Human activity recognition using limb component extraction

Jamie Boeheim

Follow this and additional works at: https://scholarworks.rit.edu/theses

Recommended Citation

This Thesis is brought to you for free and open access by RIT Scholar Works. It has been accepted for inclusion in Theses by an authorized administrator of RIT Scholar Works. For more information, please contact ritscholarworks@rit.edu.
Human Activity Recognition Using Limb Component Extraction

by

Jamie Lynn Boeheim

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering

Supervised by

Dr. Andreas Savakis
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, NY
March 2008

Approved By:

Dr. Andreas Savakis, Professor and Department Head
Primary Advisor – R.I.T. Dept. of Computer Engineering

Dr. Juan Cockburn, Associate Professor
Secondary Advisor – R.I.T. Dept. of Computer Engineering

Dr. Muhammad Shaaban, Associate Professor
Secondary Advisor – R.I.T. Dept. of Computer Engineering
Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title: Human Activity Recognition Using Limb Component Extraction

I, Jamie Lynn Boeheim, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

_________________________________
Jamie Lynn Boeheim

_________________________________
Date
Dedication

To my family who got me here, and to Patrick who kept me here.
Acknowledgements

A special thanks to everyone who helped with this project, especially my advisor Dr. Andreas Savakis, the Image Understanding research group, and my committee members Dr. Juan Cockburn and Dr. Muhammad Shaaban.
Abstract

Interest in the field of human activity recognition has existed for quite sometime, but has gained popularity in recent years for use in many areas of application. In the security industry, suspicious activities could be detected in high-profile areas. In the medical industry, systems could be trained to detect patterns of motion indicating distress or to detect a lack of motion if a person had fallen and was unable to move. However, algorithms with reliable accuracy are difficult to implement in a real-time environment due to computational complexity.

This thesis developed a new way of extracting and using data from a human figure in a video frame to determine what type of activity the subject is performing. Following background subtraction, a thinning algorithm operating on the silhouette offered a more robust limb extraction method, while a six-segment representation of the human figure offered more accuracy in deriving limb parameters, or components, such as distance from torso, and angle of displacement from the vertical axis. Neural networks or nearest neighbor classifiers used the limb components to identify a number of activities, such as walking, running, waving and jumping. This entire human activity recognition system was tested with both a MATLAB implementation (non real-time) and a C++ implementation in OpenCV (real-time). The algorithm achieved 96% classification accuracy in video feeds, which is only slightly lower than that of intensive, non real-time systems.
Table of Contents

Thesis Release Permission Form ............................................................... ii
Dedication .................................................................................................. iii
Acknowledgements ....................................................................................... iv
Abstract ......................................................................................................... v
Table of Contents ........................................................................................... vi
List of Figures ................................................................................................. vii
List of Tables ................................................................................................... viii
Glossary ............................................................................................................ ix

Chapter 1  Introduction ................................................................................. 1

Chapter 2  Background .................................................................................. 4
  2.1 Applications of Activity Recognition ..................................................... 4
  2.2 Methodology ............................................................................................ 5
    2.2.1 Human Figure Extraction ................................................................. 7
    2.2.2 Human Figure Representation ......................................................... 8
    2.2.3 Activity Data Extraction and Representation ..................................... 10
    2.2.4 Activity Classification ...................................................................... 12

Chapter 3  Limb Component Extraction ....................................................... 15
  3.1 Background Subtraction ......................................................................... 16
  3.2 Skeletonization ........................................................................................ 19
  3.3 Feature Point Extraction ........................................................................ 22
  3.4 Six-Segment Model of the Human Figure .............................................. 22
  3.5 Evaluation of Limb Components for Characterization ............................ 26

Chapter 4  Activity Classification ................................................................. 30
  4.1 Classification Algorithms ........................................................................ 31
    4.1.1 Artificial Neural Network ............................................................... 31
    4.1.2 Nearest Neighbor .......................................................................... 33
  4.2 Real-time Classification ........................................................................... 34

Chapter 5  Implementation and Integration .................................................. 37
  5.1 Background Capture ................................................................................. 39
  5.2 Video Frame Processing ........................................................................... 39

Chapter 6  Results .......................................................................................... 44
  6.1 Training and Performance Results .......................................................... 44
    6.1.1 MATLAB Testing ............................................................................ 45
    6.1.2 Real-time Environment .................................................................... 50
  6.2 Example Frames ....................................................................................... 52
    6.2.1 MATLAB Prototype ....................................................................... 52
    6.2.2 Real-time Environment .................................................................... 58

Chapter 7  Conclusions .................................................................................. 63
  7.1 Discussion of Results .............................................................................. 63
  7.2 Future Work ............................................................................................. 66

Bibliography .................................................................................................... 69
List of Figures

Figure 1: Activity Recognition Process ................................................................. 6
Figure 2: Human Figure Representation Schemes .................................................... 8
Figure 3: Skeletonization Elimination Rules (1 denotes a white pixel) ...................... 21
Figure 4: Feature Point Rules (1 denotes a white pixel) ........................................... 22
Figure 5: Multiple Endpoints on a Limb ................................................................. 23
Figure 6: Fujiyoshi’s Five-Segment Model ............................................................. 24
Figure 7: Six-Segment Model .................................................................................. 25
Figure 8: Limb Component Measurements .............................................................. 28
Figure 9: Artificial Neural Network Topology .......................................................... 32
Figure 10: System Flow Diagram ............................................................................ 38
Figure 11: Image Storage Structure ........................................................................ 40
Figure 12: Neighborhood Transformation Example (1-white, 0-black) ................. 42
Figure 13: Test Jump Activity .................................................................................. 53
Figure 14: Test Run Activity .................................................................................... 54
Figure 15: Test Walk Activity .................................................................................. 55
Figure 16: Test Wave1 Activity ............................................................................... 56
Figure 17: Test Wave2 Activity ............................................................................... 57
Figure 18: Real-time Jump Activity ......................................................................... 58
Figure 19: Real-time Run Activity ........................................................................... 59
Figure 20: Real-time Walk Activity ......................................................................... 60
Figure 21: Real-time Wave1 Activity ....................................................................... 61
Figure 22: Real-time Wave2 Activity ...................................................................... 62
List of Tables

Table 1: Activity Descriptions ......................................................................................... 46
Table 2: ANN Classification on Individual Frames, Confidence Threshold = 0.6 .......... 48
Table 3: ANN Classification on Activity Segments, Confidence Threshold = 0.6 ...... 48
Table 4: NN Classification on Individual Frames, Confidence Threshold = 50 ............. 48
Table 5: NN Classification on Activity Segments, Confidence Threshold = 50 ............. 48
Table 6: NN Classification on Individual Frames, No Confidence Threshold ............... 49
Table 7: NN Classification on Activity Segments, No Confidence Threshold .......... 49
Table 8: NN Classification on Individual Frames, Confidence Threshold = 50, Motion Weight = 30 .................................................................................................................. 49
Table 9: NN Classification on Activity Segments, Confidence Threshold = 50, Motion Weight = 30 .................................................................................................................. 49
Table 10: NN Classification on Individual Frames, No Confidence Threshold, Motion Weight = 30 .................................................................................................................. 50
Table 11: NN Classification on Activity Segments, No Confidence Threshold, Motion Weight = 30 .................................................................................................................. 50
Table 12: NN Classification on Individual Frames, No Confidence Threshold, Motion Weight = 30 .................................................................................................................. 51
Table 13: NN Classification on Activity Segments, No Confidence Threshold, Motion Weight = 30 .................................................................................................................. 51
Glossary

ANN: See Artificial Neural Network.

Artificial Neural Network: An interconnected group of artificial neurons that uses a model (e.g. mathematical or computational) for information processing. Generally it is an adaptive system that changes its structure based on information fed through the network during training.

Feature Extraction: The locating of certain characteristics within an image or signal. Features may be anything, but here they refer to the head, hands, and feet.

Feature Point: Any distinguishable point on a human figure. It is generally tracked from frame to frame. Common choices of feature points include points on the head, hands, feet, elbows, and knees.

Feature-based: An approach which uses features.

k-NN: See Nearest Neighbor.

k-Nearest Neighbor: A general term for the classification algorithm which finds the k data vectors in a set which have the smallest Euclidean distance to a test data vector.

NN: See Nearest Neighbor.

Occlusion: A hiding or partial covering of a feature or object.

Real-time: The results from the calculations are required without noticeable delay. Delays could be disastrous or cause the system to fail.

ROI: The Region of Interest in an image.
Chapter 1

Introduction

Research in the field of computer vision aims to replicate the extremely intricate and complex pattern recognition capabilities of the human brain. Human activity recognition uses image understanding techniques on video sequences to detect and recognize what a human subject is doing in the surrounding environment. This type of information is difficult to obtain both quickly and accurately, but it is of great value in many applications, particularly in the security industry. Human activity recognition is one of the complex tasks the human brain does effortlessly, but presents many difficulties when a computer system attempts to automate the process. The vast amounts of data available in video streams often make it difficult for a computer system to make classification decisions in real-time environments due to the large processing requirements.

Real-time human activity recognition systems are becoming increasingly important in the security industry, where it is important to identify suspicious actions and behaviors to avoid harm coming to others. People watching security videos may miss important occurrences, and in such situations, it would be helpful for a computer system to flag suspicious actions and behaviors and alert to possible dangerous or criminal situations. Such real-time systems are important not only to the security industry, but
also for the medical industry. Motion patterns of humans could be examined to help patients that seem to be experiencing a dangerous condition, such as a seizure, heart attack, or serious fall. Proper medical personnel would be alerted to provide care to the person. Applications of this type of system are varied, but in all situations, a human activity recognition system would provide an extra layer of vigilance and security in real-time to alert people for the possible need to respond to an important situation that may otherwise be missed.

The solution to the real-time problem then becomes how to best represent the human figure within the computer system such that the data extracted from the video is minimal, but can still be used in a classification system to accurately differentiate among activities. The data extracted from video streams should all provide independent and unique information about the scene, and this information should characterize an activity and allow for individual differences for how people perform activities differently.

Considerable research has been done on segmentation algorithms which extract objects and people from the environment for example, and research into human activity recognition has been gaining popularity by building on existing methods of image understanding. This task of human activity recognition combines both the spatial information of the target subjects within the video frames, as well as the temporal information of the target subjects as they move over time. This allows human activity recognition research to draw on a wide variety of approaches and algorithms to devise an optimal recognition system for a given set of circumstances. Offline analysis of data would allow the usage of more complex and computationally intensive algorithms than would be feasible in a real-time environment. A method must be used to separate the
human figure from the environment so as to indicate what part of each video frame is significant. Then, the position and motion data of the figure can then be extracted. Chapter 2 discusses popular approaches to these issues and how several approaches are not suitable for implementation in real-time systems.

The framework proposed in this thesis implements a real-time human activity recognition system by reducing the data required for processing and classification such that only a few parameters of body positioning, orientation, and net motion in the XY plane are required for a classification. This is accomplished by representing the human figure as a streamlined skeleton and comparing relative position and orientation data of the subject’s body and limbs and using this data in a classifier. Chapter 3 discusses how these limb components were chosen, extracted, and used in the classification process, and Chapter 4 discusses how the classification structure was developed and tested for the system. Implementation details of the system are discussed in Chapter 5 with the human activity recognition system results presented in Chapter 6. Discussion and analysis of the results, as well as paths for future work, are discussed in the final chapter.
Chapter 2

Background

Interest in the field of human activity recognition has existed for sometime, but the research has gained popularity in recent years with the desire for advanced security and surveillance systems. With growing knowledge in the field, the methods have begun to be applied to various other fields. The thorough, computationally intensive algorithms have begun to be adapted for real-time environments with somewhat less accuracy, but a much faster processing speed. However, human activity recognition is only a single stage of a multi-staged system, with each stage presenting unique difficulties in achieving a system that operates in real-time. Many approaches to the subject have been researched and tested, each having strengths and weaknesses.

2.1 Applications of Activity Recognition

Within the last several years, there has been a growing desire for increased security measures in high-profile areas such as national monuments, airports, bridges and tunnels, and office buildings in light of terrorist threats. In the past, much of surveillance work often required a person to view many streams of video simultaneously. Studies on human psychology and attention span indicate that remaining vigilant on a task of detecting stimuli is difficult if the stimuli are infrequently presented and if the subject has
been doing the task for an extended period of time [7, 15]. A psychological study on signal detection by Mackworth showed that accuracy in this type of task has been shown to dramatically fall after as little as thirty minutes, with subjects missing almost 25% of the target signals. Thus, it is difficult for people to remain attentive to video streams and detect questionable actions and behaviors for long periods of time. Airports have attempted to address this issue by mandating shift changes at security checkpoints after certain lengths of time have passed and training x-ray screeners with Threat Image Projection tests (TIP). Still, a single detection miss can have a great price in life and property. Automating the process of detecting questionable behaviors and actions would reduce the probability that the behavior or action would go undetected by a human observer.

Beyond the need in areas of security, other fields such as medicine benefit from automated systems which detect and identify human activities. Such systems can be trained to detect patterns of motion indicating distress, as would be the case if a person was having a heart attack for example, or to detect a lack of motion if a person had fallen and was unable to move. These types of system could be used in conjunction with existing systems of alert bracelets and necklaces where subjects self-report medical emergencies for aid. If the subject was unable to alert the staff independently, these systems could alert medical staffs to potential emergencies.

2.2 Methodology

Automated human activity recognition by computers attempts to accomplish what the human brain does naturally – to differentiate among the actions and behaviors of
human figures and to assign the actions and behaviors to categories or classes. Unfortunately, the complexity of pattern recognition performed by the brain is near impossible to replicate on a scale suitable for current technology. Therefore, this problem becomes a matter of finding the best way to represent an activity in a computer system and determining what type of information from the scene is necessary to accurately distinguish one activity from another [6, 16].

In general, representing the activity is accomplished with a multi-step process illustrated in Figure 1. A very popular approach to representing an activity within a computer system is to use data from the human figure performing the activity in the scene [8, 12, 22, 23]. This involves the extraction of the figure from the background which is a non-trivial task. Once the figure is extracted, the data can be used for any number of human figure representation schemes with the ultimate goal of extracting the most useful and complete information to characterize the activity. Once the activity data is extracted, it can be used in a classifier for the decision-making process.

![Figure 1: Activity Recognition Process](image)

In each stage of processing, implementation choices are made, and some implementations are not well suited for real-time environments for various reasons. The following sections outline the most common approaches to each step and comparisons to illustrate the strengths and weaknesses of each for this application.
2.2.1 Human Figure Extraction

There is broad need for robust algorithms capable of distinguishing among many different activities with a high degree of accuracy. Optical flow analysis is one approach used in activity recognition [1]. Used alone, the data captured with optical flow algorithms provide insight into the areas of the scene that are moving, but optical flow does not capture foreground objects that are stationary. This information can be just as valuable for activity classification as the information about the motion in other parts of the scene. Barron shows that more robust algorithms of optical flow can achieve very low error rates in the calculated motion vectors, but the algorithms that could realistically be used in a real-time environment show significant calculation error in non-synthetic video sequences. Optical flow algorithms will also extract the motion of any part of the scene, be it foreground or background. This creates difficulty in environments where there is noticeable motion in the background from extraneous sources.

A second approach to human activity recognition involves the extraction of the human figure from the background with a background subtraction algorithm. Such algorithms can be classified as either adaptive – where background estimation changes over time to account for variations in scenery – or non-adaptive – where the background model remains constant throughout a video stream. Adaptive algorithms allow for robust background subtraction particularly in outdoor scenes where extraneous motion from things such as animals, tree leaves, moving cars, and other people, can affect the subtraction [13, 17]. However, it becomes difficult to extract an entire human figure if parts of the body remain motionless for long periods of time. Motionless parts of the figure are typically lost, and this important information cannot be used for classification.
purposes. Non-adaptive algorithms do not perform as well in outdoor scenes, but do very well in controlled environments. In addition, they are generally less computationally intensive [18, 21, 23]. For a constrained environment for testing, the effect of the sensitivity to background noise with non-adaptive background subtraction becomes negligible.

2.2.2 Human Figure Representation

Several representations of the human figure have been presented in research. These representations range from a holistic approach to a very piece-wise approach. Each has strengths and weaknesses, but some are more suited to a real-time environment than others. With the goal of reducing the amount of data obtained from extracting the figure from the background, it becomes important to eliminate extraneous data which adds no new information to speed up processing.

![Figure 2: Human Figure Representation Schemes](a) Jin et al. [11], (b) Ozer et al. [19], (c) Mori et al. [16]
Several approaches in research have a holistic approach based on silhouettes. Blank uses solutions to the Poisson equation to extract space-time features of the human figure such as action dynamics, shape structure, and orientation for a “volumized” characterization. Jin simply uses the human figure silhouettes extracted from the background. However, these approaches can be sensitive to noise, and a great deal of error can be introduced by clothing and body size.

In the middle of the spectrum, other research uses a combination of holistic and piece-wise approaches. Chen and Ozer divide the extracted silhouettes into limb shapes, or “limb blocks” for individual analysis. These processes are also sensitive to noise, and have errors that are introduced from clothing and body size. In addition, there is the added processing requirements of accurately and quickly dividing the silhouettes into the limb blocks.

More piece-wise approaches are also explored in research. Feature points can be extracted from the human figure and used in various configurations to represent the human figure. A feature point can be any distinguishable point on the figure, and usually corresponds to major joints, such as hands, feet, elbows, and knees for example. Once feature points are extracted, they can be used in many ways. Fujiyoshi, Kim, Lu, Mori, and others use feature points to represent the body in a skeleton-like manor. Algorithms such as the skeleton [8, 12, 14, 16], lend themselves more to real-time applications due to the relatively few levels of processing needed for the algorithm to work. They are less sensitive to noise, clothing error, and body size differences. These algorithms can be computationally intensive, but with careful implementation techniques and modified
feature point configurations, the extra processing time can be more easily mitigated than in other approaches.

2.2.3 Activity Data Extraction and Representation

The method of activity data extraction and activity representation is tied closely with the method of representing the human figure. Obviously, approaches used with a silhouette representation will be ill-suited for feature point representations. Therefore, these two stages are closely linked when determining the processing for the activity recognition system.

Once a human figure can be extracted from the background, a silhouette of that figure is obtained. Many human activity recognition algorithms use silhouettes as a basis for the activity classification [11, 18, 24]. For example, template libraries can be created to represent activities. These libraries are composed of many single frames of silhouettes that are marked as representative of particular activities. Frames are extracted from a video stream and matched with library silhouette templates for a closest match. This method and other model-based learning methods, however, rely on human annotation of test video stream frames and expert knowledge of which frames would be most representative of the activity, as well as how many template frames should be included for an activity in the library [9, 14, 18]. Restricting computational intensity by limiting the size of the template library often reduces the robustness of the algorithm. Matching test frames to template frames of similar activities can be difficult and/or error-prone.

Using the entire silhouette of the human figure for classification presents difficulties due to variations in posture, body size, and clothing worn, among other
factors. Some approaches to activity recognition fit a skeleton-like structure to the silhouette to extract data from the position and movement of the figure for classification [8, 12, 14, 16]. In a derivative of template matching, the human figure extracted with the silhouette is broken down into key sections, or regions [3, 19, 20]. Such approaches may include key sections such as head, upper arm, lower arm, hand, thigh, calf, and foot regions, as well as multiple torso regions. The positions and orientations of all these regions and the “joints” which join them, relative to one another, are characteristic of the activity being performed in the scene. Temporal data is still being taken into account, as is the case with optical flow methods, but the temporal data is instead presented in the context of a specific part of the human body, rather than as part of the entire visual field. Since each single region moves as one entity, be it rotationally or along an axis, the temporal data is reduced to a single measurement per region on the human figure.

Of course, as the region sizes increase, more generalization is introduced into the measurements of the regions. This increased generalization can create temporal error. If a non-moving region becomes generalized into a moving region, the motion will be distributed over the entire generalized region. Taking into account individual differences when performing an activity, many of the important motion parameters could be averaged out. It essentially becomes an optical flow calculation with very low resolution. The advantage, however, is that the generalization can be defined to group areas of the visual field, with similar optical flow measurements, together and eliminate redundancy.

Another approach to activity recognition involves a combination of background subtraction and feature point tracking [8, 12, 22, 23]. Feature points can be defined as points of interest on the human figure that are tracked over time. A feature point is
usually represented as a single point, such as on the boundary of the silhouette or any
distinctive point on a non-thresholded extracted human figure. A small neighborhood of
pixels may be used to perform neighborhood matching to determine the motion vector
from one frame to another [5]. Popular choices for feature points include head, hands,
feet, knees, and elbows [3, 8, 23]. These parts of the body are involved in most pose
differentiation methods, and their position and orientation relative to one another provide
data to be used in classification.

Techniques involving feature point extraction and representation vary greatly. Some approaches assume known locations of manually extracted feature points in each
video frame and simply track them over time [16]. More robust algorithms use various
methods of automatic feature point extraction, such as with skeleton or extremal point-
extraction [3, 8]. From the positions and orientations of feature points in relation to the
centroid of the figure, assumptions can be made about the positions and orientations of
the rest of the limbs, thereby reducing the data involved in extraction, tracking, and
classification [8, 23].

2.2.4 Activity Classification

As processing is done on the human figures in the video sequences, the
information collected must be helpful in the classification process, that is, the data must
be sufficient for a classifier to differentiate among the activities in the system. Without
characteristic data from the video sequences, it would be impossible for correct
classification to be accomplished. To ensure the data collected is sufficient, multiple
classifiers may be considered on a data set to find the classification potential.
Several classifiers have been used in human activity recognition research with varying degrees of complexity. When differentiating among a few activities, simple thresholds on extracted features can be used for a high accuracy rate [8]. However, as activities become more numerous and complex, this approach does not work. Data will not threshold easily, and thresholding techniques often fall short when individual differences produce variances in activity parameters such as speed.

Artificial neural networks (ANNs) have been used as classifiers in many problems involving detection and differentiation. While more complex to train than other methods, they are very robust and can be customized to achieve the classification structure desired for a particular application. A drawback, however, is the retraining of the classifier when activities are added to the system, which can be extremely time consuming. Reducing the training set with a limited number of representative frames of the activity would reduce the training time, but the robustness of the classifier would suffer.

The nearest neighbor algorithm is more sensitive to outliers than ANNs, but provides similar classification characteristics with the advantage of being easier to reconfigure and adapt to the situation. The nearest neighbor algorithm can use distance measurements ranging from simple Euclidean distances to more complex weighted distance functions that emphasize specific extracted data from the human figure. The drawback to this algorithm is the training set, or ‘neighbors’, used for the distance measurements. As this set grows, the system doing computations on all samples to find the nearest neighbor would slow down. This could be mitigated by incorporating techniques such as K-means clustering to increase classification speed without a great loss in robustness.
The implementation of a human activity recognition system requires the integration of several processing steps that must be fine-tuned to the application. The human figure must be extracted from the scene, the human figure and activity must be represented in the system, and a classifier must be developed to differentiate among activities. The next several chapters will discuss these steps in detail, and how individual algorithms must be tuned to the specific application of real-time human activity recognition.
Chapter 3

Limb Component Extraction

One of the fundamental problems in activity recognition systems is how activities are represented within the system, known as the feature extraction process. Most feature extraction algorithms use combinations of human figure pose and change of position over time to define activities in the system [8, 12, 14, 16, 21, 22]. How pose and change in position over time is captured influences how successful the system will be in differentiating among activities.

A common feature extraction approach is to extract the human figure from the scene and identify positions of specific “feature points” on the figure. These points on the human figure are generally taken to give indicative data on the activity being performed. The data can be used in many ways. Fujiyoshi and Su use limb parameters, or components, such as length, position, and angle of displacement which can be determined from analysis of feature points located on figure limbs and joints. The cyclic patterns of motion of the limb parameters are used to classify an activity as either walking or running based on a simple threshold. To generalize Fujiyoshi’s approach, the system should be able to differentiate among more activities which cannot be done with a simple threshold. A new approach must be taken to use the feature point data in new ways. This chapter discusses how the system presented in this thesis represents activities
with limb parameters and under what conditions limb components provide sufficient data to classify activities.

### 3.1 Background Subtraction

To record and track the feature points on the human figure, it is necessary to first extract the figure from the background. This figure extraction can be done in several ways. The simplest static approach requires knowledge of the background scene. This generally requires obtaining a single frame of the background. The assumption is that the background frame has static elements that will not change during the course of the activity recognition processing. As such, outdoor scenes and crowded areas are not good candidates for using static background subtraction due to the high level of dynamic behavior of the background elements, such as trees blowing in the wind and cars traveling down roads.

Extracting the human figure from the scene in this way generally involves the subtraction of the known background frame from the video feed frames. In grayscale, this can be as simple as the absolute value of a pixel-by-pixel subtraction. In color images, it could be the subtraction process on each color plane, or a Euclidean distance calculation. This method is extremely fast and ultimately lends itself very well to parallelization, but again, it is not adaptive to changes and motion in the background scene.

Often times, an adaptive background subtraction approach will use running averages or infinite impulse response (IIR) filters. These processes require motion for the full figure outline to be extracted. If a person is standing still and waving, the filter will
only extract the outline around the upper torso and arm. The lower body and legs lack motion and will not be extracted. Hence, the figure outline will be incomplete, and the feet feature points cannot be extracted. While these points are not necessary for classification, they would be helpful. In many situations, lack of motion of a certain part of the body can be just as important as having motion present. Therefore, this approach was not used in this thesis.

Variations of this adaptive model exist, where probability is taken into account as to whether a given pixel is background or not. These approaches are somewhat more effective in extracting a complete human figure outline and subsequently obtaining all the feature points needed. Unfortunately, the processing speeds are a bit slow for use in a real-time environment, and the algorithms still require the use of a known background frame as an initial reference.

In an indoor testing environment, there is little need for an adaptive algorithm, as the background remains constant for long periods of time. Therefore, a non-adaptive background subtraction algorithm was used.

Results of prototyping in MATLAB yielded positive results for background subtraction on grayscale images. The extracted silhouettes of the human figures were full and well-defined. Using this approach in the real-time system did not yield robust results, however. The human subjects and their clothes were not as reliably extracted from the scene, leaving holes and ill-defined silhouettes. The same problem occurred when using the absolute difference background subtraction on each color plane of the image. If the difference was above a given threshold on any one of the color planes, the pixel was marked as part of the silhouette. This allowed for extended utilization of the
data given in the captured frames without adding much computational complexity and processing time. However, the silhouettes extracted were only slightly better than when using grayscale images, and there were many more extraneous pixels marked from light flicker and shadowing.

Instead of the absolute difference threshold on each plane, a Euclidean distance calculation was used on each pixel of the frame using color information. The color vectors of each pixel were used in the distance calculation between the background image and the video feed frames. For each pixel in the test frame $p_t$, the distance $D$ from the corresponding pixel in the background frame $p_b$ was computed.

$$D = \sqrt{(r_b - r_t)^2 + (g_b - g_t)^2 + (b_b - b_t)^2}$$

After the distance was computed for a given pixel, a threshold $t$ was applied. If the distance was greater than $t$, the pixel was set white. Otherwise, the pixel was set black. The result of this operation was a binary image which defined the areas of greatest dissimilarity between the test image and the background image. Using the lowest resolution possible for many image formats, 8-bit unsigned integers were used to store pixel data. Setting a pixel white required a 255 value, while setting it black required a 0 value.

$$P = \begin{cases} 
255, & D > t \\
0, & \text{otherwise}
\end{cases}$$

This background subtraction algorithm yielded clean, well-defined, whole silhouettes with very few extraneous pixels being marked as part of the silhouette.
To further clean the silhouette shapes, morphological processing was performed on the resulting frames. Small extraneous groups of pixels not part of the silhouette were removed by erosion, and breaks in the silhouettes were joined together using a closing operation.

3.2 Skeletonization

Once a silhouette has been successfully extracted from the background, the system can estimate the locations of the feature points needed to be extracted. Several algorithms exist which accomplish this. More basic algorithms divide the silhouette into three sections – a top, middle, and bottom – to define areas where feature points are located [21]. Feet, for example, would be located in the lower part of the bottom section. Feature points located in the middle section would probably be hands. Of course, these divisions would be based on the locations of the section boundaries relative to the human figure. This algorithm is effective at determining which feature is being analyzed, but can not precisely locate the feature point. A general region can be determined, but limb occlusions and partial occlusions make it difficult to identify whether a limb or feature point is actually present in a particular silhouette division.

An algorithm that is more effective at determining precise locations of feature points extracts local maxima from distance measurements from the centroid of a silhouette to a silhouette boundary pixel [8]. Fujiyoshi found that local maxima would be indicative of limb extremities, and those maxima could be labeled as the precise location of the feature point needed for extraction. Unfortunately, depending on the silhouette, extracting the correct local maxima proves to be difficult. Assuming that the feature
points to be extracted would be the strongest local maxima on a graph of distances, they may have to be greater than a certain threshold to be considered strong enough. However, taking only the strongest local maxima does not account for partial occlusions of the limbs. Even taking local maxima with moderate strengths would not account for partial occlusions because the shape of the local region is not considered.

An alternative approach to extracting feature points is through a skeletonization process. Skeletonization of a shape involves the gradual whittling away at the shape’s boundary pixels until nothing is left but a one-pixel-wide “skeleton” of the previous shape. This is generally accomplished with a set of masks that act as “elimination rules” for determining which pixels should be left and which should be removed. Several such rule sets exist [4, 10]. The skeletonization process has the advantage of being able to extract small variations in shapes where partial limb occlusion is present. Feet, hands, and head on the human figure are readily extractable as well. This algorithm can be computationally intensive, depending on the implementation strategy, but this can be mitigated by narrowing the search area of the figure and utilizing a lookup table for the masks.

Huang’s skeletonization algorithm was used in this thesis [10]. The algorithm is based on a set of elimination rules in the form of 3x3 pixel configurations seen in Figure 3. The pixel is ‘erased’ from the silhouette if the 3x3 neighborhood surrounding it matches a configuration in the eliminate rule set. Each pixel can be examined in parallel which allows for the possibility of greater processing speedup with additional computing resources. Continuing elimination until no pixels can be removed, the remaining pixels form a one-pixel wide ‘skeleton’ of the human figure extracted from the silhouette. This
skeleton gives the form of the human figure with defined limb endpoints – essential for extraction of limb components.

<table>
<thead>
<tr>
<th>Amount</th>
<th>Elimination Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Never</td>
</tr>
<tr>
<td>1</td>
<td>Never</td>
</tr>
<tr>
<td>2</td>
<td><img src="image1" alt="Rule Image" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image2" alt="Rule Image" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="image3" alt="Rule Image" /></td>
</tr>
<tr>
<td>5</td>
<td><img src="image4" alt="Rule Image" /></td>
</tr>
<tr>
<td>6</td>
<td><img src="image5" alt="Rule Image" /></td>
</tr>
<tr>
<td>7</td>
<td><img src="image6" alt="Rule Image" /></td>
</tr>
<tr>
<td>8</td>
<td>Never</td>
</tr>
</tbody>
</table>

Figure 3: Skeletonization Elimination Rules (1 denotes a white pixel)
3.3 Feature Point Extraction

Once the silhouette has been skeletonized, feature points on the limb extremities can be extracted. While applying the elimination rules to the pixels of the silhouette, the endpoint masks in Figure 4 developed for this thesis can also be applied to flag skeleton endpoints – an endpoint being defined as a pixel with a single, eight-connected neighbor.

![Figure 4: Feature Point Rules (1 denotes a white pixel)](image)

The skeletonization process can leave behind multiple endpoints on the skeleton for a single limb as shown in Figure 5. To determine the actual endpoint of the limb, a clustering process is used to average feature points located within certain distances of each other. If other feature points are found within a certain neighborhood of a feature point being analyzed, those points are averaged to create a clustered limb endpoint.

3.4 Six-Segment Model of the Human Figure

Once feature points have been extracted and clustered, their correspondences to parts of the human figure must be established. Also, the feature point data must be used in a way that presents unique information for each different activity to the classifier. A shortcoming of some feature point based algorithms is that the structural component of what the feature points represent is lost if they are processed independently of a structural
Figure 5: Multiple Endpoints on a Limb
reference. That is, there is information to be gained when the feature points are used in conjunction with a structural model for the human figure. Fujiyoshi used a five-segment model to represent the human figure, and data was extracted with head, hands, and feet feature points [8]. Figure 6 shows an example of the five-segment model being applied to a silhouette. A centroid is marked in the middle of the region of interest, and five feature points, extracted by a local maxima algorithm, are connected to the centroid with line segments. The length as well as the rotational angle of the line segments are computed and used for classification. These line segments simulate the limbs and torso of the human figure.

![Figure 6: Fujiyoshi’s Five-Segment Model](image)

This model has obvious error in identifying limbs, particularly apparent in the arms. The resulting segments poorly correspond to the actual orientation of the arms of a walking human figure. The body axis along the torso is nonexistent in this model by its definition. To create a more realistic model of the human figure without adding much complexity, a six-segment model was developed, seen in Figure 7. This model takes into account the body axis, as well as hip and shoulder ‘centroids’ for better estimates of limb orientation relative to the rest of the body.
After the skeleton of the human figure is created and the extracted feature points are clustered, the centroid of the region of interest is computed. This centroid is moved horizontally until it encounters a white pixel on the skeleton corresponding to the torso of the body. Based on a ratio found from averaging body proportions of test subjects, the torso centroid is moved vertically to create both a shoulder centroid and a hip centroid. Feature points can then be joined to their corresponding centroid for more accurate distance and angle measurements.

![Six-Segment Model](image)

Figure 7: Six-Segment Model

The correspondence between feature points and the joints is found simply by examining where the feature point is in relation to the shoulder and hip centroids. If the feature point is below the hip, that feature point is a foot. If the feature point is above the shoulder, that feature point is a head. Any feature points in between the shoulder and hip are hands. Exceptions to these rules are made for some postures, such as when one or both hands are raised above the head, or if the hands hang low below the hip. In the first case, the head is marked as the feature point closest to the shoulder, and others are marked as hands. In the second case, hands are marked as the feature points closest to the hip. Both exception cases are only used when the shorter-distanced head and hands
differ significantly distance-wise from the other feature points in their respective regions. This ensures exceptions are not applied incorrectly, such as when a leg is bent at the knee and appears shorter than the other leg which is fully extended.

3.5 Evaluation of Limb Components for Characterization

When the correspondence of feature points to human figure parts has been established, the data must be used in a way that could identify the activity being performed. The periodic motion patterns of limbs over time that is characteristic of many activities have been examined in current research [8, 23]. It has been found that frequencies and amplitude of the motion curves in the XY plane over time can be used to classify activities in video sequences. However, algorithms utilizing this technique are not able to classify a very wide range of activities, and some classification results are reported only for separating two activities [8]. Upon investigating this approach, it was found that the differences in frequency and amplitude among activities fall within the range of individual differences of human subjects for certain single activities. For example, the leg swing amplitude of walking for one person may be more characteristic of the ‘norm’ running characterization for leg swing amplitude.

As with any algorithm, relying on a small number of data sequences – in this case one feature point – does not provide enough data to reach a conclusion, or meaningful classification. Therefore, several samples must be used, and this gives a rationale to the selection of several feature points on the body for use in classification. Most commonly used are the head, hands, and feet, but other joints such as elbows, knees, and shoulders have also been used [8, 12, 14, 16, 21, 22, 23].
Analysis of each feature point’s motion history has been shown to provide correct classification of activities [8, 23]. However, this method relies on continuous and accurate tracking of each feature point in space-time. Due to occlusions of the limbs, such as when the arms line up with the torso when walking is viewed from the side, some feature points are temporarily lost during the video feeds. The success of this algorithm depends on a robust system for predicting the location of feature points after they have been occluded from the camera’s view, and this is a difficult problem.

This problem would have to be solved using characteristic motion patterns for each activity to be used in the tracking calculations. This would place many constraints on the scalability of the system by requiring human annotation of feature point correspondence after occlusions. It would also require increased processing when trying to find correspondence of feature points based on several motion profiles that would exist for each activity. These would not be desirable features of a real-time robust algorithm for activity recognition.

A robust algorithm should require minimal human annotation of data that would not rely on activity profiles needed to be programmed into the algorithm. By simply using a limited number of parameters extracted from each frame in whole video sequences of an activity, the need for human annotation is eliminated without requiring large amounts of processing time or memory. Feature points are used as a function of where they are in relation to the central vertical axis of the body. For example, processing a feature point does not determine what the human figure’s right hand is doing, but simply that there is a hand located on the right side of the body, regardless of which hand it actually is.
This reduces the number of problems associated with occlusion and the complexity of finding feature point correspondences. The resulting features extracted from each frame can therefore be the limb distances to the corresponding centroids and angles of displacement from the torso axis, of the head, hand found of the left side of the body, hand found on the right side or the body, foot found on the left side of the body, and foot found on the right side of the body. This gives ten parameters that provide orientation data of key parts of the body that have been shown to be key factors in differentiating among activities. Figure 8 illustrates the distance and angles measurements for the five feature points of the head, hands, and feet of the human figure.

![Figure 8: Limb Component Measurements](image)

Since all these measurements are relative to the central vertical axis of the body and are not based on the video frame itself, there is no information provided about the motion of the human figure as time progresses. To provide more data to the classifier and incorporate the temporal characteristics of the human figure as a whole, delta motion values in both X and Y directions are used in classification. These calculations are simply the difference between the position of the hip centroid in the XY plane of the
current frame and the previous frame. This allows differentiation between activities such as standing still and jumping in place.

Once data has been extracted from the human figure in the scene, the data must be used in some way to classify the activity. Chapter 4 discusses methods for using the feature data in classification, as well as two classifiers that were used in testing the validity of the limb component extraction approach.
Activity Classification

A classifier can only perform to its potential if the training data is accurate and sufficiently representative for the classification process. Compounding error in the data – due to blurring for example – can result in the classifier being unable to accurately differentiate among classes. Hands and feet may be blurred if a camera cannot capture a fast action with a sufficient number of frames, and this may cause feature points to be erroneously marked. Errors may be present in both the training set data and the data submitted for classification. The classifier may also fail if the training data is not sufficiently broad for class differentiation. Some data points may be dependent on others, and thus do not contribute enough unique information for the classification process.

Addressing these problems was a two-fold process. The data extracted from the video frames was examined for accuracy, and an error margin was determined. Additionally, the classification process was performed using two different algorithms – the artificial neural network (ANN) and the nearest neighbor (NN).
4.1 Classification Algorithms

The data collected from feature extraction consists of the distance and angle of displacement measurements of the hands, feet, and head from their corresponding shoulder or hip marker, as well as the X and Y axis delta values from the motion calculations. To test whether this data is sufficient for the classification process, two different classifiers were used in prototyping – artificial neural network (ANN) and nearest neighbor (NN).

To test the capability of classification using the extracted data from the video sequences, a MATLAB prototype of the system was implemented and used with the Blank data set [2]. This data set contains color video sequences of ten activities performed by nine individual subjects. This data set was chosen because the camera moved very little during the taping of the sequence, making non-adaptive background subtraction very easy. There were also a wide variety of taped activities performed by each subject, which would help determine if the features extracted from the video sequences have the capability of differentiating among a varied set of activities. Nine subjects performing each activity allowed for the capture of individual differences in classifier training.

4.1.1 Artificial Neural Network

The first classifier to be used was the ANN which has been used in prior activity recognition research for classifying activities [22, 24]. This classifier has the advantage of being less sensitive to outliers, and the network structure can be configured in a variety of ways, depending on the needs of the system.
For the prototyping of this method, a feed-forward backpropagation ANN structure was created using the MATLAB neural networks toolbox. The network had twelve inputs, one for each of the limb components extracted from the skeletons and two for motion parameters, and a hidden layer of fifty neurons as shown in Figure 9. The hidden layer had to be sufficiently large to accommodate the number of inputs to the structure. The number of outputs of the classifier equaled the number of activities, or classes, on which the classifier would be trained. Between network layers, the ‘tansig’ transfer function was used. This allowed for a -1 to 1 output, easily mapped to a binary mapping if needed. The network was trained with the ‘trainlm’ training function.

Data was first prepared for classifier training based on the leave-one-out method.

The limb parameters were extracted for each activity video sequence for training with
eight of the nine subjects. The classifier was then tested with the activity video sequences of the remaining subject. The test subject was rotated after each trial. This method of training and testing allowed for a reasonable estimate of the performance of the classifier with the full set of available data.

The maximum output among all outputs of the network was used as the classification result. To provide a confidence measure to the classifier output, the maximum response was considered as a valid classification result if it was above a certain threshold. If the maximum response fell below the threshold, the frame would be marked as ‘Unclassified’. The threshold was adjusted to determine the optimal value which passed the largest number of correct classifications while not passing a large number of incorrect classifications.

4.1.2 Nearest Neighbor

The second classifier considered was nearest neighbor. This classification algorithm is very simple – the Euclidean distances from the test parameters to the training set parameters were calculated, and the classification result is the activity corresponding to the smallest distance measurement (k-NN is possible). Initial calculations were made by squaring the differences between the template library parameters ($l$ subscript) and the test parameters ($t$ subscript). This resulted in five temporary results for the feature point to joint marker distances ($d$ subscript), and five temporary results for the feature point angle of displacement measurements ($a$ subscript). The temporary results were summed, and the square root was taken to determine the distance $D$ from test frame limb components to template library limb components.
\[ h_{dt_a} = (h_{dt_d} - h_{dt_t})^2 \]
\[ h_{ata} = (h_{ata_d} - h_{ata_t})^2 \]
\[ h_{nta} = (h_{nta_d} - h_{nta_t})^2 \]
\[ h_{pta} = (h_{pta_d} - h_{pta_t})^2 \]
\[ f_{nta} = (f_{nta_d} - f_{nta_t})^2 \]
\[ f_{pta} = (f_{pta_d} - f_{pta_t})^2 \]
\[ mot_{x} = (mot_{x_d} - mot_{x_t})^2 \]
\[ mot_{y} = (mot_{y_d} - mot_{y_t})^2 \]

\[ D = \sqrt{h_{dt_a} + h_{nta} + h_{nta} + h_{pta} + f_{nta} + f_{pta} + mot_{x} + mot_{y}} \]

The training data set was used for matching test video sequence frames. All template data was accumulated in a single file. When test limb parameters were extracted, the Euclidean distance was calculated between the test limb parameters and each template data entry. The activity corresponding to the smallest distance would be the result of the classification algorithm. As a confidence measure, a threshold was set on the maximum distance allowable by the algorithm. If the smallest distance was not below the threshold, the frame would be marked ‘Unclassified’.

### 4.2 Real-time Classification
The nearest neighbor classification algorithm was chosen for the real-time system implementation due to the better classification performance based on prototypes as seen in Chapter 6, and the ease of modifying the underlying dataset for testing strategies. The template data, read in from a file during system initialization, was stored in a table in memory. Once the camera has begun to capture video frames and the limb parameters are extracted, the Euclidean distances are calculated from the extracted limb parameters to each template in the table. Again, the shortest distance in classification needed to be below a threshold for the classification decision to hold. Otherwise, the frame would be marked ‘Unclassified’.

Extending the system implemented in MATLAB, it was desired to have the ability to recognize more activities. Adding more activities required a modification to the Euclidean distance formula. In the scale of the video frames being used, the delta motion value ranges are very small – on the order of two to three pixels. The motion difference calculations would be insignificant compared to the difference calculations of the extracted limb parameters of the head, hands, and feet. Therefore, the range of values of each parameter and delta value were converted to a 0 to 100 scale. This scaling accentuated the small differences in delta motion values that were otherwise obscured in the formula.

With this scaling, the delta motion values were still somewhat obscured. Accumulating differences in the ten limb parameters would overshadow the two differences of delta motion. In order to have more equal contributions of spatial error and motion error, the delta motion difference values were weighted by a factor $W$ before
being added to the sum. The equations for the motion components were adjusted as seen below, and used in the distance calculation as before.

\[ \text{mot}_x = (\text{mot}_x - \text{mot}_x_t)^2 \cdot W \]
\[ \text{mot}_y = (\text{mot}_y - \text{mot}_y_t)^2 \cdot W \]

\[ D = \sqrt{h_d + h_n + h_p + f_t + f_n + f_p + \text{mot}_x + \text{mot}_y} \]

Once the decisions have been made on the best approach, for a specific application, to each stage of the activity recognition process, the algorithms can be implemented on the desired platform. The following chapter explains the steps taken to implement the activity recognition algorithms in both a MATLAB and C++ environment.
Chapter 5

Implementation and Integration

For a real-time implementation of the human activity recognition system, both hardware and software concerns must be addressed. The system requires a hardware platform consisting of a computer console for the algorithm processing and a camera capable of two-way communication with the computer console. OpenCV libraries and a C++ compiler were needed for the algorithm development in C++.

The pan-tilt-zoom (PTZ) camera capturing the live video feed must be controlled from the computer console to best specify the field of view of the system. During the initialization phase, a clutter-free background frame is captured for use with the background subtraction algorithm. Therefore, there must be two-way communication between the computer and the camera. The modules first developed by Justin Hnatow and later revised by Andrew Mullen, provide most of the functionality needed for these requirements.

In a real-time environment, a MATLAB code base would not be sufficient for the high processing speeds necessary for streaming video. Streamlining the system requires conversion of the prototype system to a more effective programming environment. The prototype code was converted to C++ which is far more streamlined than the MATLAB
platform. Furthermore, C++ has the additional benefit of utilizing Intel’s OpenCV framework developed for computer vision and image processing systems.

![System Flow Diagram]

Figure 10: System Flow Diagram

Once the hardware and software platforms are selected, separate subsystems are integrated to form the completed structure as a stand-alone, real-time human activity recognition system. **Error! Reference source not found.** presents a diagram of the system where each stage of the algorithm is defined. On startup, the template library is read from file and stored in memory. A background frame is captured by controlling the position of the camera to incorporate the desired field of view. After capturing the background frame, the camera remains fixed and is not adjusted further. Each image frame in the video stream is processed and clustered feature points representing limbs are obtained for classification. To display the classification results, a sequence of 30 frames
is analyzed, and the resultant activity classification is based on the activity with the maximum number of classifications within the 30 frames.

5.1 Background Capture

In order to perform background subtraction, there must be a static, background frame with which to perform the comparison. This background frame should be captured while there is little – ideally no – motion present. The Camera module was incorporated into the system to control the camera’s PTZ parameters for effective capture of the background frame. In practice, the camera was adjusted to incorporate a field of view that would allow subjects as much room as possible to move for the activities. Once the camera has been adjusted for background capture, the user inputs a command from the keyboard to store the frame into a background frame container. This frame is stored in memory, and is accessed for every background subtraction calculation.

5.2 Video Frame Processing

Several OpenCV functions were used to optimize the processing performed on the captured video frames stored in memory. Since available memory storage is not an issue in this system, many image containers were created to store the images resulting from intermediate processing steps throughout the algorithm. It is possible to perform in place computations, so that a container could be used for multiple steps if memory did become an issue. Separating each step of the algorithm has the additional benefit of facilitating debugging and giving a visual screen output of how the algorithm is performing.
The background subtraction algorithm is simply a thresholding of the Euclidean distance between the RGB color vectors of the background frame and the video stream frame for each pixel. OpenCV by default stores the images in an interleaved BGR structure as seen in Figure 11: Image Storage Structure. Each pixel value is stored as an 8-bit unsigned integer, which is the lowest possible resolution. First in the Euclidean distance calculation, one vector is subtracted from the other. The result is then squared, and then the individual squared vector entries are added, with a square root taken after the addition. Once the difference is taken between the pixel RGB color vectors, the interleaved BGR planes are split into three individual color planes and converted to 32-bit floating point values to complete the distance calculation. The result is stored in a single channel grayscale image containing the Euclidean distance measurements from the background frame to the video stream frame. Each pixel is then thresholded and converted back to 8-bit unsigned integers that display a bitonal image with the white
silhouette of the human figure extracted on a dark background. Values above the threshold are set to 255, or white, and values below are set to 0, or black.

Once the image is thresholded, morphological processing is performed to remove stray white clusters one or two pixels wide by erosion with two passes of a 3x3 square structuring element. Then the silhouette is dilated to close holes in the silhouette and rejoin sections that had not been extracted wholly from the background. An erosion follows to remove the enlarging effects of the dilation. The final dilation and erosion used 3x3 square structuring elements. The dilation was performed five times and the erosion three times, to account for the initial erosion to remove small stray white clusters.

In practice the system becomes difficult to test due to changes in the background when the entire camera field of view is considered for the skeletonization algorithm. Often other figures walk by, or a chair is moved by the computer console to observe the streaming output. In order to facilitate users being near the camera without influencing the algorithm, the left and right most quarters of the field of view are disregarded during skeletonization. These boundaries appear as yellow vertical lines on the skeletonization/model-fitting streaming output.

In implementing the skeletonization algorithm, it is possible to have a highly parallelized structure, as each pixel can be processed independently upon each iteration. In this system, the masks are applied to each pixel iteratively. To speed up processing, the morphological masks are stored in a lookup table in memory. Each 3x3 neighborhood surrounding a white pixel is transformed into an 8-bit sequence representing the presence or absence of white pixels in that neighborhood as binary flags. To reduce the size of the lookup table, the center pixel is excluded from this sequence as
it is already known to be white. Figure 12 illustrates a sample transformation of the neighborhood to 8-bit sequence. This 8-bit sequence is used as the ‘address’ in the lookup table which corresponds to the elimination rule value. If the center pixel is to be removed, a ‘true’ value will be returned from the table lookup. Otherwise a ‘false’ will be returned. The feature point detection masks are implemented in a similar lookup table structure.

![Figure 12: Neighborhood Transformation Example (1-white, 0-black)](image)

Based on the above methods, the skeletonization, feature point extraction, and feature point clustering are completed on the binary image of the human silhouette extracted from the background. This allows the fitting of the six-segment model to the human figure for limb component extraction. For visualization of the algorithm, the human silhouette and skeleton are displayed as video streams on the monitor. On the skeleton video stream frames, the center of the skeleton, or region of interest (ROI), is marked by a green circle. This point is the basis of the body ratio calculations that determine the shoulder and hip locations on the human figure. The shoulder and hip markers are shown with yellow circles, and the clustered feature points are joined to their
corresponding joint markers with red line segments. The line segments, along with the feature points and joint markers, illustrate the fitted six-segment model to the human figure. Thus, it becomes quite easy to see on the display how the six-segment model is calculated and fitted to the skeleton of the extracted human figure silhouette.
Chapter 6

Results

In determining the capabilities of limb component extraction approach in this thesis, a MATLAB prototype was first created to determine classification performance for a variety of activities. Once the prototype was tested, the algorithms were implemented in a real-time environment on live video streams from a PTZ camera.

6.1 Training and Performance Results

The performance of the human activity recognition systems developed was tested on individual frame classification accuracy and video segment classification accuracy. The streaming video, clustered feature points, and system classification results were saved to file during the experiments, so the system’s performance could later be analyzed. The saved video frames were examined to determine what activity was occurring in each frame, and the corresponding data extracted from each frame were checked for the classification results.

Individual frame classification was assessed by comparing the activity performed in the saved video frames to the classification results saved in the log file. The percentage of frames correctly classified was recorded for each activity. Activity video segment classification was accomplished by majority vote, i.e. by examining the results
of all frames in video segments where a single activity was performed, and calculating which activity was selected most often. An activity frequency histogram was created, and each activity bar was incremented by one each time a frame was classified as the corresponding activity. After all frames of the activity sequence were classified, the activity with the largest number of frame classifications was recorded as the classification result for the activity sequence.

6.1.1 MATLAB Testing

The MATLAB prototype was tested with the Blank data set for human actions [2] in order to determine the performance of the proposed activity recognition method. The data set contains nine subjects, and the activities chosen for testing were jumping (\textit{pj}ump), run, walk, \textit{wave}1, and \textit{wave}2.

Table 1 gives a description of each activity and how each was performed in the recording. These five activities were sufficiently different from one another, but some shared similar posture and orientation components that can make their differentiation difficult. For example, ‘walking’ is often very similar to ‘running’ in posture, but the speed at which the activity takes place is different. \textit{Wave}1 and \textit{wave}2 are one-handed waving and two-handed waving, respectively. These activities test the feature point analysis and correspondence having the arms raised over the head. \textit{Pjump} is jumping in place, and is renamed to \textit{jump} for simplicity. This activity shared similar postures with ‘walking’ when the arms and legs all align to form a single segment along the torso. Nine subjects, each performing the activities, comprise the data set.
<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Pjump</em></td>
<td>Subject is standing in place, stationary, then jumps up and down</td>
</tr>
<tr>
<td><em>Run</em></td>
<td>Subject jogs either from left to right, or right to left, in the camera’s view</td>
</tr>
<tr>
<td><em>Walk</em></td>
<td>Subject walks either from left to right, or right to left, in the camera’s view</td>
</tr>
<tr>
<td><em>Wave1</em></td>
<td>Subject stands stationary while waving either the right or left hand, from shoulder height to over the head</td>
</tr>
<tr>
<td><em>Wave2</em></td>
<td>Subject stands stationary while waving both hands, from shoulder height to over the head</td>
</tr>
</tbody>
</table>

Table 1: Activity Descriptions

An artificial neural network (ANN) was first used for the classification of frames. This network was trained and tested using the leave-one-out method. The network was trained on all of eight subjects’ activities and tested on the activities of the ninth subject. If the maximum output of the classifier was below 0.6, the frame was marked as unclassified. The test subject was rotated after each trial. Table 2 presents the results of the ANN classification on individual frames with a confidence threshold of 0.6. Table 3 presents the ANN classification results on the activity segments. Of the 2828 individual frames classified, 68% were classified correctly, and of the 45 activity segments classified, 96% were classified correctly.

To test the performance of the nearest neighbor (NN) classifier, similar tests were done on the data set. The training libraries were created using all the frames of all the activities of eight subjects. The classifier was tested on the frames of the ninth subject’s activities and the test subject was rotated after each trial. Table 4 presents the results of the NN classification on individual frames. A distance threshold of 50 was used as a
confidence measure. Table 5 presents the NN classification results on the activity segments. The tests were repeated without a confidence threshold, and the results are presented in Table 6 and Table 7. With the confidence threshold, 60% of individual frames and 89% of activity segments were classified correctly. Without the confidence threshold, 69% of individual frames and 89% of activity segments were classified correctly.

The results from nearest neighbor algorithm in Table 6 and Table 7 used a non-weighted Euclidean distance for the shortest distance calculation. Due to the camera resolution, the motion data gathered from the system were only deltas of very small numbers of pixels, on the order of 1 to 5 pixels. These measurements were overshadowed by the difference values calculated from the other limb components. It was decided that a weighting factor should be applied to the difference of the motion data during the library comparisons to better scale the classifier input. The tests were repeated with a motion weighting value of 30. Table 8 presents the results of NN classification with a confidence threshold of 50, and Table 9 presents the corresponding results for the activity segments. Table 10 presents the results of NN classification with no confidence threshold, and Table 11 presents the corresponding results for the activity segments. With the confidence threshold, 64% of individual frames and 93% of activity segments were classified correctly. Without the confidence threshold, 72% of individual frames and 96% of activity segments were classified correctly.
<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Test Results</th>
<th>Classified Activity</th>
<th>Number of Tests</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jump</td>
<td>Run</td>
<td>Walk</td>
<td>Wave1</td>
</tr>
<tr>
<td>Jump</td>
<td>306</td>
<td>9</td>
<td>28</td>
<td>58</td>
</tr>
<tr>
<td>Run</td>
<td>9</td>
<td>232</td>
<td>62</td>
<td>1</td>
</tr>
<tr>
<td>Walk</td>
<td>22</td>
<td>49</td>
<td>478</td>
<td>4</td>
</tr>
<tr>
<td>Wave1</td>
<td>26</td>
<td>1</td>
<td>17</td>
<td>480</td>
</tr>
<tr>
<td>Wave2</td>
<td>21</td>
<td>6</td>
<td>42</td>
<td>57</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: ANN Classification on Individual Frames, Confidence Threshold = 0.6

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Test Results</th>
<th>Classified Activity</th>
<th>Number of Tests</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jump</td>
<td>Run</td>
<td>Walk</td>
<td>Wave1</td>
</tr>
<tr>
<td>Jump</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Run</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Walk</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>Wave1</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>Wave2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: ANN Classification on Activity Segments, Confidence Threshold = 0.6

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Test Results</th>
<th>Classified Activity</th>
<th>Number of Tests</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jump</td>
<td>Run</td>
<td>Walk</td>
<td>Wave1</td>
</tr>
<tr>
<td>Jump</td>
<td>302</td>
<td>23</td>
<td>95</td>
<td>35</td>
</tr>
<tr>
<td>Run</td>
<td>20</td>
<td>108</td>
<td>70</td>
<td>3</td>
</tr>
<tr>
<td>Walk</td>
<td>38</td>
<td>56</td>
<td>515</td>
<td>15</td>
</tr>
<tr>
<td>Wave1</td>
<td>33</td>
<td>16</td>
<td>84</td>
<td>404</td>
</tr>
<tr>
<td>Wave2</td>
<td>18</td>
<td>6</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: NN Classification on Individual Frames, Confidence Threshold = 50

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Test Results</th>
<th>Classified Activity</th>
<th>Number of Tests</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Jump</td>
<td>Run</td>
<td>Walk</td>
<td>Wave1</td>
</tr>
<tr>
<td>Jump</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Run</td>
<td>0</td>
<td>7</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Walk</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>Wave1</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Wave2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: NN Classification on Activity Segments, Confidence Threshold = 50
### Table 6: NN Classification on Individual Frames, No Confidence Threshold

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Test Results</th>
<th>Classified Activity</th>
<th>Number of Tests</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump</td>
<td>Jump 304 24 97 76 37 0 538</td>
<td>57%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td>Run 20 264 73 5 12 0 374</td>
<td>71%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Walk</td>
<td>Walk 39 60 515 17 8 0 639</td>
<td>81%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wave1</td>
<td>Wave1 37 18 94 426 78 0 653</td>
<td>65%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wave2</td>
<td>Wave2 25 10 55 82 452 0 624</td>
<td>72%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>2828</td>
<td>69%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 7: NN Classification on Activity Segments, No Confidence Threshold

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Test Results</th>
<th>Classified Activity</th>
<th>Number of Tests</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump</td>
<td>Jump 8 0 0 0 0 1 9</td>
<td>89%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td>Run 0 7 2 0 0 0 9</td>
<td>78%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Walk</td>
<td>Walk 0 0 9 0 0 0 9</td>
<td>100%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wave1</td>
<td>Wave1 0 0 0 8 1 0 9</td>
<td>89%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wave2</td>
<td>Wave2 0 0 1 0 8 0 9</td>
<td>89%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>45</td>
<td>89%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 8: NN Classification on Individual Frames, Confidence Threshold = 50, Motion Weight = 30

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Test Results</th>
<th>Classified Activity</th>
<th>Number of Tests</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump</td>
<td>Jump 366 8 41 45 25 53 538</td>
<td>68%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td>Run 8 160 62 3 2 139 374</td>
<td>43%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Walk</td>
<td>Walk 28 45 526 26 11 3 639</td>
<td>82%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wave1</td>
<td>Wave1 73 5 66 400 43 66 653</td>
<td>61%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wave2</td>
<td>Wave2 39 1 28 62 364 130 624</td>
<td>58%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>2828</td>
<td>64%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 9: NN Classification on Activity Segments, Confidence Threshold = 50, Motion Weight = 30

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Test Results</th>
<th>Classified Activity</th>
<th>Number of Tests</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump</td>
<td>Jump 8 0 0 0 0 1 9</td>
<td>89%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td>Run 0 8 1 0 0 0 9</td>
<td>89%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Walk</td>
<td>Walk 0 0 9 0 0 0 9</td>
<td>100%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wave1</td>
<td>Wave1 0 0 0 8 1 0 9</td>
<td>89%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wave2</td>
<td>Wave2 0 0 0 0 9 0 9</td>
<td>100%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td>45</td>
<td>93%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Actual Activity</td>
<td>Test Results</td>
<td>Jump</td>
<td>Run</td>
<td>Walk</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>Jump</td>
<td>369</td>
<td>8</td>
<td>42</td>
<td>86</td>
</tr>
<tr>
<td>Run</td>
<td>8</td>
<td>277</td>
<td>74</td>
<td>4</td>
</tr>
<tr>
<td>Walk</td>
<td>29</td>
<td>45</td>
<td>527</td>
<td>26</td>
</tr>
<tr>
<td>Wave1</td>
<td>74</td>
<td>5</td>
<td>70</td>
<td>435</td>
</tr>
<tr>
<td>Wave2</td>
<td>42</td>
<td>12</td>
<td>35</td>
<td>115</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10: NN Classification on Individual Frames, No Confidence Threshold, Motion Weight = 30

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Test Results</th>
<th>Jump</th>
<th>Run</th>
<th>Walk</th>
<th>Wave1</th>
<th>Wave2</th>
<th>Unclassified</th>
<th>Number of Tests</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>89%</td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>89%</td>
<td></td>
</tr>
<tr>
<td>Walk</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>Wave1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>Wave2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>9</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>45</td>
<td>96%</td>
<td></td>
</tr>
</tbody>
</table>

Table 11: NN Classification on Activity Segments, No Confidence Threshold, Motion Weight = 30

### 6.1.2 Real-time Environment

The MATLAB prototype was recoded using C++ in order to utilize the OpenCV framework for image processing and to better meet real-time requirements for the system. The camera captured video frames from the test environment in real-time for processing and classification. Test subjects performed the same activities as in the Blank data set, and in addition to the five activities, a sixth activity for standing (*stand*) was added to the classification process. This activity was added to account for any idling in pose of a subject when in between activity actions. Four subjects, each performing the activities, comprise the data set.
The weighted version of the nearest neighbor classifier was used in the implementation of the real-time system, and no confidence threshold was specified. To test the validity of the real-time classifier, a library was created on all the frames of a single subject. The classifier was tested with the frames of the remaining subjects’ activities. Table 12 presents the results of the NN classification on individual frames. Table 13 presents the NN classification results on the activity segments. Results indicated that 52% of individual frames and 60% of activity segments were classified correctly.

### Table 12: NN Classification on Individual Frames, No Confidence Threshold, Motion Weight = 30

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Jump</th>
<th>Run</th>
<th>Stand</th>
<th>Walk</th>
<th>Wave 1</th>
<th>Wave 2</th>
<th>Unclassified</th>
<th>Number of Tests</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump</td>
<td>46</td>
<td>2</td>
<td>17</td>
<td>19</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>97</td>
<td>47%</td>
</tr>
<tr>
<td>Run</td>
<td>16</td>
<td>115</td>
<td>5</td>
<td>124</td>
<td>7</td>
<td>22</td>
<td>0</td>
<td>289</td>
<td>40%</td>
</tr>
<tr>
<td>Stand</td>
<td>72</td>
<td>11</td>
<td>149</td>
<td>60</td>
<td>2</td>
<td>39</td>
<td>0</td>
<td>333</td>
<td>45%</td>
</tr>
<tr>
<td>Walk</td>
<td>29</td>
<td>191</td>
<td>12</td>
<td>307</td>
<td>7</td>
<td>22</td>
<td>0</td>
<td>568</td>
<td>54%</td>
</tr>
<tr>
<td>Wave1</td>
<td>27</td>
<td>30</td>
<td>14</td>
<td>33</td>
<td>265</td>
<td>46</td>
<td>0</td>
<td>415</td>
<td>64%</td>
</tr>
<tr>
<td>Wave2</td>
<td>95</td>
<td>13</td>
<td>9</td>
<td>40</td>
<td>11</td>
<td>183</td>
<td>0</td>
<td>351</td>
<td>52%</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2053</td>
<td>52%</td>
</tr>
</tbody>
</table>

### Table 13: NN Classification on Activity Segments, No Confidence Threshold, Motion Weight = 30

<table>
<thead>
<tr>
<th>Actual Activity</th>
<th>Jump</th>
<th>Run</th>
<th>Stand</th>
<th>Walk</th>
<th>Wave 1</th>
<th>Wave 2</th>
<th>Unclassified</th>
<th>Number of Tests</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>Run</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0%</td>
</tr>
<tr>
<td>Stand</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>33%</td>
</tr>
<tr>
<td>Walk</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>100%</td>
</tr>
<tr>
<td>Wave1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>100%</td>
</tr>
<tr>
<td>Wave2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>67%</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
<td>60%</td>
</tr>
</tbody>
</table>
6.2  Example Frames

Select frames of the video sequences were captured to illustrate the processing steps taken in the activity recognition system. Section 6.2.1 presents frames taken from the test video sequences in the MATLAB prototype system, and section 6.2.2 presents frames taken from the real-time video streams in the OpenCV system.

6.2.1  MATLAB Prototype

The original frames are shown in (a), the silhouetted frames in (b), and the skeletonized frames in (c).
Figure 13: Test Jump Activity
Figure 14: Test Run Activity
Figure 15: Test Walk Activity
Figure 16: Test Wave1 Activity
Figure 17: Test Wave2 Activity
6.2.2 Real-time Environment

The original frames are shown in (a), the silhouetted frames in (b), the skeletonized frames in (c), and the six-segment model with feature points in (d).

Figure 18: Real-time Jump Activity
Figure 19: Real-time Run Activity
Figure 20: Real-time Walk Activity
Figure 21: Real-time Wave1 Activity
Figure 22: Real-time Wave2 Activity
Conclusions

The results of the tests on the MATLAB prototype indicate that the limb parameters extracted in each frame provide effective features for accurate classification of the activity performed. Both the ANN and k-NN classification algorithms yielded high classification accuracies of 96% for whole activity segment testing, and up to 72% classification accuracy on individual frames.

Furthermore, testing indicated that the algorithm lends itself well to a real-time environment. The algorithm operated in real-time with a slight processing delay needed to extract the limb components from the video frames and classify the frames as the corresponding activity. While limited testing indicated the classification accuracy was lower in the real-time environment due to space constraints of performing the activities and variations in the background subtraction results, the system correctly recognized activities in 60% of whole activity segments and 52% of individual frames.

7.1 Discussion of Results

The MATLAB prototype of the real-time system performed well on the test data set. The background was easily subtracted due to the plain background on which the test video sequences were filmed. The filming was also done with very little jitter of the
camera, making a non-adaptive background subtraction algorithm feasible. This allowed for the extraction of well-formed silhouettes and skeletons of the human figures, and led to limb components being extracted predictably and reliably for the classification process.

The performances of the two classifiers used were surprising good in overall accuracy. The ANN was developed to have a confidence threshold of 0.6 on the output of the classifier. This produced a 68% classification on individual frames and 96% classification on activity sequences. Removing the confidence threshold distance on the k-NN output increased the percentage of correct classifications of individual frames from 60% to 69%. With this in mind, it became apparent from looking at best match distances that the confidence threshold distance would have to be sufficiently high and greater than most smallest distance measurements, if not nonexistent, to ensure that a significant number of correct, smallest-distance classifications were not marked ‘Unclassified’.

This trend continued when a weighting factor was added to the motion difference calculations in the k-NN Euclidean distance calculations. Individual frame classifications rose from 64% to 72% with the removal of the confidence threshold distance, and on activity sequence classifications, accuracy rose from 93% to 96%. For this particular application of the NN algorithm, it appeared that the confidence threshold was not helpful to the classification process.

Including a weighting factor in the motion differences for the NN Euclidean distances calculations produced the most prominent effect on classification accuracy. Since the motion difference values were on such a small scale compared to the difference values of the other limb components, this weighting factor assisted in scaling the effects of each extracted value for more equal contributions from each parameter. Thus, the
motion contributions to the NN algorithm were not obscured. Adding the motion weighting factor to the calculations increased accuracy from 60% to 64% on individual frames and from 89% to 93% on activity sequences for a confidence threshold distance of 50. Corresponding increases in accuracies were made, 69% to 72% and 89% to 96%, for the no confidence threshold configuration.

In the real-time environment, the classification accuracy was lower than in the MATLAB prototype system. This can be attributed to a few factors. The MATLAB prototype system had very reliable background subtraction due to the plain background used in capturing the video sequences. The background used in the real-time system testing environment included objects, was less plain and had many more variations in color. This caused errors in the background subtraction algorithm where the head and forearms were not extracted due to the coloration of the door. Legs were sometimes not extracted due to the coloration of the filing cabinets, rug, and chair upholstery. It is possible that the library of known activities did not include enough variation to account for individual differences in how activities are performed.

The addition of the stand activity is very close in posture to jump, and it was seen that stand was often classified as jump in the testing environment. The posture in these two activities is quite similar with only vertical motion being a solid indicator of which activity is being performed. Because the background subtraction would often miss the head, the ratio for determining shoulder and hip centroid positions would change when a subject would be standing still, and this slight change would register as motion causing a jump classification.
There was also much more confusion between the *walk* and *run* activity in the real-time system. The number of frames classified correctly was about the same number of frames classified as the alternate activity. This did not provide good confidence that the classifier functions correctly. This was partly due to the background subtraction problems which cut off the head and forearms in some frames. These errors artificially created centroid motion, as in the case of *jump* and *stand*, which caused misclassification. Background subtraction in *walk* created artificial vertical motion, which is more characteristic of the *run* activity. Also, individual differences with the speed at which the two activities being performed may have further influenced the misclassification rates. A *run* performed at a slower pace may have been matched to a *walk* library template particularly if the motion values were unstable due to the background subtraction errors.

### 7.2 Future Work

There are a large number of potential uses for this type of human activity recognition system, and due to the nature of how this system was constructed, the limb parameter approach to representing the human figure can be extended to multiple figures in the frame, and to inanimate objects with which a human subject is interacting.

As the single human subject within a video stream can be classified as performing a particular activity, it is possible to use the same limb components approach to characterize multiple human subjects within a video stream, and classify each person’s activity. These activities can be done independently of the other subjects, such as the activities used in this research, or the activities can be inter-person, with interaction occurring among the subjects in the video. Analysis on the positioning of limb
components of each person could indicate if the activity is interactive in nature. This type of information could be useful in detecting situations where violence occurs, possibly one person assaulting another, or if there is some sort of object exchange between two people, which may indicate suspicious behaviors such as stealing, passing of weapons or drugs, and money exchanges.

In the security field, there is a demand for activity recognition systems that can detect human-object interactions. This type of activity detection system could determine if a person is taking an object away from the environment or if the person is leaving an object behind in the environment. Taking an object away from the environment could indicate that a person is stealing, and a detection system of this sort would be particularly helpful for security systems in consumer stores and where high-priced goods are small and could easily be hidden away. Leaving an object behind in the environment could indicate that a harmful object has been purposefully placed by a person with malicious intentions. This information would be especially helpful to security systems encountering and preventing terrorist activities. Even with false positives of detection, this type of human-object interaction detection would assist people with surveillance by alerting them to possible suspicious activities, and to reduce the potential of a crime going unnoticed.

There is still much to be done in the area of human activity recognition. Algorithms that must function in a real-time environment must possess the speed of processing necessary for such an environment, as well as the accuracy to correctly detect what type of behavior or activity a person is performing. Accuracy is crucial, and these detection systems may be the only warning of a threatening situation taking place if the
event goes unnoticed by surveillance personnel. Refining algorithms to streamline processing is the key to accomplishing this task. By analyzing data being extracted from video frames, only the necessary and sufficient information is required for the classification task. Refining what is the necessary and sufficient information will shape future fast and accurate human activity detection systems, and assist humans with important tasks of vigilance and surveillance.
Bibliography