6-30-2005

Vision-based hand shape identification for sign language recognition

Jonathan Rupe

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

Recommended Citation
Vision-Based Hand Shape Identification for Sign Language Recognition

by

Jonathan C. Rupe

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

Supervised by

Dr. Juan Cockburn
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, NY
April 2005

Approved By:

_____________________________________________
Dr. Juan Cockburn
Primary Advisor – R.I.T. Dept. of Computer Engineering

_____________________________________________
Dr. Andreas Savakis
Secondary Advisor – R.I.T. Dept. of Computer Engineering

_____________________________________________
Dr. Roxanne Canosa
Secondary Advisor – R.I.T. Dept. of Computer Science
Title: Vision-Based Hand Shape Identification for Sign Language Recognition

I, Jonathan C. Rupe, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

_________________________________
Jonathan C. Rupe

Date
Acknowledgements

I would like to take this opportunity to acknowledge the help of many people during the work on this thesis. I would like to thank my advisor, Dr. Juan Cockburn, for his continued guidance, support, and patient supervision throughout this thesis. I would also like to thank my committee members Dr. Andreas Savakis and Dr. Roxanne Canosa for their assistance. I am also very grateful for the many helping hands – literally! Thank you very much Yuriy Luzanov, Jared Holsopple, Sergey Katsev, Justin Hnatow, David Howig, Joseph Labatte, Jennifer Braun, Ruwan Budhagoda, Johnathan Ruppert, Eugene Pokhis, Stephen Joralemon, and Julie Christopher. Your time and the many, many images of your hands are greatly appreciated! I would also like to thank Jared and Yuriy a second time for their much-appreciated assistance with Support Vector Machines.
Abstract

This thesis introduces an approach to obtain image-based hand features to accurately describe hand shapes commonly found in the American Sign Language. A hand recognition system capable of identifying 31 hand shapes from the American Sign Language was developed to identify hand shapes in a given input image or video sequence. An appearance-based approach with a single camera is used to recognize the hand shape. A region-based shape descriptor, the generic Fourier descriptor, invariant of translation, scale, and orientation, has been implemented to describe the shape of the hand. A wrist detection algorithm has been developed to remove the forearm from the hand region before the features are extracted. The recognition of the hand shapes is performed with a multi-class Support Vector Machine. Testing provided a recognition rate of approximately 84% based on widely varying testing set of approximately 1,500 images and training set of about 2,400 images. With a larger training set of approximately 2,700 images and a testing set of approximately 1,200 images, a recognition rate increased to about 88%.
# Table of Contents

THESIS RELEASE PERMISSION FORM ................................................................. II
ACKNOWLEDGEMENTS ......................................................................................... III
ABSTRACT ................................................................................................................ IV
TABLE OF CONTENTS ............................................................................................ V
LIST OF FIGURES ..................................................................................................... VII
LIST OF TABLES ......................................................................................................... VIII
GLOSSARY ................................................................................................................ IX

CHAPTER 1 INTRODUCTION ............................................................................. 1

CHAPTER 2 BACKGROUND AND RELATED WORK ........................................ 2
2.1. AN OVERVIEW OF SIGN LANGUAGE RECOGNITION ................................ 2
2.2. GENERIC HAND GESTURE RECOGNITION SYSTEM ............................... 4
  2.2.1 Hand Gesture Modeling ............................................................... 5
    2.2.1.1 Three-Dimensional Hand Models ........................................... 6
    Volumetric Models ................................................................. 6
    Skeletal Models .................................................................. 7
  2.2.1.2 Appearance-Based (View-Based) Models ................................. 9
  2.2.1.3 Hand Model Parameters for Sign Language Recognition .......... 10
  2.2.2 Hand Gesture Analysis .............................................................. 12
    2.2.2.1 Feature Detection .............................................................. 12
    Hand Localization ............................................................... 12
    Hand from Arm Segmentation ............................................... 13
    Feature Extraction and Parameter Computation ....................... 14
  2.2.3 Hand Gesture Recognition .............................................................. 15
  2.3. SUPPORT VECTOR MACHINES ........................................................... 17
    2.3.1 Two-Class Support Vector Machines ..................................... 18
      2.3.1.1 Training ........................................................................ 18
    2.3.1.2 Classification ................................................................. 21
    2.3.2 Multi-Class Classification .................................................... 21
      2.3.2.1 Combining Binary Classifiers ......................................... 22
      2.3.2.2 Considering all Data at Once ......................................... 24
  2.4. GENERIC SHAPE REPRESENTATION TECHNIQUES ......................... 25
    2.4.1 Global Contour Shape Representation Techniques ................... 27
      2.4.1.1 Simple Descriptors ......................................................... 27
      2.4.1.2 Shape Signatures .......................................................... 27
      2.4.1.3 Statistical Moments ...................................................... 28
      2.4.1.4 Scale Space ................................................................. 28
      2.4.1.5 Spectral Descriptors ...................................................... 29
      2.4.1.6 Alternatives ................................................................. 30
    2.4.2 Region-Based Shape Representation Techniques ...................... 30
List of Figures

Figure 1: Stokoe’s Phonetic Breakdown of ASL ............................................................... 3
Figure 2: Vision-based Gesture Recognition System ....................................................... 5
Figure 3: Spatial Hand Models ......................................................................................... 6
Figure 4: Skeletal Structure of the Human Hand............................................................... 8
Figure 5: Hand Gesture Analysis Stage ........................................................................... 12
Figure 6: Wrist Segmentation ......................................................................................... 14
Figure 7: Classes that are separable with linear hyperplane ......................................... 18
Figure 8: Classes that are linearly separable but contain errors ...................................... 20
Figure 9: Classes that are not linearly separable ............................................................ 20
Figure 10: The Directed Acyclic Graph for a Four Class Problem ................................. 24
Figure 11: Breakdown of Shape Representation Techniques ......................................... 26
Figure 12: (a) A Shape in Polar Space; (b) The Polar-raster Sampled Shape From (a) .... 33
Figure 13: Block Diagram of Pre-Processing Stage ....................................................... 39
Figure 14: Block Diagram of the Wrist Detection Stage ................................................ 41
Figure 15: Valid Wrist Candidate Locations .................................................................. 42
Figure 16: Wrist Candidates Disregarded Due to Distance From Maximum Wrist Width ...................................................................................................................... 43
Figure 17: Block Diagram of Feature Extraction Stage ................................................... 44
Figure 18: (a) A Sample Hand Image with the Forearm Removed; (b) Its Resulting Polar-raster Sampled Image ...................................................................................... 45
Figure 19: Block Diagram of Classification Stage .......................................................... 46
Figure 20: Hand Shape Test Set ....................................................................................... 50
Figure 21: Images Which Show Variations in Perspective Angle .................................. 51
Figure 22: Images that Show Variations in the Scale ..................................................... 52
Figure 23: Images that Show Variations in the Orientation ........................................... 52
Figure 24: Successful Wrist Detection ............................................................................ 54
Figure 25: Borderline Failed Wrist Detection ............................................................... 54
Figure 26: Illustration of the Wrist Detection’s Tolerance to Noise and Blurriness ...... 55
Figure 27: An Example Case Where No Wrist is found Due to No Forearm Showing .... 56
Figure 28: Example Where the Wrist is found with Sufficient Forearm ......................... 56
Figure 29: Successful Wrist Detections at Different Scales ............................................ 57
Figure 30: Successful Wrist Detection with Little Curvature at Wrist ............................. 58
Figure 31: Wrist Detection Failures When Fingers Overlap Wrist ................................. 59
Figure 32: Wrist Detection Failure When Wrist is Bent and Hand Mass is Greater than Forearm Mass ............................................................................................... 60
Figure 33: Successful Wrist Detection When Wrist is Bent But Forearm Mass is Greater than Hand Mass ............................................................................................... 60
Figure 34: Wrist Detection Failure When Insignificant Forearm is Showing and Other Possible Wrist Locations Exist ............................................................................. 60
Figure 35: Wrist Detection Accuracy .............................................................................. 61
List of Tables

Table 1: High-Level Block Diagram of Hand Shape Recognition System ................. 38
Table 2: Recognition Rates as Radial and Angular Frequency are Varied. Tested with RBF Kernel, $\gamma = 1/k$, $C = 1$, Sampling Factor = 5. ......................................................... 63
Table 3: Visual Effects of Varying the Sampling Factor for Polar Hand Images .......... 64
Table 4: Effect of the Sampling Factor on the Recognition Rate .............................. 65
Table 5: Effect of the Sampling Factor on Feature Extraction Time ......................... 65
Table 6: Effects of $\gamma$ in the RBF and Polynomial Kernels ................................. 68
Table 7: Effects of Cost Parameter ($C$) in the RBF and Polynomial Kernels .......... 68
Table 8: Comparison of Classification Approaches .................................................. 69
Table 9: Recognition Rates for Each Hand Shape Using Cramer & Singer Approach w/RBF Kernel ........................................................................................................... 70
Table 10: Timing Results for Each Stage .................................................................. 72
Table 11: Summary of Results of Kernel Types, Classification Approaches, and Sampling Factor ........................................................................................................... 73
Glossary

Phoneme

The smallest phonetic unit in a language that is capable of conveying a distinction in meaning.

Analysis-by-synthesis

A technique used to estimate the hand’s posture by synthesizing a 3-D model of the hand and then varying its parameters until the projection of the model in the image plan and the real hand image appear as the same visual image.

Neural Network (NN)

A device modeled after the human brain, in which several interconnected elements process information simultaneously, adapting and learning from past patterns. NNs are commonly used for modeling and recognizing patterns found in data collections.

Hidden Markov Model (HMM)

A probabilistic network capable of finding patterns that appear over time. It is like a finite state machine, but where the transitions and output are probabilistic. HMMs are commonly used in pattern recognition systems.

Support Vector Machine (SVM)

A supervised learning classification technique that separates classes by a maximal-margin hyperplane. Highly used in pattern classification.

Generic Fourier Descriptor (GFD)

A descriptor obtained by extracting various scaled Fourier coefficients from the 2D Fourier transform of the image obtained when circularly sampling an object in an image up to its maximum radius.

Sampling Factor

A parameter used when circularly sampling an object in an image to obtain the polar-raster sampled image (polar image). This parameter determines the width of this polar image based on the maximum radius found in the source image. This, in effect, determines the circular sampling frequency.
Polar-Raster Sampled Image (Polar Image)

The image obtained after circularly sampling an object in an image up to its maximum radius.

Radial Frequency

A parameter used when extracting the GFD. It determines, in conjunction with the angular frequency, which coefficients are extracted from the 2D Fourier transform of the polar image.

Angular Frequency

A parameter used when extracting the GFD. It determines, in conjunction with the radial frequency, which coefficients are extracted from the 2D Fourier transform of the polar image.

Inflection Points

Points on a function where the concavity changes from down-to-up or up-to-down.

Euclidean Norm

A norm computed by summing the squares of all data points and taking the square root.
Chapter 1   Introduction

Humans use gestures in communication every day. They are used not only in conjunction with speech to further reinforce its meaning, but they are also used to describe things that cannot be portrayed by speech alone. Research in automatic recognition of human communication has predominantly been focused on speech recognition and handwriting, while research in the automatic recognition of gestures has lagged behind. However, there has been a recent surge in the interest of the automatic recognition of human gestures. Some of the more structured forms of gestures are those that are performed in sign language. Sign language is a specific area of human gesture communication and a full-fledged complex language that is used by various Deaf communities around the world. In fact, more than half a million people within the United States use American Sign Language (ASL), many as their primary method of communication. Unfortunately, there is a tremendous lack of non-Deaf people who have an in-depth knowledge of sign language, which leads to the social isolation of the Deaf community. This has brought forth motivation for the development of a computational system capable of automatically interpreting sign language. Since sign language is a structured form of gesture communication, the development of such a system would also be beneficial in human-computer interaction (HCI), virtual reality, and robotics.
Chapter 2  Background and Related Work

2.1. An Overview of Sign Language Recognition

One of the biggest challenges in sign language recognition (SLR) is to find a modeling technique that is powerful enough to capture the language, yet it can scale to large vocabulary size. There are approximately 6000 cataloged signs in American Sign Language, and many signs can appear in many different forms depending on subject, object, and numeric agreement [1]. With the different forms in which the same sign can appear, the number of possible cases to consider increases to a degree much larger than 6,000. In many hand gesture recognition systems the entire gesture is modeled, however, in sign language recognition, where the language is very large, it is not feasible to model each sign. It is more feasible to break down signs into a limited set of primitive parts, phonemes, that can be combined to make up the entire set of signs in ASL. This procedure enables each of the phonemes to be modeled separately as is done in speech recognition [2].

Since the early 1960’s research has been done on the linguistics of American Sign Language. Fortunately, one of the most significant findings of the properties of ASL, first proposed by William Stokoe, was that American Sign Language could be broken down into phonemes [3]. Stokoe defined three types of phonemes: location – where on the body the sign takes place, hand shape – how the fingers are articulated, and movement – how the hands move [3]. Figure 1 depicts each of these types of phonemes. Figure 1 (A) depicts the area in which signs take place (location), Figure 1 (B) depicts a
small set of hand shapes (the ASL alphabet), and Figure 1 (C) depicts a small set of movements performed when signing various signs in ASL.

Figure 1: Stokoe’s Phonetic Breakdown of ASL

(A) Depicts the area in which signs take place. (B) Depicts a small set of hand shapes (the ASL alphabet). (C) Depicts a small set of movements: (a) upward (b) downward (c) rightward (d) leftward (e) toward signer (f) away signer (g) nod (h) supinate (i) pronate (j) up and down (k) side to side (l) twist wrist (m) circular (n) to and fro. Additional movements not depicted consist of synchronous and asynchronous two-handed movements of the hand and finger movements (e.g. wiggling fingers). The solid ellipse, dashed ellipse and dashed arrow represent the initial hand location, the final location and the path taken respectively.

Since Stokoe’s findings, additional theories have been proposed on the linguistics of ASL. The most significant of these discoveries include two additional types of phonemes: hand orientation and facial expressions [2].

Thus, in order to develop a system capable of interpreting sign language, it is important for the system to be able to determine the position, orientation, shape, and movement of the hand. Before attempting to develop a system capable of recognizing sign language, it is helpful to first understand the components of a generic hand gesture recognitions system.

---

2.2. **Generic Hand Gesture Recognition System**

In recent years, there has been a tremendous amount of research on hand gesture recognition. Some of the earlier gesture recognition systems attempted to identify gestures using glove-based devices that would measure the position and joint angles of the hand [4]. However, these devices are very cumbersome and usually have many cables connected to a computer. This has brought forth the motivation of using non-intrusive, vision-based approaches for recognizing gestures. Vision-based approaches involve using one or more video cameras to capture a person gesturing and using computer vision techniques to interpret each particular gesture. A vision-based gesture recognition system can be broken down into three main components: *hand gesture modeling*, *hand gesture analysis*, and *hand gesture recognition*. The gesture model describes how the hand gesture is to be represented. The type of application desired has a significant impact on the type of model that must be chosen. If an application with only a small number of gestures is needed, then a simple model can be used. However, in an application with a large gesture set, such as sign language recognition, a more detailed model will be required. However, there are trade-offs in choosing a more complex model as will be discussed later. After choosing a model, analysis is performed to compute the model parameters from the image features that are extracted from the video input streams. The analysis stage is followed by the recognition phase, which classifies the model parameters, representative of a specific gesture, while taking into consideration the model and in some cases grammar. The following figure shows a diagram of a generic vision-based gesture recognition system.
The following sections describe the three main components of a vision-based gesture recognition system in more detail.

### 2.2.1 Hand Gesture Modeling

Gestures, particularly in sign language, involve significant motion of the hands. Thus, in developing a sign language recognition system, it is important to model both the motion (temporal characteristics) and shape (spatial characteristics) of the hand. While modeling the motion of the hand is imperative for sign language recognition, it is out of the scope of this thesis and is left for future work. Since only the spatial characteristics of the hand is of concern, temporal modeling of the hand will not be described here.

---

2 Image taken from [4]
Discussions on temporal hand modeling can be found in [4] and [8]. Spatial modeling can be divided into two major categories: 3-D model-based approaches and appearance-based or view-based techniques, as shown in the following figure. The following sections describe these two model types in greater detail.

**Figure 3: Spatial Hand Models**

### 2.2.1.1 Three-Dimensional Hand Models

Three-dimensional models attempt to infer the 3-D pose of the hand. 3-D hand models are classified into two major groups: volumetric models and skeletal models.

**Volumetric Models**

Volumetric models aim to describe the 3-D appearance of the hand, as it would appear in real-life. They are commonly used in computer animation but have been recently used in computer vision applications [4]. Volumetric models are employed in vision-based hand gesture recognition by the approach of *analysis-by-synthesis*. *Analysis-by-synthesis* estimates the hand’s posture by synthesizing the 3-D model of the hand, and then varying its parameters until the projection of the model on the image plane and the real hand image appear as the same visual image [4], [8]. Some volumetric models represent the surface of the hand with B-splines [10]. These are most popular in

---

3 Image taken from [4]
the field of computer animation since they are quite realistic [4]. However, these are too complex to be rendered in real-time gesture recognition applications. An alternative to these is in the use of geometric shapes such as cylinders, spheres, ellipsoids, and hyper-rectangles to approximate the parts of the hand. These parts can then be combined to model the entire hand.

Several systems [10], [11], [12], [23] have been proposed which use the approach of analysis-by-synthesis. Among these, the fastest frame rate achieved is 27 Hz. However, this was only achievable with a PC cluster consisting of six, Pentium-III, 600 MHz PC’s. While these models give a fairly realistic representation of the hand, they require many parameters. Obtaining these parameters with computer vision based techniques can be quite complex and time-consuming, which generally restricts these types of models from real-time use. In addition, these hand models are user dependent, since the model should be calibrated for each user, and thus they can only give approximate estimations [8].

**Skeletal Models**

Skeletal models represent the 3-D structure of the hand. While volumetric models require many parameters to accurately represent the actual appearance of the hand, skeletal models use a greatly reduced set of parameters to describe the structure of the hand. In order to understand skeletal models, it is first important to understand the structure of the human hand, shown in Figure 4.
Figure 4: Skeletal Structure of the Human Hand

The human hand consists of 5 fingers, each of which contains three joints. Except for the thumb, there are two degrees of freedom for metacarpophalangeal (MCP) joints and one degree of freedom for proximal interphalangeal (PIP) joints and distal interphalangeal (DIP) joints [8]. Taking into account all degrees of freedom for each joint and also considering global hand pose, the human hand has roughly 27 degrees of freedom [7], [8]. In skeletal models, each finger is represented as a kinematics chain where the palm is its base reference frame, the fingertips are the end-effectors, and inverse kinematics is involved in computing the joint angles [8]. However, a unique solution to the inverse kinematics problem cannot be guaranteed [8] and computation is rather complex.

4 Image taken from [8]
Several systems have been proposed which use skeletal models [24], [25], [26]. Among systems that reported their operating rates, frame rates ranged between 8 fps and 0.02 fps (45 minutes per frame)! Due to the computational complexity, skeletal models are not suited well for real-time gesture recognition applications.

2.2.1.2 Appearance-Based (View-Based) Models

The second major type of hand models is known as appearance-based models. Appearance-based models are derived directly from the information contained in the images. There are a variety of appearance-based models: those based on deformable templates, those that use hand image sequences, and those that use other image features such as shape representation features and image eigenvectors.

Some appearance-based models are based on deformable templates. Deformable templates are the set of points on the outline or region of an object, used for interpolation to approximate the outline or region of an object [7]. Deformable templates consist of internal and external parameters. Internal parameters consist of an average set of points that describes the shape along with variability parameters that allow the shape to be deformed. External parameters are used to describe the global motion of the hand, which is generally described with rotations and translations. Two such systems which incorporate deformable models include [27] and [28]. In [28], a deformable template is used for hand tracking. However, the system suffers from scale and rotation confusion and “implausible model shapes” [28]. In [27] the hand shape is represented with a 3-D deformable template (Point Distribution Model). The results averaged angle differences of the 3-D hand posture of about 10° – 20°, which is sufficiently accurate for sign language recognition [27]. However, the system requires that the model and the hand in
the image overlap [27]. The method has also not been applied to sign language recognition.

Some appearance-based models are based on hand image sequences. Gestures, with these types of models, are depicted by a sequence of images themselves [7]. Motion history images (MHIs), images formed by the combination of the motion of pixels in a series of images over time, are one example. While systems that use these types of models may be good for small gesture sets, it would not be feasible to use such a model for a system such as sign language recognition that has a very large gesture set.

Most appearance-based models are based on parameters of the hand image. Models under this category use parameters such as shape contours, image moments, and image eigenvectors. The position of fingertips in the image has also been used to help distinguish between hand gestures. These types of parameters have been widely used for sign language recognition. Some examples of systems that use these types of appearance-based parameters are described in the following section.

2.2.1.3 Hand Model Parameters for Sign Language Recognition

Although 3-D hand models capture more detailed information about the hand, due to their complexity and high computation costs, they have not been a practical hand model for real-time sign language recognition. In addition, knowledge of the exact hand shape parameters seems unnecessary for sign language recognition [7]. Also, since 3-D model-based approaches capture 3-D data, they often require the use of multiple cameras (two cameras for stereo vision or multiple cameras usually positioned orthogonally to each other) [19], [24], [41]. The use of multiple cameras, in addition to making a more
A pricey system, usually involves camera calibration, which results in great computational costs. Appearance-based approaches rarely involve using multiple cameras.

As a result, several sign language recognition systems have leaned towards the use of appearance-based models [29–35]. For representing the shape of the hand [29] and [31] used the distances of all the fingers of the hand to each other. However, these systems used gloves with colored markers on the fingertips, palm of the hand, and back of the hand to easily locate the hand and fingertips. The area of the colored markers was also used to give an indication of hand shape. Other parameters were used to represent the location and orientation of the hand. The system developed in [29] was capable of recognizing 97 signs with an accuracy of 91.7% and [31] was able to recognize 262 signs with an accuracy of 94%. In [30] several parameters were used to help distinguish between hand shapes. These parameters included: the flatness of the hand region, the area of the hand region, and the number of protrusions of the hand. Other parameters were used to represent the location and orientation of the hand. This system was capable of recognizing 70 signs with an accuracy of 91.4%. In [32] the hand’s area, length of the eigenvector of the hand blob, and eccentricity of the bounding ellipse were used as appearance-based hand model parameters. This system also used other parameters to represent the location and orientation of the hand. It achieved a recognition accuracy of 92% and 98% depending on whether the point of view of the camera was from the desk or attached to the signer’s hat. In [33] the eccentricity of the bounding ellipse, area, and length of the major eigenvector were used as hand-model parameters. With the use of other parameters to describe the hand’s location and orientation, this system achieved a recognition accuracy of 91.3% with 40 signs. In [34] the length of the hand contour was
used to represent the hand shape. In [35] a wavelet-based feature set was used to estimate the pose of the hand. It was reported to have very high accuracy (99.9%) for detecting the 24 static hand shapes of the ASL alphabet. However, testing details were not given and the speed was not reported.

2.2.2 Hand Gesture Analysis

The goal of the analysis stage is to detect and measure features from the video-sequence images in order to estimate the model parameters. The analysis stage can thus be broken down into two major components: feature detection and parameter estimation. The following figure shows a block diagram of the analysis stage.

![Figure 5: Hand Gesture Analysis Stage](image)

2.2.2.1 Feature Detection

The main task in the analysis stage is to detect relevant image features to be used to compute the model parameters. Before these hand features can be obtained, the hand in the image must first be detected and located in a process called localization or segmentation. After the hand region is detected, various features are computed, which are then used to determine the model parameters.

**Hand Localization**

The localization of the hand in the image can be done in several ways or in a combination of ways. One of the more popular ways of localizing the hand in the image is to use color segmentation. Because of the characteristic color of human hand, it can
often be used solely to segment the hands in images. In the cases where gloves are worn, this color segmentation is trivial and very effective. There are many different techniques used to perform color segmentation including color histogram matching [36], look-up tables, and the use of probabilistic models. Alternatively, the motion of a user’s hand in images can be used to help detect the hand region from images. Several systems use standard background subtraction [38] and other techniques [36], [37] to detect areas of motion in images in order to locate the region of the hands. Color segmentation, motion, and other visual indicators can be combined in another approach known as fusion. By combining these techniques, a more robust hand localization technique can be developed. In [36] color histogram segmentation and motion information is fused to locate the hands in images.

Hand from Arm Segmentation

Often times, images of hands contain portions of the user’s wrist. To prevent the estimation of the hand shape from being altered by existence of wrist information, it is important to separate the hand from the arm. In many systems, this task is avoided by wearing long-sleeved shirts [15], [16]. However, it is desirable to implement a method to segment the hand from the arm so that it is not necessary to pose any additional restrictions on the system. Previously used techniques to distinguish the hand from the arm are based on the width of the wrist and the contour segmentation. One width-based wrist locating technique is based on the observation that the width of the arm is thinnest at the location of the wrist [17]. This can be shown in the following figure.
Another width-based wrist location technique is based on the observation that there is a sudden change in the width of the arm at the location of the wrist, as can also be seen in Figure 6 [18]. By detecting the location of this sudden change, the location of the wrist can be identified. The wrist can also be located using a contour-based method. In a contour-based method, it can be seen that there is a sharp turn in the contour at the location of the wrist on the thumb-side of the hand [18].

**Feature Extraction and Parameter Computation**

After locating the hands in the images, it is necessary to extract the required features to be used for computing the model parameters. Although various models have different parameters, the same feature can sometimes be used to compute several parameters. Some common features extracted include hand silhouettes [12], [41], contours [34], key points distributed along hand (fingertips, joints, palm) [25], [42], [43], [45], and distance-transformed images [44]. Silhouettes have been used as features for parameter computation in both 3-D based models [12], [41] and appearance based models. In [41], the silhouette features from multiple viewpoints are combined to construct a “voxel model” which is then used to estimate the hand’s joint angles.

---

5 Image taken from [18]
Contours can also be used to compute both 3-D model-based parameters and appearance-based model parameters. Image-contour to model-contour matching is another way contours can be used in 3-D model-based approaches [7]. In appearance-based approaches, contours are often used to compute shape signatures. Key hand point features are also used in appearance-based models and 3-D based models. In [43], a 3-D model-based approach, eight key hand points (5 fingertips, wrist, middle finger joint, and thumb joint) are used to estimate the hand’s joint angles. In [29], an appearance-based approach, the locations of colored markers located at the fingertips, palm, and back of hand are used to compute fingertip and hand distance parameters. In [44], a distance-transformed image is used to compute palm size and hand direction parameters. In some appearance-based model approaches, the features extracted are treated as the actual hand parameters used in the recognition stage. For example, in [34], the hand contour feature is used directly as a parameter for the recognition phase.

### 2.2.3 Hand Gesture Recognition

In the gesture recognition phase, the parameters computed from the analysis stage are recognized as a specific gesture. Methods for recognizing gestures can be divided down into two major categories: rule-based approaches and learning-based approaches [2]. In rule-based approaches, humans find relations between gesture parameters and manually encode their relationships [2]. Given a set of parameters derived from an input gesture, the parameters are compared to the encoded rules to determine the gesture. A major disadvantage of this technique is that it is very difficult for humans to find relations between parameters in a high-dimensional space. Machine learning-based approaches help overcome this limitation by finding the relations between the high-dimensional
parameters and the gestures. Generally, for static gesture recognition, a technique derived from vector quantization is used to cluster the parameters; where an $n$-dimensional space is partitioned into convex sets using $n$-dimensional hyper-planes based on training examples [7]. Some metric, such as Euclidean distance, is used to determine the nearest neighbor for classification. However, in cases where the parameters are clustered into non-convex sets, a nonlinear clustering technique is used.

Neural Networks (NNs) and Hidden Markov Models (HMMs) are machine learning-based classification approaches that have been widely used in gesture recognition. A neural network is a data-modeling tool capable of capturing and representing both linear and non-linear input/output relationships. Neural networks can recognize patterns found in a given data collection and are capable of modeling that data. A Hidden Markov Model is a probabilistic network capable of finding patterns that appear over time. The goal of HMM is to predict what is observed within an underlying hidden system. In the example of someone trying to predict the weather based on the state of a piece of seaweed, the observation would be the state of the seaweed and the actual weather would be the hidden system. HMMs consist of hidden states and observable states. In the seaweed example, a HMM would describe the weather in relation to the state of the seaweed and would also include a sequence of seaweed state observations. The hidden states (the weather) consist of probabilities of transitioning from itself to the next state, and is only dependent upon the current state and probability. The observable states (the state of the seaweed) produce outcomes during transitions between the hidden states or while the model is in one of its hidden states.
Both Neural Networks and Hidden Markov Models have been applied to the gesture recognition problem. In [16], [31], and [33] HMMs have shown promising results, however, they are not without problems. One problem arises in defining the topology of the HMM – that is the number of states and transitions. The topology of HMMs relies on trial and error or educated guesses, which can be rather difficult when modeling a large gesture set [2]. Additionally, the HMM assumes that transitions occur from one state to the next. This results in poorer recognition rates for processes that do not follow this order [7]. While Neural Networks have not been as widely used for gesture recognition as Hidden Markov Models, they have shown to work nearly as well as HMMs for gesture recognition [49].

2.3. Support Vector Machines

Support Vector Machines (SVM) are a supervised learning classification technique that has not been given a lot of attention in gesture recognition systems. However, SVMs have been receiving increasing attention, in recent years, as an approach to high performance pattern classification. Support Vector Machines were originally designed as binary classifiers, but later extended to perform multi-class classification. An example binary classification may be to determine whether a test element is a face or not. In this case, there are two classes: faces and not faces. The SVM determines a function that separates the two classes. Later the function is used in classification of unknown images. First, an overview of binary SVM will be described. Then, a short overview of how SVM can be extended for multi-class classification will be given.
2.3.1 Two-Class Support Vector Machines

In order to perform classification with SVM a function that separates the classes has to be obtained, this is called the training stage. Only after the training stage can classification be performed.

2.3.1.1 Training

During the training phase, the SVM is presented with a large set of training data (training examples) consisting of a pair: a vector \( x \) and the class label \( y \) of the training element. The vector \( x \) consists of features or attributes to describe the training element. For example, it could be a vector of pixel values in an image. The class labels would often be either a 1 or a \(-1\) indicating whether or not the given training element is of the desired class or not. A training set of \( l \) elements can be represented as follows:

\[
T = \{(x_1, y_1), \ldots, (x_l, y_l)\}, \quad \text{where } x \in \mathbb{R}^n, \ y \in \{1,-1\}
\]  

(1.1)

Using this training data, the SVM creates a model that can be used during testing to predict the class of a given test element. In order to find a model capable of classification, the SVM finds a linear hyperplane, which separates the training data into two separate classes as shown in the following figure.

![Figure 7: Classes that are separable with linear hyperplane.](image-url)
There are several hyperplanes, which can separate the data, however, SVM find the one optimal hyperplane that maximizes the margin (maximizes the distance between the separating hyperplane and the nearest data point(s) (training vectors) of each class), as shown in the figure above. The training points \( x \) that lie on separating hyperplane satisfy:

\[
<w, x> + b = 0
\]  

(1.2)

Where \( w \) is the normal to the hyperplane, \( <*,*> \) denotes inner product, \(-b/\|w\|\) is the perpendicular distance from the hyperplane to the origin, and \( \|w\| \) is the norm of \( w \) [47]. The training vectors, which lie on the two hyperplanes, H1 and H2, are called support vectors (the support vectors are circled in the diagram above). Only these support vectors are required for the solution, since the removal of any of the support vectors would alter the solution found [47]. However, any other training points (vectors) can be thrown out. These support vectors are then used later during the testing phase for classification.

In the above description it is assumed that the training examples are linearly separable. However, when the training data is not linearly separable, which is usually the case as shown in Figure 8 and Figure 9 an additional approach must be taken. There are two main approaches to handling training data that is not linearly separable. The choice of the approach is dependent upon the prior knowledge of the training data. In the case where the data is expected to be generally linearly separable, as shown in Figure 8, an additional cost function associated with the misclassification errors can be added [5]. A user definable parameter, often called \( C \), allows the user to assign a penalty to these errors. The larger the chosen value \( C \), the higher the penalty of errors [47]. A more detailed explanation of the introduction of this cost function can be found in [47] and [5].
In the case where the training data is not expected to be linearly separable, as shown in the following figure, the data is mapped into a higher dimensional space where it can be linearly separated.

This non-linear mapping is affected by what’s known as a kernel function. Common kernel types in addition to the linear kernel, include polynomials, and radial basis functions [47].
2.3.1.2 Classification

During the testing phase, the model obtained during training is used to classify a particular test element. Classification uses the model to determining which side of the hyperplane the test example lays. Based on which side the test example resides, the appropriate class label is assigned to the test element. The sign of the following formula, sometimes called the decision function, is used for classification in the general case:

\[ f(x) = \sum_{i=1}^{N_s} \alpha_i y_i K(s_i, x) + b \]  

(1.3)

Where \( N_s \) is the number of support vectors, \( \alpha_i \) are the Lagrange multipliers determined during training while maximizing the margin, \( K(s_i, x) \) is the kernel used to provide an efficient encoding of the inner product in a higher dimensional space where the data is linearly separable, and \( s_i \) are the support vectors, also determined during training. The existence of \( b \) in the above formula is determined by the kernel type. The result of the above function is a sign: positive or negative indicating the predicted class of the test element.

2.3.2 Multi-Class Classification

There are two types of techniques used for multi-class SVM. One is by combining several binary classifiers while the other is by considering all the data in one optimization formula.
2.3.2.1 Combining Binary Classifiers

Some of the major methods based on solving multiple binary classifiers include the one-against-all-method, the one-against-one method, and the Directed Acyclic Graph Support Vector Machines (DAGSVM) method.

The one-against-all method (also referred to as the one-versus-rest method) is the standard method for multi-class SVM. It creates \( k \) models, one for each of the \( k \) classes. For each class, \( i \), the \( i \)th of \( k \) SVM is trained with all elements of its own class as positive test elements and all other classes as negative test elements [53]. A decision function like the one in Eq. (1.3) is used for each of the \( k \) classes, except the sign of the function is not taken. A given test element is classified as belonging to the class that has the largest value of the decision function. One disadvantage of the one-against-all method is that the training time is very large due to the fact that each SVM is trained with all the training elements.

The one-against-one method also combines two-class classifiers; it creates a model for every pair of classes. This results in \( k(k-1)/2 \) models, where \( k \) corresponds to the number of classes. The following example is given to better illustrate this idea. Consider an SVM consisting of the following 5 classes: \{A, B, C, D, and E\}. An SVM with these classes will train \( 5(5-1)/2 = 10 \) models consisting of all possible (but non-repeating) pairs of classes. These pairs would include: AB, AC, AD, AE, BC, BD, BE, CD, CE, and DE. There are different ways for classifying a test element using the constructed models. One possible method of classification is to take the sign of the decision function like the one in Eq. (1.3) with a given test element, for each of the \( k(k-1)/2 \) decision functions. If the result of the sign says it is in the first class, then a tally or
vote is incremented for that class. If the result of the sign says it is in the other class, then a vote is incremented for that other class. After going through all class-pairs, the class with the highest vote is predicted as being the class of the given test element. This classification scheme is known as the Max Wins method [48]. An advantage of the one-against-one over the one-against-all method is in the greatly reduced training time. Although many more models are trained, since each class requires significantly fewer training elements (only those for the two classes being trained), the overall training time is significantly less (in most cases). [48] shows a “typical case” where all of the one-against-one SVM can be trained in about the same time as two one-against-all SVM.

Directed Acyclic Graph Support Vector Machines (DAGSVM) is another multi-class classification approach that combines multiple binary classifiers. It is similar to the one-against-one method in that it uses the same training phase; however, the classification phase uses a tree like structure – a “Rooted Binary DAG (Directed Acyclic Graph).” [48]. The graph has $k(k-1)/2$ nodes, one for each of the classifiers, and $k$ leaves corresponding to the $k$ classes. When classifying a test element, the classifier at the root node is evaluated. Based on the decision of the evaluation, the test element will either follow the tree along the left edge or right edge to the next node. The classifier at this node is then evaluated sending the test element along a path to another classification node. When the test element reaches a leaf node, the evaluation terminates and the predicted class becomes the class of the leaf node reached. A total of $k-1$ decisions are required to classify a test object. An example DAG for a four class system is shown in the figure below.
Although the manner in which the tree is constructed matters, the order of the classes in the list does not \cite{48}. One of the advantages of DAGSVM is that the testing time is less than that of the one-against-one method, and yet it still retains comparable results. This is due to the fewer number of decisions required to classify an object.

### 2.3.2.2 Considering all Data at Once

An alternative approach to multi-class classification is to combine all the training data into one optimization problem as opposed to several two-class problems. This approach is similar to the one-against-all approach where each class is separated out from all the other classes, and there are also \( k \) decision functions \cite{53}. However, all of this information is combined into solving one problem. The classification is the same as the one-against-all method where the class with the largest decision value is chosen as the predicted class. A disadvantage of considering all the training data at once is that it suffers from a much greater training time \cite{53}. However, this approach often results in fewer support vectors and thus generally faster classification times \cite{53}.

\footnote{Image captured from source [XX DAGSVM]}
2.4. **Generic Shape Representation Techniques**

Since the complexity of 3-D hand model-based approaches generally prevents their use in real-time applications, an appearance-based hand modeling approach seems better for use in a gesture recognition system, such as sign language recognition. Therefore, it is helpful to first investigate possible ways in which generic shapes can be represented in attempts to find a technique that may be applied for hand shape representation for sign language recognition. The following section discusses various ways for representing the shape of arbitrary objects.

There are numerous ways of representing the shape of various objects. A recent and very extensive review of shape representation techniques has been done by Zhang in [46]. He found that they could largely be classified into contour-based representation techniques and region-based representation techniques, which can each be further subdivided down into global and local based approaches. Contour-based representation techniques only capture information about the boundary of an object, while region-based representation techniques capture information about the entire region. Structural shape representation techniques depict the shape of a region by breaking it down into sub-parts, while global representation techniques describe the shape of the region as a whole [46]. The following figure shows this taxonomy:
Since structural shape representation techniques represent the shape of objects with small primitive parts, there are several drawbacks from these types of representation techniques. One disadvantage is that the types of primitive parts of the shape of the object must be known ahead of time – the success of these structural methods depends on this knowledge [46]. Another drawback of structural shape representation techniques is that it fails to capture the global shape of an object. Therefore, two objects, which are similar from a structural perspective, may look completely different in the global aspect. Global shape representation techniques do not have this problem. Another major drawback of structural representation approaches is that they are not robust. A small variation in the boundary of an object can result in very large changes to the structural description of that object. This problem is not as significant for global shape representation methods. Additionally, structural approaches often involve high

---

Figure 11: Breakdown of Shape Representation Techniques

Image captured from source [46].
computational complexities and difficult matching [46]. Because of the drawbacks of structural shape representation techniques, they are not a practical representation scheme for use in sign language recognition. While structural shape representation techniques are not covered in this paper, a review of these techniques can be found in [46]. The following sections will cover some global contour-based and region-based shape representation techniques and discuss some of their advantages and disadvantages.

2.4.1 Global Contour Shape Representation Techniques

There are numerous global contour shape representation techniques. Some of these descriptors include simple descriptors, signatures, statistical moments, scale space, and spectral descriptors.

2.4.1.1 Simple Descriptors

Some simple descriptors include boundary length, curvature - the rate of change of slope, and bending energy – the energy necessary to bend a “rod” to the desired shape [40]. However, these are not very accurate and can only distinguish shapes to a limited degree.

2.4.1.2 Shape Signatures

An alternative way to represent an image contour is with a shape signature. Shape signatures are one-dimensional representations of the boundary of a shape. There are many different types of shape signatures. One of the easiest ways to represent the 2-D contour as a signature is to represent each coordinate pair as a complex number such that

\[ s(k) = x(k) + jy(k) \]  

(1.4)
Where \( s(k) \) is the signature and \( x(k) \) and \( y(k) \) are the point coordinates [22]. This signature, and generally most other signatures, is translation invariant [22]. Many other signatures are scale invariant and can also be presented in a rotation invariant way [46]. Since shape signatures are sensitive to noise and slight changes in the boundary can cause misclassification, the use of shape signatures alone is not desirable.

2.4.1.3 Statistical Moments

Statistical moments such as the mean, variance, and higher-order moments can also be used to describe the contour of objects [22]. Given a contour shape signature \( z(i) \), the \( r^{th} \) moment \( m_r \) and the \( r^{th} \) central moment \( \mu_r \) can be estimated as [40]:

\[
m_r = \frac{1}{N} \sum_{i=1}^{N} [z(i)]^r
\]

\[
\mu_r = \frac{1}{N} \sum_{i=1}^{N} [z(i) - m_1]^r
\]

The basic idea behind this approach is to reduce the description into a one-dimensional function. Statistical moments can be translation, rotation, and scale invariant. The equations given above can be normalized as shown in [40] to achieve invariance to translation, rotation, and scaling. Moments are also easy to implement and they carry a “physical” representation of the boundary shape, but not when the moment is of high order [22], [46].

2.4.1.4 Scale Space

Scale space offers an alternative representation of shape that deals with problems that arise from the fact that shape description varies with scale and different representations result at different resolutions. In a curve or boundary, some inflection
points may appear at one resolution, but disappear at another resolution. The scale space
approach tracks these inflection points after applying a unique Gaussian smoothing
kernel to the 1-D curvature function over a range of sizes and then differentiates the
result twice [40]. Inflection points that remain present in the representation are said to be
“significant.” The main drawback from the scale space representation technique is that
the implementation and matching of these descriptors are rather complex. [46]

2.4.1.5 Spectral Descriptors

Shape descriptors such as the Fourier descriptors (FD) and wavelet descriptors
(WD), that analyze shape in the spectral domain, fall under this category. Spectral
descriptors have the advantage that, since they are analyzed in the spectral domain, they
are less sensitive to noise and boundary variations. Given a shape signature $s(k)$, the
discrete Fourier transform (DFT) of $s(k)$ is [22]:

$$a(u) = \frac{1}{K} \sum_{k=0}^{K-1} s(k) e^{-j2\pi ku/K} \quad \text{for } u = 0, 1, 2, \ldots, K - 1 \quad (1.7)$$

The complex coefficients of $a(u)$ are the Fourier descriptors [22]. Different
shape signatures, such as those that use complex coordinate as explained in the shape
signature section above, can be used in the computation of Fourier descriptors. By
themselves, Fourier descriptors are not invariant to geometric changes, however they can
be made invariant to translation, rotation, and scaling [40]. Fourier descriptors have
several advantages that make them attractive; they are easy to compute, are simple to
normalize and thus match, have a physical meaning, and capture local and global features
[46]. While wavelet descriptors have the advantage over the Fourier descriptors in that
they are able to analyze data at different resolutions better [40], there are drawbacks
which make WD’s less attractive for use in shape representation. The most significant disadvantage is that the wavelet representation uses a complex matching procedure making this type of application less appealing [46].

2.4.1.6 Alternatives

Several other shape representation techniques exist, such as elastic matching and stochastic method, as mentioned in [46], however, these methods suffer from implementation and matching complexity and expensive operations.

2.4.2 Region-Bases Shape Representation Techniques

Region-based shape representation techniques take into consideration the interior of objects as well as the border. Global region-based shape representation techniques are less affected by noise. Some region-based shape representation techniques include simple shape descriptors, regular and geometric moments, Zernike moments, and the generic Fourier descriptor.

2.4.2.1 Simple Descriptors

Some simple descriptors include area, compactness (circularity) – defined as \( \frac{(\text{perimeter} \cdot \text{length})^2}{\text{area}} \), and eccentricity – the ratio of major and minor axes [40]. As with simple contour descriptors, they can only distinguish shapes to a limited degree.

2.4.2.2 Moments

Moment representations interpret a normalized gray-level image function as a probability density of a two-dimensional random variable whose properties can be described using statistical characteristics [40]. Moments capture global information
about an object without requiring closed boundaries. Regular moments are defined as follows:

If \( f(x, y) \) is a digital image, the moment of order \((p + q)\), dependent on scaling, translation, rotation, and gray-level transformations is given by [22], [40]:

\[
m_{pq} = \sum_{x=-\infty}^{\infty} \sum_{y=-\infty}^{\infty} x^p y^q f(x, y) \quad \text{for } p, q = 0, 1, 2, \ldots
\]  

(1.8)

A set of translation, rotation, and scaling invariant moments can be derived by using nonlinear combinations of the lower order regular moments – a set of moments usually called geometric moments [39]. However, only a few invariant moments derived from the lower order regular moments is not sufficient for accurately describing the shape of an object, and the desired higher order invariant moments are quite difficult and computationally expensive to derive [39], [6].

To overcome these problems associated with regular moments, Zernike moments and other orthogonal moments, based on the theory of orthogonal polynomials, have been introduced [39]. Orthogonal moments are simply extensions of regular moments where the \( x^p \) and \( y^q \) in Eq. (1.8) are replaced with orthogonal polynomials [46]. Orthogonal moments are capable of shape reconstruction with good accuracy and are able to do so with little information redundancy [39].

In [39] studies have been done to compare the capability of image representation of orthogonal and non-orthogonal moments and also their sensitivity to image noise and their aspects of information redundancy. It was concluded that higher order moments are more sensitive to noise and orthogonal moments such as Zernike moment are better than other types of moments in terms of information redundancy. In terms of overall
performance, the Zernike moments outperformed the others. For this reason, it has been adopted by MPEG-7 as a region-based shape descriptor.

2.4.2.3 Generic Fourier Descriptor

Although the Zernike moment descriptor (ZMD) is a robust shape descriptor and has been reported to perform very well in comparison to many other region-based shape descriptors [20], it has several drawbacks. One of the drawbacks of Zernike moment descriptors is that its kernel is difficult to compute. In addition, the shape must be normalized into a unit disk before computing the moments. When compared to Fourier descriptors, the performance of the ZMD was significantly worse [49] in terms of computational speed. Additional drawbacks are described in [46]. The Generic Fourier Descriptor (GFD), proposed by Zhang and Lu [20] is said to overcome these drawbacks. The GFD is simpler to compute than the ZMD, the features of the GFD are derived in the spectra domain, and the GFD has better retrieval performance due to the nature of multi-resolution analysis in the radial and circular directions of the shape [46].

The generic Fourier descriptor is obtained by extracting various scaled Fourier coefficients from the 2-D Fourier transform of the polar-raster sampled image [46]. The polar-raster sampled shape image is an image that is created by circularly sampling the object or area in an image. The dimensions of this image correspond to the radius of the object/area, $r$, and the circular (angular) sampling frequency, $\theta$. A sample polar-raster sampled image is shown the following figure.
The purpose of converting the spatial hand image into a polar image is to achieve both translation invariance and rotation invariance. The rotation of an image in Cartesian space (i.e. the rotation of the hand in the original image) results in a circular shift in polar space (i.e. a shift of the hand in the polar image) \[20\]. Based on the translation property of the Fourier transform, a shift in the spatial domain results in a phase change in the frequency domain \[20\]. In other words, a rotation of the original hand image, results in phase change in the frequency domain due to the conversion of the original image into a polar image. If the phase is then ignored, by only retaining the magnitudes of the Fourier coefficients, then rotation invariance can be achieved. Translation invariance is achieved by using the centroid as the origin in polar space when converting the original image into a polar image.

After a polar-raster sampled image is obtained, the DFT of this polar image is then computed. This following formula describes this \[46\]:

\[
PF_x(\rho, \phi) = \sum_r \sum_i f(r, \theta_i) e^{j2\pi \left(\frac{r}{\rho^2 + \theta_i^2} \frac{2\phi}{T} \right)}
\]  

(1.9)

where \(0 \leq r < R\) and \(\theta_i = i(2\pi / T)\) \((0 \leq i < T)\); \(0 \leq \rho < R\), \(0 \leq \phi < T\). \(R\) and \(T\) are the radial frequency resolution and angular frequency resolution respectively. Subsets of the

---

\(^8\) Image captured from \[46\]
Fourier coefficients obtained when computing the DFT are used to obtain the GFD. The following equation describes which coefficients are used to extract the Generic Fourier Descriptor (GFD) and how these features are derived from the Fourier coefficients.

\[
GFD = \begin{bmatrix}
PDFT(0,0) & PDFT(0,1) & PDFT(0,n) & PDFT(m,0) & PDFT(m,1) & PDFT(m,n)
\end{bmatrix}
\begin{bmatrix}
\text{Area}^{-1} & PDFT(0,0) & PDFT(0,0) & \ldots & PDFT(0,0) & PDFT(0,0) \\
PDFT(0,0) & PDFT(0,0) & PDFT(0,0) & \ldots & PDFT(0,0) & PDFT(0,0)
\end{bmatrix}
\] (1.10)

In the above equation, \( PDFT \) corresponds to the Fourier coefficients from the DFT of the polar image, \( m \) is the maximum number of radial frequencies selected, and \( n \) is the maximum number of angular frequencies selected. The selection of \( m \) and \( n \) determines how coarsely or finely the image is represented; the larger the selection of \( m \) and \( n \), the more finely the image is represented. The total number of features extracted is equal to \( m \times n \). As previously stated, by only using the magnitudes of the coefficients, rotation invariance is achieved since the phase information is ignored. Scale invariance is achieved by dividing the magnitude of the DC component \( PDFT(0,0) \) by the area of the polar image and by dividing all the other coefficients by the magnitude of the DC component.

2.4.2.4 Grid and Shape Matrix Based Techniques

There are several alternative ways of representing shape, such as grid based methods and shape matrices, as are discussed in [46]. With the grid based method, a grid of cells is placed over a shape and a bitmap is created by scanning the cells and assigning a 1 to the cells covered by the shape and a 0 to the cells not covered by the shape [46]. The shape is first normalized by scaling the shape to the size of the grid, shifting it into the upper left corner, and rotating it about its major-axis [46]. While this method provides a simple shape representation, it suffers due to the unreliability of its major-axis
rotation normalization and is not rotation invariant for region-base shapes [46]. The shape matrix technique is similar to the grid-based technique, except that circular grid is used instead of a rectangular grid. The shape is normalized by the maximum radius of the circle providing scale, rotation, and translation invariance. While this technique overcomes some of the problems with the grid-based technique, it lacks robustness with sparse sampling and expensive shape matching [46].

2.4.3 Summary of Shape Representation Techniques

Contour-based methods provide an effective way of representing the shape of an object and are more popular than region-based methods. However, there are several drawbacks to contour-based shape representation approaches. One drawback is that contour-based methods are sensitive to noise since only the boundary of the shape is used. Also, in many applications, the interior of the object is valuable, and sometimes contains information that is more valuable than the contour. Region-based methods can overcome these drawbacks. Region-based shape representation techniques do not suffer as much from noise as contour-based methods since the interior information of the object is taken into consideration. Even though region-based methods take the interior of the region into consideration, they are not necessarily more complex than contour-based methods.

Shape representation methods that work in the spatial domain suffer from noise sensitivity and high dimensionality. These drawbacks can be resolved by using histograms, moments, scale space, and spectral transforms [46]. Although the use of histograms and scale space helps to alleviate the effects of noise and high dimensionality, it comes at the cost of expensive matching [46]. Moments offer a robust and compact
representation, but higher order moments lack physical meaning. Solutions in the spectral domain, especially the Fourier transform, offer the most promising solution.

2.4.4 Shape Representation Techniques for Sign Language Recognition

In sign language recognition, it is desirable to use a shape representation technique that will sufficiently describe the shape of the hand while also being capable of fast computations, enabling the recognition to be done in real-time. It is also desirable for the technique to be invariant to translation, rotation, and scaling. In addition, a method that will allow for easy matching would be beneficial.

In [20] Zhang performed comprehensive tests comparing some of the more prominent contour-based descriptors and region-based descriptors. His studies compared the shape descriptors in terms of good retrieval accuracy, compact features, general application, low computational complexity, robust retrieval performance, and hierarchical coarse-to-fine representation. His tests involved matching shapes from the MPEG-7 database that have undergone changes in translation, orientation, and scale. His research concluded that for contour-based descriptors, the Fourier descriptor was the best of the techniques tested. For region-based shape descriptors the Zernike moment descriptor and the generic Fourier descriptor were the best approaches with regards to the fore mentioned aspects.

Since the interior of the hand is important for distinguishing between many hand shapes, especially closed-fist hand shapes, a region-based shape descriptor is favorable over a contour-based shape descriptor for hand shape recognition, because region-based shape descriptors encapsulate information about the interior of objects. Among the
region-based shape descriptors, the Generic Fourier Descriptor is one of the more promising descriptors due to its ability to accurately describe shape and its speed and ease of computation. The following section describes the system developed to identify hand shapes commonly found in American Sign Language using the region-based Generic Fourier Descriptor.
Chapter 3 System Description

3.1. System Overview

A system was developed to recognize hand shapes from American Sign Language. A new wrist detection algorithm was developed to remove the forearm from the hand in the input images. An appearance-based hand-modeling paradigm using the region-based Generic Fourier Descriptor (GFD) shape representation technique proposed by Zhang in [20] was implemented to describe the shape of the hand. The classification of the hand shapes was performed with Support Vector Machines.

The hand shape recognition system that was developed is composed of four major stages: pre-processing, wrist detection, feature extraction, and classification. Pre-processing is done first on the image to remove noisy pixels. This is followed by wrist detection where the forearm is removed from the hand to allow for a more accurate representation of the hand shape. The wrist detection is followed by feature extraction where various features are extracted from the image and used to represent the hand shape. With these features, the hand shape is then classified as a certain hand shape. The following figure shows a high-level block diagram of the system.

Table 1: High-Level Block Diagram of Hand Shape Recognition System
3.1.1 Pre-Processing

Before wrist detection is done on the hand image and before features are extracted from the hand shape, pre-processing is done to condense the representation of the image and to clean up the image and eliminate noise. The pre-processing of the input image consists of color to grayscale conversion, segmentation of the hand from the background, and opening. This is shown in the following block diagram.

![Block Diagram of Pre-Processing Stage](image)

The system takes, as input, color images and converts them to grayscale. Following the color-to-grayscale conversion is the segmentation process where the hand is separated from the background of the image by a simple threshold operation with a constant value. Following the segmentation is opening. This removes random noise in the background of the image, which would corrupt the wrist detection and feature extraction stages.

3.1.2 Wrist Detection

The wrist detection stage detects the location of the wrist in the cleanly segmented hand image and then removes the forearm from the hand at this wrist location. The forearm is removed from the image so that the forearm does not obstruct the hand shape recognition process.

The wrist detection procedure works by finding the locations along the hand region where there is a downward to upward change in concavity, while searching the
hand region in the direction from the tip of the hand to the end of the forearm. Points on a function where the concavity changes from down-to-up or up-to-down are known as inflection points. Using the second derivative of a function, points of upward and downward concavity and thus inflection points can be found. This can be illustrated by introducing the Concavity and Inflection Point Theorems. The Concavity Theorem says that if the function $f$ is twice differentiable at $x = c$, then the graph of $f$ is concave upward at $(c, f(c))$ if $f''(c) > 0$ and concave downward if $f''(c) < 0$ [50]. The Inflection Point Theorem says that if $f'(c)$ exists and $f''(c)$ changes sign at $x = c$, then the point $(c, f(c))$ is an inflection point of the graph of $f$. If $f''(c)$ exists at the inflection point, then $f''(c) = 0$ [50]. Thus, inflection points can be found where sign of the second derivative of a function changes at the boundaries of a point, and the second derivative of this point is zero.

Since the hand is wider than the wrist, and the forearm is thinnest at the wrist and widest near the elbow, then the wrist location is one of these inflection points. Other points of inflection may appear in other areas of the hand region depending on how the hand is shaped. Erroneous inflection points are removed via means described below. The predicted wrist location is chosen out of the remaining inflection points (wrist candidates) based on their width and location along the hand region.

The following figure shows a block diagram of the steps in the wrist detection stage.
Before a wrist width function is obtained to find inflection points in, the hand region is dilated. This is done to close any partially opened regions for the following hole filling operation, and also to fill cracks so that a clean wrist width function can be obtained. Next, holes are filled and the orientation of the hand region is computed. Using the dilated hole-filled binary hand image and the orientation of the hand in the image, a function is obtained representing the width along the entire hand region. This width function is then used to find the inflection points. Since inflection points are found by taking the second derivative of a function, the second derivative of the wrist function was taken to find candidate wrist points. However, since the width function is generally noisy and taking derivatives also amplifies noise, the wrist function and its first and second derivative are filtered. The filtering of each of these functions is critical in order to eliminate many false candidates caused by noise. The filtering is done with a simple averaging filter given a specified window. This window size is dynamically computed based on a third order polynomial. This polynomial was determined by testing the filter with various window sizes using a test set consisting of about 125 images: 5 images of each hand shape found in the ASL alphabet (minus J). Hand images in this test set ranged in an overall size of approximately 35 to 250 pixels from the tip of the hand to the
end of the forearm. The following third order polynomial was calculated to best fit these test results and is used to determine the window size.

\[ f(x) = -0.0000001x^3 + 0.000006x^2 - 0.013x + 1.888, \quad x = \text{hand region width} \quad (1.11) \]

After computing and filtering the second derivative of the wrist width function, the inflection points were found. However, only inflection points with a concavity in the down-to-up direction were considered due to the fact that the width of the hand region is smaller at the wrist location and larger at the hand and forearm, instead of larger at the wrist location and smaller at the hand and forearm.

In most hand images, there is more than one wrist candidate (inflection point). In order to eliminate false candidates, only candidates that are to the forehand side of the maximum width point are considered. This is demonstrated in the following figure:

![Figure 15: Valid Wrist Candidate Locations](image)

Restricting the candidates to the forehand side of the point of the maximum width makes the assumption that the maximum width occurs where the hand is located. This is the case for all signs in the alphabet and numbers 1 through 10 and is almost always true. Wrist detection will fail in the rare cases where the forearm is greater in width than any width in the hand, such as the case where the hand is in a flat position and the wider portion of the forearm is visible in the hand image.
In addition, all candidates that have a width that is less than 25% the maximum width in the hand image are immediately eliminated to remove erroneous candidates. Any candidates less than 25% the value of the maximum width of the hand could not possibly be a wrist, even with the hand shape of the number five stretched to the max.

Candidates whose location is further than 96% of the distance between the maximum width and the end of the forearm are eliminated as potential wrist candidates. This is shown in the following image.

![Figure 16: Wrist Candidates Disregarded Due to Distance From Maximum Wrist Width](image)

The removal of these candidates is done to eliminate false candidates that may appear way at the ‘end’ of the forearm. The value of 96% was determined based on testing experiments performed on the 125-image test set described above. The only case in which a valid wrist candidate would be removed is in the case that there is very little forearm showing. However, in this case, the small amount of forearm showing would not have a significant impact on the hand detection procedure and thus is not an issue.

After eliminating the invalid candidates, the final predicted wrist location is chosen based on its width and its distance to the maximum width location. A width ratio and a location ratio are computed for each candidate. These are defined as follows:

\[
Width\_ratio = \frac{\text{candidate\_width}}{\text{maximum\_width}} \quad (1.12)
\]
\[
Location\_ratio = \frac{candidate\_location - max\_width\_location}{end\_forearm\_location - max\_width\_location}
\] (1.13)

The Euclidean or L-2 Norm of these ratios is computed for each candidate. The Euclidean Norm is computed as follows:

\[
|\mathbf{x}| = \sqrt{|x_1|^2 + ... + |x_n|^2}, \text{ where } \mathbf{x} = (x_1, x_2, ..., x_n) \text{ on } \mathbb{R}^n
\] (1.14)

The candidate with the smallest Euclidean Norm is chosen as the final predicted wrist location. After the wrist location has been found, the forearm is removed based on that wrist location. Pseudocode for the wrist detection procedure is listed in the Appendix.

### 3.1.3 Feature Extraction

The feature extraction stage consists of obtaining features of the hand shape in the input image. The features are computed from the hand region after the wrist detection stage has completed so that the features are representative of the actual hand shape and not corrupted by the forearm. The Generic Fourier Descriptor (GFD) is used as the hand feature. The GFD is obtained by taking the 2-D Fourier Transform of the polar-raster sampled hand image. A particular subset of the Fourier coefficients is scaled and extracted as the hand features. A detailed explanation of Generic Fourier Descriptor can be found in Section 2.4.2.3.

The following figure shows a block diagram of the feature extraction stage.

![Block Diagram of Feature Extraction Stage](image)

Figure 17: Block Diagram of Feature Extraction Stage
The first step in the feature extraction stage is to get the centroid and the maximum radius so that the polar-raster sampled image can be obtained. The maximum radius and the centroid can then be used to sample the segmented hand image in a radial direction, where the centroid is the origin and is sampled to a maximum radius found, to obtain a polar-raster sampled image. The segmented hand image is sampled such that a polar image is obtained with a height equal to the radius of the hand and a width equal to a factor of that radius. The optimal factor is determined during testing based on the recognition accuracy obtained with various scaling factors. The following figure shows a sample hand image and its resulting polar image using a width multiplication factor of 5.

![Sample Hand Image and Polar Sampled Image](image)

**Figure 18:** (a) A Sample Hand Image with the Forearm Removed; (b) Its Resulting Polar-raster Sampled Image

After the polar-raster sampled image has been obtained, the 2-D Fourier transform of the polar image is computed in following step. In the last step of the feature extraction stage, the actual hand features (the GFD) are computed from the Fourier transform of the
polar image. The features that are extracted are based on the specified radial and angular frequencies and these features are all scaled by the area image or the DC component. The optimal radial and angular frequencies were determined based on testing experiments comparing the angular and radial frequency to the recognition rate. These testing results can be found in Section 4.4.

A polar-raster sampled image is used in order to allow for rotation invariance. By taking the magnitude of the Fourier coefficients in combination with the use of the polar image, rotation invariance is attained. In addition, by using the centroid as the origin in the polar-raster sampled image, translation invariance is achieved. The scaling of the coefficients of the 2-D Fourier transform of the polar-raster sampled image results in scale invariance. Details on the rotation, scale, and translation invariance of the GFD can be found in Section 2.4.2.3.

### 3.1.4 Classification

The classification stage predicts the class of a given test element. Classification is done with Support Vector Machines. Since there are many different hand shapes, a multi-class SVM is used where each hand shape corresponds to a different class. Section 2.3 presents an overview of two-class and multi-class Support Vector Machines. The following figure shows a block diagram of the classification stage.

![Block Diagram of Classification Stage](image)

Figure 19: Block Diagram of Classification Stage
The classification stage predicts the class of the test element based on the features obtained from the image and the SVM function parameters (SVM model) and minimum and maximum feature values both obtained during the training phase. The training phase is described in Section 3.1.5. When a test hand image is input, the system extracts hand image features from the test image as described in the feature extraction section above. These features are then scaled between –1 and 1 based on the minimum and maximum feature values obtained during training. These minimum and maximum feature values are stored in the scale file. The scaled features are then passed on to the SVM classifier along with the required SVM function parameters (support vectors, etc…) needed for classification with the Support Vector Machine. These parameters are stored in the model file that was also created during the training phase. Details on how the SVM classification is done are described in Section 2.3.

3.1.5 Training

In order to be able to classify a hand shape using Support Vector Machines, the system must be trained. Training, as described in Section 2.3.1.1, teaches the SVM how to separate the hand classes given testing examples. The training of the SVM requires a series of features to describe each hand shape as well as a class label to define the class where the hand shape belongs. For each image in the training set, hand features were extracted via the feature extraction stage as described in the above section, and these values along with the appropriate hand class were written to a training file. After generating this training file, the support vectors in the file were scaled between –1 and 1 depending on minimum and maximum values of each of the features across all support vectors. This was done to improve the performance of the SVM. Using the scaled hand
features and associated class labels, the Support Vector Machine was then trained to generate a model for use during the classification stage.

The training of the SVM was done with both LIBSVM [51] and BSVM [52], libraries for classification with Support Vector Machines. The LIBSVM library uses the one-against-one approach for multi-class classification as described in Section 2.3.2.1 above, and the BSVM library uses a combined classification (all-at-once) approach as described in Section 2.3.2.2. Both classifiers were tested to determine the optimal classification technique for this system. These results are shown in Section 4.5.
Chapter 4 Results

The hand recognition system is composed of four major stages: pre-processing, wrist detection, feature extraction, and classification. Testing results were obtained and are reported for each of these stages. The testing setup is described in the following section. Following the description of the test setup are explanations of the results obtained from each stage.

4.1. Test Setup

Although the system was developed to recognize 31 hand shapes, a test set was created consisting of 33 hand shapes - 25 of the 26 letters (A-Z, minus J) of the ASL alphabet, and the numbers 1, 2, 3, 4, 5, 7, 8, and 10 (eight number hand shapes). The hand shape J is the same as the hand shape for I, except for the sign for J involves motion. Since this system does not take motion into consideration, this letter was disregarded. In addition, the hand shapes for the numbers 0, 6, and 9 are the same as the hand shapes for the letters O, W, and F respectively. Because of this, these numbers were combined with their associated letters. The following figure shows each of these hand shapes.
Figure 20: Hand Shape Test Set
For each of the 33 hand shapes, 10 images of each hand shape were captured from 13 different people. This resulted in 13 image sets each composed of 330 images per set (10 images per hand shape times 33 hand shapes). Overall, the entire database of hand shapes consisted of 4,290 images (13 image sets times 330 images per set). Each of these images was taken with a black background to simplify the segmentation process.

Each of these images was taken at different angles, scales, and orientations. The hand images were captured at different angles where the angle of the hand varied in both the front-to-back direction and side-to-side direction. Examples of these differences in angles are shown in the following images, all of the letter A.

![Figure 21: Images Which Show Variations in Perspective Angle](image)

Large variations in the angle of the hand were used to help generate a more robust and practical database for training and testing.

Images were also taken where the scale of the hand in the image varied. Examples of images that vary in scale are shown in the following figure.
Figure 22: Images that Show Variations in the Scale

Both of these hand shapes are of the letter A, but they are both taken at different scales.

Images were also taken at arbitrary orientations. The following figure demonstrates the different orientations that the testing and training images were captured at. Each image in the example is of the letter C, however, each hand shape is tilted at different orientations. Also notice that the differences in angles at which the images were captured at.

Figure 23: Images that Show Variations in the Orientation
The images can be captured at different scales and orientations without causing problems due to the scale, translation, and rotation invariance of the GFD as described in Section 2.4.2.3.

The hand recognition system was tested on a Dual 2.66 GHz Pentium-4 Xeon machine with 2 GB of RAM. The system was running the Gentoo Linux operating system. The OpenCV [14] Image Processing Library from Intel was used to provide common image functionality. Implementations of the Support Vector Machine training and classification algorithms were used from the LIBSVM and BSVM libraries [51], [52].

4.2. Pre-Processing

The pre-processing stage simply consisted of grayscale color conversion, hand from background segmentation (a simple threshold operation), and opening. These operations provided sufficient pre-processing in order to completely eliminate all unwanted background pixels and prepare the image for the wrist detection stage. The pre-processing stage had a very fast execution rate of approximately 380 fps.

4.3. Wrist Detection

The wrist detection stage locates the wrist location in a hand image and removes the forearm from the hand. The wrist detection procedure worked very well in most cases. It is tolerant to noise and blurriness, works well regardless of the amount of forearm shown, works well with both large and small hand regions, and detects wrist locations even when there is very little change in the width of the hand at the location of the wrist. The wrist detection suffers in cases where the fingers are bent down over in
front of the wrist, where the hand is bent significantly at the wrist while only a small amount of forearm showing, and when there is not enough forearm showing to detect a wrist while other wrist candidates could be found in other locations of the hand. Examples of all of these cases are demonstrated below, but first an explanation of what is considered successful wrist detection vs. unsuccessful wrist detection is given.

### 4.3.1 Successful Vs. Unsuccessful Wrist Detection

The determination of successful or unsuccessful wrist segmentation was determined based on human observation. To demonstrate what is considered successful and unsuccessful wrist detection, the following images are used. The following figure demonstrates cases in which the wrist detection was completed successfully.

**Figure 24: Successful Wrist Detection**

The following figure demonstrates cases in which the wrist detection was borderline failure.

**Figure 25: Borderline Failed Wrist Detection**
4.3.2 Robustness of the Wrist Detection Procedure

The wrist detection procedure works very well even under poor conditions. This is because of the preprocessing done on the image before the wrist detection procedure and the filtering done on the wrist width function extracted. The following figure illustrates the wrist detection procedure’s tolerance to noise and blurriness.

![Figure 26: Illustration of the Wrist Detection’s Tolerance to Noise and Blurriness](image_url)

Even with very blurry and noisy images, the wrist detection worked very well. As can be seen in the images on the right hand side, there is much less noise in the images where the forearm is removed due to the preprocessing. Any additional noise that was
not cleaned up during the preprocessing stage is eliminated during the filtering step of the wrist detection stage to allow for robust wrist detection.

The wrist detection algorithm was generally not affected by the amount of forearm that is visible. In some rare cases, however, the amount of visible forearm did have a negative impact on the wrist detection procedure. This is described in Section 4.3.3. In cases where very little forearm is visible, as is shown in the following images, the wrist detection does not find a wrist location and the hand as left as is. In this case, since there is such little forearm, it would not make a difference as to whether or not the forearm was removed.

Figure 27: An Example Case Where No Wrist is found Due to No Forearm Showing

However, even if only a small portion of forearm were showing; if this amount was significant, the wrist detection procedure would locate the wrist and remove the forearm as shown in the following figure.

Figure 28: Example Where the Wrist is found with Sufficient Forearm
The wrist detection algorithm was also developed to work well with images where the size of the hand region is in the range of approximately 50 pixels to 250 pixels. Hand regions larger and smaller than this still work, but as the hand region size increases significantly beyond these ranges, the accuracy of the wrist detection declines. To tolerate significantly larger or smaller images, the polynomial defined for determining the filtering window size (Eq. (1.11)) would need to be recalibrated. Wrist detection worked extremely well in hand images where the size of the hand region was towards the higher end of the range. Although the wrist detection still worked well with images where the size of the hand region was towards the lower end of the scale, the wrist detection did not work quite as well as those with larger sized hand regions. The following images show successful wrist detection in cases where the hand image is very large and very small.

Figure 29: Successful Wrist Detections at Different Scales
In Figure 29 (c) & (d) the original images were resized for testing the wrist detection with small hand sizes. The approximate size (in length) of the hand regions (not image dimensions) in the above (source) images are 325 pixels, 350 pixels, 45 pixels, and 31 pixels for images (a), (b), (c), and (d) respectively.

The wrist detection procedure also worked well even with hand images where there was very little change in the width at the location of the wrist as compared to the widths of the forearm and hand. This is demonstrated with the following illustration.

Figure 30: Successful Wrist Detection with Little Curvature at Wrist

4.3.3 Problems with the Wrist Detection Procedure

Although the wrist detection procedure works very well in a wide range of cases, it is not without problems. The wrist detection procedure fails when the hand is in a shape such that the fingers point down in front of the forearm. This is demonstrated in the following figure.
Figure 31: Wrist Detection Failures When Fingers Overlap Wrist

Notice in these cases, the wrist is usually hidden behind the hand and/or fingers. The wrist detection fails because of the obstruction of the wrist function due to the location of the fingers.

The wrist detection also fails in cases where the wrist is bent downward significantly and the mass of the hand is greater than the mass of the forearm, as demonstrated in the following images.

Figure 32: Wrist Detection Failure When Wrist is Bent and Hand Mass is Greater than Forearm Mass

In these cases, the orientation of the hand is assumed to go in the incorrect direction, and thus the wrong side of the hand region is removed. The wrist location is also found using
an incorrect search direction. However, in cases where the mass of the forearm is greater than that of the hand, the wrist detection works successfully as shown in the following figure. In these cases, the orientation is correctly computed and thus the wrist is searched for in the proper direction and the correct portion of the arm is removed.

![Figure 33: Successful Wrist Detection When Wrist is Bent But Forearm Mass is Greater than Hand Mass](image)

The wrist detection occasionally has problems when the wrist location was not detected due to an insignificant amount of forearm showing while other wrist candidate locations were found in other acceptable regions of the hand. In these cases, an incorrect wrist candidate is predicted as the actual wrist location. An example of this is demonstrated in the following figure.

![Figure 34: Wrist Detection Failure When Insignificant Forearm is Showing and Other Possible Wrist Locations Exist](image)

Notice in the above images, the slight curve in the middle of the hand is interpreted as a wrist candidate. This candidate is not eliminated during the wrist detection procedure because the maximum width in the hand region is above that wrist candidate, and since
the actual wrist location was not detected due to an insignificant amount of forearm showing, the wrist detection chose an incorrect wrist candidate.

4.3.4 Wrist Detection Accuracy

The wrist detection procedure was tested with all the training and test images used in the hand recognition system. This resulted in an overall accuracy of 93.4% over 4,290 images. Since the hand shapes for letters P and Q often involve bending the fingers in front of the forearm, the wrist detection did not work well for these hand shapes. Without taking the hand shapes for letters P and Q into consideration, the overall accuracy increases to 96.57%. Even better wrist detection could be obtained by taking into consideration the motion of the hands in the image. The following figure shows the wrist detection accuracy of each hand shape.

Figure 35: Wrist Detection Accuracy
4.3.5 Wrist Detection Speed

The wrist detection stage operated at a satisfactory speed. On average, the procedure operated at about 17 fps. The parts of the wrist detection stage, which took significant portions of the execution time were the parts where the wrist width function was obtained and where the forearm was removed at the found wrist location; both parts involving many floating-point operations to compute pixel orientations.

4.4. Feature Extraction

The features that are extracted from the hand image have a tremendous impact on the overall system performance. The features extracted are determined based on the radial and angular frequencies specified and are also impacted by the sampling factor chosen when converting the image into a polar image. The following discusses the effects of the radial and angular frequencies as well as the sampling factor.

4.4.1 Effects of the Radial and Angular Frequencies

The features that are extracted from the hand images have a tremendous impact on the overall system performance and are dependent on the radial and angular frequencies chosen, as described in Section 2.4.2.3. Various combinations of radial and angular frequencies were tested, resulting in a different number of features each with a different hand recognition rate. The radial frequency was varied between 3 and 11 while the angular frequency was varied from 9 to 19, 22, and 30 depending on the results. The following table summarizes the recognition rate of the entire testing set for each combination of radial and angular frequencies, as tested with the RBF kernel with default parameters ($\gamma = 1/k$, $C = 1$, where $k =$ the number of features in the input data). The
results were also based on a sampling factor of 5 when converting the hand image into a polar hand image. Details on the sampling factor are described below. The blank entries are those combinations that were not tested. The bold numbers indicate the highest value in their respective category.

<table>
<thead>
<tr>
<th>Radial Frequency</th>
<th>Angular Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>73.94</td>
</tr>
<tr>
<td>10</td>
<td>73.88</td>
</tr>
<tr>
<td>11</td>
<td>74.34</td>
</tr>
<tr>
<td>12</td>
<td>74.93</td>
</tr>
<tr>
<td>13</td>
<td>75.66</td>
</tr>
<tr>
<td>14</td>
<td>75.46</td>
</tr>
<tr>
<td>15</td>
<td>75.73</td>
</tr>
<tr>
<td>16</td>
<td>75.79</td>
</tr>
<tr>
<td>17</td>
<td>75.53</td>
</tr>
<tr>
<td>18</td>
<td>75.86</td>
</tr>
<tr>
<td>19</td>
<td>75.40</td>
</tr>
<tr>
<td>20</td>
<td>75.53</td>
</tr>
<tr>
<td>21</td>
<td>75.92</td>
</tr>
<tr>
<td>22</td>
<td>74.80</td>
</tr>
<tr>
<td>23</td>
<td>74.60</td>
</tr>
<tr>
<td>24</td>
<td>74.21</td>
</tr>
<tr>
<td>25</td>
<td>74.21</td>
</tr>
<tr>
<td>26</td>
<td>73.81</td>
</tr>
<tr>
<td>27</td>
<td>73.35</td>
</tr>
<tr>
<td>28</td>
<td>72.96</td>
</tr>
<tr>
<td>29</td>
<td>72.43</td>
</tr>
<tr>
<td>30</td>
<td>71.57</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>74.54</td>
</tr>
</tbody>
</table>

Table 2: Recognition Rates as Radial and Angular Frequency are Varied. Tested with RBF Kernel, \( \gamma = 1/k, C = 1, \) Sampling Factor = 5.

Notice that generally, after an angular frequency of 4, the recognition rate decreases as the angular frequency increases. The radial frequency, on the other hand, has relatively consistent recognition rates across the board. However, the best combination of radial and angular frequencies occurs at a radial frequency of 4 and an angular frequency of 20. This combination results in only 80 features.
4.4.2 Effects of the Sampling Factor

Another important factor that influences the recognition rate is based on how the hand image is converted into a polar image. To convert the hand image to a polar image, the image is sampled in a circular direction while varying the radius up to the maximum radius of the hand region. Sections 2.4.2.3 and 3.1.3 describe in greater detail how the hand image is converted into a polar image. The resulting polar image has the dimensions $\text{radius} \times \text{radius} \cdot \text{sampling factor}$, a size dependent on the radius and the sampling factor. The following figure shows an input image (a), its corresponding forearm-removed hand image (b), and the resulting polar images from the forearm-removed image, where the sampling factor is 5 (c) and 2 (d).

![Images of hand images and polar images]

Table 3: Visual Effects of Varying the Sampling Factor for Polar Hand Images

The sampling factor, which corresponds to the circular sampling rate, impacts the features that are extracted, and thus the overall system performance. The effect of the
circular sampling rate was tested to determine the optimal sampling rate in terms of both speed (image size) and accuracy. The following table shows the effects of the sampling factor on the recognition rate.

<table>
<thead>
<tr>
<th>Sampling Factor (Sampling Rate/Radius)</th>
<th>Recognition Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>77.84%</td>
</tr>
<tr>
<td>2</td>
<td>78.17%</td>
</tr>
<tr>
<td>3</td>
<td>78.03%</td>
</tr>
<tr>
<td>4</td>
<td>78.10%</td>
</tr>
<tr>
<td>5</td>
<td>78.30%</td>
</tr>
<tr>
<td>6</td>
<td>78.36%</td>
</tr>
<tr>
<td>8</td>
<td>78.03%</td>
</tr>
</tbody>
</table>

Table 4: Effect of the Sampling Factor on the Recognition Rate

As can be seen, variations in this sampling factor have little effect on the overall recognition results. However, the sampling factor has a significant impact on the speed of the feature extraction stage. The following table shows the differences in the execution time of the feature extraction stage when using a sampling factor of 2 vs. a sampling factor of 5.

<table>
<thead>
<tr>
<th>Sampling Factor</th>
<th>Feature Extraction Time (s)</th>
<th>Frame Rate (fps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>.0610</td>
<td>16.39</td>
</tr>
<tr>
<td>2</td>
<td>.0165</td>
<td>60.61</td>
</tr>
</tbody>
</table>

Table 5: Effect of the Sampling Factor on Feature Extraction Time

Based on both speed and accuracy, it was decided that a sampling factor of 2 was optimal.

### 4.4.3 Feature Extraction Speed

The feature extraction stage executed very quickly with a sampling factor of 2, resulting in an execution time of .017 seconds (60.6 fps). This shows that the GFD can be quickly extracted for real-time applications. The computation of the 2-D Fourier transform of the polar-raster sampled image was done with the Discrete Fourier
Transform (DFT) algorithm provided by the OpenCV Image Processing Library. A significant decrease in execution time above that already attained could be obtained by using a Fast Fourier Transform (FFT) algorithm instead of the DFT algorithm.

4.5. Classification

The classification of the hand shapes was performed with Support Vector Machines. Since the wrist detection worked un-reliably for the hand shapes for letters P and Q, these hand shapes were not included in the set of recognizable hand shapes. Therefore, the hand recognition system was developed to recognize the hand shapes for letters A–Z, minus I, P, and Q and the numbers 1, 2, 3, 4, 5, 7, 8, 10 of ASL, resulting in a total of 31 hand shapes (classes). In addition, the hand shapes that failed the wrist detection significantly were not used for training or testing. Thus the total number of test and training images was approximately 3,900.

As described in Section 2.3, there are two main approaches to performing multi-class classification with Support Vector Machines. One technique is to simply combine binary-classifiers, while the other technique involves considering all the class information at once. Both of these classification techniques were tested. For the case where binary classifiers are combined, the one-against-one approach was used. For the all-at-once approach, an approach by Cramer & Singer [53] was tested. The LIBSVM and BSVM libraries [51], [52] by Chang and Linn, which provide implementations of the one-against-one and Cramer & Singer classification techniques respectively, were used for classification with SVM.

Before a Support Vector Machine is capable of classification using either a combined classification approach or an all-at-once approach, the system must be trained.
with examples. When training a Support Vector Machine, it is important to evaluate how well the training data can model the given data, and also how quickly it is capable of determining a given class. The ability of the SVM to model the data and the speed at which it does so is affected largely by the kernel type chosen, the kernel parameters, the classification approach chosen, and the actual training data. The speed of the classification is affected also by the number of support vectors, which is dependent on the number of training vectors – the size of the training data. When training the system, both the ability of the SVM to model the data and the speed at which this was done were taken into consideration when choosing both the classification approach and the kernel type and parameters.

4.5.1 Kernel Parameters

The RBF kernel and the polynomial kernel were both used to train the system. To determine the optimal kernel parameters for each kernel type, numerous tests were run to evaluate the performance of the system while varying these kernel parameters. For both of these kernels, the parameters $\gamma$ and $C$ (cost) were varied to evaluate their effect on the recognition rate of the system and also the number of support vectors. The following table shows the recognition rates and number of support vectors when varying the parameter $\gamma$ with a default cost parameter of 1. The results were tested with a radial frequency of 4, an angular frequency of 20, and a sampling factor of 5. The polynomial kernel was tested with a degree of 2 and coefficient of 1.
Table 6: Effects of $\gamma$ in the RBF and Polynomial Kernels

As can be seen in Table 6, the gamma value of $2^{-3}$ was optimal in terms of the recognition rate for the RBF kernel, while a gamma value of $2^1$ or greater was optimal for the polynomial kernel.

Next, the effect of the cost parameter was examined. Using the optimal gamma values obtained, the cost parameter (C) was varied for the RBF and polynomial kernels to determine its optimal value. The following table shows these results tested with a radial frequency of 4, an angular frequency of 20, and a sampling factor of 5. The polynomial kernel was again tested with a degree of 2 and coefficient of 1.

Table 7: Effects of Cost Parameter (C) in the RBF and Polynomial Kernels
As can be seen in Table 7, an optimal cost parameter for the RBF kernel is 1, however, the cost parameter had no effect on the recognition rate for the polynomial kernel.

### 4.5.2 Classification Approach

The effects of the classification technique on the speed and accuracy of the system was also tested. The one-against-one and Cramer & Singer all-at-once classification approaches were compared to examine these effects. Tests were run using the one-against-one and the all-at-once classification approaches with both the RBF kernel and the polynomial kernel. The tests were performed using the optimal kernel parameters as determined previously. The following table compares the classification rate, classification speed, and number of support vectors of the SVM trained with different kernel types using different classification methods. Once again, a radial frequency of 4, angular frequency of 20, and sampling factor of 5 were used for these tests.

<table>
<thead>
<tr>
<th>Classification Technique</th>
<th>Kernel</th>
<th>Classification Rate [s] (Frame Rate [fps])</th>
<th># Support Vectors</th>
<th>Classification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-against-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RBF</td>
<td></td>
<td>.9479 (1.05)</td>
<td>2,119</td>
<td>84.50%</td>
</tr>
<tr>
<td>Poly</td>
<td></td>
<td>.8167 (1.22)</td>
<td>1,845</td>
<td>81.07%</td>
</tr>
<tr>
<td>All-at-Once</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RBF</td>
<td></td>
<td>.7606 (1.31)</td>
<td>1,902</td>
<td>84.04%</td>
</tr>
<tr>
<td>Poly</td>
<td></td>
<td>.5762 (1.74)</td>
<td>1,434</td>
<td>77.11%</td>
</tr>
</tbody>
</table>

Table 8: Comparison of Classification Approaches

As can be seen in Table 8, the classification rates using the different classification approaches were very similar using the RBF kernel. However, using the polynomial kernel, the classification rates using the all-at-once approach were poorer. For both the
RBF kernel and polynomial kernel, the classification time and number of support vectors was less for the all-at-once approach as compared to the 1-v-1 approaches.

4.5.3 Hand Recognition Rates

Hand recognition rates of 84.43% and 87.57% were achieved with training sets of 2,400 and 2,700 images and testing sets of 1,500 and 1,200 images respectively. Most hand shapes were identified with high recognition rates. However, those hand shapes that are very similar in appearance, such as M, N, S, and T suffered from higher misclassification rates. These hand shapes are all closed-fist hand shapes. Those hand shapes that are not closed-fist had better recognition rates. The following figure shows the recognition rates for each of the 31 hand shapes.

Table 9: Recognition Rates for Each Hand Shape Using Cramer & Singer Approach w/RBF Kernel
The above results were tested using the Cramer & Singer classification approach with RBF kernel with $\gamma=2^{-3}$ and $c=2^1$, a radial frequency of 4, angular frequency of 20, and a sampling factor of 2.

### 4.5.4 Classification Speed

The classification speeds using both the 1-v-1 and all-at-once classification approaches and with the RBF and polynomial kernels were all too slow for real-time execution. Frame rates ranged from 1.05 fps to 1.74 fps for the classification stage alone, as shown in Table 11. While the classification performed more quickly with the Cramer & Singer approach, its speed was still not fast enough to achieve real-time rates. A significant improvement would be needed in the classification stage to achieve real-time recognition.

### 4.6. Summary of Results

The system was capable of identifying 31 hand shapes – letters A-Z minus, P, Q, and J and numbers 1, 2, 3, 4, 5, 7, 8, and 10. In the given test setup, the one-against-one classification approach yielded very similar recognition rates to that of the Cramer & Singer all-at-once approach. However, the classification speed of the all-at-once approach was slightly greater than that of the one-against-one technique. Using the RBF kernel, the hand identification system yielded a recognition rate of approximately 84% with both the one-against-one and all-at-once classification approaches using a training set of approximately 2,400 images and a testing set of approximately 1,500 images. Using a larger training set of approximately 2,700 images and a testing set of about 1,200 images, the recognition rate increased to almost 88%. Recognitions rates for most hand shapes worked very well. Hand shapes for B, D, E, K, L, O/0, R, U, W/6, Y, 1, 5 and 10
all had recognition rates greater than 90% for both training and test sets. Hand shapes B and L had 100% recognition rates for test set 1, and hand shapes B, L, and U had 100% recognition rates for test set 2. However, hand shapes that are very similar in appearance, such as M, N, S, & T had poorer results. The recognition rates for these hand shapes were 76%, 44%, 51%, and 59% respectively for test set 1 and 80%, 42.50%, 56.76%, and 66.67% for test set 2.

The speed of the system was good for all stages except for the classification stage where the speed was very slow. The preprocessing stage executed very quickly at a frame rate of approximately 381 fps. Wrist detection operated at a reasonable frame rate of 16.6 fps, on average. Feature extraction ran quickly at a rate of about 60.6 fps with a sampling factor of 2. However, the average classification rate was too slow for real-time execution with a frame rate of approximately 1.31 fps. Combining all these stages, the entire system operates at a rate of about 1.2 fps. The following table shows the average execution times for each of the stages.

<table>
<thead>
<tr>
<th></th>
<th>Preprocessing</th>
<th>Wrist Detection</th>
<th>Feature Extraction</th>
<th>Classification</th>
<th>Total (w/out Classification)</th>
<th>Total (w/ Classification)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Execution Time (s)</strong></td>
<td>.0026</td>
<td>.0604</td>
<td>.0165</td>
<td>.7632</td>
<td>.0795</td>
<td>.8427</td>
</tr>
<tr>
<td><strong>Frame Rate (fps)</strong></td>
<td>380.9</td>
<td>16.55</td>
<td>60.61</td>
<td>1.31</td>
<td>12.57</td>
<td>1.19</td>
</tr>
</tbody>
</table>

**Table 10: Timing Results for Each Stage**

Tests were performed to determine the optimal radial and angular sampling frequencies, sampling factor, kernel type, kernel parameters, and classification approach. The following table summarizes these results with a radial frequency of 4, angular frequency of 5 and using the kernel parameters of $\gamma=2^{-3}$ and $C=2^{1}$ for the RBF kernel and $\gamma=2^{1}$ and $C=2^{-2}$ for the polynomial kernel with a degree of 2 and a coefficient of 1.
<table>
<thead>
<tr>
<th>Classification Technique</th>
<th>Kernel</th>
<th>Sampling Factor</th>
<th>Classification</th>
<th>Total (w/out Classification)</th>
<th>Total (w/ Classification)</th>
<th># Support Vectors</th>
<th>Classification Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-v-1 RBF</td>
<td></td>
<td>5</td>
<td>.9479 (.105)</td>
<td>.1243 (8.05)</td>
<td>1.072 (.933)</td>
<td>2119</td>
<td>84.50%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>.9438 (.106)</td>
<td>.0795 (12.58)</td>
<td>1.023 (.977)</td>
<td>2114</td>
<td>84.25%</td>
</tr>
<tr>
<td>Poly</td>
<td>5</td>
<td>.8167 (1.22)</td>
<td>.1240 (8.06)</td>
<td>.9407 (1.06)</td>
<td>1845</td>
<td>81.07%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>.8197 (1.22)</td>
<td>.0797 (12.55)</td>
<td>.8994 (1.11)</td>
<td>1907</td>
<td>80.61%</td>
<td></td>
</tr>
<tr>
<td>All-at-Once RBF</td>
<td></td>
<td>5</td>
<td>.7606 (1.31)</td>
<td>.1242 (8.05)</td>
<td>.8847 (1.13)</td>
<td>1902</td>
<td>84.04%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>.7632 (1.31)</td>
<td>.0795 (12.57)</td>
<td>.8427 (1.19)</td>
<td>1910</td>
<td>84.43%</td>
</tr>
<tr>
<td>Poly</td>
<td>5</td>
<td>.5762 (1.74)</td>
<td>.1239 (8.07)</td>
<td>.7001 (1.43)</td>
<td>1434</td>
<td>77.11%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>.5766 (1.73)</td>
<td>.0796 (12.56)</td>
<td>.6562 (1.52)</td>
<td>1434</td>
<td>77.57%</td>
<td></td>
</tr>
</tbody>
</table>

Table 11: Summary of Results of Kernel Types, Classification Approaches, and Sampling Factor

As can be seen, in terms of both speed and the classification rate, the all-at-once classification technique using the RBF kernel with a sampling factor of 2 provides for the optimal setup.
Chapter 5  Conclusion

A vision-based hand identification system was developed to recognize 31 hand shapes from the American Sign Language – the letters A–Z, minus J, P, and Q, and the numbers 1, 2, 3, 4, 5, 7, 8, and 10. A wrist detection algorithm was developed to eliminate the forearm from the hand region to prevent dissimilarities of the same hand shape from occurring because of the varying amounts of forearm visible. The wrist detection algorithm provided a detection accuracy of approximately 97% on the 31 hand images. An appearance-based approach was used to extract features from the hand image. These hand features were obtained from the forearm-removed hand image by computing the Generic Fourier Descriptors. The descriptors that were extracted were determined by the radial and angular sampling frequency defined. The optimal radial and angular frequencies in terms of the recognition rate and the number of features were 4 and 20 respectively. The classification of the given hand shape, based on the extracted features was performed using Support Vector Machines. After testing various kernels with various kernel parameters, it was found that the RBF kernel with $\gamma=2^{-3}$ and $C=2^1$ gave the best results. The one-against-one combined binary classification approach was compared to the Cramer & Singer all-at-once classification approach. In the given test setup, the one-against-one approach yielded very similar recognition rates to that of the all-at-once approach. However, classification speed was slightly greater with the Cramer & Singer approach. Using the RBF kernel with optimized kernel parameters and using the Cramer & Singer classification approach, the hand identification system yielded a recognition rate of approximately 84.5% on a training set with 2,400 images and a testing set with 1,500 images. A recognition rate of about 87.57% was obtained when increasing
the training set to about 2,700 images and using a testing set of about 1,200 images. Due to the classification of the system with Support Vector Machines the overall speed of the system was too slow to achieve real-time rates. Without the classification stage, the average execution rate for the system including preprocessing, wrist detection, and feature extraction was approximately 12.6 fps. However, with the average classification rate of 1.31 fps, the overall execution rate of the system slowed down to 1.2 fps.

### 5.1. Future Work

To improve the speed of the system, a reduced set of support vectors or a training algorithm capable of providing a fewer number of support vectors while still maintaining a high recognition rate would be valuable. An alternative speed up approach would be to greatly reduce the number of features while still maintaining a comparable recognition rate or to implement the classification in parallel or with hardware. Making the hand shape recognition view-independent would also be a valuable addition. After improving the speed of the system, adding the ability to capture the motion and the locations of the hands would allow for the recognition of full-motion sign language.
Chapter 6  Appendix

6.1. Pseudocode for Main Wrist Detection Routine

// Main function for finding the wrist location and removing the
// forearm at that wrist location
Function wristDet( img_input, img_no_forearm ) {
  dilate(img_input);
  fill_holes(img_input);

  orientation = getOrientation( img_input );

  // Calculate angle needed to adjust image so that it will lie
  // horizontally
  if( orientation < 0 ) {
    rotation_angle = orientation*-1;
  } else if( orientation == 0 ) {
    rotation_angle = 90;
  } else {
    rotation_angle = 180 - orientation;
  }

  // Get function of width across hand region (see sub-function
  // below)
  wrist_width_function = getWristWidthFunction( img_input, rotation_angle )

  // calculate filter window size based on 3rd order polynomial
  // derived: f(x)=-.00000001x^3+.000006x^2-.0013x+.1888
  filter_window_size = calcFilterWindowSize();

  // Find second derivative, but be sure to filter between all
  // steps
  filter( wrist_width_function, filter_window_size );
  fir_der_wrist_function = derivative( wrist_width_function );
  filter( fir_der_wrist_function, filter_window_size );
  sec_der_wrist_function = derivative( fir_der_width_function );
  filter( sec_der_wrist_function, filter_window_size );

  max_width = getMaxWristWidth();

  // Get all the wrist candidates based on the second derivative
  // of the width function and the maximum width in the wrist
  // function. The max width is used to eliminate false
  // candidates. See sub-function below for details.
  getWristCandidates( sec_der_wrist_function, max_width );

  // if there are wrist candidates, find the best one and remove
  // the forearm at that wrist location
  if( wrist candidates exist ) {
    for( i = 0 to number_wrist_candidates – 1 ) {
      width_ratio = computeWidthRatio();
      location_ratio = computeLocationRatio();
      closeness = sqrt( width_ratio^2 + location_ratio^2 );
    }
if( closeness is minimum of all candidates ) {
    wrist_location = candidate_wrist_location;
} // end if
} //end for

img_no_forearm = remove_forearm( img_input, wrist_location );
} // otherwise, if no wrist location was found, set the return
// image to NULL indication no wrist location was found
else {
    img_no_forearm = NULL;
}
} // End Function wristDet

### 6.2. Pseudocode for getWristWidthFunction() Subroutine

// This function obtains the wrist width function
Function getWristWidthFunction( img_input, rotation_angle ) {

    // go through entire image and calculate the number of pixels
    // that are 'on' in each column after rotating each pixel by the
    // rotation_angle
    for( c = 0 to input_image_width - 1 ) {
        wrist_width_function[c] = 0;
        for( r = 0 to input_image_width - 1 )
            rotate_curr_pixel( rotation_angle, rot_r, rot_c );
        if( img_input[rot_r,rot_c] > 0 )
            wrist_width_function[c] += 1;
    } // end for r
} // end for c

// remove leading and trailing zero values
trimWristFunction( wrist_width_function );

return wrist_width_function;
} // End Function getWristWidthFunction
6.3. **Pseudocode for getWristCandidates() Subroutine**

// This function gets all the wrist candidates  
Function getWristCandidates( sec_der_wrist_function, max_width ) {  
    num_candidates = 0;  
    // only consider candidates from them max width location to the  
    // end of the forearm  
    for( i = max_width_location to end_hand_region ) {  
        // inflection points where concavity change from  
        // down-to-up are wrist candidates, but candidates  
        // with a width < 25% of max_width are ignored.  
        if( isDownToUpInflectionPoint( sec_der_wrist_function, i )  
            & width_wrist_function(i) > max_width*.25 ) {  
            // Only inflection points with a width > 25% the max  
            // width and that are located within X percent of the  
            // hand region are acceptable wrist candidates. Here  
            // X = 96%.  
            if( isWithinXPercentRegion(i,96)  
                wrists candidates[num_candidates] = i;  
                num_candidates++;  
            } // end if  
        } // end if  
    } // end for  
} // End Function getWristCandidates
Bibliography


79


