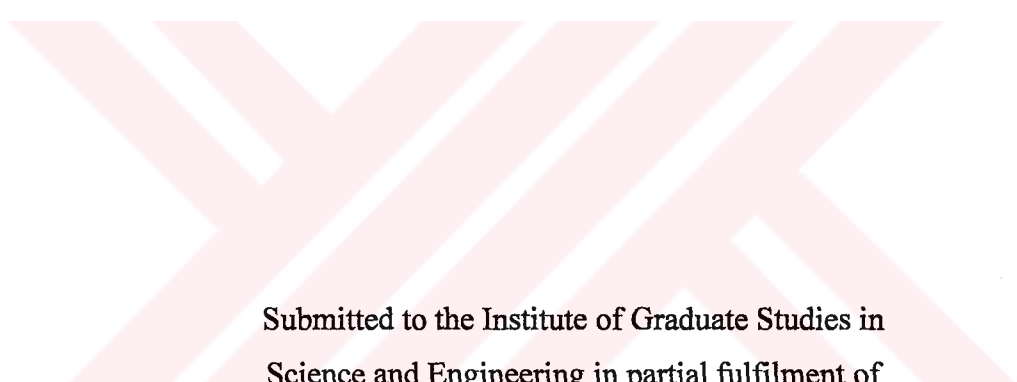


HAND GESTURE RECOGNITION USING ARTIFICIAL NEURAL NETWORKS

by
GÖRKEM GÖKNAR



Submitted to the Institute of Graduate Studies in
Science and Engineering in partial fulfilment of
the requirements for the degree of
Master of Science
in
Electrical and Electronics Engineering

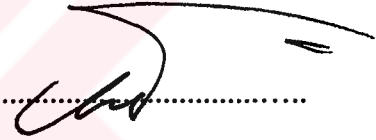
Yeditepe University
2005

HAND GESTURE RECOGNITION USING ARTIFICIAL NEURAL NETWORKS

APPROVED BY:

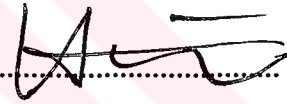
Assoc. Prof. Dr. Tlay Yıldırım
(Thesis Supervisor)

.....



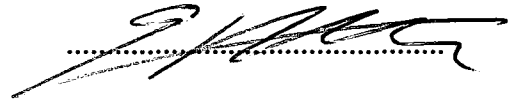
Assist. Prof. Dr. Soner zgnel

.....



Assist. Prof. Dr. Cem nsalan

.....



DATE OF APPROVAL: 03.06.2005

...TO MY FAMILY



ACKNOWLEDGMENT

I would like to express my sincere appreciation to Assoc. Prof. Dr. Tlay Yıldırım and also Assist. Prof. Dr. Cem nsalan for their guidance and support, which made this work possible.

I surely thank Prof.Dr. Cevdet Acar and all members of Electrical and Electronics engineering department for their support in my short academic career.

Finally, I thank to her for making me work.



ABSTRACT

HAND GESTURE RECOGNITION USING ARTIFICIAL NEURAL NETWORKS

In order for humans to interact with computers, a fast and easy way is to use hand gestures. Although using hand gestures in computer interaction was very cumbersome and needed special glove and computer hardware in the past, nowadays in any personal computer, simple cameras are available and there is enough processor power to do the expensive computations done in the past. With this feature, not only controlling computer with hand is possible but also some simple interpreters for sign language recognition can be made.

In this thesis a hand gesture recognition system, using an inexpensive camera and a personal computer is proposed. The system uses visual image as input and computes the geometric features, such as invariant moments and signature, of the extracted hand image for classification in a Multi-layer Perceptron Artificial Neural Network. The gestures used in the system are American Sign Language Manual Alphabet Gestures and Turkish Sign Language Manual Alphabet Gestures.

ÖZET

EL İŞARETLERİNİN YAPAY SİNİR AĞLARI İLE ALGILANMASI

İnsanlar birbirleriyle konuşmanın yanında farklı hareketli işaretler kullanarak anlaşmaktadır. Buna örnek olarak birisine dur derken, el ile dur işareti yapmak verilebilir. Benzer yöntem kullanılarak bilgisayar insan iletişiminde de el işaretleri kullanılabilir. Eskiden bunu yapmak kısıtlı işlemci gücü ve pahalı kamera sistemleri yüzünden zor olmasına rağmen günümüz araştırmaları, bu yöntem ile insan-bilgisayar iletişimi üzerinde durmaktadır. El işaretlerinin kullanılabileceği alanlar, bilgisayarı görsel olarak el ile yönetmek veya bir robota “sağa, sola git” gibi komutlar göndermek olabilir. Bir diğer önemli alan ise engelliler için işaret dili tanıma sistemidir, bu sayede işaret dili bilmeyen kişiler ile rahatça anlaşabilirler.

Bu tezde basit bir kamera ve ev bilgisayarı ile el işaretlerinin bilgisayar tarafından algılanması denenmiştir. Kullanılan sistem kamera ile elin de içinde bulunduğu görüntüyü almakta, daha sonra bu görüntüden elin görüntüsü çıkarılmaktadır. Bulunan el görüntüsünün, daha sonra çeşitli görüntü işleme teknikleri kullanılarak geometrik öznitelik vektörleri çıkarılmakta, sınıflandırıcı sisteminde ise çok-katman algılayıcı yapısındaki yapay sinir ağları kullanılmaktadır.

LIST OF FIGURES

Figure 2.1. RGB Color Cube.....	4
Figure 2.2. HSI Model.....	5
Figure 2.3. Dilation and Erosion	9
Figure 2.4. Open and Close Operations.....	10
Figure 2.5. Boundary of Sign V	10
Figure 2.6. Color probability thresholding of face image	17
Figure 2.7. Color Distance thresholding of face image	18
Figure 2.8. Color probability thresholding Hand image 1	18
Figure 2.9. Color distance thresholding Hand image 1	19
Figure 2.10. Color probability thresholding, Hand image 2.....	19
Figure 2.11. Color distance thresholding, hand image 2	20
Figure 3.1. Signature and LP Filtered Signature of Sign “V”	25
Figure 4.1. A basic neuron structure.....	32
Figure 4.2. Artificial Neuron Structure	33
Figure 4.3. General Mathematical Neuron Model.....	34
Figure 4.4. Activation Functions	36
Figure 4.5. Signal-flow Model of One Neuron Learning Process.....	38
Figure 5.1. Probability image	48
Figure 5.2. Open Operation	48
Figure 5.3. Largest Area After Labeling.	49
Figure 5.4. Signature plot of Figure 5.3.	50
Figure 5.5. Low-Pass filtered Signature of Figure 5.3.	50
Figure 5.6. Approximate Finger Information	51
Figure 5.7. Magnitude of Signature, Filtered Signature and Finger Information.....	51
Figure 5.8. Different “Y” sign from same person	52
Figure 5.9. Sign “C” from 4 people.....	53
Figure 5.10. Sign “W” from 4 people.....	53
Figure 5.11. American Sign Language Manual Alphabet	54
Figure 5.12. “E” Sign from same person.....	56
Figure 5.13. Processed “E” Signs of Figure 5.12	56

Figure 5.14. “F” sign from 4 people.....	56
Figure 5.15. Turkish Sign Language Manual Alphabet	57
Figure 5.16. ASL dataset trained with Gradient Descent Algorithm.	59
Figure 5.17. ASL 9x15x5 Network MSE graph.....	60
Figure 5.18. ASL 9x30x5 network MSE graph.....	61
Figure 5.19. ASL 9x15x30x5 MSE Graph.....	62
Figure 5.20. ASL 9x30x15x5 MSE Graph.....	63
Figure 5.21. ASL 9x30x15x5 MSE Graph (Delta inc=1.3, Delta dec=0.3)	64
Figure 5.22. ASL 9x30x15x5 MSE Graph (Delta inc=1.9, Delta dec=0.9)	64
Figure 5.23. TSL 9x90x30x5 MSE Graph, One Hand	67
Figure 5.24. TSL 9x90x30x5 MSE Graph, Two Hands	68



LIST OF TABLES

Table 5.1. ASL 9x15x5 network, 9 vector input	59
Table 5.2. ASL 9x30x5 network	60
Table 5.3. ASL 9x15x30x5 Network Results.....	61
Table 5.4. ASL 9x30x15x5 Network Results.....	62
Table 5.5. ASL 9x30x15x5 Network Results.....	63
Table 5.6. ASL Network 5 Training results	65
Table 5.7. ASL Network 5 Test results	65
Table 5.8. TSL 9x90x30x5 Network Results, One Hand	67
Table 5.9. TSL 9x90x30x5 Network Results, Two Hand	68
Table 5.10. TSL 50x30x5 Network Results, One Hand	69
Table 5.11. TSL 50x30x5 Network Results, Two Hands.....	69
Table 5.12. TSL 50x90x30x5 Network Results, One Hand	69
Table 5.13. TSL 50x90x30x5 Network Results, Two Hands.....	70
Table 5.14. TSL 360x400x5 Network Results, One Hand	70
Table 5.15. TSL 84x160x20x5 Network Results, One Hand	71
Table 5.16. TSL 84x160x20x5 Network Results, Two Hands.....	71
Table 5.17. TSL 79x100x20x5 Network Results, One Hand	71
Table 5.18. TSL 79x100x20x5 Network Results, Two Hands.....	72
Table 5.19. TSL 79x120x50x5 Network Results, One Hand	72
Table 5.20. TSL 79x120x50x5 Network Results, Two Hands.....	72
Table 5.21. TSL Two-hand Network 8, 80x100x40x15x5.....	73
Table 5.22. TSL Network 8, Training Data Recognition Results	74
Table 5.23. TSL Network 8, Test Data Recognition Results	74

LIST OF SYMBOLS / ABBREVIATIONS

ANN	Artificial Neural Networks
ASL	American Sign Language
TSL	Turkish Sign Language
RGB	Red,Green,Blue color triple
HSV	Hue,Saturation,Value color triple
θ	Angle or Threshold Value
A^c	Complement of Set A
\notin	Not element of a set
\emptyset	Empty set
\cup	Union of sets
\cap	Intersection of sets
\oplus	Dilation operator
\ominus	Erosion operator
\circ	Opening operator
\bullet	Closing operator
p	Pixel (Picture element) p
$N_4(p)$	4-neighbourhood of pixel p
$N_8(p)$	4-neighbourhood of pixel p
(x,y)	Cartesian coordinate pair x and y
$f(x,y)$	Image pixel value at coordinates x and y
$f_R(\cdot)$	Image red component function
$f_G(\cdot)$	Image green component function
$f_B(\cdot)$	Image blue component function
\mathbf{m}	Mean vector
\mathbf{z}	Vector z
$D(z,m)$	Euclidean distance of vector z to mean vector m
$(\cdot)^T$	Complement Operator
C	Covariance Matrix
C^{-1}	Matrix inverse of C

$P(rgb)$	Probability distribution of rgb triple
$P(rgb skin)$	Probability of skin colored pixels in rgb triple
$P(rgb \sim skin)$	Probability of non-skin colored pixels in rgb triple
M_{00}	2D Moment of order 0
M_{pq}	2D Moment of order $p+q$
μ_{pq}	2D Central moment of order $p+q$
$\bar{x} = x_c, \bar{y} = y_c$	Centroid of image x and y positions
A	Area of region
P	Perimeter of region
C	Compactness of region
E	Eccentricity of region
ϕ	Invariant moment vector
u_k	Linear combiner output of neuron k
y_k	Output signal of neuron k
$\phi(\cdot)$	Activation function
$e_k(n)$	Error signal of neuron k
$d_k(n)$	Target signal of neuron k
$\xi(n)$	Cost function (Mean Square Error) at time-step n
$w_{kj}(n)$	Value of synaptic weight of neuron k excited by input element $x_j(n)$
$\Delta w_{kj}(n)$	Weight adjustment value of neuron k for element $x_j(n)$
$\frac{\partial \xi(n)}{\partial w_{ji}(n)}$	Derivative of cost function to synaptic weight
η	Learning parameter
$\delta_j(n)$	Local gradient
$\Delta_{ij}(n)$	Weight Update value
η^-	Delta decrease parameter
η^+	Delta increase parameter

DEDICATION.....	iii
ACKNOWLEDGEMENTS.....	iv
ABSTRACT	v
ÖZET	vi
LIST OF FIGURES	vii
LIST OF TABLES.....	ix
LIST OF SYMBOLS / ABBREVIATIONS	x
1. INTRODUCTION	1
1.1. Background on Gesture Recognition.....	1
1.2. Limitations of Hand Gesture Recognition Systems	2
1.3. Layout of Thesis	3
2. IMAGE PROCESSING FUNDAMENTALS	4
2.1. Color Image Processing.....	4
2.1.1. RGB Model.....	4
2.1.2. CMY Color Model.....	4
2.1.3. HSI Color Model	5
2.1.4. RGB to HSI Conversions	5
2.2. Morphological Image Processing	7
2.2.1. Basic Set Theory.....	7
2.2.1. Logic Operations in Binary Images	8
2.2.2. Dilation	8
2.2.3. Erosion.....	8
2.2.4. Opening and Closing	9
2.2.5. Boundary Extraction.....	10
2.2.6. Labeling Connected Components.....	11
2.3. Skin Region Segmentation and Tracking	13
2.3.1. Introduction to Segmentation	13
2.3.2. Color Image Thresholding.....	13
2.3.3. Euclidean and Mahalanobis Distance in Color Space	14
2.3.4. Color Probability Distribution Model.....	15
2.3.5. Skin Color Thresholding Tests	17
2.4. Cam-Shift Color Tracker	21

3. FEATURE EXTRACTION.....	23
3.1. Compactness.....	23
3.2. Orientation	23
3.3. Eccentricity.....	24
3.4. Signatures	25
3.5. Moment Invariants.....	27
4. ARTIFICIAL NEURAL NETWORKS.....	30
4.1. Introduction	30
4.2. Historical background.....	30
4.3. Neural Networks Versus Conventional Computation	31
4.4. Biological Neuron.....	32
4.5. Mathematical Model.....	33
4.5.1. Types of Activation Functions	35
4.5.1.1. Threshold Function.....	35
4.5.1.2. Piecewise-Linear Function	36
4.5.1.3. Sigmoid Function	36
4.5.2. Learning.....	37
4.5.3. Multi Layer Perceptron.....	39
4.5.3.1. Introduction to MLP	39
4.5.3.2. Backpropogation Algorithm	40
4.5.3.3. Mathematics of Backpropogation.....	40
4.5.3.3. Backpropogation Models.....	43
5. EXPERIMENT RESULTS AND DISCUSSION	45
5.1. Experiment Process	45
5.1.1. Image input.....	45
5.1.2. Image processing	45
5.1.3. Feature extraction	46
5.1.4. Neural network classifier.....	47
5.2. Sample Gesture Processing	48
5.3. American Sign Language Manual Alphabet	52
5.4. Turkish Sign Language Manual Alphabet.....	55
5.5. ASL Results.....	58

5.5.1. ASL Network 1	59
5.5.2. ASL Network 2	60
5.5.3. ASL Network 3	61
5.5.4. ASL Network 4	62
5.5.5. ASL Network 5	63
5.5.6. ASL Network 6	65
5.5.6. Discussion on ASL results	66
5.6. TSL Results	66
5.6.1. TSL Network 1	66
5.6.2. TSL Network 2	68
5.6.3. TSL Network 3	69
5.6.4. TSL Network 4	70
5.6.5. TSL Network 5	70
5.6.6. TSL Network 6	71
5.6.7. TSL Network 7	72
5.6.8. TSL Network 8	73
5.6.9. TSL Results Discussion	75
5.7. Conclusion and Future Work	76
6. APPENDIX A: DATABASE	78
7. REFERENCES	79

1. INTRODUCTION

1.1. Background on Gesture Recognition

Humans communicate with each other not only by talking but also using hand and body gestures. Example gestures can be given as “the table over there” (pointing the table) or “stop please” (making a stop sign). A more complex version is sign language where talking is not possible, instead this system uses complex hand and body gestures. But not everyone understands sign language; hence an “interpreter” must be used. Although there is not a complete system fully capable of understanding yet, it is possible to capture some of the static signs (letters) of the sign language alphabet. Using analysis of locations of the previous sign and next sign it is possible to understand “words”. In every country different sign language systems are used hence it is not possible to derive a universal system, some countries use only one hand (eg. USA) and some (eg. Turkey, Japan) use both hands. For this purpose “Hand Gesture Recognition” is a relatively new and growing subject in signal processing area.

Hand gestures have a wide area of usability. Computer system can be taught to interpret sign language alphabet or can be used as a controller. A vision controlled painting system can be made or you can command a robot to follow or stop by you. This general system is widely called as Human-Computer Interaction (HCI).

Prior systems used sensor gloves to capture the hand motion data [1], also Kessler et al. tried the glove system for direct computer interface [2]. Although these systems are already being used in simulation and virtual reality systems, it is not eligible for sign language or simple control systems for the requirement of hardware and gloves.

Later systems used color gloves to explicitly identify fingers (hence easy feature extraction) for visual input. Each finger was assigned a special color, hence making the image processing part easier to identify finger location and shape.

Another system is used to control the mouse pointer via visual hand gestures. This system uses dynamic position of the hand by motion recognition, and in fact is not a static hand posture recognition system. Akyol et al. [3] proposed a hand gesture control interface

for automotive. This system used infra-red cameras as input and images were processed in a similar way in this thesis, using 2D hand projection (or silhouette). They used manually constructed 20 and 6 class datasets, which classes were barely separable. Juan [4] used hand gestures to control a simple robot, the gesture were first processed and then fed into a fuzzy classifier system. Then he assigned each output to commands like stop, turn left, turn right or forward.

Marcel [5] used direct gray-scale hand images for classification in a Hidden Markov Model classifier and obtained good results for a 6-class database, and then he improved the system for dynamic classification [6]. Freeman [7] used orientation histogram based approach to differentiate hand postures. In this model he used partial derivatives of edge images for obtaining feature information. A wide area of people working on hand posture classification can be found on Kohler's page [8], each with different features and classifiers.

1.2. Limitations of Hand Gesture Recognition Systems

Complete recognition of hand gestures is in fact a challenging subject. Since hand gestures are rich in types of use; they have multi meanings and these can change in space and time. Also human hand is a complex non-rigid object, looking from one view is different than other.

Visual tracking of hand is in fact the most important part of the system. It is not possible, yet, for any computer system to correctly classify every hand gesture in every condition. First one of these conditions is different illumination. The illumination changes during the day. If there is only artificial light, the body of the person or number of people in the room can change the illumination taken by the camera. This problem can be achieved by first taking a sample color region of the object to be tracked, as proposed in this thesis.

System must be able to know where the head and hand is, initially. This can be done by setting the person in a specified position in front of the camera, or just taking only hand to the camera region.

A good feature vector must be used and this must be computationally efficient. Putting the whole captured image to the classifier without any preprocess will sometimes give good results, but if the image is very big, system will not be stable and fast. Most of the features of hand can be recognized by geometrical properties, as proposed in this thesis. Also edge information of the hand can give good results.

Classifier system must be good and robust in order to work on all conditions. A classical clustering approach will not work for higher number of classes (or number of gestures to be recognized). In literature Hidden-Markov Models for dynamical classification and Artificial Neural Networks for statistical classification are found to be robust. A hybrid model can be used for generalization.

1.3. Layout of Thesis

Main objective of this thesis is recognition of visual hand postures (static gestures) using geometrical properties of binary hand images and classifying them with the help of Multi-layer Perceptron Artificial Neural Network. This work presented here proposes a computer efficient and inexpensive system for static hand gesture recognition using contour model of hand for feature extraction and Multi-layer Perceptron neural network as a classifier.

Input images for system are taken from a cheap USB or TV-Card camera. First image processing fundamentals needed for meaningful parts of image extraction from input image are described. Then, the mathematics of features extracted from the binary image are described, followed by basics of Artificial Neural Networks and Multi-Layer Perceptron Network model. Finally, system process information and results are described with the conclusion of the proposed model.

2. IMAGE PROCESSING FUNDAMENTALS

2.1. Color Image Processing

2.1.1. RGB Model

Hardware (cameras, monitors) usually use RGB (red, green, blue) model [9]. This model is based on Cartesian coordinate system and uses primary colors red, green and blue for processing. Images represented in RGB color model consist of three component images for each primary color.

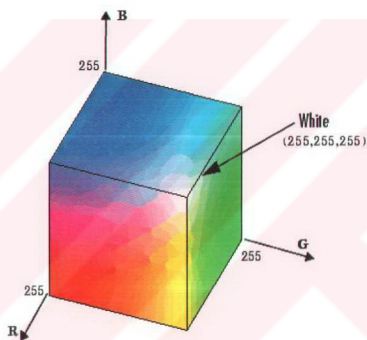


Figure 2.1. RGB Color Cube

2.1.2. CMY Color Model

Uses cyan, magenta and yellow colors which are secondary colors derived from primary colors. Mainly printing devices uses this model. Also an additional black value (denoted as "K") is used for printing pure black values, making the model CMYK.

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2.1)$$

2.1.3. HSI Color Model

This model is the best model for human eye as it is a kind of *describing* color. When humans view a color object, we describe it by its hue, saturation and brightness. Hue is a color attribute that describes a pure color and saturation gives a measure of the degree to which a pure color is diluted by white light. Intensity value gives how much an object is illuminated by means of gray-scale value. HSI is similar to the human way of saying that, this dress is dark yellowish green, meaning it has a hue going from green to yellow with high saturation and low intensity.

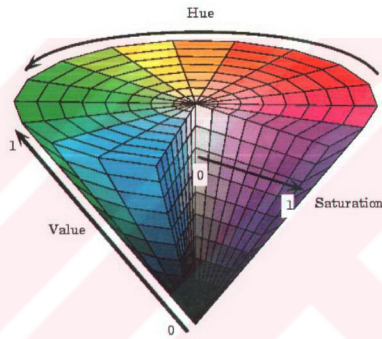


Figure 2.2. HSI Model

2.1.4. RGB HSI Conversions

Given an image in RGB format , H component of each RGB pixel is obtained by :

$$H = \begin{cases} \theta & , \text{if } B \leq G \\ 360 - \theta & , \text{if } B > G \end{cases} \quad (2.2)$$

with,

$$\theta = \cos^{-1} \left\{ \frac{\frac{1}{2} [(R-G) + (R-B)]}{\sqrt{(R-G)^2 + (R-B)(G-B)}} \right\} \quad (2.3)$$

The saturation component is calculated as:

$$S = 1 - \frac{3}{(R + G + B)} [\min(R, G, B)] \quad (2.4)$$

And the intensity (value) component is given by:

$$I = \frac{1}{3}(R + G + B) \quad (2.5)$$

where R, G and B are red, green and blue values of color image respectively.

These calculations have an assumption that R, G and B values are normalized on the range [0, 1] instead of regular [0, 255] interval, and angle θ is measured with respect to the red axis of HSI space. Hue can be normalized to the range [0, 1] by dividing all values in hue equation to 360.

2.2. Morphological Image Processing

Mathematical morphology is a used for extracting image components that are useful in representation and description of region shape, such as boundaries, skeletons and convex hull. These techniques are used for filtering, thinning or getting a desired region from the image. Our work here will be mainly used in binary images, which in fact are 2 dimensional integer space Z^2 .

2.2.1. Basic Set Theory

Most of the morphological operators use the set theory as base. The set of all pixel coordinates that do not belong to set A , denoted by A^c , is given by

$$A^c = \{w \mid w \notin A\} \quad (2.6)$$

Where $w = (x, y)$ is the coordinate of point in set A (an area location A in the image). Then points in image which are not A is the complement of set A .

The union of two sets,

$$C = A \cup B \quad (2.7)$$

Will give all elements in sets A and B , similarly the intersection of two sets A and B is

$$C = A \cap B \quad (2.8)$$

Which gives elements that both exist in sets A and B .

2.2.1. Logic Operations in Binary Images

Majority of applications based on morphological concepts involve binary images, which involve pixel values of either 1 or 0. The principal logic operations used in image processing are AND (“ \cap ” or “&”), OR (“ \cup ” or “|”) and NOT (complement or “ \sim ”). These operators are used pixel-by-pixel operations between two images and it has the same idea of use as in the set theory. The first image can be mask and second image can be the structuring element.

2.2.2. Dilation

One of the fundamental operations in binary images is dilation. Mathematically dilation is defined in terms of set operations. With A and B as sets in Z^2 , the dilation of A by B , denoted by $A \oplus B$ is defined as

$$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\} \quad (2.9)$$

where \emptyset is the empty set and B is commonly referred as the structuring element in dilation, as well as in other morphological operations. Dilation of A by B is the set consisting of all the structuring element origin locations where the reflected and translated B overlaps at least some of A . Dilation is similar to convolution theory in communication systems. One of the simplest applications of dilation is for bridging gaps in the image.

2.2.3. Erosion

For sets A and B in Z^2 the erosion of A by B , denoted as $A \ominus B$ is defined as:

$$A \ominus B = \{z \mid (B)_z \cap A^c \neq \emptyset\} \quad (2.10)$$

In words, erosion of A by B is the set of all structuring element origin locations where the translated B has no overlap with the background A . Erosion simply removes unconnected or any user defined shape type (the structuring element) of pixels, thus shrinking the

image.

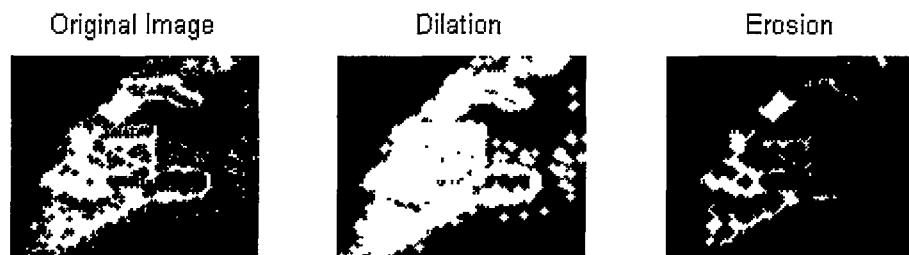


Figure 2.3. Dilation and Erosion

2.2.4. Opening and Closing

Opening generally smooths the contour of objects, while breaking narrow gaps. Closing in contrast eliminates small holes and fills gaps in the contour [10].

The morphological opening of set A by structuring element B, denoted $A \circ B$ is defined as

$$A \circ B = (A \ominus B) \oplus B \quad (2.11)$$

Thus, the opening A by B is the erosion of A by B, followed by a dilation of the result by B. The morphological closing of set A by structuring element B, is defined as

$$A \bullet B = (A \oplus B) \ominus B \quad (2.12)$$

That is closing of A by B is dilation of A by B, followed by erosion of the result by B.

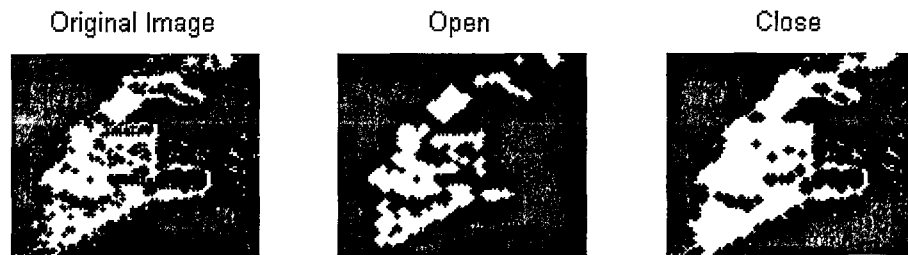


Figure 2.4. Open and Close Operations

2.2.5. Boundary Extraction

The boundary of set A , denoted by $B(A)$, can be obtained by first eroding A by B , then performing the set difference between A and its erosion, that is :

$$B(A) = A - (A \ominus B) \quad (2.13)$$

where B is a suitable structuring element or marker, and the original image A is assumed to be a filled image. The boundary of the object will be used to calculate the signature of the object and will be described in feature extraction section.

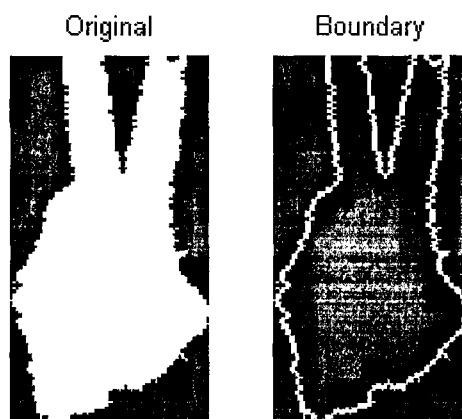


Figure 2.5. Boundary of Sign V

2.2.6. Labeling Connected Components

Connected component analysis allows separating unconnected objects from the image. This method is used in binary images, to obtain the mask image for desired object. Once the skin color objects are found in the image, there should be a proper way to identify the hand region from the image. The main approach widely used is body-object centered model, which assumes that head is in the middle of the screen and two hands are in left and right sides of the screen respectively, this leads to three largest skin areas in the image. Even only one hand is assumed to be in the image, there will be some noise, like some other small regions and pixels that have same skin probability but not skin data. For this reason segmentation requires objects to be separated. Then we can use an area threshold level or if one hand is used; the maximum region area that is greater than certain threshold. First the connectivity must be defined:

A foreground pixel p (which corresponds to 1 if background pixels are assumed to be 0) at coordinates (x,y) has two horizontal and two vertical neighbors whose coordinates are: $(x+1,y)$, $(x-1,y)$, $(x,y+1)$ and $(x,y-1)$, and this set of 4-neighbors of p is denoted $N_4(p)$. The four diagonal neighbors of p have coordinates $(x+1,y+1)$, $(x+1,y-1)$, $(x-1,y+1)$ and $(x-1,y-1)$ denoted as $N_d(p)$. The union of $N_4(p)$ and $N_d(p)$ are the 8-neighbours of p denoted by $N_8(p)$.

Two pixels p and q are said to be *4-adjacent* if $q \in N_4(p)$, and similarly *8-adjacent* if $q \in N_8(p)$. A path between pixels p_1 and p_n is a sequence of points p_1, p_2, \dots, p_n such that p_k is adjacent to p_{k+1} for $1 < k < n$. If 4-adjacency is used path can be 4-connected, and 8-connected if 8-adjacency is used. These two types of connectivity are the mainly used types for 2D binary images.

Two foreground pixels p and q are said to be *4-connected* if there exists a 4-connected path between them, consisting entirely of foreground pixels. For any foreground pixel, p , the set of all foreground pixels connected to it, is called the connected component containing p . Then this set is assigned a label, which is mainly the component number.

A simple algorithm to find 8-connected components [11] of image $f(u, v)$, H and W being height and width of the image is :

procedure *IncrementalLabeling*

nextLabel = 2

Create collision set $C = \{\}$

for $v=1$ to H , and $u=1$ to W **do**

if $f(u, v)=1$ **then**

if all neighbors $n_i=0$ **then**

Assign $f(u, v)$ to *nextLabel*

nextLabel = *nextLabel* + 1

else

Choose any neighbor $n_k > 1$

Assign $f(u, v)$ to n_k

for all neighbors $n_i > 1$ **do**

if $n_i \neq n_k$ **then**

Set (n_i, n_k) to collision set C

end if

end for

end if

end if

lmax = *nextLabel* - 1

end for

for all $(u, v) \in C$ **do**

A = labels containing u

B = Find labels containing v

if $A \neq B$ **then**

$A = A \cup B$

end if

end for

for all (u, v) **do**

if $f(u, v) > 1$ **then**

S = Label sets containing $f(u, v)$

Set $f(u, v)$ to first set

end if

end for

end procedure

2.3. Skin Region Segmentation and Tracking

2.3.1. Introduction to Segmentation

A classifier is good as long as it has meaningful data in its inputs. Thus to correctly classify hand images online if possible, one must be able to get hand image region properly. Most off-line systems for gesture recognition use manual segmentation of hand images from their database. This work uses a less cumbersome semi-automatic process which uses color skin probability model for segmentation. Also for online recognition system the segmentation of hand will be in real-time with no user input thus the system must be able to understand the hand image properly.

2.3.2. Color Image Thresholding

One obvious way to extract an object from the image is by looking up the colors corresponding to that object. Since a color image consists of three gray image levels this can be done with a simple thresholding. If a color object (in this case, skin regions) is to be segmented from an image one basic approach is simple thresholding. If RGB model is used color object region can be described by threshold parameters. Such an example will be:

$$f(x, y) = (f_R(x, y) > R_t) \& (f_G(x, y) > G_t) \& (f_B(x, y) > B_t) \quad (2.14)$$

where $f(x, y)$ is the thresholded mask image, R_t , G_t and B_t are threshold values for red, green and blue components, and $f_R(\cdot)$, $f_G(\cdot)$, $f_B(\cdot)$ are red, green and blue components for the color image respectively. The thresholded mask image is a binary image (with 0 and 1 values only). The logical operators can be any combination. The mask image multiplied by original image will give “color of interest” image.

Although this system works fine under same illumination and background conditions it is not a proper way to extract color regions in most conditions. Threshold value can be used adaptively to improve performance, by setting a convergence parameter then calculating threshold parameters automatically. This method is called adaptive thresholding but in the skin model case this will not work also.

2.3.3. Euclidean and Mahalanobis Distance in Color Space

Given a set of sample color points representing of a color (in this case skin color range) of interest, we obtain an estimate of the “average” or “mean” color that we wish to segment [12]. Let this average color be RGB column vector \mathbf{m} . To segment the image pixels according to each pixel being in the selected range, one of the simplest measures is Euclidean distance.

Let z denote the arbitrary point ($z=f_{RGB}(x, y)$, which is calculated in the iterative loop) in RGB space. We say that z is similar to \mathbf{m} if distance between them is less than a threshold, T .

The Euclidean distance between z and \mathbf{m} is given by

$$\begin{aligned} D(z, \mathbf{m}) &= \|z - \mathbf{m}\| \\ &= \sqrt{[(z - \mathbf{m})^T (z - \mathbf{m})]} \\ &= \sqrt{[(z_R - m_R)^2 + (z_G - m_G)^2 + (z_B - m_B)^2]} \end{aligned} \quad (2.15)$$

Where $\| \cdot \|$ is the norm of the argument and subscripts R, G, B denote red, green and blue components of vectors \mathbf{m} and z . The locus point $D(z, \mathbf{m}) \leq T$ is a solid sphere of radius T .

A useful generalization of equation (2.15) is a distance measure of the form:

$$D(z, \mathbf{m}) = \sqrt{[(z - \mathbf{m})^T C^{-1} (z - \mathbf{m})]} \quad (2.16)$$

where $(\cdot)^T$ is the complement operator and C is the covariance matrix of the samples representative of the color we wish to segment, and is computed as:

$$C = \frac{1}{K} \sum_{k=1}^K (z_k - m_k)(z_k - m_k)^T \quad (2.17)$$

where K is the number of points in the set and z_k being the RGB point operated.

This distance is commonly referred to as the Mahalanobis distance. The locus of points $D(z,m) \leq T$ describes a solid 3D elliptical body with its principal axes oriented in the direction of maximum data spread. When $C=I$, the identity matrix, Mahalanobis distance reduces to Euclidean distance, giving a 3D sphere.

2.3.4. Color Probability Distribution Model

In the skin color case, basic threshold approaches fail in complex backgrounds. Thus a more robust approach must be used. The fact about human skin color is in fact most of the human skins around the world fall in a region of RGB space under same illumination conditions. But it is nearly impossible to provide same illumination every time; each minute of the day has different illumination parameters. A robust approach is to take a sample region of skin color from the image (or from the first frame of movie in online scenario) then using a probability lookup table (LUT), compare each image if it contains a skin color region. This approach is mainly developed by Jones et al. [13] and called as skin color probability model. Many new color tracker systems use this method as a base.

To build the lookup table, first sample skin color regions must be selected. In this study, the thresholding is done by constructing a lookup table for each person from three or four images and then applied to all images of that person, thus instead of manually thresholding each image, a semi-automatic and robust thresholding method is proposed.

An RGB image in hardware has a maximum of $256*256*256$ different colors (excluding alpha channel) in today's vision systems. For each primary color there exist 256 possible levels, which is also called 256 bin. But as Jones et al., stated 256 bin system uses a large lookup table thus decreases the computing time, a better way is to use a 32 bin system, which in fact quantizes every 8 neighborhood images to 1 pixel.

Once the histogram is constructed the probability distribution is obtained using:

$$P(rgb) = \frac{c(rgb)}{T_c} \quad (2.18)$$

Where $c(rgb)$ gives the count in the histogram bin associated with the RGB color triple and T_c total count obtained by summing the point counts in all of the bins. By this definition skin color and non-skin color distribution will be:

$$P(rgb | skin) = \frac{s(rgb)}{T_s} \quad (2.19)$$

$$P(rgb | \sim skin) = \frac{n(rgb)}{T_n} \quad (2.20)$$

where $s(rgb)$ is pixel count in RGB of skin histogram, $n(rgb)$ is equivalent count from non-skin histogram and T_s and T_n are total counts contained from skin and non-skin color respectively.

After constructing these lookup tables then color threshold for skin color objects can be obtained as:

$$\frac{P(rgb | skin)}{P(rgb | \sim skin)} > \theta \quad (2.21)$$

where $0 < \theta < 1$ is a desired threshold value.

When experimenting with this system, it is observed that most light and most dark value pixels are misclassified by the system hence these regions (RGB(0,0,0) and RGB(32,32,32)) are set to 0 after constructing the LUT and also non-skin pixels are ignored for fast computation, this leads to:

$$P(rgb | skin) > \theta \quad (2.22)$$

This system here is found to be a robust way of segmenting skin-color pixels, also this method is a base system for CAMSHIFT online skin color tracker algorithm that will be described later.

2.3.5. Skin Color Thresholding Tests

These examples show a rough description of how Mahalanobis distance and color probability pictures are constructed. Below pictures are thresholded after training the probability database with 20 human skin color images that are cropped manually. Up left image in each Figure is either probability or distance color map of the original image down right.

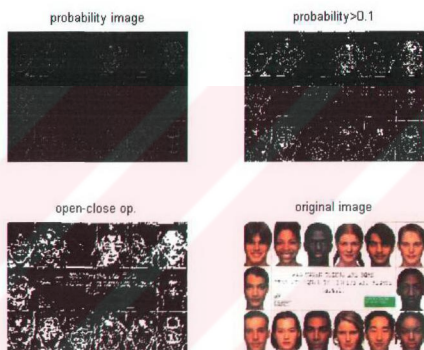


Figure 2.6. Color probability thresholding of face image

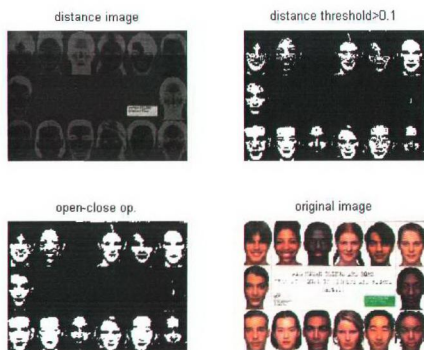


Figure 2.7 Color Distance thresholding of face image

Some faces from the original image of Figure 2.6 is included in the training. As can be seen, nearly all faces are extracted except the dark skin tones, which are not used in training.

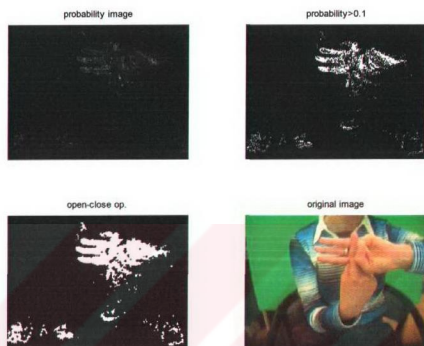


Figure 2.8. Color probability thresholding Hand image 1

Original image in Figure 2.8 was not in the training set, but it was properly extracted. Both color probability (Figure 2.8) and distance thresholders (Figure 2.9) give proper results.

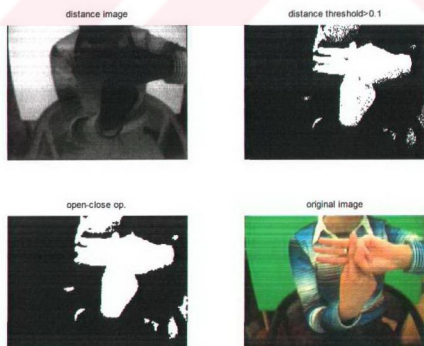


Figure 2.9. Color distance thresholding Hand image 1

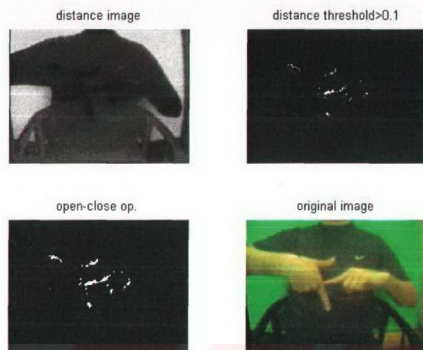


Figure 2.10. Color probability thresholding, Hand image 2

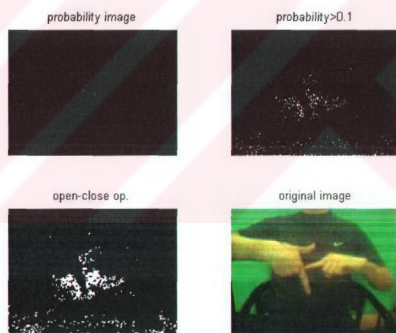


Figure 2.11. Color distance thresholding, hand image 2

As can be seen in the original images of Figure 2.10 and Figure 2.11, that are not included in training, skin regions cannot be segmented. This is in fact, due to the camera calibration failure in that image, not every time a camera gets good pictures.

The last figure is not also trained. As can be seen, although the distance probability image seems to capture the hand region (black values), a different color threshold value

different then the used one must be used. The color-probability system doesn't capture a clue at all. Just training 20 images is not enough but more data must be trained.

Generally a Mahalanobis color thresholder can track the hand images but there will be other regions as well due to the elliptic nature, thus is not in fact a good color object tracker. Whenever enough training is used color-probability tracker is a better choice. This tracker is best worked on local probability thresholding, take one probability distribution from the first image then the tracker can find desired regions in the similar images (same time, same person skin).

In the proposed model, the color-probability tracking system is used to semi-automatically threshold skin regions from a person at a time. Each person has at least 75 images in a dataset. In two or more cases, more images from different classes are used to train the probability tracker. After thresholding, hand skin segmentation is performed automatically using connected component analysis for that person.

2.4. Cam-Shift Color Tracker

This algorithm is based on a robust non-parametric technique for climbing density gradients to find the peak of probability distributions called the mean-shift algorithm [14]. The mean-shift algorithm is meant to be used in static mode, taking only the first color distribution of video sequence, thus fails in video sequences which have illumination changes. Therefore, mean-shift is modified to deal with dynamically changing color probability distributions derived from video frame sequences. This modified algorithm is called the Continuously Adaptive Mean Shift (CAMSHIFT) algorithm and developed by Bradski [15]. Algorithm is meant to be a color object tracker, thus an object with a color distribution or skin (flesh) color can also be used.

This algorithm first takes a manual area of the object (in this case a skin area), after this a color-probability histogram is created. Each image in video is then compared with histogram and area desired (if there is only one hand that will be the object) is chosen, after that another color-probability for the tracked area is recreated and system continues to work like this until stopped.

Main algorithm of mean-shift is:

1. Choose a search window size
2. Choose the initial location of search window
3. Compute the mean location (centroid) in the search window
4. Center the search window at the mean location computed in Step 3
5. Repeat Steps 3 and 4 until a certain conversion threshold is reached.

For discrete 2D image probability distributions the mean location (the centroid) within the search window is obtained below. Zeroth moment of a 2D function $f(x,y)$, x and y being the cartesian coordinates, is :

$$M_{00} = \sum_x \sum_y f(x,y) \quad (2.23)$$

The first moment for x and y is:

$$M_{10} = \sum_x \sum_y xf(x,y) , M_{01} = \sum_x \sum_y yf(x,y) \quad (2.24)$$

Then the mean search window location (the centroid of the area) is:

$$x_c = \frac{M_{10}}{M_{00}} \quad (2.25)$$

$$y_c = \frac{M_{01}}{M_{00}} \quad (2.26)$$

where $f(x,y)$ is the pixel (probability) value at position (x,y) in the image and x and y range over the search window.

CAMSHIFT algorithm uses mean-shift algorithm as base:

1. Choose the initial location of the search window
2. Mean shift as above; store the zeroth moment (M_{00})
3. Set the search window size equal to a function of M_{00} found in Step 2
4. Repeat Step 2 and 3 until convergence (certain threshold level)

Experiments are performed with the CAMSHIFT tracker and it is observed that it is good as a tracker for skin color sequences. Some videos showing CAMSHIFT at work for skin-color can be found on the appendix. In the future, an online system with this tracker for gesture recognition could be made.



3. FEATURE EXTRACTION

Feature extraction is a general term of getting mathematical descriptions from an image. Instead of using full image data as input some properties of meaningful regions (in this case the hand regions) are extracted. This section shows the geometrical features used in this work.

3.1 Compactness

Compactness is a measure of how deformed an object shape is. It is measured as the ratio of object area to square of objects perimeter. When the ratio is exactly one the object is a circle.

Area of the object can be found as summing all the pixels within the object and in fact it is the zeroth moment M_{00} .

$$A = M_{00} = \sum_x \sum_y I(x, y) \quad (3.1)$$

The perimeter in turn can be found as summing all the boundary pixels of the object, denoted by P . Then the compactness, C , is:

$$C = \frac{4\pi A}{P^2} \quad (3.2)$$

3.2. Orientation

To obtain the degree of movement of object in z -axis (along x, y plane), orientation can be calculated by image moments. Second moments are:

$$M_{20} = \sum_x \sum_y x^2 I(x, y) \quad (3.3)$$

and

$$M_{02} = \sum_x \sum_y y^2 I(x, y) \quad (3.4)$$

Then the orientation angle is

$$\theta = \frac{1}{2} \arctan \left(\frac{2 \left(\frac{M_{11}}{M_{00}} - x_c y_c \right)}{\left(\frac{M_{20}}{M_{00}} - x_c^2 \right) - \left(\frac{M_{02}}{M_{00}} - y_c^2 \right)} \right) \quad (3.5)$$

3.3. Eccentricity

The major length and width of the object can be found as described [16]. Let

$$a = \frac{M_{20}}{M_{00}} - x_c^2 \quad (3.6)$$

$$b = 2 \left(\frac{M_{11}}{M_{00}} - x_c y_c \right) \quad (3.7)$$

and

$$c = \frac{M_{02}}{M_{00}} - y_c^2 \quad (3.8)$$

Then length l and width w from the centroid are

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}} \quad (3.9)$$

$$w = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}} \quad (3.10)$$

l and w are major length and width of the ellipse to be fit in the segmented object, it can be used for ellipse fitting. Their ratio will also give the “eccentricity”, E , of the object which is in fact how linear or circular the object is.

$$E = \frac{l}{w} \quad (3.11)$$

3.4. Signatures

A signature is a 1D functional representation of the boundary. A simple calculation is to plot the distance from the centroid of the object to the boundary as a plot of angle [12], or the polar coordinate transform of the boundary image from the Cartesian space can be performed. The basic idea behind is to reduce the 2D image to a 1 dimensional form.

Signatures generated by this approach are invariant to translation, but depend on rotation and scaling. Normalization with respect to maximum length can make the signature scale invariant. But rotation can be a problem. One way to solve this problem is to take every boundaries' starting point similar. This can be the first point after rotating the image in its major eigen axis (simply the axis which has the longest length within the image).

Another method to gain rotation invariance is to take Fourier transform of the signature points. After this we use a simple low-pass filter to disable some high frequency responses.

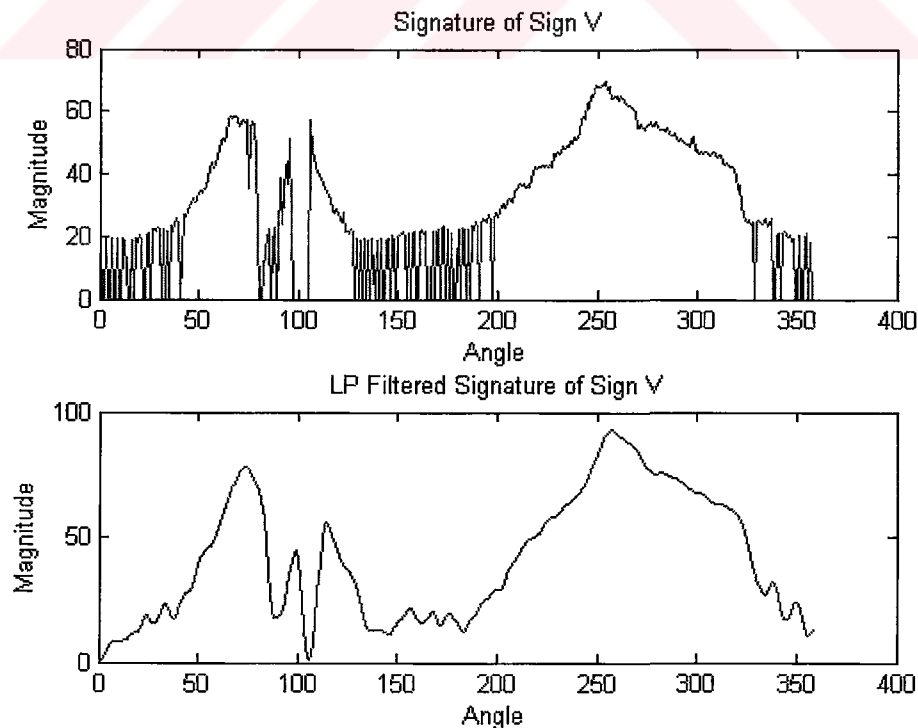


Figure 3.1. Signature and LP Filtered Signature of Sign “V”

For hand images, the signature is a good choice for extracting features. It gives the rough shape of the hand (Figure 3.1) and, in theory even 360 points for 360 degrees can describe the shape properly. But if the object image is not good enough (if it is very noisy, or not good segmented) this approach cannot describe the shape alone.

Using the signature transform, we can count the number of fingers in an image. Given the signature of the hand is normalized in scale $[0, 1]$, basically when we fit a circle in a radius of 0.5, and mask the image the 1-0 and 0-1 crossings mostly will give the finger locations. If working with only one hand, this method can track whole classes, which are dependent on finger locations. But in the two-hand case this method fails, the fingers don't define the class as alone.



3.5. Moment Invariants

The 2D moments work in similar way as 1D moments of a function. If all of the moments (infinite) are taken into account, these can define the shape exactly. But this doesn't simplify the image. From the image moments it is possible to calculate: centroid, area, fitted ellipse location, or other geometric descriptors. But to obtain a translation, scale and rotation invariant measures of the image another method must be used.

Widely used features of an image used in pattern recognition are moment invariants. Seven of the invariant calculations is obtained by Hu in 1967 [17].

The 2D moment of order $(p+q)$ of a digital image $f(x,y)$ is defined as

$$M_{pq} = \sum_x \sum_y x^p y^q f(x,y) \quad (3.12)$$

For $p,q=0,1,2 \dots$ where summations are over values in spatial coordinates x and y spanning the whole image (in this case the segmented area) and x and y , $f(x,y)$ being the value of the pixel coordinate (x,y) . $f(x,y)$ will be 0 or 1 if image is binary, 0 to 255 if image is unsigned integer (8 bit, or grayscale). The corresponding central moment is

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x,y) \quad (3.13)$$

where

$$\bar{x} = x_c = \frac{M_{10}}{M_{00}} \quad (3.14)$$

$$\bar{y} = y_c = \frac{M_{01}}{M_{00}} \quad (3.15)$$

The normalized central moment of order $(p+q)$ is defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma} \quad p, q = 0, 1, 2, \dots \quad (3.16)$$

$$\gamma = \frac{p+q}{2} + 1 \quad p+q = 2, 3, \dots \quad (3.17)$$

A set of seven 2D moment invariants that are insensitive to translation, scale change, mirroring and rotation can be derived from these equations. These are:

$$\varphi_1 = \eta_{20} + \eta_{02} \quad (3.18)$$

$$\varphi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \quad (3.19)$$

$$\varphi_3 = (\eta_{20} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \quad (3.20)$$

$$\varphi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \quad (3.21)$$

$$\begin{aligned} \varphi_5 = & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \quad (3.22)$$

$$\begin{aligned} \varphi_6 = & (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ & + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \end{aligned} \quad (3.23)$$

$$\begin{aligned} \varphi_7 = & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \quad (3.24)$$

The corresponding seven invariant moment vector will be:

$$\varphi = [\varphi_1 \varphi_2 \varphi_3 \varphi_4 \varphi_5 \varphi_6 \varphi_7] \quad (3.25)$$

Calculated moment scales will most likely have a wide dynamic range (e.g. maximum could have 100 and minimum could have a 0.00001 value). To reduce the dynamic range, logarithm of the moment can be taken, and since logarithm of a negative value will give a complex number, the absolute value of the logarithm is taken, which is:

$$\varphi_M = |\log(\varphi)| \quad (3.26)$$

The calculated results in theory are invariants but in practice there is a small change if the image is rotated or rescaled due to quantization errors in the image, since there is a finite number of pixels to be dealt with (finite bandwidth).



4. ARTIFICIAL NEURAL NETWORKS

4.1. Introduction

In Artificial Neural Network (ANN) is a way of solving problems in the sense of human (or animal) biology [18]. This can be simulated by the function of brain, which processes the information gathered from whole body, using neurons for acquisition and transfer. Neurons are a complicated network around the whole body, acquiring information via body transducers (heat, sound, pain etc.) then converts these information to the brain via chemical reactions, similar to electrical information. All information gathered previously are, in fact learnt in a storage (brain), then processed for proper reaction if a similar information is retrieved. This reaction can be “remembering a person that we have just met a day ago”. This learning process corresponds to adjustments in synaptic connections that exist between neurons. Artificial neural networks take this system into account and tries to simulate the natural processing in an artificial or computer aided fashion.

4.2. Historical background

Although neural networks appear to be a fashion of research nowadays, this field was established before the advent of computers.

The modern era of neural networks began with the early works of McCulloch, a neuroanatomist and Pitts [19], a mathematician in 1943. They introduced logical calculations by using parallel synaptic connections and producing a very simple neural network model. This work influenced von Neumann to use idealized switch-delay elements in the construction of the EDVAC (Electronic Discrete Variable Automatic Computer) which was an improved version of ENIAC. In 1986, the back-propagation algorithm was first reported to be developed by Rumelhart, Hinton and Williams [20] . Nowadays, with the relatively fast speed of personal and research computers, the research on both artificial neural network models and their implementations are studied in a growing fashion. Nowadays, neural networks can be implemented in hardware.

4.3. Neural Networks Versus Conventional Computation

Neural networks, with their ability to derive meaning from complicated data, can be used to extract patterns and detect information that is too complex to be noticed by other computer techniques. In fact there is no exact explanation of how a network understands the system. A trained neural network can be thought in the category of information it has given to analyze, this can be a complex function approximation or a person face detection system. Then this “expert” network provides projections given new situations and answers the “what if” questions.

Some advantages of Artificial Neural Networks can be:

- **Adaptive learning:** An ability to learn how to perform tasks based on the data given for training or initial experience.
- **Self-Organization:** An ANN can create its own organization or representation of the information it receives during learning phase.
- **Real Time Operation:** ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- **Fault Tolerance via Redundant Information Coding:** Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

Conventional computers use an algorithmic approach, following a set of instructions in order to solve a problem. If there is a missing step, the computer cannot solve the problem. Thus one must in fact already know the answer to the problem, to state the problem to the computer.

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. Neural networks learn by example, thus cannot be programmed to perform a specific task. Neural network is successful in solving as long as it has carefully choosing examples for input, else the results may be corrupted. The disadvantage is that because the network finds out how to

solve the problem by itself, its operation can be unpredictable. On the other hand, conventional computers use programs created by human to solve problems, thus the whole operation and results are predictable; if anything goes wrong it is due to a software or hardware fault.

Neural networks and conventional algorithmic in fact complement each other. There are tasks that are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks require systems that use a combination of the two approaches in order to perform at maximum efficiency.

4.4. Biological Neuron

In the human brain, a typical neuron collects signals from others through host structures called *dendrites*. The neuron sends out spikes of electrical activity through a long, thin way known as an *axon*, which splits into a set of branches. At the end of each branch, a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity from the axon, into electrical effects that inhibit or excite activity in the connected neurons. When a neuron (Figure 4.1) receives excitatory input that is sufficiently large compared with its inhibitory input (like a threshold voltage), it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes, thus changing the threshold value.

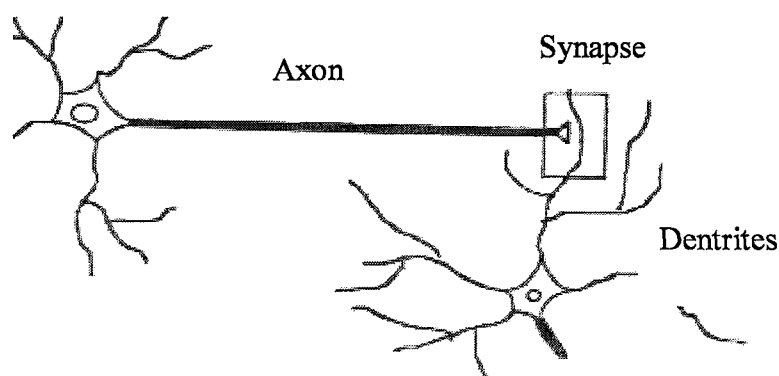


Figure 4.1. A basic neuron structure

These biological neural network model is translated into a computational model (Figure 4.2), making it an artificial neural network. Although not all the information about the biological model is known, the basic artificial model is a gross idealization of real networks of neurons.

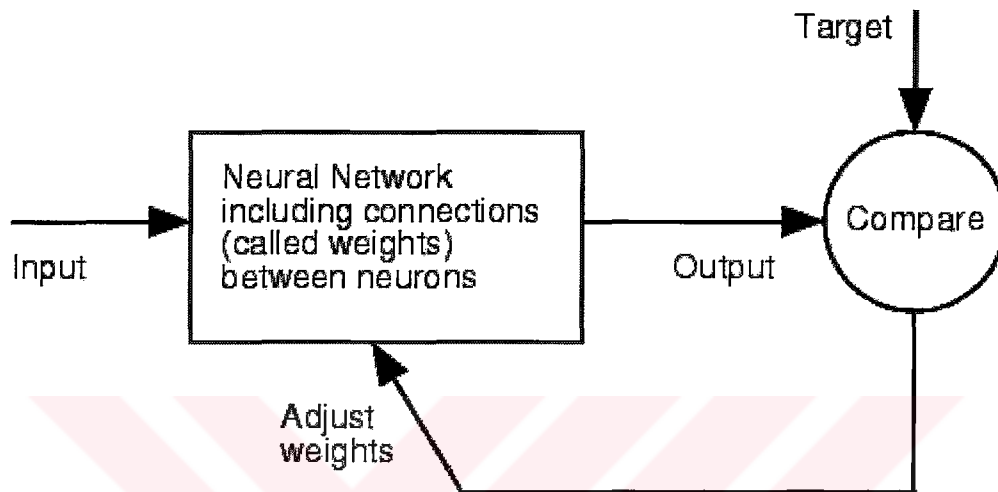


Figure 4.2. Artificial Neuron Structure

4.5. Mathematical Model

Each neuron performs a simple computation. It receives signals from its input links and it uses these values to compute the activation level (or output) for the neuron, using an activation function. This value is passed to other neurons via its output links. The input value received of a neuron is calculated by summing the weighted input values from its input links.

In mathematical terms, a neuron k may be described by:

$$u_k = \sum_{j=1}^m w_{kj} x_j \quad (4.1)$$

and

$$y_k = \varphi(u_k + b_k) \quad (4.2)$$

Where x_1, x_2, \dots, x_m are the input signals; $w_{k1}, w_{k2}, \dots, w_{km}$ are the synaptic weights of neuron k ; u_k is the linear combiner output due to the input signals; b_k is the bias; $\varphi(\cdot)$ is the activation function; and y_k is the output signal of the neuron. The use of bias has the effect of an affine transformation to the output u_k , making the output shift. When the bias is included in the model by assuming x_0 to be the bias ($w_{k0}=b_k$), and v_k can be used as bias plus output (Figure 4.3).

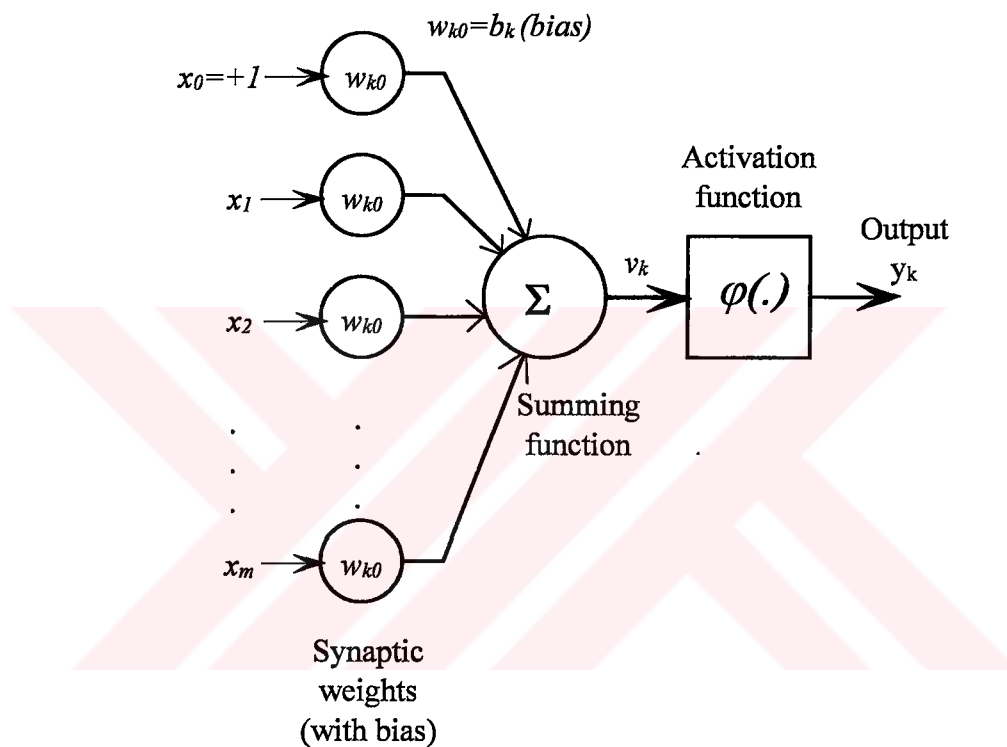


Figure 4.3. General Mathematical Neuron Model

4.5.1. Types of Activation Functions

4.5.1.1. Threshold Function

Threshold Function can be described as

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \quad (4.3)$$

This function is also called *hard limiter*. The output neuron k using this function is expressed as

$$y_k = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \quad (4.4)$$

where v_k is the induced local field of the neuron, meaning

$$v_k = \sum_{j=1}^m w_{kj} x_j + b_k \quad (4.5)$$

This type of neuron is referred to in literature as *McCulloch-Pitts model*. The output is 1 induced local field of that neuron is nonnegative and 0 otherwise. This describes the *all-or-none* property of McCulloch-Pitts model.

4.5.1.2. Piecewise-Linear Function

Piecewise-Linear Function has the property

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v, & +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \quad (4.6)$$

where the amplification factor, thus the slope of the line, is assumed to be one. This function compresses the range of output in a linear fashion, and can be viewed as an approximation of a non-linear amplifier.

4.5.1.3. Sigmoid Function

Sigmoid Function is the most common form of activation used in ANN research. This function is a balance between linear and nonlinear behavior. The infinite range of values is compressed to a maximum finite number, and the range is divided exponentially. An example is the *logistic function* (or logsig, or log-sigmoid) as defined:

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \quad (4.7)$$

Where a is the slope parameter of the sigmoid function. A sigmoid function assumes a continuous range of values from 0 to 1, and since it is continuous (on the contrary to threshold or linear functions) it is differentiable.

When negative values are needed in output, not only 0 and 1 but -1 as well then *hyperbolic tangent function* (tansig, or tan-sigmoid) can be used as:

$$\varphi(v) = \tanh(v) \quad (4.8)$$

Plots of these sample activation functions can be seen in Figure 4.4.

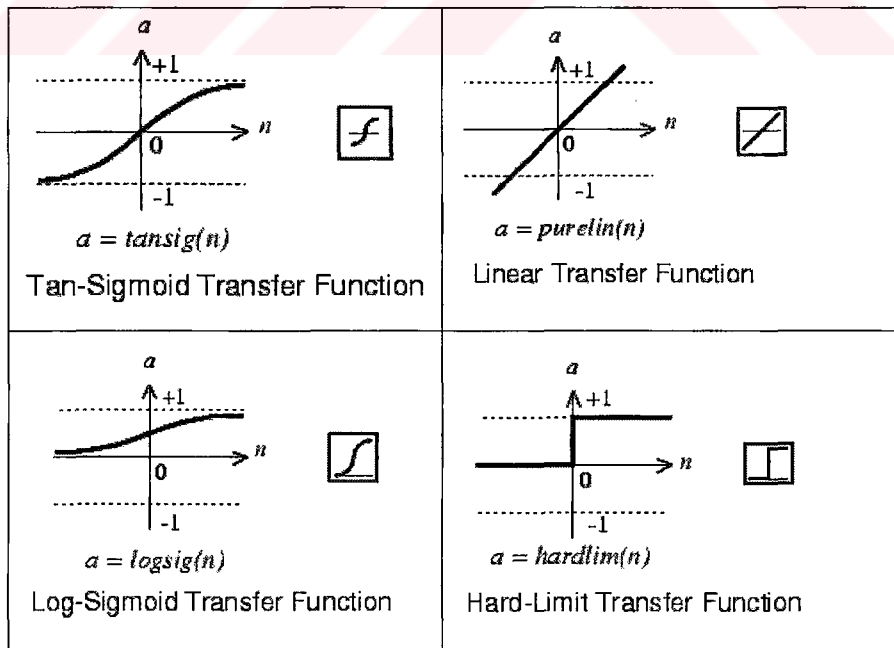


Figure 4.4 Activation Functions

4.5.2. Learning

ANN learns by training, and to train network there must be some targets or desired responses to give the network. Then the network compares its own calculated output with the target data (target output), if there is a mistake (error signal) it must redesign itself (recalculate the synaptic weights). Assuming a single neuron k , $d_k(n)$ being the target output error at time step n , can be formulated as:

$$e_k(n) = d_k(n) - y_k(n) \quad (4.9)$$

This error signal is a control mechanism, to correct the weights of neuron k , making the output signal $y_k(n)$ closer to the target $d_k(n)$ step by step in time (Figure 4.5). This is achieved by minimizing a general cost function $\xi(n)$ to calculate the performance:

$$\xi(n) = \frac{1}{2} e_k^2(n) \quad (4.10)$$

which is the instantaneous value of the error energy. The step-by-step adjustment is continued until the system reaches a desired steady state (until a given error threshold). This is thus an error-correction mechanism.

To redesign the synaptic weights, information from the error must be gained. The basic rule for correction is *Delta rule* or *Widrow-Hoff rule* named after its creators in 1960. Let $w_{kj}(n)$ denote the value of synaptic weight of neuron k excited by element $x_j(n)$ of the signal vector $\mathbf{x}(n)$ at time step n . According to the Delta rule the adjustment $\Delta w_{kj}(n)$ applied to the synaptic weight w_{kj} at time step n is:

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n) \quad (4.11)$$

Where η is a positive constant that determines the rate of learning in that step, and called as learning rate parameter. When the adjustment is computed, the updated value of w_{kj} will be:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \quad (4.12)$$

These *old* and *new* terms can also be written as

$$w_{kj}(n) = z^{-1} [w_{kj}(n+1)] \quad (4.13)$$

where z^{-1} is the *unit-delay operator* representing a *storage element* (memory).

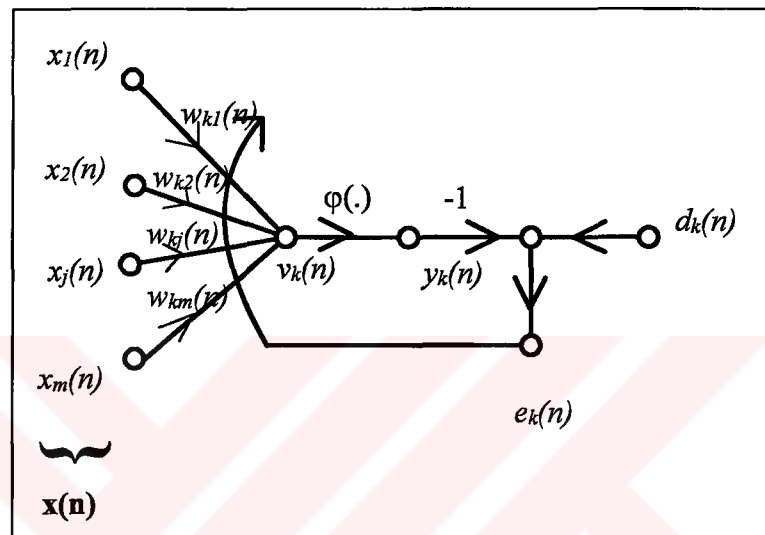


Figure 4.5. Signal-flow Model of One Neuron Learning Process

This kind of training that has a target dataset is called *Supervised learning*. The simplest one (having an only one layer) is called *Perceptron*. Each trained output has a *target class* in the target database, when the computed response has no proper class for a given input in the test phase (after training) then either there is no such target for that information or the network is falsely trained. Training which has no initial target dataset is called *unsupervised learning*. The target classes are automatically constructed by network and each new output that has not a similar target class is assigned a new target class.

4.5.3. Multi Layer Perceptron

4.5.3.1. Introduction to MLP

The network consisting of a set of source nodes (neurons) that constitute the input layer, one or more hidden layers of computation nodes, and an output layer of computations are called *multilayer feed forward networks*. The input signal is processed to the output one layer at a time. These ANN types are commonly named as *multilayer perceptrons* (MLP), which represents a generalization of a perceptron.

These kinds of networks are highly successful in difficult problems by training in a supervised manner with the popular *error backpropogation algorithm*, based on *error-correction learning rule*.

4.5.3.2. Backpropogation Algorithm

The error backpropogation learning consists of a two pass through the different layers of network: a forward pass and a backward pass. In the forward pass, an input vector is applied to the nodes of the network computing the effect in each layer, and an output is found. During the forward pass the synaptic weights are fixed. During the backward pass, the synaptic weights are all adjusted according to an error-correction rule. The actual response for the network (the output) is subtracted from a target response to produce an error signal. Then the error is propagated through the network backwards, adjusting all the weights according to a formula. This algorithm is also called the *backpropogation algorithm*.

4.5.3.3. Mathematics of Backpropogation

The error signal at the output of neuron j at time step n (n th training example) is defined by:

$$e_j(n) = d_j(n) - y_j(n) \quad (4.14)$$

Since all neurons have instantaneous error energy, the instantaneous total energy is computed as summing all the neurons in the output layer:

$$\xi(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n) \quad (4.15)$$

Where set C includes all neurons in the output layer. When there are N_{total} number of examples the *average squared error energy* will be

$$\xi_{av} = \frac{1}{N} \sum_{n=1}^N \xi(n) \quad (4.16)$$

For a given training set ξ_{av} represents the cost function, as a measure of learning performance. The objective of learning is to minimize the cost function, when the cost function approaches zero, the network will be able to detect and classify all the inputs similar to the training set correctly.

The backpropagation algorithm applies a correction of $\Delta w_{ji}(n)$, which is proportional to the partial derivative of error to the synaptic weight, and can be written according to the chain rule:

$$\frac{\partial \xi(n)}{\partial w_{ji}(n)} = \frac{\partial \xi(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)} \quad (4.17)$$

And with some differentiation with respect to error, output and the sum function, this equation is minimized to:

$$\frac{\partial \xi(n)}{\partial w_{ji}(n)} = -e_j(n) \varphi_j'(v_j(n)) y_j(n) \quad (4.18)$$

The correction applied to the synaptic weight is defined by the modified delta rule as:

$$\Delta w_{ji}(n) = -\eta \frac{\partial \xi(n)}{\partial w_{ji}(n)} \quad (4.19)$$

Where η is the learning rate parameter, and minus sign accounts for gradient descent (a direction) in weight space. Finally the correction is:

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_j(n) \quad (4.20)$$

Where the local gradient $\delta_j(n)$ is defined as

$$\delta_j(n) = e_j(n) \varphi_j'(v_j(n)) \quad (4.21)$$

In the application of backpropagation algorithm, two passes are processed.

In the forward pass the weights remain unchanged through the network, and output is calculated by neuron-by-neuron basis.

$$y_j(n) = \varphi_j(v_j(n)) \quad (4.22)$$

where $v_j(n)$ is the induced local field of neuron j and computed as:

$$v_j(n) = \sum_{i=0}^m w_{ji}(n) y_i(n) \quad (4.23)$$

where $w_{ji}(n)$ is the synaptic weight connecting neuron i to neuron j at time n , and $y_i(n)$ is the input to the neuron j . If neuron j is first layer then the input is the general input of the network, if it is in a hidden layer then its input is the output of the previous layer, and calculations are done in a standard way. But when the j^{th} neuron is in the output layer it is compared to the desired response then the backward pass occurs.

The backward pass starts output layer by passing the error signals leftward through the network, layer by layer, and recursively computing local gradient for each neuron. For a neuron in the output layer, the local gradient δ is simply the error signal of that neuron multiplied by the first derivative of its nonlinearity.

If the activation function is logistic function then:

$$\varphi_j(v_j(n)) = \frac{1}{1 + \exp(-av_j(n))}, \quad a > 0 \text{ and } -\infty < v_j(n) < \infty \quad (4.24)$$

Differentiating with respect to $v_j(n)$, we get

$$\varphi_j'(v_j(n)) = \frac{a \exp(-av_j(n))}{[1 + \exp(-av_j(n))]^2} \quad (4.25)$$

with $y_j(n) = \varphi_j(v_j(n))$, the exponential term can be eliminated, and the derivative is expressed by :

$$\varphi_j'(v_j(n)) = ay_j(n)[1 - y_j(n)] \quad (4.26)$$

For a neuron j in the output layer, $y_j(n) = o_j(n)$, hence the local gradient for neuron j is :

$$\begin{aligned} \delta_j(n) &= e_j(n)\varphi_j'(v_j(n)) \\ &= a[d_j(n) - o_j(n)]o_j(n)[1 - o_j(n)] \end{aligned} \quad (4.27)$$

Where $o_j(n)$ is the function signal at the output of neuron j .

For an arbitrary hidden neuron j , the local gradient is expressed as

$$\begin{aligned} \delta_j(n) &= \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n) \\ &= ay_j(n)[1 - y_j(n)] \sum_k \delta_k(n) w_{kj}(n) \end{aligned} \quad (4.28)$$

The whole one process learning of a training of one set inputs is called one epoch.

4.5.3.4. Backpropagation Models

The general backpropagation algorithm described above is also called as *batch-gradient descent algorithm*, and called in Matlab as **traingd** [21].

For a more general approach to maximize the learning capability a positive momentum term is included in the adjustment term, making the synaptic weight adjusting term as:

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_j(n) \quad (4.29)$$

The momentum term α is an adjustment to control the learning rate, minimizing the probability of any oscillation when learning-parameter η is chosen large to gain a faster synaptic change. When the momentum term is used the algorithm is called as *Batch Gradient Descent with Momentum*, and expressed in Matlab as **traingdm**.

Multilayer networks typically use sigmoid transfer functions in the hidden layers. Sigmoid functions are characterized by the fact that their slope must approach zero as the input gets large. This causes a problem when using gradient descent to train a multilayer network with sigmoid functions, since the gradient can have a very small magnitude, but instead making oscillations in the output.

The resilient backpropagation (or Rprop as called in literature) training algorithm suggested by Riedmiller and Braun in 1993 [22], has the effect to eliminate the harmful effects of the magnitudes of the partial derivatives, also causing the backpropagation system to compute faster. Only the sign of the derivative is used to determine the direction of the weight update; the magnitude of the derivative has no effect on the weight update. The size of the weight change is determined exclusively by a weight-specific update value.

$$\Delta w_{ij}(n) = \begin{cases} -\Delta_{ij}(n), & \text{if } \frac{\partial \xi(n)}{\partial w_{ji}(n)} > 0 \\ +\Delta_{ij}(n), & \text{if } \frac{\partial \xi(n)}{\partial w_{ji}(n)} < 0 \\ 0, & \text{else} \end{cases} \quad (4.30)$$

The second step of Rprop is determining the new update-values:

$$\Delta_{ij}(n) = \begin{cases} \eta^+ \Delta_{ij}(n), & \text{if } \frac{\partial \xi(n-1)}{\partial w_{ji}(n)} \frac{\partial \xi(n)}{\partial w_{ji}(n)} > 0 \\ \eta^- \Delta_{ij}(n), & \text{if } \frac{\partial \xi(n-1)}{\partial w_{ji}(n)} \frac{\partial \xi(n)}{\partial w_{ji}(n)} < 0 \\ \Delta_{ij}(n-1), & \text{else} \end{cases} \quad (4.31)$$

where $0 < \eta^- < 1 < \eta^+$. Every time the partial derivative of the corresponding weight changes its sign, the update-value $\Delta_{ij}(n)$ is decreased by the factor η^- , likewise if the derivative does not change its sign the update is slightly increased. If the derivative is zero, then the

update value remains the same. By default η^- (delta decrease) is set to 0.5, and η^+ (delta increase) is set to 1.2. The Rprop function name used in Matlab is **trainrp** and it has promising convergence rates and speed in pattern recognition applications.

For a ANN system to be trained properly multiple epochs with different input-target couples must be done. Whenever the network is trained with more training data the cost function will reduce, making the network more stable to similar test inputs (which are in fact the inputs to be recognized) .



5. EXPERIMENT RESULTS AND DISCUSSION

5.1. Experiment Process

Once recognized by the computer system, hand gestures can be used to command any machine, it can type a letter from the alphabet or just send a control sign to a robot to turn left. Proposed system in this thesis is an off-line method for recognition of hand gestures, thus does not perform on-line realtime gesture recognition. Source code for the system in Matlab and database gestures can be found in Appendix A. With some modifications on-line recognition can be implemented in future work. These modifications are in fact real-time video acquisition, a hand posture flag recognizer (thus the system will know that there is a hand sign input to feed the neural network), and an output interpreter (to do the meaning of the sign, e.g. if sign is A (class 1) in the output of classifier, type "A" on the screen).

5.1.1. Image input

Hand image is obtained with a simple USB camera, the gesture database is constructed directly from Matlab. To construct a database for a person, the system shows the guide sign for the person and 5 gestures for that letter is entered.

5.1.2. Image processing

Obtained images are process to get the hand shape image (silhouette).

1. For each person a skin probability matrix is constructed, using maximum 5 images from that person's hand database. An area of skin region is taken from each 5 image and the system is trained to obtain skin color probability matrix. Then each image in the database is compared with this matrix. The compared values which are greater than skin probability threshold value 0.1, gets the binary skin image, which includes pure hands or face plus some noise.

2. The noise in the image is mostly eliminated by performing open-close operation on the binary image. Open-close operation also makes the small gaps disappear in the image.
3. After open-close operation, connected components analysis is performed. Each connected component is labeled with a number value.
4. The area of each connected component is calculated and the one which has the largest area greater than a certain threshold is taken as binary hand image. Properties of hand region (area, eccentricity, location, number of holes) are stored.
5. Filled hand region (to discard the holes) is processed for feature extraction.
6. Then each sign is assigned to its class directory under the directory named after that person.

5.1.3. Feature extraction

Binary hand image features are extracted as followed:

1. Invariant calculation function is used on hand region to obtain 7 Hu moment invariants. Absolute value of the logarithm of invariant moments are taken to widen the dynamic range.
2. Compactness and eccentricity features are calculated from the image moments.
3. For the ASL system these 9 values are composed to perform a 9x1 vector input for neural network.
4. For the TSL system signature of the hand region is calculated, then filtered through a low-pass filter to eliminate high frequency components. Filtered signature is normalized by its maximum value to gain scale invariance. Fourier transform of filtered signature is taken to gain rotation invariance. Then the absolute value of the

Fourier transform is taken to eliminate complex numbers. Maximum signature is a 360x1 vector. But since the high frequency components are eliminated it is seen that only a finite number of first components of the Fourier transform are efficient as good as the whole signature vector. Different test has been performed for 40 and 60 Fourier components as well as 360 full components.

5. Another feature vector, additional approximate finger location information, can also be calculated from the signature information. The normalized signature whose threshold is greater than 0.5 will give approximate finger locations.

5.1.4. Neural network classifier

Gradient descent, gradient descent with momentum and resilient backpropagation algorithms (Rprop) are tested with the system with different layers and different number of neurons. For each class the target vectors are assumed to be incremental binary values, e.g. for class 1 (sign A) the target is [0 0 0 1], class 2 target is [0 0 1 0], class 6 target is [0 1 1 0] and so on. This assumption has advantage that when the classes are increased target vectors are automatically increased, also this is an easier way to compute the error response of the system. Since the gradient descent algorithm does not give good results, Rprop algorithm was used in all tests, because of its fast convergence, memory and computation efficiency.

5.2. Sample Gesture Processing

Original image was read and compared with skin color probability lookup table and probability image was constructed (Figure 5.1)

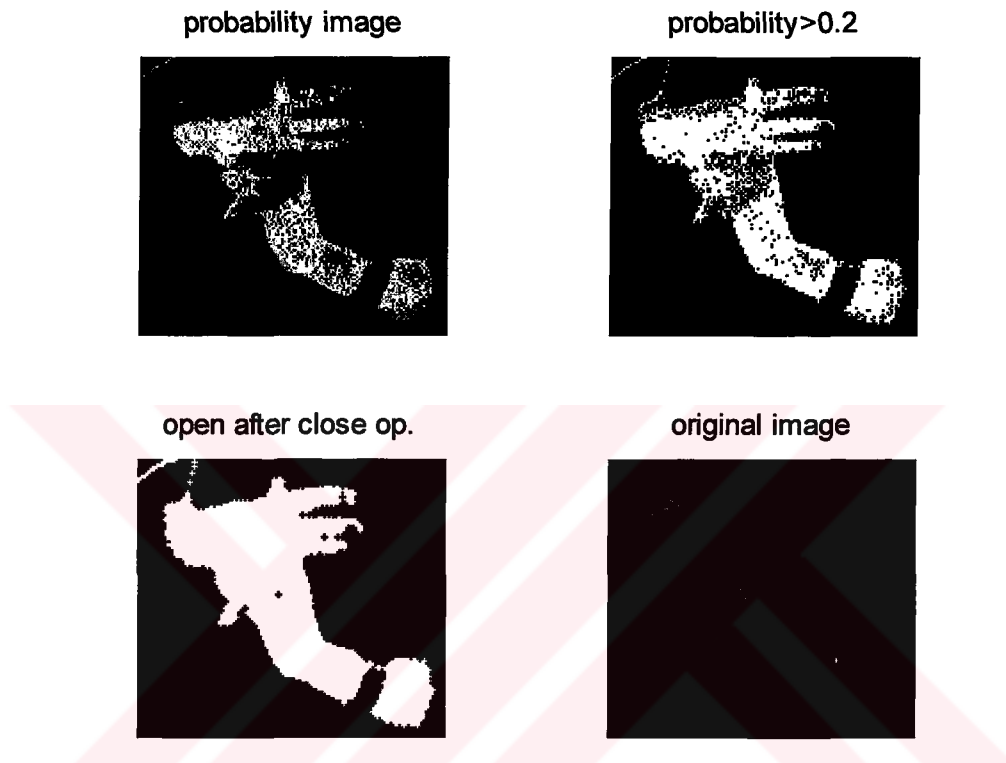


Figure 5.1. Probability image

Open operation was performed after thresholding the image (Figure 5.2) then using connected component labeling largest area is obtained (Figure 5.3).



Figure 5.2. Open Operation



Figure 5.3. Largest Area After Labeling.

Additional feature information is calculated from the segmented image. Area is found to be 4284 and perimeter 416.73, then the compactness is:

$$C = \frac{A}{P^2} = \frac{4284}{(416.73)^2} = 0.024 \quad (5.1)$$

Eccentricity is found to be 0.78 from the image moments. Also the 7 invariant moments are:

$$\varphi_m = [0.2469 \quad 0.0125 \quad 0.0072 \quad 0.0007 \quad 0.0000 \quad 0.0001 \quad 0.0000] \quad (5.2)$$

To widen the dynamic scale with positive values absolute value of the logarithm of the moments were taken:

$$|\log(\varphi_m)| = [1.3986 \quad 4.3855 \quad 4.9394 \quad 7.2218 \quad 13.3721 \quad 9.4225 \quad 14.3220] \quad (5.3)$$

Signature plot (Figure 5.4) is computed for 360 degrees of rotation around the centroid of the image.

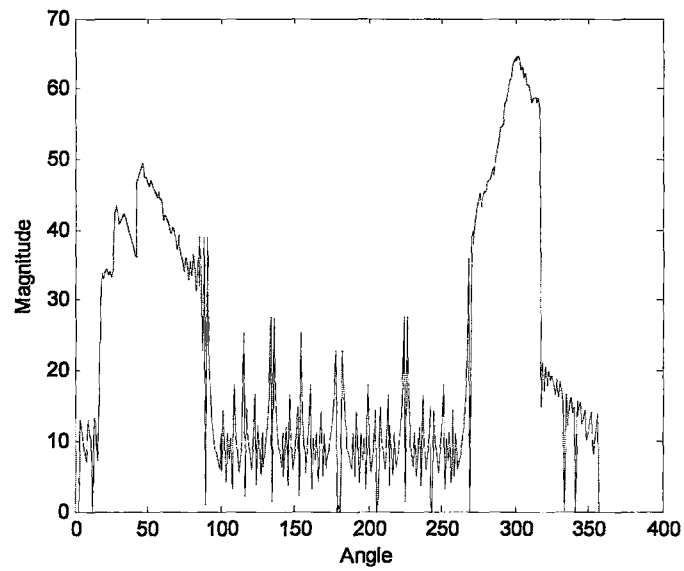


Figure 5.4. Signature plot of Figure 5.3

Filter 0.2 Hz normalized low-pass filter operation (Figure 5.5) was performed on signature plot to remove the noisy data.

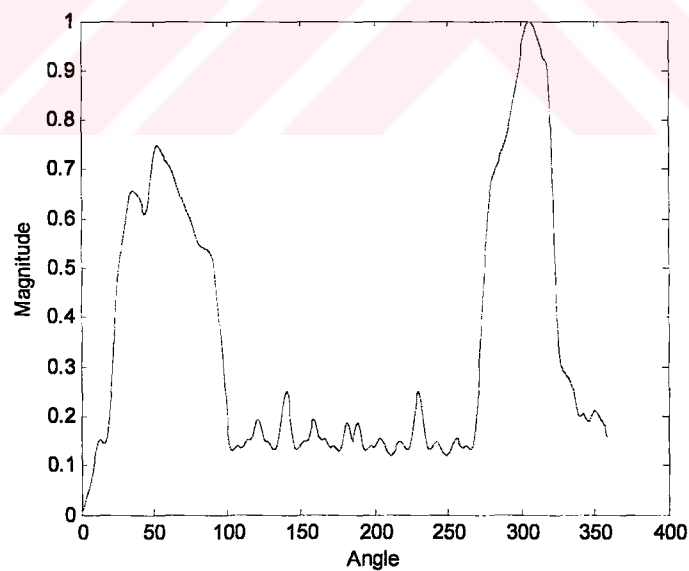


Figure 5.5. Low-Pass filtered Signature of Figure 5.3

Additional approximate finger information (Figure 5.6) was calculated by setting a 0.5 threshold on filtered signature.

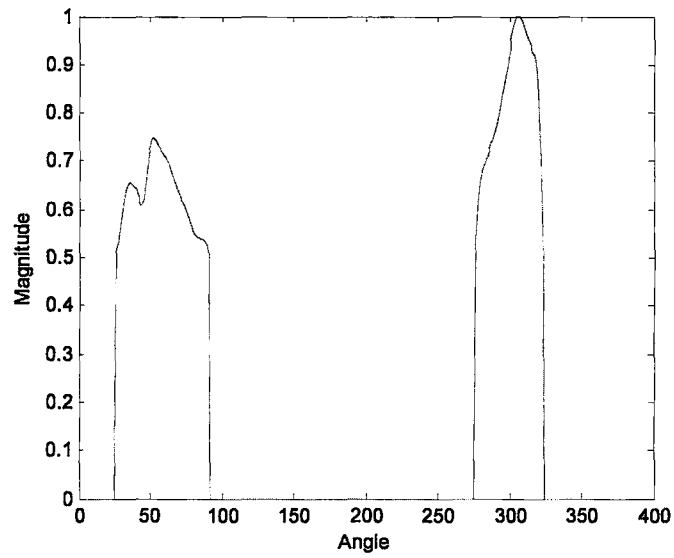


Figure 5.6. Approximate Finger Information

Magnitude of the Fourier Spectrum of Signature and filtered signature and finger information can be seen on Figure 5.7.

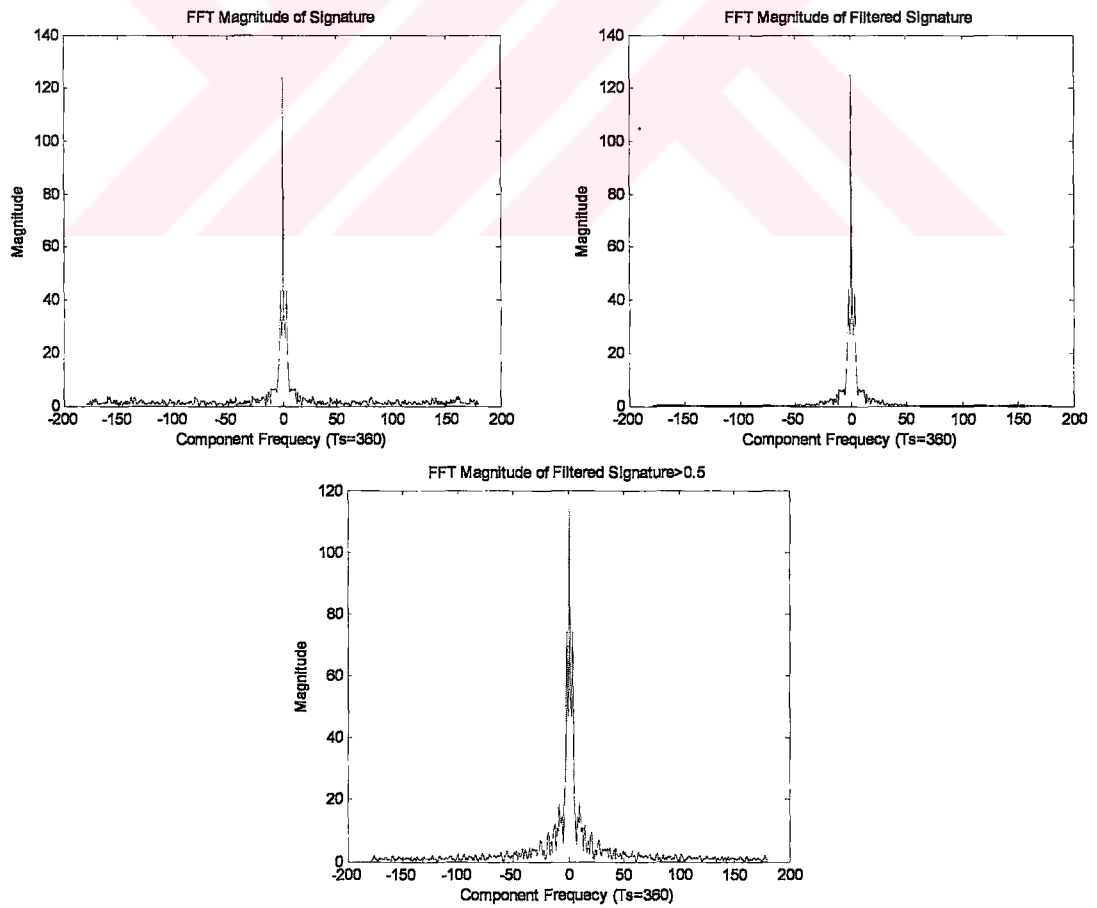


Figure 5.7. Magnitude of Signature, Filtered Signature and Finger Information

5.3. American Sign Language Manual Alphabet

This system used one hand gestures to represent letters of the alphabet (Figure 5.11). Sign J and Z are dynamic signs which need movement of hand in the shape of the letter. Some letters such as R, D, G and Z are so much similar in shape that it is almost impossible to classify each letter independently using only geometric features. This also applies to similar letters of K and V. For the system proposed in this thesis includes just eight of these signs, which are: A, B, C, D, L, V, W and Y.

The database is constructed from 15 people with each person doing the same gesture 5 times. There are a total of 75 gestures for each class, having total number of 600 gestures in the database.

A person does the same gesture 5 times but these 5 gestures differ in orientation and picture as well. Figure 5.8 shows two of the “Y” sign from the same person. Figure 5.9 shows the “C” sign taken from 4 different people, and Figure 5.10 showing “W” sign likewise.

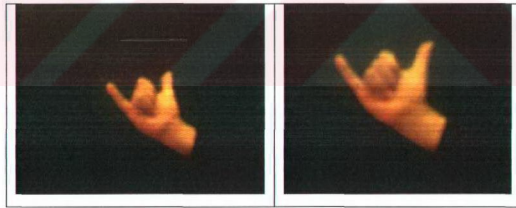


Figure 5.8. Different “Y” sign from same person

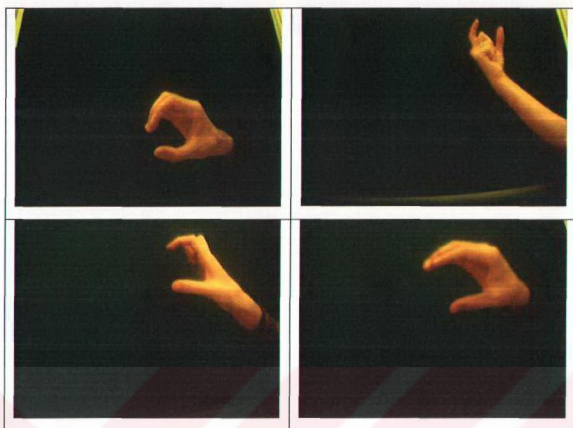


Figure 5.9. Sign "C" from 4 people

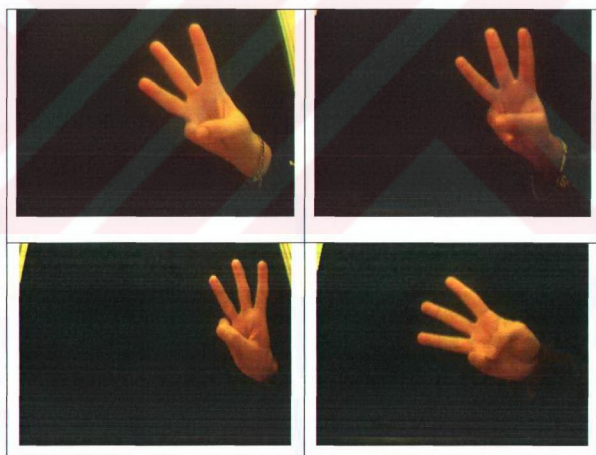


Figure 5.10. Sign "W" from 4 people

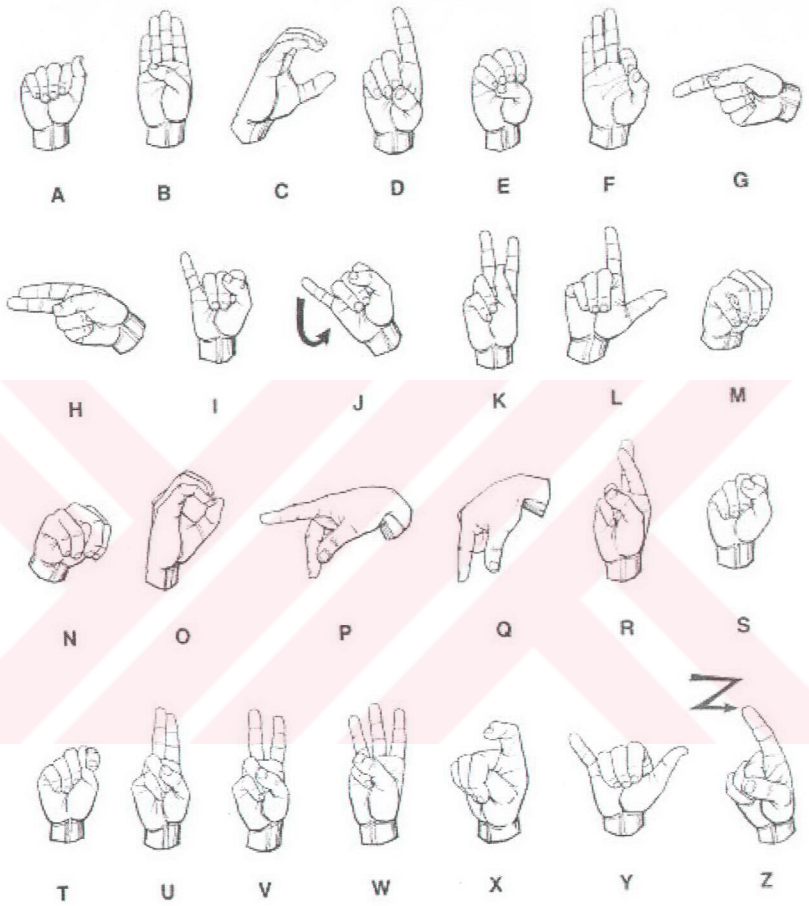


Figure 5.11. American Sign Language Manual Alphabet

5.4. Turkish Sign Language Manual Alphabet

The TSL consists of two hand gestures (Figure 5.15), but these two hand gestures are in fact mixture of the like gestures of one hand gestures in ASL. For a proper Turkish database construction, the letter gesture movies from sign language page from Koc University [23] is also shown to some of the performers, which had no initial experience with sign language. Again some of the gestures in TSL are dynamic and discarded in the system, also similar gestures in shape are discarded as well.

The database is constructed from 15 people with each person doing the same gesture 5 times. Total 21 alphabet letters were used. The dynamic gestures were discarded. When constructing the database, a person did the same gesture 5 times but these 5 gestures differed in orientation and rotations as well. Figure 5.12 shows two of the “E” sign from the same person and Figure 5.14 shows “F” sign of 4 people. Figure 5.13 shows image processing of the “E” sign of the person that was shown in Figure 5.12.

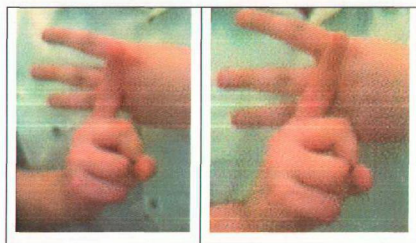


Figure 5.12. "E" Sign from same person



Figure 5.13. Processed "E" Signs of Figure 5.12

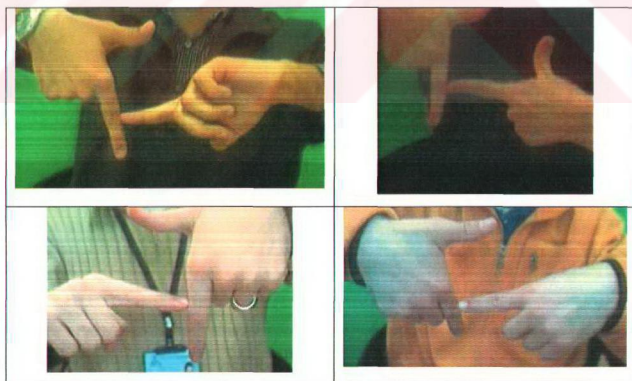
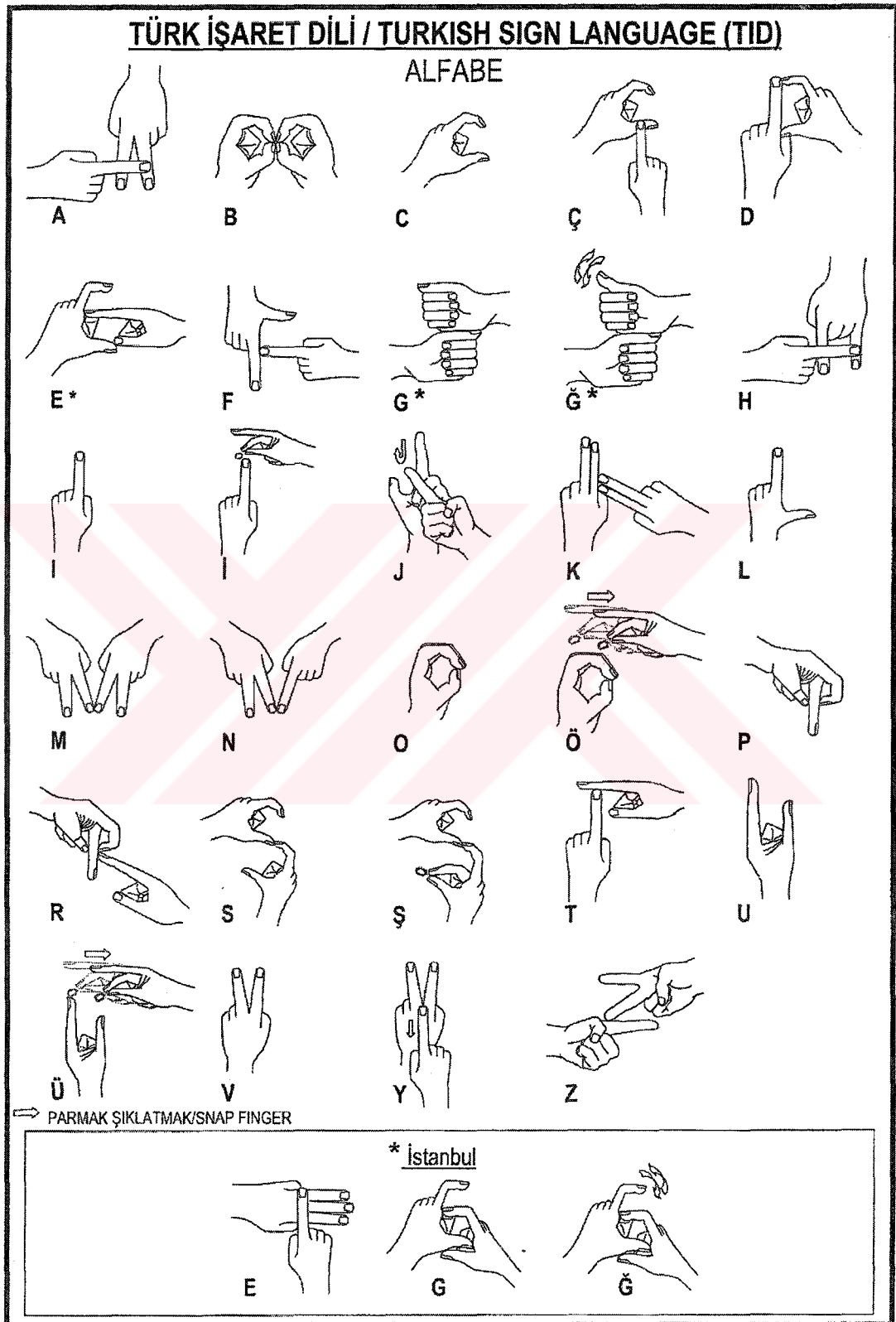


Figure 5.14. "F" sign from 4 people



Approved by the Turkish National Federation of the Deaf December 2002 Türkiye İşitme Engelliler Millî Federasyonu tarafından 12.2002'de kabul edilmiştir

Figure 5.15. Turkish Sign Language Manual Alphabet

5.5. ASL Results

In this test results, an M input N output neural network with K hidden layers will be presented as $M \times K \times N$ network, meaning network has M node (M neurons) input layer, K node hidden layer, and N node output layer . Output layer does not contain any activation function, all operations are performed with input and hidden layers, but mean square error (MSE, general error of the system) is calculated from the output by looking at the difference from the target vector.

The target vector is chosen as 5 column binary vector. Meaning, first class has a target vector as [0 0 0 0 1], the second class has [0 0 0 1 0], third class has [0 0 0 1 1]; which are in fact 5 bit binary numbers in an increasing motion. This choice makes the system more general and the class numbers can be increased automatically.

There are a total of 600 gestures in ASL dataset, in most cases %10 of this total is used as test input (60 gestures) and the rest (540 gestures) is used for training. In some of the networks %15 of the total dataset is used as test input. Tables show the number of test and train vectors used in each class and also the correct and false classes. 9 vector input is generally used for one hand gestures. This 9 vector input includes: compactness, eccentricity and 7 invariant moments.

For comparison Gradient Descent Backpropogation is tested on the ASL database, but it is seen that even modifying momentum term or learning parameters doesn't give good results. One example for the training process of ASL dataset is shown in Figure 5.16. As can be seen, MSE can not fall below 0.1 after 1000 epochs and does the system can not recognize its own training set. Rprop backpropogation is found to be more robust than the Gradient Descent algorithm, also this algorithm is the most memory efficient and fast algorithm compared to other backpropogation algorithms like Levenberg-Marquardt or Quasi-Newton Algorithms [17]. All the networks below use Rprop algorithm for training, with different input and hidden layer numbers.

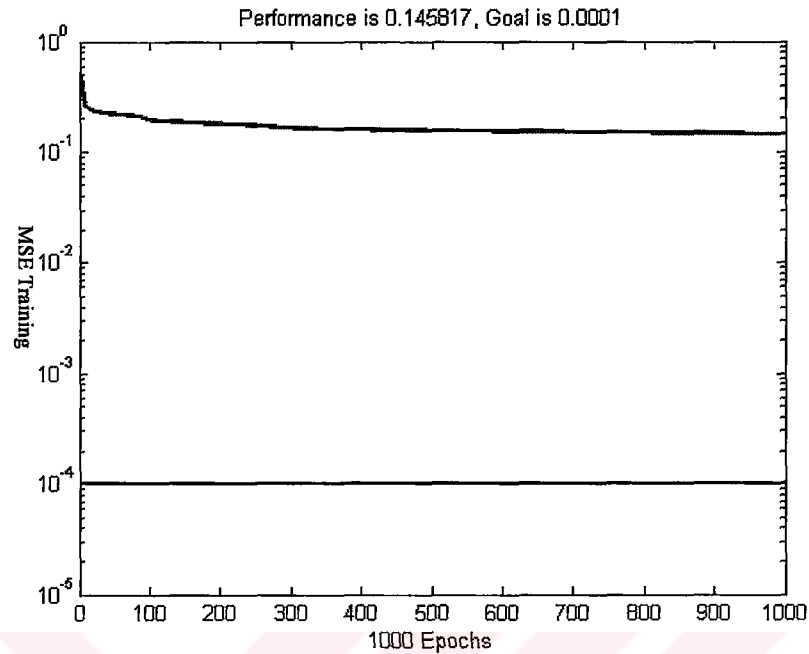


Figure 5.16. ASL dataset trained with Gradient Descent Algorithm.

5.5.1 ASL Network 1

9 column input vector with features ; compactness, eccentricity and 7 invariant moments are used. 1 hidden layer with 15 nodes is used. In 1000 epochs MSE reached 0.032 (Figure 5.17) with Rprop algorithm. As can be seen in Table 5.1, 80 per cent of the training set was correctly classified, and 79 per cent of the test set was correctly classified.

	A	B	C	D	L	V	W	Y
Train Correct	66	44	49	49	52	51	64	54
Train False	9	16	26	11	23	9	11	6
Test Correct	6	6	7	5	7	5	6	6
Test Fail	3	1	1	1	1	2	2	1

Table 5.1. ASL dataset, 9x15x5 network, 9 vector input

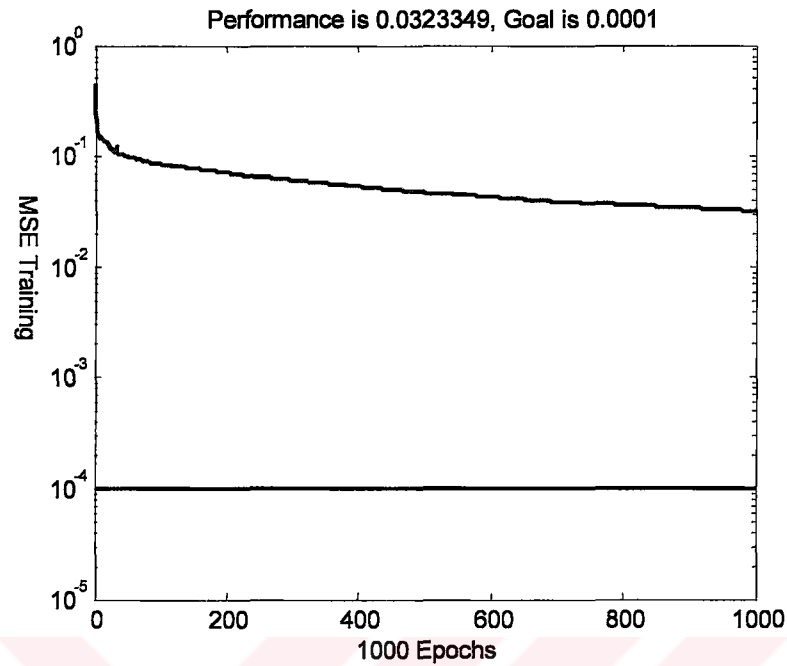


Figure 5.17. ASL 9x15x5 Network MSE graph

5.5.2. ASL Network 2

9 column input vector with features ; compactness, eccentricity and 7 invariant moments are used. 1 hidden layer with 30 nodes is used. In 1000 epochs MSE reached 0.028 (Figure 5.18) with Rprop algorithm. 82 per cent of the training set was correctly classified, and 71 per cent of the test set was correctly classified (Table 5.2). It seems that A and D signs were the bare misclassified classes.

	A	B	C	D	L	V	W	Y
Train Correct	68	45	51	45	59	48	72	56
Train False	7	15	24	15	16	12	3	4
Test Correct	5	7	6	3	6	3	7	6
Test Fail	4	0	2	3	2	4	1	1

Table 5.2. ASL 9x30x5 network

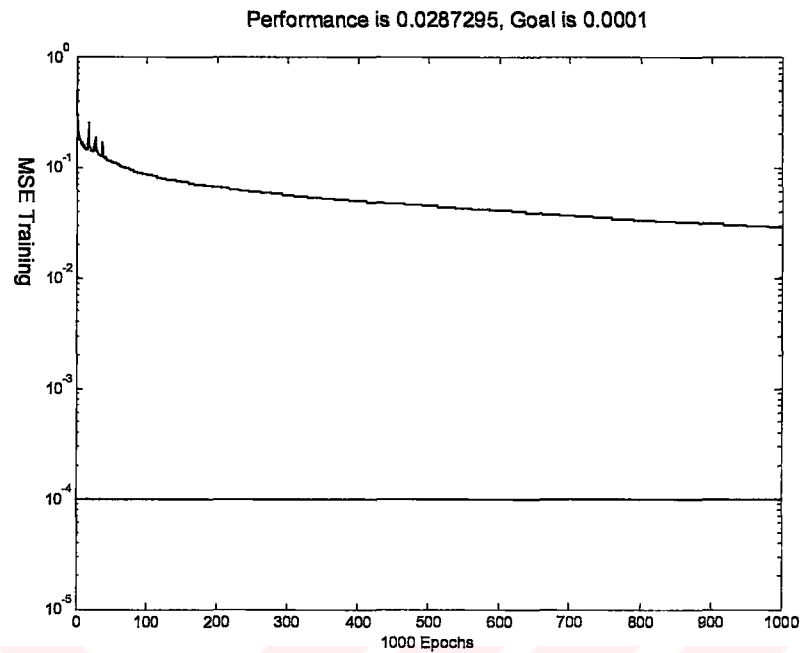


Figure 5.18. ASL 9x30x5 network MSE graph

5.5.3. ASL Network 3

9 column input vector with features ; compactness, eccentricity and 7 invariant moments are used. 2 hidden layers with 15 and 30 nodes was used. In 1000 epochs MSE reached 0.016 (Figure 5.19) with Rprop algorithm. 92 per cent of the training set was correctly classified, and 81 per cent of the test set was correctly classified (Table 5.3).

	A	B	C	D	L	V	W	Y
Train Correct	73	56	69	54	63	56	75	55
Train False	2	4	6	6	12	4	0	5
Test Correct	7	6	7	4	7	5	8	5
Test Fail	2	1	1	2	1	2	0	2

Table 5.3. ASL 9x15x30x5 Network Results

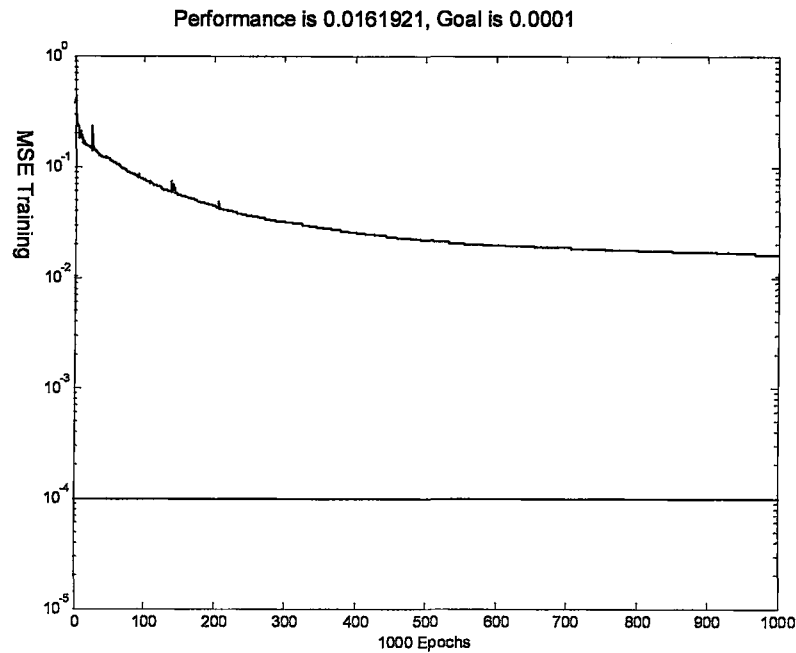


Figure 5.19. ASL 9x15x30x5 MSE Graph

5.5.4. ASL Network 4

9 column input vector with features; compactness, eccentricity and 7 invariant moments are used. 2 hidden layers with 30 and 15 nodes were used, instead of 15 and 30 in the network 4. In 1000 epochs MSE reached 0.007 (Figure 5.20) with Rprop algorithm. 96 per cent of the training set was correctly classified, and 81 per cent of the test set was correctly classified (Table 5.4). Changing the order of hidden layers seems to have a good effect.

	A	B	C	D	L	V	W	Y
Train Correct	70	59	74	56	73	58	74	56
Train False	5	1	1	4	2	2	1	4
Test Correct	6	5	6	5	6	6	8	7
Test Fail	3	2	2	1	2	1	0	0

Table 5.4. ASL 9x30x15x5 Network Results

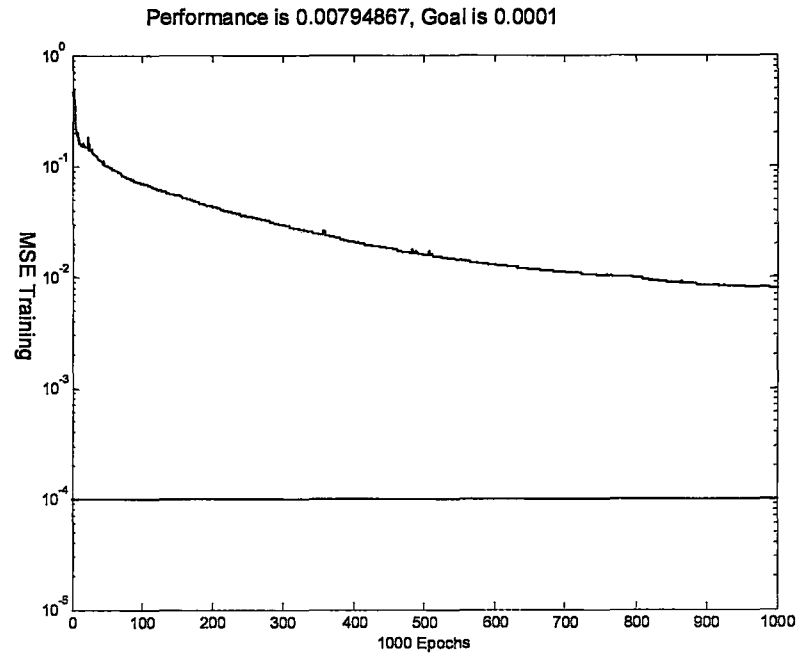


Figure 5.20. ASL 9x30x15x5 MSE Graph

5.5.5. ASL Network 5

In all the above networks, delta increase factor of Rprop was 1.3 and delta decrease was 0.3, when we modify these factors and make them 1.1 and 0.1, the network performance drops down (Figure 5.21), also results are also not good (Table 5.5). Also increasing these factors (1.9 and 0.9) makes the network worse (Figure 5.22). It seems that delta increase and delta decrease are best set to 1.3 and 0.3 respectively for proper results.

	A	B	C	D	L	V	W	Y
Train Correct	68	49	42	34	42	43	64	42
Train False	7	11	33	26	33	17	11	18
Test Correct	6	6	6	4	5	3	5	5
Test Fail	3	1	2	2	3	4	3	2

Table 5.5. ASL 9x30x15x5 Network Results modified delta

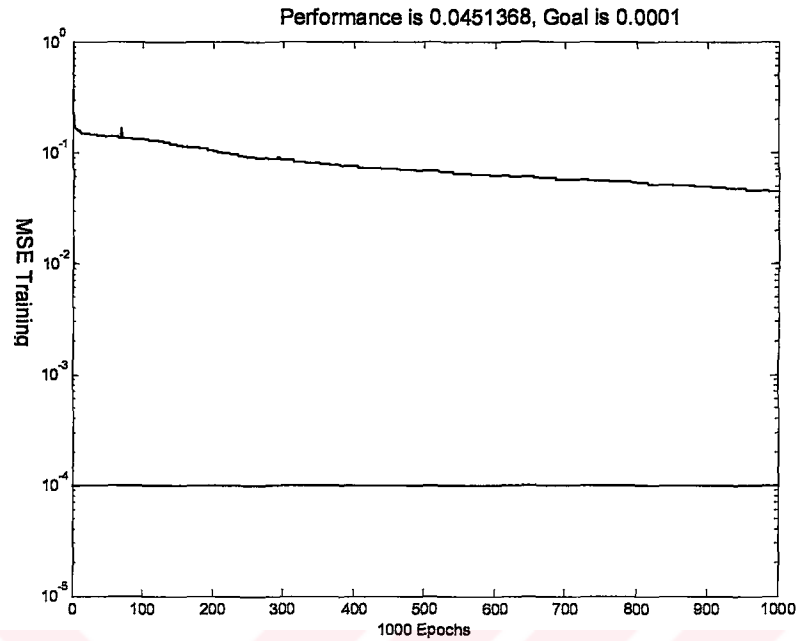


Figure 5.21. ASL 9x30x15x5 MSE Graph (Delta inc=1.3, Delta dec=0.3)

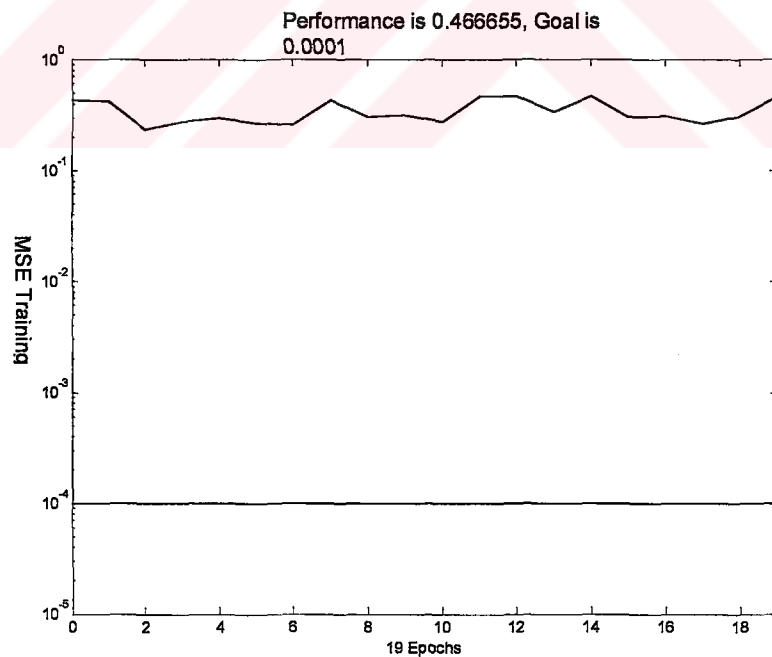


Figure 5.22. ASL 9x30x15x5 MSE Graph (Delta inc=1.9, Delta dec=0.9)

5.5.6. ASL Network 6

In this test 3 hidden layer structure was used. First hidden layer has 30 nodes, second 20 nodes and the third one 15 nodes. Input was 9x1 input, including moment invariants, compactness and eccentricity. This test achieved a training correct ratio of 94 per cent and training correct ratio of 89 per cent, which was the best result comparing to other ASL tests. Table 5.6 shows the output of network for each class of the training set, and Table 5.7 shows the output of network for each class of the test set.

Table 5.7 shows that mainly letter “D” is misclassified as B and letter “W” has a misclassification of being either “L”, “D” or “V”. In same cases letters “B” and “Y” could not be separated to any class.

Classes	CLASSIFIED AS								
	None	A	B	C	D	L	V	W	Y
A	1	73	0	0	0	0	1	0	0
B	1	0	58	1	0	0	0	0	0
C	0	1	0	72	0	0	0	2	0
D	0	0	0	0	58	2	0	0	0
L	0	0	0	0	0	70	0	5	0
V	0	0	1	0	0	3	55	1	0
W	0	0	0	0	0	2	1	72	0
Y	4	1	1	0	0	0	0	1	53

Table 5.6. ASL Network 5 Training results

Classes	CLASSIFIED AS								
	None	A	B	C	D	L	V	W	Y
A	1	16	0	0	0	0	0	0	0
B	3	0	29	1	0	0	0	0	0
C	0	0	0	17	0	0	0	0	0
D	0	0	3	0	27	0	1	1	0
L	0	0	0	0	0	16	0	2	0
V	0	0	2	0	0	2	27	1	0
W	0	0	0	0	0	0	0	18	0
Y	2	0	1	0	0	1	0	0	29

Table 5.7. ASL Network 5 Test results

5.5.7. Discussion on ASL results

For one hand ASL signs, provided that the gestures are barely different (as in the case), invariant moments with Rprop neural network training gives good results. Recognition rates can be improved by using more training data.

5.6. TSL Results

TSL database is built in the same manner as ASL. But this time there are both two hand and one hand gestures to be considered. It is assumed that one hand gestures can be classified in an independent network or manner than two hand ones. Each hand can be tracked independently then if these hands regions collapse then the system can assume there is two hand input for the classifier.

There are 525 one hand signs (7 classes, 15 people, each person has 5 gestures in a class) and 1050 two hand signs (14 classes, 15 people, each person has 5 gestures in a class) in the database. 1575 Overall gestures in TSL database. %16 of the total number (87 for one hand and 175 for two-hand case) is used for test and rest (438 for one hand and 875 for two hands) is used for training. Selected gesture for one hand set were: C,I,L,O,P,U and V; and for two hand set: A, B, D, E, F, G, K, M, N, R, S, T, Y, Z.

For comparison with the ASL features, a 9 column input: compactness, eccentricity and invariant moment vector is also tested with two hand dataset. But it is found that this information is not enough when increasing the number of classes.

Other features considered are Fourier transform of signature information, and approximate finger position vector (which can be found from the signature plot).

5.6.1. TSL Network 1

9 column input vector with features ; compactness, eccentricity and 7 invariant moments are used to compare the dataset with ASL recognition. 2 hidden layers with 90 and 30 nodes was used. For the one hand gestures, in 1000 epochs MSE reached 0.003

(Figure 5.23). Although 96 per cent of the training set was correctly classified, only 71 per cent of the test set was classified correctly (Table 5.6). This is in fact due to the shape of the signs. Filled Images of L,U and V, C are in fact very similar.

	C	I	L	O	P	U	V
Train Correct	60	63	63	61	62	61	62
Train False	3	0	0	1	0	1	1
Test Correct	8	9	9	8	8	11	9
Test Fail	5	4	4	4	4	1	3

Table 5.8. TSL 9x90x30x5 Network Results, One Hand

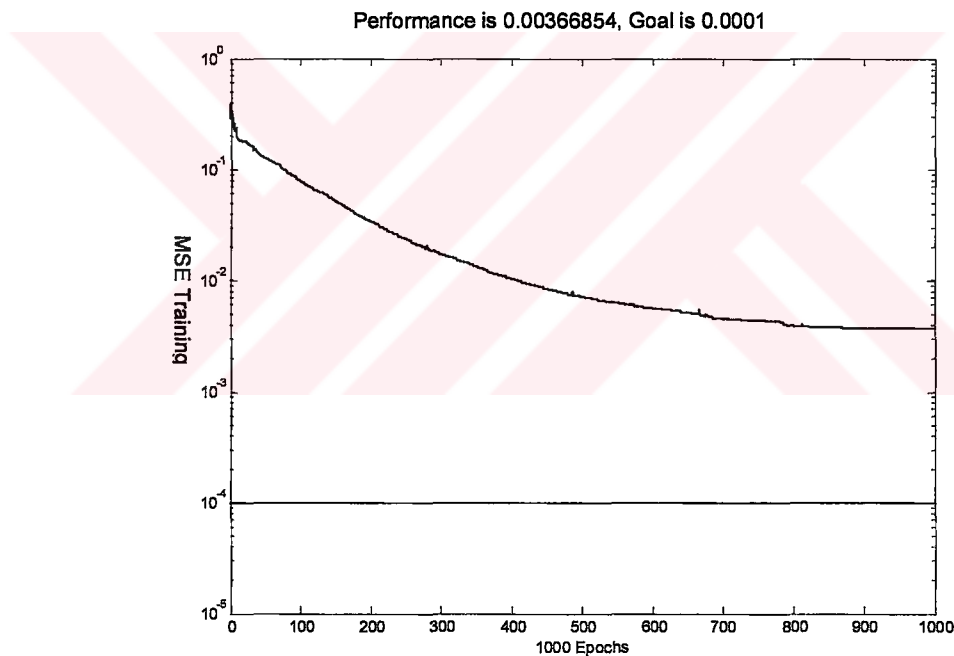


Figure 5.23. TSL 9x90x30x5 MSE Graph, One Hand

Using the same network on two hand signs gives near 0.06 MSE performance (Figure 24) and thus cannot correctly classify even the training set (Table 5.7).

	A	B	D	E	F	G	K	M	N	R	S	T	Y	Z
Train Correct	58	31	53	32	49	48	56	32	50	43	30	21	63	34
Train False	17	19	22	18	26	2	19	18	25	7	45	29	12	16
Test Correct	8	4	8	8	5	7	8	4	6	4	5	7	6	4
Test Fail	7	6	7	2	10	3	7	6	9	6	10	3	9	6

Table 5.9. TSL 9x90x30x5 Network Results, Two Hand

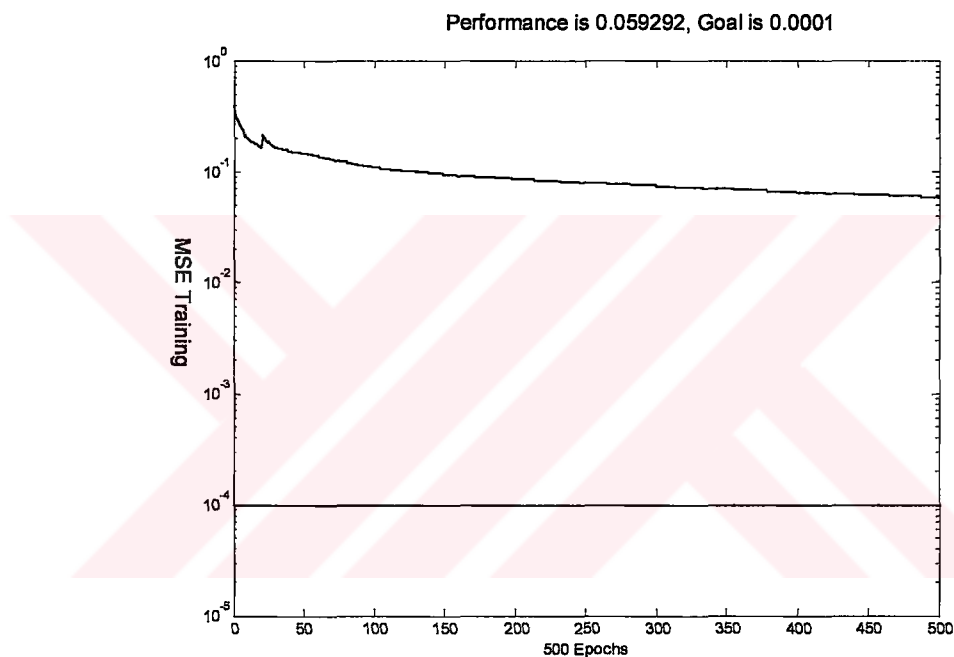


Figure 5.24 TSL 9x90x30x5 MSE Graph, Two Hands

5.6.2. TSL Network 2

Next network types used additional signature information for training. It is seen that in nearly 300 epochs all of these types achieve MSE less than 0.001, thus this gives training set recognition rate above 95 per cent.

In TSL Network 2, 50 column input vector with features; compactness, eccentricity, 7 invariant moments with first 41 values of the Fourier of the signature signal were used. 1 hidden layer with 30 nodes was used.

In the one hand dataset, 98 per cent of the training set was correctly classified, but test set was just on the 58 per cent recognition ratio (Table 5.8). Although near all the training data was correctly classified, also the two hand test classification was bad giving a 34 per cent overall ratio (Table 5.9).

	C	I	L	O	P	U	V
Train Correct	65	66	67	66	65	66	67
Train False	2	0	0	0	2	1	0
Test Correct	5	5	5	8	4	3	4
Test Fail	4	4	3	0	4	5	4

Table 5.10. TSL 50x30x5 Network Results, One Hand

	A	B	D	E	F	G	K	M	N	R	S	T	Y	Z
Train Correct	65	66	62	61	64	65	66	62	65	65	64	66	66	65
Train False	2	1	5	5	3	2	1	4	1	2	3	1	0	2
Test Correct	4	5	2	1	2	4	2	4	4	2	3	1	1	5
Test Fail	5	4	7	8	6	4	6	4	4	6	5	7	7	3

Table 5.11. TSL 50x30x5 Network Results, Two Hands

5.6.3. TSL Network 3

Increasing the hidden layers did not give any improvements. 2 hidden layers with 90 and 30 nodes was used, but test recognition overall ratio was 44 per cent for one hand signs and 21 per cent for two hand signs (Table 5.10 and Table 5.11).

	C	I	L	O	P	U	V
Train Correct	65	66	66	66	66	65	67
Train False	2	0	1	0	1	2	0
Test Correct	6	5	3	4	3	1	4
Test Fail	3	4	5	4	5	7	4

Table 5.12. TSL 50x90x30x5 Network Results, One Hand

	A	B	D	E	F	G	K	M	N	R	S	T	Y	Z
Train Correct	66	64	64	66	67	67	67	64	65	65	66	67	65	67
Train False	1	3	3	0	0	0	0	2	1	2	1	0	1	0
Test Correct	4	1	3	1	2	0	1	3	2	2	3	2	0	1
Test Fail	5	8	6	8	6	8	7	5	6	6	5	6	8	7

Table 5.13. TSL 50x90x30x5 Network Results, Two Hands

5.6.4. TSL Network 4

In this example full 360 column signature vector (360 degrees) was used as input. And test results was again disappointing for one hand set (Table 5.12).

	C	I	L	O	P	U	V
Train Correct	41	46	52	34	56	55	58
Train False	22	17	11	28	6	7	5
Test Correct	4	6	8	7	5	6	1
Test Fail	9	7	5	5	7	6	11

Table 5.14. TSL 360x400x5 Network Results, One Hand

5.6.5. TSL Network 5

In this test a 84 column vector with invariant moments, compactness, every 10th average of signature vector, with additional number of holes in the image features were used.

By using 2 hidden layers with 160 and 20 nodes; one hand set gave a test recognition ratio of 73 per cent (Table 5.13) , and two hand set gave 60 per cent ratio (Table 5.14). The signs that were mixed in two hand set was mostly B, D, F, S and Z, due to fact that B is like D and S is like Z.

	C	I	L	O	P	U	V
Train Correct	63	63	61	62	62	62	63
Train False	0	0	2	0	0	0	0
Test Correct	7	10	11	10	10	8	8
Test Fail	6	3	2	2	2	4	4

Table 5.15. TSL 84x160x20x5 Network Results, One Hand

	A	B	D	E	F	G	K	M	N	R	S	T	Y	Z
Train Correct	75	49	73	48	72	50	75	50	74	46	70	49	75	47
Train False	0	1	2	2	3	0	0	0	1	4	5	1	0	3
Test Correct	10	4	7	7	7	8	11	7	11	5	5	9	8	5
Test Fail	5	6	8	3	8	2	4	3	4	5	10	1	7	5

Table 5.16. TSL 84x160x20x5 Network Results, Two Hands

5.6.6. TSL Network 6

This example we used additional approximate finger information instead of signature. Also the invariant moments were used as input. With this modification and using 2 layer hidden networks one hand set gave 70 per cent test recognition (Table 5.15) and two hand set gave an improved 73 per cent recognition. Near 100 per cent of all training set was recognized. From Table 5.16 it seems that most mixed signs were E and R.

	C	I	L	O	P	U	V
Train Correct	61	63	63	62	62	61	63
Train False	2	0	0	0	0	1	0
Test Correct	4	11	8	10	9	11	8
Test Fail	9	2	5	2	3	1	4

Table 5.17. TSL 79x100x20x5 Network Results, One Hand

	A	B	D	E	F	G	K	M	N	R	S	T	Y	Z
Train Correct	75	50	75	50	75	50	75	50	75	49	75	50	74	49
Train False	0	0	0	0	0	0	0	0	0	1	0	0	1	1
Test Correct	10	7	11	6	11	7	10	10	12	6	12	7	12	7
Test Fail	5	3	4	4	4	3	5	0	3	4	3	3	3	3

Table 5.18. TSL 79x100x20x5 Network Results, Two Hands

5.6.7. TSL Network 7

Again using additional finger information and moments, this time node number of hidden layers were increased to 120 and 50 instead of 100 and 20. Slight improvement of 74 per cent was seen on two hand set (Table 5.18), but one hand set failed to 66 per cent (Table 5.17).

	C	I	L	O	P	U	V
Train Correct	63	63	63	62	62	62	63
Train False	0	0	0	0	0	0	0
Test Correct	6	9	12	11	8	8	4
Test Fail	7	4	1	1	4	4	8

Table 5.19. TSL 79x120x50x5 Network Results, One Hand

	A	B	D	E	F	G	K	M	N	R	S	T	Y	Z
Train Correct	73	50	75	48	75	50	75	50	75	49	72	49	75	50
Train False	2	0	0	2	0	0	0	0	0	1	3	1	0	0
Test Correct	12	7	13	8	10	8	10	9	11	6	10	9	10	7
Test Fail	3	3	2	2	5	2	5	1	4	4	5	1	5	3

Table 5.20. TSL 79x120x50x5 Network Results, Two Hands

5.6.8 TSL Network 8

In this test input was fed as every 5th average of 360 degree signature vector with additional invariant moment information, eccentricity and compactness. Input was an 80x1 vector. Three hidden layers with node sizes 100, 40 and 15 were used respectively. 80 per cent of TSL two-hand dataset were used as training and the rest 20 per cent were used as test. In this test results just 65 per cent of the test data were recognized correctly. Table 5.21 shows the letters being correctly and incorrectly detected for training and test data. Table 5.22 and Table 5.23 shows detailed recognition results for training and test data respectively. It is seen that in B, D, E and G letters misclassify with each other, and E, F, K, D letters has also a misclassification in themselves. Also letter Z is sometimes misclassified as T or Y.

	A	B	D	E	F	G	K	M	N	R	S	T	Y	Z
Train Correct	60	60	60	59	60	60	60	60	60	60	60	60	59	60
Train False	0	0	0	1	0	0	0	0	0	0	0	0	1	0
Test Correct	10	10	9	9	11	10	9	11	10	12	10	7	7	9
Test Fail	5	5	6	6	4	5	6	4	5	3	5	8	8	6

Table 5.21. TSL Two-hand Network 8, 80x100x40x15x5

Classes	CLASSIFIED AS														
	None	A	B	D	E	F	G	K	M	N	R	S	T	Y	Z
A	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	59	0	1	0	0	0	0	0	0	0	0
F	0	0	0	0	0	60	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	60	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	60	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	60	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	60	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	60	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0
Y	0	0	0	0	0	0	0	0	0	1	0	0	0	59	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	60

Table 5.22. TSL Network 8, Training Data Recognition Results

Classes	CLASSIFIED AS														
	None	A	B	D	E	F	G	K	M	N	R	S	T	Y	Z
A	0	10	0	0	2	2	0	0	0	0	0	1	0	0	0
B	1	0	10	1	1	0	2	0	0	0	0	0	0	0	0
D	1	0	1	9	1	0	2	0	0	0	0	1	0	0	0
E	0	0	0	1	9	0	3	0	0	0	0	0	2	0	0
F	0	0	0	1	0	11	0	3	0	0	0	0	0	0	0
G	0	0	1	0	2	0	10	0	0	0	2	0	0	0	0
K	0	0	0	2	1	2	1	9	0	0	0	0	0	0	0
M	0	0	0	0	0	1	0	0	11	1	0	0	2	0	0
N	0	1	0	0	0	1	0	0	2	10	0	0	0	1	0
R	1	0	0	0	0	0	1	0	0	0	12	1	0	0	0
S	2	1	0	0	0	0	0	0	0	1	1	10	0	0	0
T	1	0	0	0	1	1	0	0	2	0	1	0	7	0	2
Y	0	1	0	0	0	1	1	0	0	2	1	0	0	7	2
Z	0	0	0	0	0	0	1	1	0	1	1	0	2	0	9

Table 5.23. TSL Network 8, Test Data Recognition Results

5.6.9. TSL Results Discussion

From the overall results it seems that best input for a two hand set was approximate finger information which gave good results. One hand set was not correctly classified, this in fact due to fact that C, U, V and L signs are similar; which all have just 2 finger location information. Another reason is the semi-automatic training construction system which in fact was not manually cut and pasted, in contrary to most works in the literature.

It is well known that when the training set was manually created by cropping the hand regions manually, when the test inputs are from that set system will in fact have a recognition rate of 90 per cent and above.



5.7. Conclusion and Future Work

The system proposed here is a fast and easy way of recognizing geometrically separable hand gestures. The training gestures used in this system are semi-automatically constructed, instead of manually cropping hand area from each image (as the case in the literature). Because of this feature this system can grow itself easily for adapting class sizes or people.

In the ASL dataset with 8 classes more than 81 per cent correct recognition was achieved with only a 9x1 geometrical feature vector, this was due to the fact that 8 classes was geometrically separable; meaning that their shape were not similar.

In the TSL one hand dataset, it is seen that C, V, U and L signs all have nearly same geometrical features (only two finger positions used to identify), so system was not good at recognizing these shapes with just image moments. On the other hand a 74 per cent correct recognition rate of 14 different two hand gestures is good. Additional finger information vector seems to have good results.

The drop-down in recognition rate can be described by the training sets robustness. Since the training set is semi-automatically constructed in some cases not all the hand information can be obtained, this gets noisy pixels in the binary image. A solution to this problem can be training the color-probability tracker with a lot more than 20 the class images in the dataset.

Most of the work on the literature seems to work on one hand gestures, using either finger positions or geometric properties. Rarely researcher work on two-hand recognition systems, those who do that use dynamic models for two hand recognition. System proposed here uses the features used in static one-hand gesture recognition literature for two-hand gesture recognition. 74 per cent recognition rate is a nice introduction for two hand gesture recognition systems.

Off-line gesture recognition system proposed in thesis is a flexible system. As introduced by using CAMSHIFT color tracker, with some hand gesture flags, this system can easily work in an on-line fashion. This can provide us to navigate a robot, or navigate mouse pointer and use keyboard with visual hand gestures.

Future work on this gesture recognition system can be adding dynamic hand gesture recognition system, which can be classified as time –delayed neural networks or Hidden Markov Models.



APPENDIX A: DATABASE

In this thesis all the experiments are done using Matlab 7.0.1 computer program. Image capture is done with a simple USB camera. The gesture database was constructed and Matlab program was tested on a Pentium IV 3 GHz. Computer with 1 GB memory. Gesture database constructed can be found in the delivered via CD.



REFERENCES

- [1] C. Lee and Y. Xu, "Online, Interactive Learning of Gestures for Human/Robot Interfaces", *IEEE International Conference on Robotics and Automation*, vol. 4, pp 2982-2987, Minneapolis, 1996
- [2] G. D. Kessler, L.F. Hodges and N. Walker. "Evaluation of the CyberGlove(TM) as a Whole Hand Input Device", *ACM Transactions on Computer-Human Interaction*, vol. 2, No. 4, pp. 263-283, Dec. 1995
- [3] S. Akyol, U. Canzler, K. Bengler and W. Hahn,. "Gestensteuerung für Fahrzeugbordsystem", *Mustererkennung 2000, 22. DAGM-Symposium*, pp. 139-146, Kiel, September 2000
- [4] W. Juan, *Hand Gesture Telerobotic System using Fuzzy Clustering Algorithms*, MSc. Thesis, Ben-Gurion University of the Negev, Israel , 2001
- [5] S. Marcel S., "Hand posture recognition in a body-face centered space", *Proceedings of the Conference on Human Factors in Computer Systems (CHI)*, 1999
- [6] S. Marcel, "Hand Gesture Recognition Using Input-Output Hidden Markov Models", *Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 456, Grenoble, France, 2000
- [7] W.T. Freeman and M. Roth, "Orientation Histograms for Hand Gesture Recognition", *IEEE International Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland, 1995.
- [8] M. Kohler, "Vision Based Hand Gesture Recognition Systems" [webpage]. Available at HTTP: <http://ls7-www.cs.uni-dortmund.de/research/gesture/vbgr-table.html>
- [9] R.C. Gonzales and R. E. Woods, *Digital Image Processing*, 2nd Ed., Prentice-Hall, 2002.

- [10] Matlab 7.0.1 Image Processing Toolbox, Users Guide, Version 5, The MathWorks Inc., Natick, MA, 2004.
- [11] W. Burger, "Connected Component Labeling Algorithms", [webpage]. Available at HTTP: <http://webster.fh-hagenberg.at/staff/burger/lva/dim/uebungen/region-labeling.pdf>
- [12] R.C. Gonzales, R. E. Woods and S.L. Eddins, *Digital Image Processing using Matlab*, Prentice-Hall, 2004.
- [13] M.J. Jones and J.M. Rehg, "Statistical color models with application to skin detection", *Proc. Of the CVPR '99*, vol.1 , pp. 274-280, 1999.
- [14] D. Comaniciu and P. Meer, "Robust analysis of feature spaces: Color image segmentation", *International Conference on Computer Vision and Pattern Recognition*, pp. 750-755, San Juan, Puerto Rico, 1997
- [15] "Open Source Computer Vision Library", [webpage] (2005, May) Available at HTTP: <http://www.intel.com/research/mrl/research/opencv>
- [16] W.T. Freeman, K. Tanaka, J. Ohta, and K. Kyuma, "Computer Vision for Computer Games", *Int. Conf. On Automatic Face and Gesture Recognition*, pp.100-105, 1996.
- [17] M.K.Hu, "Visual pattern recognition by moment invariants", *IRE Trans. Inform. Theory*, vol. IT-8, pp. 179-187, Feb. 1962.
- [18] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd Ed., Prentice-Hall, 1999.
- [19] W.S. McCulloch and W.H. Pitts, "A logical calculus of the ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, pp. 115-133, 1943

- [20] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation", in *Parallel Distributed Processing*, ch.8, MIT Press, Cambridge, MA, 1986
- [21] Matlab 7.0.1 Neural Networks Toolbox, Users Guide, Version 4.0.1, The MathWorks Inc., Natick, MA, 2004.
- [22] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm", *Proceedings International Conference on Neural Networks*, pp. 586-591. San Francisco, 1993.
- [23] "Turkish Sign Language", [webpage] (2005, May) Available at HTTP: <http://turkisaret dili.ku.edu.tr>