) & \ldots & D_2(x_{ds}) & \ldots & E_2(x_{de}) & E_2(y_{de}) \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
A_M(x_{ds}) & A_M(y_{ds}) & \ldots & B_M(x_{ds}) & \ldots & C_M(x_{ds}) & \ldots & D_M(x_{ds}) & \ldots & E_M(x_{de}) & E_M(y_{de})
\end{bmatrix}$$

Figure 3.1: Training and testing sets
As stated in subsection 1.4.1, PCA can cluster our feature vectors into clusters with similar patterns. This can be used to distinguish actions for recognition.

In the first setup, a training set, $S_{training}$, and testing set, $S_{testing}$ are used for PCA. The training set consists of frames from all ten actions we examine. The testing set is comprised of frames that are classified into actions and projected on the training set. This does not utilize the sequential information or frame order. Each frame stands alone as an identifier of an action. Frames of the testing set are projected onto the full PCA space of the training set. A testing point is determined to be part of an action based on a distance metric. The $k$-nearest neighbor algorithm is used for classification. The five nearest neighbors are found and a winner take all method is applied to determine frame identity. Figure 3.2 shows the projection of the training set to observe if the groups are clustered. We expect a distinct classification of certain actions and an overlap of actions that share the same pose. This foundation step is used as a basis to improve the algorithm.
3.3 Separate Activity Cluster

The second setup divides the frames and maintains the sequential temporal information. Each action of a training set is projected into its own subspace and clustered according to a k-means algorithm. The k-means algorithm uses a set of initial clusters that are created and each point is assigned to a cluster. The cluster centroid is deemed the mean point of the cluster. The assignment of clusters and recalculation of the mean are repeated in an iterative process until the solution converges. Therefore points are assigned due to the distance measurement to a cluster. There are ten action subspaces each with its own number of clusters as shown in Table 3.1. The minimum distance from each cluster for the points in a sequence is summed to determine which is the best cluster match for a point. A sequence is deemed a member of a subspace if it has the smallest distance measure. The separate activity clusters for each action are shown in Figure 3.4.

3.3.1 K-Nearest Neighbor Classifications

When projecting the testing set into the subspace created with the training set, K-nearest neighbor is used to decide to which action cluster a frame belongs. The training points are given IDs based on their action. This is shown in Figure 3.3

K-nearest neighbor relies on calculating the distance function between two points, where a metric is the function that gives the distance between the points.

The Euclidean distance for $d$ dimensions is defined as:

$$ D(\mathbf{a}, \mathbf{b}) = \left( \sum_{k=1}^{d} (a_k - b_k)^2 \right)^{1/2} $$

(3.1)

We use the Euclidean distance measure.
3.4 Minimally Encoded Neural Networks for Action Recognition

The final evaluation methodology utilizes neural networks. Multiple configurations are considered for action recognition. The first configuration uses one neural network which targets 10 IDs that are equivalent to the ten actions. The second configuration accounts for all clusters described in Table 3.1.

3.4.1 Bilateral Symmetry Metric BSM

The next setup uses bilateral symmetry of an image to separate the actions based on whether the action is captured with respect to sagittal or coronal plane of the individual. A bilateral difference image $I_B$ is calculated by finding the widest distance between two white pixels in the bottom 40% of the frame. The midpoint of the distance is found and used to slice the image in half. Figure 3.5a and 3.5e show the splicing effect of Algorithm 3.1. The two images, $I_L$ (3.5b and 3.5f) and $I_R$ (3.5c and 3.5g) are respectively the right and left images. $I_R$ is mirrored about the
Figure 3.4: Clusters
vertical line and the bilateral difference image, $I_B$ (3.5d and 3.5h) is calculated. The symmetry metric, $\Lambda$, is the percentage of white pixels compared to the total area of the image. Actions a4 (jumping jack) and a10 (bend) are used to illustrate the bilateral symmetry metric. The average symmetry metric is an identifier which is represented by a threshold that can be applied to the images. Coronal perspective images tend to have a low symmetry metric whereas sagittal perspective images do not.

The mean and the variance of these values are used to determine whether an image sequence is bilaterally symmetric. Table3.2 shows the values for each action. If it is bilaterally symmetric the sequence of frames is given an additional label.

The second set of experiments use the bilateral symmetry parameters in conjunction with two neural networks based on the camera perspective.
Algorithm 3.1 Bilateral Symmetry

1. For each image $I \in \{0, 1\}^{m \times n}$, calculate the bilateral difference image $I_B$

2. For each row, $\nu_i$ where $0.6 \cdot m < i \leq m$, find column indices $l_i$ and $r_i$ where

\[ c_{l_i} = 1 | \forall k < l_i, c_k = 0 \]

and

\[ c_{r_i} = 1 | \forall k > r_i, c_k = 0. \]

(a) Let row width $w_i = r_i - l_i$

3. Let silhouette center

\[ \zeta = \left( \max_i w_i \right) \frac{2}{2} + l_i \]

4. Let left side image $I_L \subset I$ be the left side of the image about column $c_\zeta$

5. Let right side image $I_R \subset I$ be the right side of the image about column $c_\zeta$ such that

\[ I_R(x, y) = I(n - x, y) \]

6. Modify $I_L$ or $I_R$ by adding zero-value elements such that $|I_L| = |I_R|$.

7. Let $I_B = I_L - I_R$

8. Calculate symmetry metric

\[ \Lambda = \frac{\sum I_B(x, y)}{mn} \]

where $I_B \in \{0, 1\}^{m \times n}$. 
### Table 3.2: Bilateral symmetry

<table>
<thead>
<tr>
<th>ACTIONS</th>
<th>μ</th>
<th>σ²</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SAGITTAL PERSPECTIVE ACTIONS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WALK (a1)</td>
<td>0.366</td>
<td>0.0159</td>
</tr>
<tr>
<td>RUN (a2)</td>
<td>0.451</td>
<td>0.00984</td>
</tr>
<tr>
<td>SKIP (a3)</td>
<td>0.437</td>
<td>0.00684</td>
</tr>
<tr>
<td>JUMP (a5)</td>
<td>0.455</td>
<td>0.00888</td>
</tr>
<tr>
<td>BEND (a10)</td>
<td>0.328</td>
<td>0.00433</td>
</tr>
<tr>
<td><strong>CORONAL PERSPECTIVE ACTIONS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JACK (a4)</td>
<td>0.139</td>
<td>0.00314</td>
</tr>
<tr>
<td>PJUMP (a6)</td>
<td>0.215</td>
<td>0.00244</td>
</tr>
<tr>
<td>SIDE (a7)</td>
<td>0.351</td>
<td>0.0261*</td>
</tr>
<tr>
<td>WAVE 1 (a8)</td>
<td>0.178</td>
<td>0.00395</td>
</tr>
<tr>
<td>WAVE 2 (a9)</td>
<td>0.134</td>
<td>0.00173</td>
</tr>
</tbody>
</table>
CHAPTER IV

EXPERIMENTAL RESULTS AND DISCUSSION

In this chapter we evaluate the proposed hull convexity defect algorithm for human activity recognition. The results sections will cover several experimental setups for the action recognition system; each setup builds on the results of the previous, leading to a final system analysis. Our goal is to test different configurations of the system so as to reach an ideal parametrization of the classification.

**Clustering:** Action clustering in one subspace is examined to compare our frame-by-frame analysis to other methodologies. Clustering also plays a role when establishing the training of the neural network described in subsequent sections.

**Bilateral Symmetry Metric:** The Bilateral Symmetry Metric described in Section 3.4.1 forms an essential preprocessing step, as well as a parameters associated with our feature vectors. It serves as an accurate initial coarse level classifier.

**Neural Networks:** Three neural network configurations can considered and used to train the frames. In combination with cluster assignment, the frames are given a numerical identifier. This identifier is dependent on the number of clusters used to train the neural network systems. They differ in their use of the bilateral symmetry
metric mentioned above. A hybrid neural network is also introduced as an alternative system.

4.1 Weizman Database

To test the proposed algorithm, we consider the Weizman database, shown in Table 4.1. It consists of nine test subjects performing ten actions (bend, hop, run, side jump, one-handed wave, two-handed wave, walk, skip, pogo jump, and jumping jacks) in an outdoor environment. For a majority of the videos, background images are provided for each individual-action combination along with masks of the actions. The Hull Convexity algorithm uses an original background segmentation method, discussed in subsequent sections instead of the provided masks. Figure 2.1 illustrates examples of each action. In this database each frame contains single individuals in which the test subjects perform actions either in the sagittal plane or in the coronal plane. Where, the coronal plane actions are ones the subject is facing the camera and sagittal plane actions are ones the subject is profile to the camera. In all cases camera is stationary and the video provided has a frame rate of 25 frames/sec with a resolution of 180 × 144; this is a relatively low quality video. This limited spatial resolution creates a number of challenges including more difficult discrimination among features and the amplification of background noise.

The Weizman method uses a statistical distance measure between video frames to cluster and classify events as well as space-time volumes. Both these methods are applied to the Weizman Database and the confusion matrices for both are shown in Table 4.2. The confusion matrix results are calculated frame-by-frame. The sequence analysis is completed on all actions except for skip. Weizman reports 95.5% accuracy without including skip, a3. The evaluation method uses a sliding window, as well as a subsequence in certain instances.
<table>
<thead>
<tr>
<th><strong>WEIZMAN DATABASE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SUBJECTS</strong></td>
</tr>
<tr>
<td><strong>ACTIONS</strong></td>
</tr>
<tr>
<td><strong>LOCATION</strong></td>
</tr>
<tr>
<td><strong>CAMERA</strong></td>
</tr>
<tr>
<td><strong>FRAME RATE</strong></td>
</tr>
<tr>
<td><strong>RESOLUTION</strong></td>
</tr>
</tbody>
</table>

Table 4.1: Weizman

4.2 Experimental Setup

For Hull Convexity algorithm analysis, we broadly classify test subjects as performing actions either in a sagittal or coronal plane. The methodology in [2] utilizes subsequences and sliding windows to achieve invariance to action duration. Our novel approach uses complete sequences of varying lengths, and an effective resolution of 110 × 250.

4.3 Frame-by-Frame Analysis

4.3.1 Base Results

To determine how actions are similar we need to understand the manner in which they may be classified. To begin with we examine a system where actions are labeled by frame. A sequence of actions such as 'bend', a10, are labeled from the 'standing' position to 'bending' to 'standing' once again as one identifier. The methods described in [2] use a subset of actions frames wherein a sliding window is used to pick out a subset of frames from an action. In our experiments, we test to see if actions can be recognized strictly on a frame-by-frame basis, with no sequential or temporal information.

The Weizman database has nine test subjects; we created nine testing and nine
training sets, using a leave-one-out method. In each case the training set has approximately 2000 images and the testing set has approximately 200 images. The training data points in the PCA space are given an ID based on class membership, and for a given probe frame we use a nearest neighbor method to determine the five IDs to which the frame is closest. We use a simple five-nearest neighbor method to determine the five IDs to which a frame is closest. A match is established using the frequency of occurrence of a frame ID. If a 'tie' occurs, the cumulative distance of the frames to one another is used as a tie breaker. Using this method we are able to understand if frames share a subset or have similar properties.

The Convexity Hull algorithm correctly classified the actions as shown in Table 4.2(c). Note that the actions labeled as translational actions have a lower performance classification than the stationary actions. The same is true for the sagittal perspective, which incorporates a majority of the translational actions.

In a frame by frame analysis, the Hull Convexity algorithm performed better than the Zelnik-Manor method for all actions except jumping jack (a4). Compared to the Blank method [2], our algorithm classified the action one-handed wave (a8), and two handed wave (a9), at a higher rate. We also observe that actions which share similar substates are often misclassified with one another. In this way we can determine that sequential information is useful in determining an action recognition decision.

When analyzing individual frames, we can compare the temporal relationships between detected features in translational cyclic actions and stationary actions. The stationary actions of bend, jumping jack, pjump, one-handed wave and two-handed wave are consistently grouped into separable clusters. Figure 4.1 shows the result of plotting the principal components $\alpha_1$ and $\alpha_2$ of the detected features of these actions. While there are distinct separations, nevertheless some actions are grouped into multiple clusters. This can be attributed to the sub-states that together make
<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a2</td>
<td>0.0</td>
<td>0.98</td>
<td>0.02</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a3</td>
<td>0.0</td>
<td>0.029</td>
<td>0.971</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a4</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a5</td>
<td>0.0</td>
<td>0.108</td>
<td>0.0</td>
<td>0.892</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.009</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a10</td>
<td>0.0</td>
<td>0.0</td>
<td>0.009</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

(a) Blank et al. [2]

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>0.824</td>
<td>0.008</td>
<td>0.024</td>
<td>0.0</td>
<td>0.144</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a2</td>
<td>0.02</td>
<td>0.347</td>
<td>0.51</td>
<td>0.0</td>
<td>0.102</td>
<td>0.0</td>
<td>0.02</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a3</td>
<td>0.014</td>
<td>0.406</td>
<td>0.435</td>
<td>0.0</td>
<td>0.087</td>
<td>0.0</td>
<td>0.058</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a4</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.955</td>
<td>0.0</td>
<td>0.008</td>
<td>0.0</td>
<td>0.008</td>
<td>0.015</td>
<td>0.015</td>
</tr>
<tr>
<td>a5</td>
<td>0.138</td>
<td>0.138</td>
<td>0.262</td>
<td>0.0</td>
<td>0.292</td>
<td>0.0</td>
<td>0.169</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.128</td>
<td>0.0</td>
<td>0.849</td>
<td>0.0</td>
<td>0.0</td>
<td>0.012</td>
<td>0.012</td>
<td>0.0</td>
</tr>
<tr>
<td>a7</td>
<td>0.032</td>
<td>0.048</td>
<td>0.238</td>
<td>0.0</td>
<td>0.175</td>
<td>0.0</td>
<td>0.508</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.009</td>
<td>0.0</td>
<td>0.07</td>
<td>0.0</td>
<td>0.296</td>
<td>0.626</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.009</td>
<td>0.0</td>
<td>0.009</td>
<td>0.0</td>
<td>0.443</td>
<td>0.579</td>
<td>0.190</td>
<td>0.0</td>
</tr>
<tr>
<td>a10</td>
<td>0.0</td>
<td>0.0</td>
<td>0.045</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.036</td>
<td>0.054</td>
<td>0.866</td>
<td>0.0</td>
</tr>
</tbody>
</table>

(b) Zelnik-Manor et al. [2]

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>0.85</td>
<td>0.01</td>
<td>0.03</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
</tr>
<tr>
<td>a2</td>
<td>0.07</td>
<td>0.55</td>
<td>0.25</td>
<td>0.0</td>
<td>0.06</td>
<td>0.0</td>
<td>0.04</td>
<td>0.0</td>
<td>0.02</td>
<td>0.01</td>
</tr>
<tr>
<td>a3</td>
<td>0.04</td>
<td>0.15</td>
<td>0.51</td>
<td>0.02</td>
<td>0.23</td>
<td>0.01</td>
<td>0.0</td>
<td>0.03</td>
<td>0.01</td>
<td>0.0</td>
</tr>
<tr>
<td>a4</td>
<td>0.07</td>
<td>0.0</td>
<td>0.02</td>
<td>0.72</td>
<td>0.02</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.17</td>
</tr>
<tr>
<td>a5</td>
<td>0.02</td>
<td>0.0</td>
<td>0.08</td>
<td>0.01</td>
<td>0.76</td>
<td>0.06</td>
<td>0.0</td>
<td>0.03</td>
<td>0.04</td>
<td>0.0</td>
</tr>
<tr>
<td>a6</td>
<td>0.01</td>
<td>0.0</td>
<td>0.0</td>
<td>0.02</td>
<td>0.93</td>
<td>0.0</td>
<td>0.02</td>
<td>0.02</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a7</td>
<td>0.06</td>
<td>0.07</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.02</td>
<td>0.73</td>
<td>0.01</td>
<td>0.04</td>
</tr>
<tr>
<td>a8</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.01</td>
<td>0.0</td>
<td>0.0</td>
<td>0.98</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a9</td>
<td>0.02</td>
<td>0.0</td>
<td>0.0</td>
<td>0.02</td>
<td>0.05</td>
<td>0.0</td>
<td>0.09</td>
<td>0.81</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>a10</td>
<td>0.0</td>
<td>0.0</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.97</td>
<td>0.0</td>
</tr>
</tbody>
</table>

(c) Convexity Hull Algorithm

Table 4.2: Confusion matrices
up each action. The action cluster exhibits further separability when plotted along principal component $\alpha_3$ as shown Figure 4.1b.

The translational cyclic actions of jump, run, side, skip, and walk are more difficult to classify. Since these actions share similar sub-states, they tend to overlap in the PCA space. Run is misclassified as skip for 25% of the frames, skip with jump for 23% of the frames, and skip with run for 15% of the frames with the Convexity Hull method- whereas the Zelnik-Manor method incorrectly classifies run (a2) and skip (a3) for a majority of the time.

### 4.4 Multiple Action Subspace

In this section we show that establishing a separate subspace for each action can improve recognition performance. To verify this hypothesis we take the known actions in the Weizman database and project them in their own subspace. The testing set now consists of a sequence of a single action and the training set as eight sequences of said actions projected and clustered in a space (Figure 3.4). Each point in a test action is projected to the space, and again the five nearest neighbor averages are calculated and used as a distance measure and the average distance per frame for a sequence is found. The same sequence is projected over the remaining nine actions. The minimum distance metric of the sequence on the set of actions is considered a match.

Figure 4.3 illustrates the sequence of bend (a10), projected in the bend (a10) (Figure 4.3a), walk (a1) (Figure 4.3b), and wave2 (a9) (Figure 4.3c), subspaces. It appears that in 2 principal components space both bend (a10) and wave2 (a9) are a match. When looking at the third principal component, the bend sequence on the walk subspace is further separable. In addition, when examining the similarity matrix, Table 4.4, the average distance of bend (a10) in the wave2 (a9) subspace is
Figure 4.1: Actions in subspace
Figure 4.2: Bilateral actions in PCA subspace

5.69, whereas when bend is projected in its own subspace the average distance per frame is 3.02.

We note several trends shown numerically in Table 4.4 and graphically in Figure 4.4. First, in the similarity matrix, actions walk (a1), and run (a2), are distinguishable but share a similar PCA space. When walk (a1) is projected into the run (a2) space, the average distance is 2.01, whereas walk (a1) is 1.96 (less than a 3% difference); hence the two actions are readily mistaken for one another. In testing set one, the sequence of run is indeed mistaken for walk (Table 4.3).

We observe that an additional metric can be added to the feature vector to further distinguish between these actions. This leads us to consider a bilateral symmetry metric which may be added to the feature vector or used as a preprocessing step—or a hybrid of both.

4.5 Neural Networks for Human Activity Recognition

The third classification mode for the final system uses a neural network to train approximately 2000 feature vectors and identify them as specific actions. The neural network setup is shown in Table 4.5. Each training vector is assigned a 9-dimensional or 27-dimensional identity label (depending on the modality of the system). All
Figure 4.3: Bend projections

Figure 4.4: Similarity matrix
### Table 4.3: Confusion matrix training set one

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a2</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a4</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a5</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a6</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a7</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a8</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a9</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a10</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 4.4: Similarity matrix

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>2.97</td>
<td>4.14</td>
<td>-</td>
<td>4.43</td>
<td>5.01</td>
<td>5.70</td>
<td>5.13</td>
<td>6.09</td>
<td>5.46</td>
<td>5.31</td>
</tr>
<tr>
<td>a2</td>
<td>2.01</td>
<td>1.96</td>
<td>-</td>
<td>2.91</td>
<td>2.60</td>
<td>3.17</td>
<td>2.99</td>
<td>3.57</td>
<td>3.26</td>
<td>2.97</td>
</tr>
<tr>
<td>a3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a4</td>
<td>4.19</td>
<td>5.05</td>
<td>-</td>
<td>2.85</td>
<td>5.14</td>
<td>5.31</td>
<td>4.71</td>
<td>5.52</td>
<td>4.95</td>
<td>4.86</td>
</tr>
<tr>
<td>a5</td>
<td>1.98</td>
<td>2.50</td>
<td>-</td>
<td>2.43</td>
<td>1.64</td>
<td>2.04</td>
<td>2.49</td>
<td>2.51</td>
<td>2.38</td>
<td>2.43</td>
</tr>
<tr>
<td>a6</td>
<td>2.98</td>
<td>4.03</td>
<td>-</td>
<td>2.15</td>
<td>2.16</td>
<td>1.27</td>
<td>1.85</td>
<td>2.38</td>
<td>2.79</td>
<td>2.15</td>
</tr>
<tr>
<td>a7</td>
<td>2.32</td>
<td>2.7</td>
<td>-</td>
<td>2.03</td>
<td>3.78</td>
<td>3.31</td>
<td>1.40</td>
<td>3.42</td>
<td>4.06</td>
<td>2.33</td>
</tr>
<tr>
<td>a8</td>
<td>3.65</td>
<td>4.56</td>
<td>-</td>
<td>4.04</td>
<td>3.55</td>
<td>4.40</td>
<td>4.62</td>
<td>1.54</td>
<td>3.62</td>
<td>2.42</td>
</tr>
<tr>
<td>a9</td>
<td>4.07</td>
<td>5.11</td>
<td>-</td>
<td>3.31</td>
<td>3.68</td>
<td>4.05</td>
<td>4.96</td>
<td>4.10</td>
<td>1.92</td>
<td>4.40</td>
</tr>
<tr>
<td>a10</td>
<td>5.33</td>
<td>6.12</td>
<td>-</td>
<td>4.65</td>
<td>5.57</td>
<td>5.55</td>
<td>5.13</td>
<td>4.73</td>
<td>5.69</td>
<td>3.02</td>
</tr>
</tbody>
</table>

**WALK (a1)  RUN (a2)  SKIP (a3)  JACK (a4)  JUMP (a5)  PJUMP (a6)  SIDE (a7)  WAVE 1 (a8)  WAVE 2 (a9)  BEND (a10)**
actions, with the exception of skip (a3) are inputs to the neural network. A correct match is determined based on maximum occurrence of frame membership within a sequence. We first test a system with 9 clusters (one per action) and without the BSM. Then in subsequent tests the BSM is added as a feature in our feature vector. The mean, median and mode of the BSM across the sequence are the 31st-33rd features. The next step is to use the BSM as a preprocessing step to divide the actions into a sagittal perspective group (a1,a2,a5, and a10) and a coronal perspective group (a4,a6,a7,a8,a9). The fourth configuration uses the BSM as both a feature component and a preprocessing step. Finally a hybrid technique is attempted in which BSM is used as a preprocessor to divide the actions where the sagittal perspective group uses the BSM as a feature component and the coronal perspective group maintains its original feature vector. Our neural network can then be tuned with the following parameters:

- Number of Clusters
- Bilateral Symmetry Metric as a feature
- Bilateral Symmetry Metric as preprocessing step

<table>
<thead>
<tr>
<th>Modality</th>
<th>Clusters</th>
<th>Neural Networks</th>
<th>BSM as a Feature</th>
<th>BSM as a Preprocessing Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>2</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>2</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>2</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>2</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>2</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>25</td>
<td>2</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 4.5: Neural network setup
• Hybrid of BSM as a feature and BSM as a preprocessing step

4.5.1 Neural Network Modalities

Modality One: N=9 clusters, Bilateral Metric, 1NN: The first modality has nine clusters assigned to each action. Table 4.6 shows the confusion matrix for this method. We note that run, (a2) is misclassified as walk (a1) for \( \frac{3}{9} \) sequences and walk (a1) is misclassified as jack (a4), and side (a7). Introducing the BSM as a feature correctly classifies \( \frac{9}{9} \) times. It also increases the correct identification rate for a1, a2, and a10. We conclude introducing the BSM as a feature component improves classification rate.

Modality Two: N=9 clusters, Bilateral Metric, 2NN: The second modality again has nine clusters assigned to each action. The actions are now separated...
<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>0.777</td>
<td>0</td>
<td>-</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a2</td>
<td>0.333</td>
<td>0.555</td>
<td>-</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a4</td>
<td>0.111</td>
<td>0</td>
<td>-</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a5</td>
<td>0</td>
<td>0.111</td>
<td>-</td>
<td>0</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a6</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0.111</td>
<td>0</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a7</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a8</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.888</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a9</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a10</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0.111</td>
<td>0.111</td>
<td>0.111</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0.666</td>
</tr>
</tbody>
</table>

(a) Without bilateral symmetric

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a2</td>
<td>0.111</td>
<td>0.666</td>
<td>-</td>
<td>0</td>
<td>0.222</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a4</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a5</td>
<td>0.111</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a6</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a7</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a8</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a9</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a10</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0.888</td>
</tr>
</tbody>
</table>

(b) With Bilateral symmetric

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a2</td>
<td>0.111</td>
<td>0.666</td>
<td>-</td>
<td>0</td>
<td>0.222</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a4</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a5</td>
<td>0.111</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a6</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a7</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a8</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a9</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a10</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0.888</td>
</tr>
</tbody>
</table>

Table 4.6: Confusion matrix 1st modality
based on their perspective to the camera. Table 4.6 shows the confusion matrix for the system. We observe that again run (a2) is misclassified as walk (a1) for $\frac{3}{9}$ sequences and walk (a1) is misclassified as run (a2) (Table 4.7a). Introducing the BSM as preprocessing step a feature correctly classifies the input data $\frac{8}{9}$ times. This also increases the correct identification rate for a1, a4, and a7 (Table 4.7b). The classification rate for run (a2) remains the same; occurrence of misclassification is distributed to other actions. We conclude that introducing the BSM as a preprocessing component aids in the identification of actions that are in different perspective planes that are often mistaken for one another. This leads us to consider a hybrid system that takes BSM as a preprocessor and also as a feature for certain actions. Figure 4.5 shows the flow for a hybrid system for both cluster layouts.

**Modality Three: N=9 clusters, Bilateral Metric, 2NN, Hybrid:** The hybrid system allows us to use the strong feature specific to each action. For example, the BSM feature helps distinguish between the actions performed in the coronal perspective, whereas the BSM as a preprocessor helps distinguish the sagittal actions. By allowing actions to be classified coarsely as such, and then including feature components in the feature vector for certain action, we can improve overall classification performance. Figure 4.6 shows the performance of the three systems described above. This new configuration can accurately identify $\frac{6}{9}$ actions correctly 100% of the time. The remaining actions of walk (a1), run (a2), and jump (a3) still have performance issues. We suggest that this is due to the shared substates among these actions. We further posit that this deficiency may be remedied with a more advanced action labeling mechanism. From this we conclude that several actions require further subclassification. This leads us to the new modality in which these changes are incorporated into the learning process of the neural network.
<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>0.777</td>
<td>0.222</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a2</td>
<td>0.333</td>
<td>0.666</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a4</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a5</td>
<td>0</td>
<td>0.111</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a6</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a7</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a8</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a9</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a10</td>
<td>0.111</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.888</td>
</tr>
</tbody>
</table>

(a) Without bilateral symmetric

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>0.888</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a2</td>
<td>0.111</td>
<td>0.666</td>
<td>-</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a4</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a5</td>
<td>0.222</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0.666</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a6</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a7</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a8</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a9</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a10</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.888</td>
</tr>
</tbody>
</table>

(b) With bilateral symmetric

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>0.888</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a2</td>
<td>0.111</td>
<td>0.666</td>
<td>-</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a4</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a5</td>
<td>0.222</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0.666</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a6</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a7</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a8</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a9</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a10</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.888</td>
</tr>
</tbody>
</table>

Table 4.7: Confusion matrix 2nd modality
Modality Four: N=25 clusters, Bilateral Metric, 1NN: This modality incorporates the use of the subclusters shown in Figure 3.4, giving us twenty-five subclusters that inherently describe a substate of an action. Again, we begin by putting all actions through a single neural network without using the BSM to set a basis for our improvements. Given BSM as a feature, we observe that the actions a1, a2, a4, and a7 perform better than the original neural network. However, we see a decrease in performance for jump (a5).

Modality Five: N=25 clusters, Bilateral Metric, 2NN: The fifth modality also has twenty-five clusters assigned to each subaction, each of which is now separated based on its perspective to the camera. This method increases the correct identification rate for walk (a1). (Table 4.10) The classification rate for run (a5) and bend (a10) are distributed to other actions. We conclude that introducing the BSM as a preprocessing component aids in the identification of run, but not other actions.
In this way, the BSM as a preprocessor alone may not be sufficient.

**Modality Six: N=25 clusters, Bilateral Metric, 2NN, Hybrid:** The sixth modality combines the use of the BSM as a preprocessor and as a feature component. We note that actions in the coronal perspective benefit from using the BSM as a feature whereas actions in the sagittal plane do not. Combining both systems into a hybrid system, we see that again the hybrid system allows us to use features specific to each action.

This new configuration can accurately identify $7/9$ of the actions correctly 100% of the time, whereas the actions of walk (a1) and run (a2) still have performance issues. We again assume this is due to overlapping substates for these actions; these clusters may need to be regrouped and assigned a commonality measure. We conclude from these results that these actions are accurately identifiable using the convex hull algorithm and a hybrid neural network.
### Table 4.9: Confusion Matrix 4th Modality

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>0.666</td>
<td>0.111</td>
<td>-</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a2</td>
<td>0.333</td>
<td>0.555</td>
<td>-</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a4</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
</tr>
<tr>
<td>a5</td>
<td>0</td>
<td>0.111</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a6</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0.111</td>
<td>0</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a7</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a8</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a9</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a10</td>
<td>0.111</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
<td>0.111</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a2</td>
<td>0.222</td>
<td>0.666</td>
<td>-</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a4</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a5</td>
<td>0.111</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.777</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a6</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0.888</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a7</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a8</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a9</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a10</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0.111</td>
<td>0.111</td>
<td>0.111</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0.666</td>
</tr>
</tbody>
</table>

(b) With Bilateral Symmetric

Walk (a1) Run (a2) Skip (a3) Jack (a4) Jump (a5) PJump (a6) Side (a7) Wave 1 (a8) Wave 2 (a9) Bend (a10)
<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>0.666</td>
<td>0.333</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a2</td>
<td>0.222</td>
<td>0.777</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a10</td>
<td>0.111</td>
<td>0.222</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.666</td>
</tr>
</tbody>
</table>

(a) Without bilateral symmetric

<table>
<thead>
<tr>
<th></th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>a5</th>
<th>a6</th>
<th>a7</th>
<th>a8</th>
<th>a9</th>
<th>a10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a2</td>
<td>0.111</td>
<td>0.777</td>
<td>-</td>
<td>0</td>
<td>0.111</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>a4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a5</td>
<td>0.111</td>
<td>0.111</td>
<td>-</td>
<td>0</td>
<td>0.666</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.111</td>
</tr>
<tr>
<td>a6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a10</td>
<td>0.111</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.222</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.666</td>
</tr>
</tbody>
</table>

(b) With bilateral symmetric

Walk (a1) Run (a2) Skip (a3) Jack (a4) Jump (a5) PJump (a6) Side (a7) Wave 1 (a8) Wave 2 (a9) Bend (a10)

Table 4.10: Confusion matrix 5th modality
Figure 4.7: Cluster of \( n=25 \) for neural network

Table 4.11: Hybrid data \( N=25 \) clusters
4.5.2 Inclusion of Skip

The authors of [2] present the results of their algorithm with the removal of skip, and indicate 95.4% accuracy. They categorize the “natural” action of skip as a combination of walk, run, and jump. Including skip in the analysis reduces the accuracy across most actions. The results (including skip) in a ten cluster, one-step neural network are shown in Table 4.12. The actions, run, (a2) jump (a5), and bend, (a10) are all misclassified as skip. In turn skip (a3) is confused for walk (a1), run (a2), jack (a4), and jump (a5).

4.6 Discussion

We tested our novel approach through a series of three experimental setups utilizing clustering, the bilateral symmetry metric and neural networks. Our algorithm works with full sequences of varying length and multiple cycles of an action. The training system does not use a subsequence of frames; all frames are used for training and
<table>
<thead>
<tr>
<th>Approach</th>
<th>Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convexity Defects</td>
<td>97.5</td>
</tr>
<tr>
<td>Sun et al. [40] SIFT + ZNK</td>
<td>97.8</td>
</tr>
<tr>
<td>Sun et al. [40] 2D + 3D SIFT</td>
<td>90.3</td>
</tr>
<tr>
<td>Sun et al. [40] FRM ZNK + MEI ZNK</td>
<td>94.6</td>
</tr>
<tr>
<td>Schindler and Gool [32]</td>
<td>100.0</td>
</tr>
<tr>
<td>Fathi and Mori [16]</td>
<td>100.0</td>
</tr>
<tr>
<td>Weinland and Boyer [45]</td>
<td>100.0</td>
</tr>
<tr>
<td>Gorelick et al. [7]</td>
<td>99.6</td>
</tr>
<tr>
<td>Jhuang et al. [20]</td>
<td>98.8</td>
</tr>
<tr>
<td>Wang et al. [44]</td>
<td>97.8</td>
</tr>
<tr>
<td>Thurau and Hlavac [42]</td>
<td>94.4</td>
</tr>
<tr>
<td>Jia and Yeung [21]</td>
<td>90.9</td>
</tr>
<tr>
<td>Liu et al. [26]</td>
<td>89.3</td>
</tr>
<tr>
<td>Klaser et al. [22]</td>
<td>84.3</td>
</tr>
</tbody>
</table>

Table 4.13: Performance comparisons [40]

testing a system. Our frame-by-frame analysis, not using a sliding window, yields comparable results to [2]. The secondary clustering approach of separate subspaces yields 7/81 sequences being misclassified for an overall 92% accuracy of the algorithm. The third system which includes a hybrid neural network yields a final result of 2/81 or 97.5% accuracy. A comparison of the results and other methods is shown in Table 4.13.

The complexity of our program is on the order of $O(n \log n)$. We conclude that the neural network with bilateral symmetry method is an accurate and fast algorithm with demonstrably less complexity than similar systems.
CHAPTER V

CONCLUSION AND FUTURE WORK

Human activity recognition is a rapidly developing field in computer vision. Accurate algorithmic modeling of action recognition can introduce a multitude of challenges. Computer vision and pattern recognition algorithms can aid in action identification. In recent years research has been emphasized in recognizing action using features extracted from more complex actions. Simple cases of action recognition of one individual executing an action are the foundation for developing these complex scenario in real environments. This can be especially useful for surveillance of public locations such as subways, shopping centers, or parking lots to reduce crime, monitor traffic flow, and offer security in general.

We developed a taxonomic shape driven algorithm to solve the problem of human action recognition and a new feature extraction technique using hull convexity defects. To test and validate this approach, we used silhouettes of subjects performing ten actions from a commonly used video database by action recognition researchers. A morphological algorithm has been used to filter noise from the silhouette. A convex hull is then created around the silhouette frame, from which convex defects are used as the features for analysis. A complete feature consists of thirty individual values which represent the five largest convex hull defects areas. A consecutive sequence of these features form a complete action. Action frame sequences are preprocessed to separate
the data into two sets based on perspective planes and bilateral symmetry. Features are then normalized to create a final set of action sequences. We then formulated and investigated three methods to classify ten actions from the database. Training and testing of the nine test subjects was performed using a leave one out strategy. Classification utilized both PCA and minimally encoded neural networks. Performance evaluation results show that the Hull Convexity Defect Algorithm provides comparable results with less computational complexity. The algorithm was designed to address the following issues in actions recognition: image noise, size invariance, spatial invariance, registration, and temporal components.

This research can lead to the creation of an action recognition system method capable of distinguishing complex actions and multiple person interaction. It is computationally inexpensive and consistently accurate. More specifically it can be used as a foundational step for further activity recognition research.
LIST OF PUBLICATIONS


