ABSTRACT

CLASSIFICATION OF HAND SHAPES AND LOCATIONS IN CONTINUOUS SIGNING

by
Swetha Karri

Sign language for the deaf and hearing impaired replaces speech with manually produced signs. Each sign has been categorized as being combinations of handshape, movement, orientation, location, and facial expressions. Of the five sign parameters, this thesis focuses on classification of two of the main parameters, the hand shapes and locations, in continuous signing.

Since the nature of hand shapes is transient and not static, neural networks was used as a classifier for hand shapes. And since locations in sign language are defined by linguistic variables rather than by hard core position values, fuzzy logic was used as a classifier for locations. Two models have been developed using neural networks and fuzzy logic toolboxes in Matlab that showed the possibility of classification of hand shapes and locations, in continuous signing.

Results show that neural network was able to classify hand shapes accurately at every instant when tested with the trained data and reasonably well with testing data. The proposed model for classification of locations was able to classify locations accurately.

CLASSIFICATION OF HAND SHAPES AND LOCATIONS IN CONTINUOUS SIGNING

by
Swetha Karri

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Biomedical Engineering

Department of Biomedical Engineering

August 2008

APPROVAL PAGE

CLASSIFICATION OF HAND SHAPES AND LOCATIONS IN CONTINUOUS SIGNING

Swetha Karri

8/27/08

Dr. Richard A. Foulds, Thesis Advisor — Date
Associate Professor of Biomedical Engineering, NJIT

8/27/08

Dr. Sergei Adamovich, Committee Member — Date
Assistant Professor of Biomedical Engineering, NJIT

8/27/08

Dr. Max Roman, Committee Member — Date
Assistant Research Professor of Biomedical Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Swetha Karri

Degree: Master of Science

Date: August 2008

Undergraduate and Graduate Education:

- Master of Science in Biomedical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2008

- Bachelor of Science in Biomedical Engineering,
  Padmasri Dr. B. V. Raju Institute of Technology, Hyderabad, India, 2006

Major: Biomedical Engineering

Dedicated to the entire Karri and Cheeti family,

who believe education is invaluable.

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Richard Foulds, my thesis advisor for giving me an opportunity to work on sign language and for his constant support, encouragement and for all the trust he had on me in completing this work. Special thanks are given to Dr. Sergei Adamovich and Dr. Max Roman for their guidance, support and for actively participating in my committee.

I am grateful to the Pfeiffer Research Foundation for supporting this study, which was conducted at the Neuromuscular Engineering Laboratory, NJIT.

I would like to extend my gratitude to my peers in the Neuromuscular Engineering Laboratory (Amanda, Kate, Sally, Diego, Qinyin, Darnell, Kai, Rico, Chaitali, David and Kunal) whose friendship made this experience immense.

My sincerest appreciation to all my teachers and friends throughout my years at St. Alphonsas High School, BVRIT College, and NJIT for providing me with a strong background and a wholesome educational experience.

I wish to express my deepest gratitude to my family for their never-ending support. To my parents, Rama Prasad Karri and Padmaja, for all the sacrifices they have made and for their encouragement. To my brother, Madhusudhan and sister, Jahnavi.

-All glory to GOD, who makes all things possible-

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

1.1 Background

1.1.1 Sign Language

Sign language is one of the several natural languages, which uses manually produced signs by combining hand shape, movement, orientation, location and facial expressions to convey meaning rather than by speech. It is mainly developed among the deaf community, which includes interpreters, friends and families of the deaf people in addition to the people who are deaf or hard of hearing themselves.

Sign language has its own phonetics (the distribution and patterning of signs), semantics (relationship between signs and what they denote), syntax (patterns of formation of sentences and phrases from signs), pragmatics (connection between sentence context and interpretation) and morphology (sign structure). It is iconic in nature, meaning it represents a whole idea at a time rather than by defining it through a sequence of words as in oral communication, where only one sound can be made or received at a time. (Swisher, 1988).

According to National center for health statistics estimate, 28 million of Americans, almost 10 percent of the population have a hearing loss of some degree. Of these 20 million people, about 2 million are classified as Deaf, who cannot hear speech even with the use of a hearing aid. And of these 2 million, about 10 percent were born without the ability to hear; the other 90 percent lost their hearing later in life. Therefore in response to the practical need for a communication system specialized for the American and Canadian deaf community, a signed language emerged with grammatical structure

dissimilar to that of spoken language. American Sign Language (ASL) is that native signed language. It is a natural and autonomous language, used by more than half a million people in the US. It is object-oriented and therefore differs from both written and spoken English.

1.1.2 Sign Parameters

In biomechanical terminology, a sign is a unique combination of hand shape, location, orientation, movement, and facial expression (*Stokoe*, 1976). These five *Stokoe*'s parameters have been incorporated into various sign languages, including American Sign Language, and they form the categorical foundation of signs. They are defined as follows:

1. **Handshape**- formation of the joints in the hand, excluding wrist angles
2. **Location**- position of the hands in signing space
3. **Orientation**- direction the palm is facing
4. **Movement**- change in handshape or location during a sign (including linear, circular, arced and repeated trajectories)
5. **Facial expression**- any expression of the face used to describe or reinforce concept being conveyed

Figures 1.1, 1.2, and 1.3 demonstrates the way individual signs have been categorized with these five sign parameters and the way each sign is differed either by location or hand shape or movement or by facial expression.

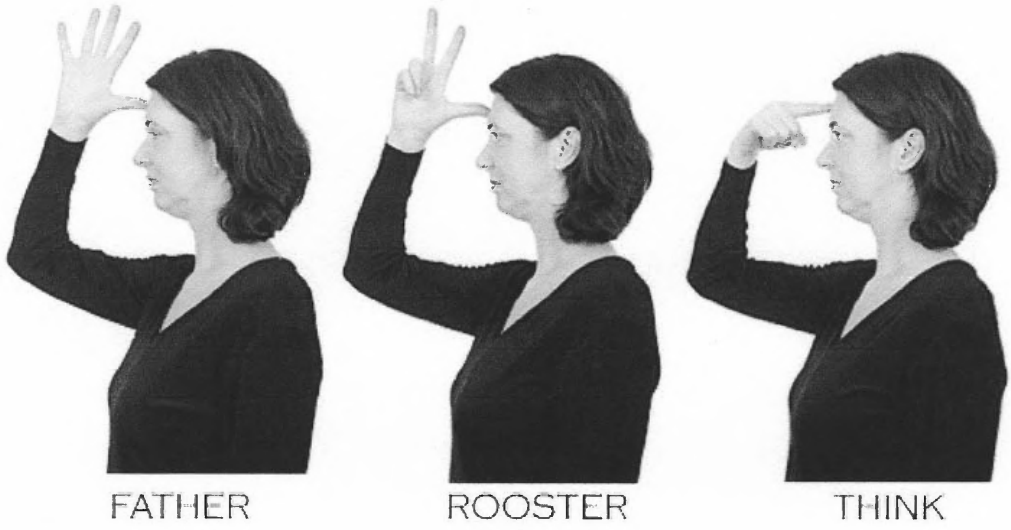


Figure 1.1 ASL signs for 'father', 'rooster', and 'think'.
(Source: www.talkinghandsbook.com)

The individual signs for 'father', 'rooster', and 'think' (Figure 1.1) have same locations but differ in hand shapes.

Figure 1.2 ASL signs for 'summer', 'ugly', and 'dry'.
(Source: www.talkinghandsbook.com)

The individual signs for 'summer', 'ugly', and 'dry' have same hand shape and movement, but differ in locations.

FATHER IS HOME.

IS FATHER HOME?

Figure 1.3 ASL sign for 'father home' and ' Is father home?'
(Source: www.talkinghandsbook.com)

Difference in facial expressions changes the meaning from 'father is home' to a question 'Is father home?' in sign language (Figure 1.3).

1.2 Objective

So far progress in classification of hand shapes has been limited to static hand shapes and locations to only two dimensions. Since the nature of hand shapes is transient (transition from one sign to another (see figure 1.4)) and not static (as represented by figures), we need a classifier like neural networks that can be applied on continuously moving data. Since locations in sign language are defined by linguistic variables like 'in front of the chest', 'near the shoulder', 'at the chin', 'below the ear', etc rather than by hard core position values like X, Y and Z, we need a system like fuzzy logic that can identify these fuzzy variables. The main objective of this thesis was to classify the two key sign parameters, hand shapes and locations, in continuous signing.

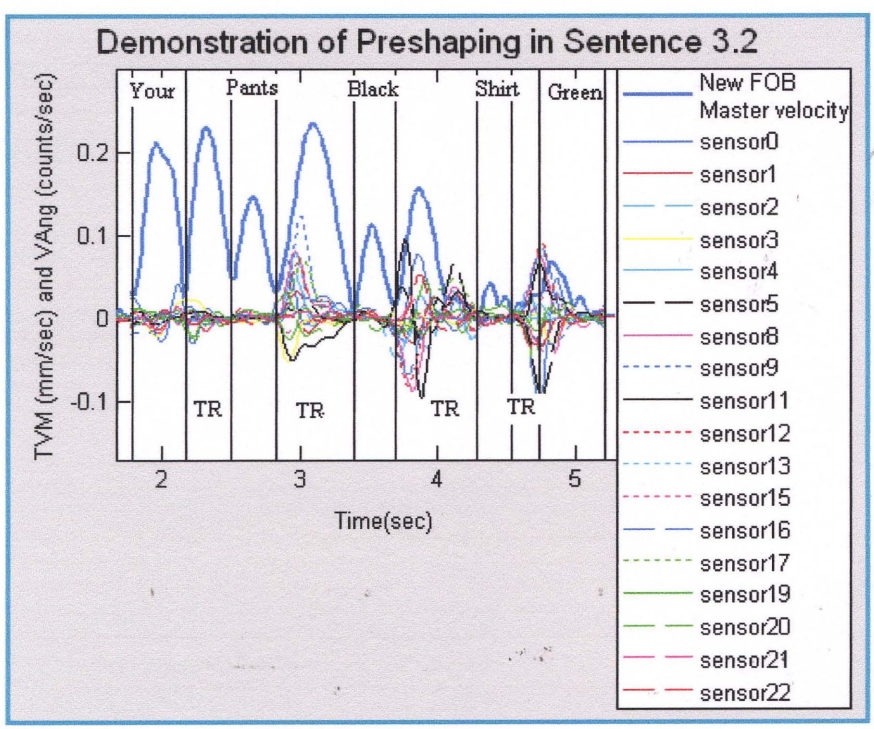


Figure 1.4 Figure demonstrating preshaping of the fingers (between PANTS and BLACK and between BLACK and SHIRT). (TR-Transition)
(Source: Chemuttaai Koech, 2007)

CHAPTER 2

METHODOLOGY

2.1 Overview

The primary goals of this research were 1) to classify hand shapes at every instant using neural networks and 2) to identify locations using fuzzy logic, in continuous signing.

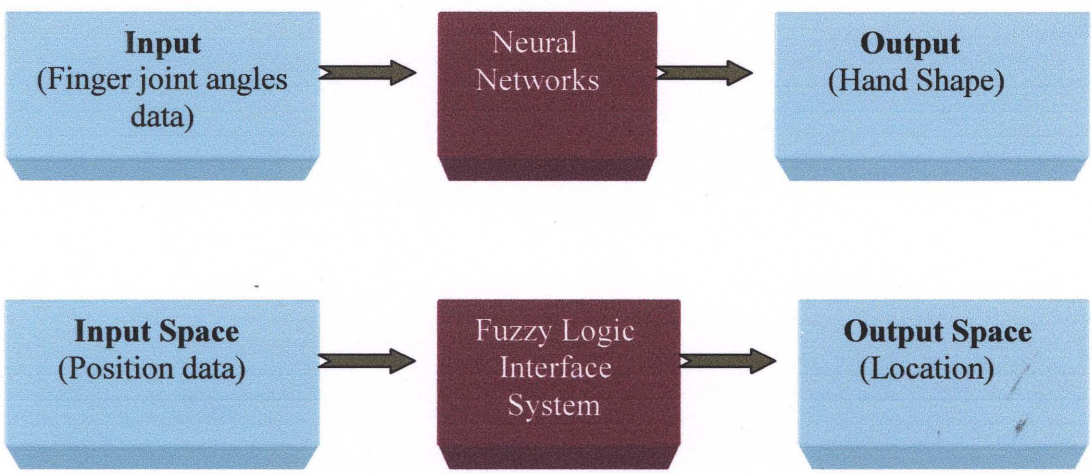


Figure 2.1 Thesis overview.

Hand shape in sign language is defined as the formation of the joints in the hand excluding wrist angles. There are about 42 basic hand shapes in sign language of which this thesis focuses on classification of 28 hand shapes.

For hand shape classification, 15 finger joint angle data from the right hand (signing hand) were collected using Cyber Glove® device for a sequence of 28 different hand shapes (alphabets and first ten digits) in sign language. Figure 2.2 shows the hand shapes for these alphabets and numbers.

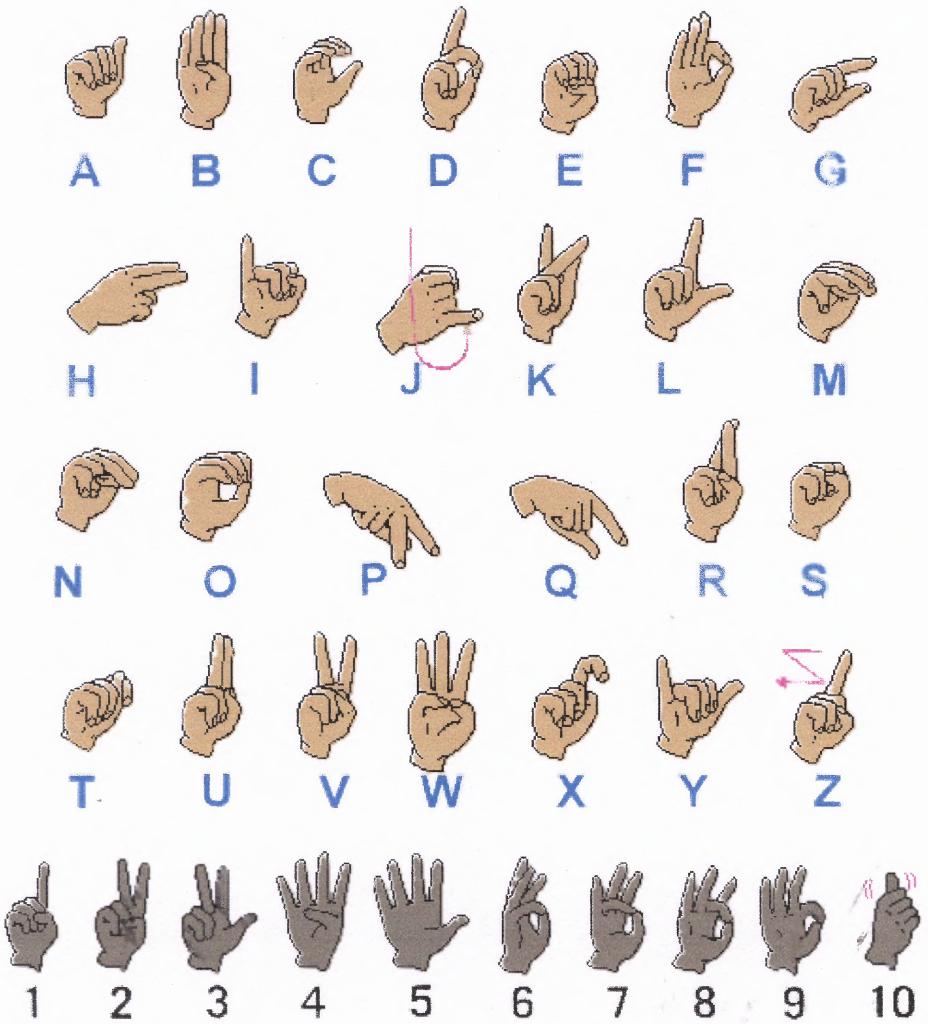
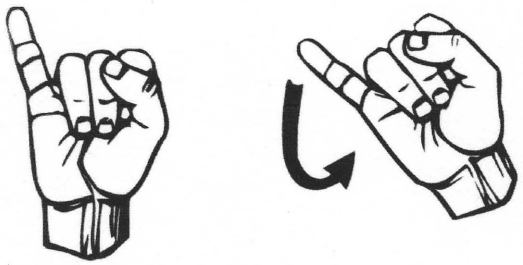


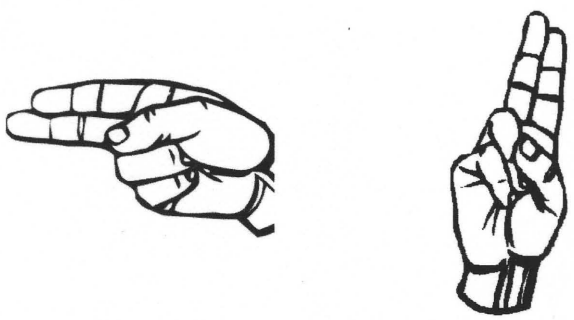
Figure 2.2 Alphabets and first ten digits in sign language.
(Source: http://www.iidc.indiana.edu/)

Hand shapes for 'h', 'j', 'p', 'q', 'z', '2', '6', '9' are eliminated for training as they possess same hand configurations as that of 'u', 'i', 'k', '1', 'v', 'w', 'f', respectively. They either differ in movement or orientation. For example, 'i' and 'j' has the same hand configuration, but differ in movement. Similarly 'h' and 'u' has the same hand shape, but differ in orientation (see figures 2.3 and 2.4).



I J

Figure 2.3 Figure showing alphabets i and j that have same hand configurations but differ in movement.
(Source: www.enchantedlearning.com)



H U

Figure 2.4 Figure showing alphabets h and u that have same hand configurations but differ in orientation.
(Source: www.enchantedlearning.com)

Joint angle data were collected from the author of this thesis. Altogether 5 trials have been made, each trial time elapse being 40 seconds at a frequency of 100 samples/ second. The collected data consisted of a transition phase indicating change in hand shape and static phase indicating a target hand shape. The finger joint angular velocity plot in the figure 3.7 demonstrates the transition phase and static phase of these hand shapes. The first trial samples (about 4000 samples) were utilized for training the neural network.

The remaining trial data samples were tested on the trained network and it was observed that the network was able to classify about 50% of the 28 target hand shapes in all the cases.

Location in sign language is defined as position where the hand is reached within the signing space with respect to the body. A signing space is the space in front of the body that extends from the top of the head down to the waist and from shoulder to shoulder. Although there are about 20 locations, this thesis demonstrates a model that can classify seven of the main locations, forehead, nose, chin, left shoulder, right shoulder, chest and hip (see figure 2.5).

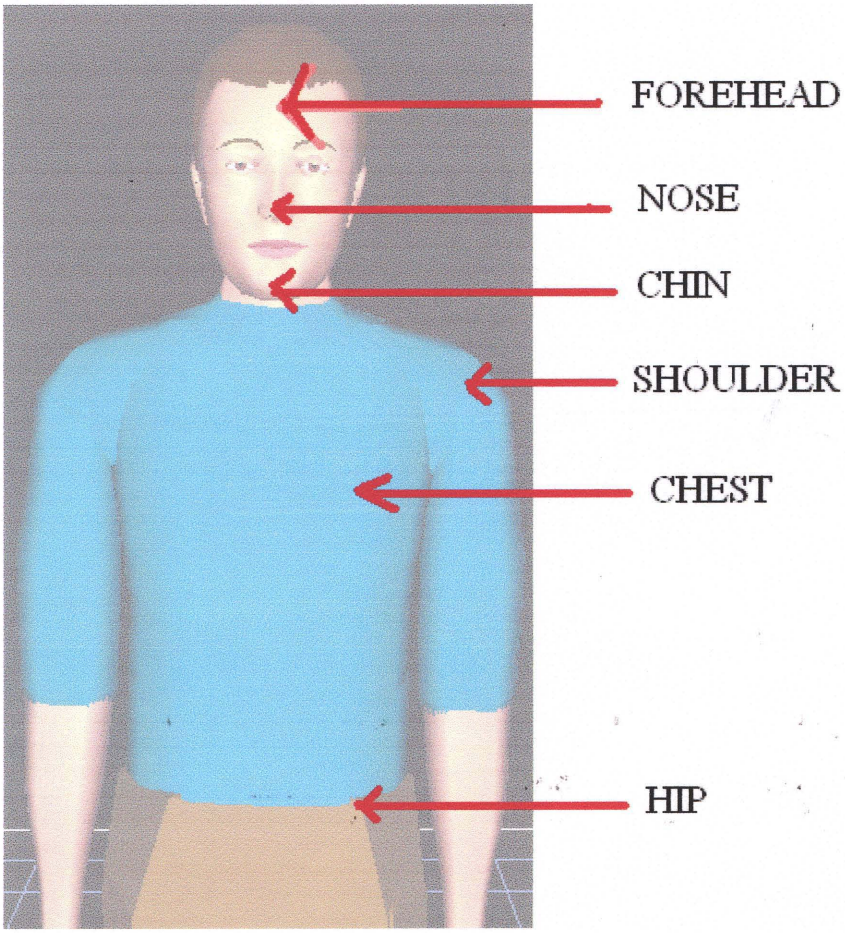


Figure 2.5 Locations in sign language.

It was assumed that the subject is fixed and the position data of these seven locations were collected using Nest of Birds® equipment with the sensor placed directly on the body. The range of motion of the signing hand in all the three dimensions, X, Z and Y, of the signing area are marked, as these parameters were later required to define the membership functions for the input and output variables while developing a Fuzzy Logic Interface System (FIS). The developed FIS was able to identify these seven locations when the position values of X, Z and Y of the hand is inputted. It also showed the possibility of including the remaining locations as well. The structure of FIS is shown in the figure 4.2.

Two models were developed using Neural Network and Fuzzy Logic toolboxes in Matlab for classification of hand shapes and locations, respectively. Figures 2.6 and 2.7 show the step by step procedure for classification of hand shapes and locations, respectively.

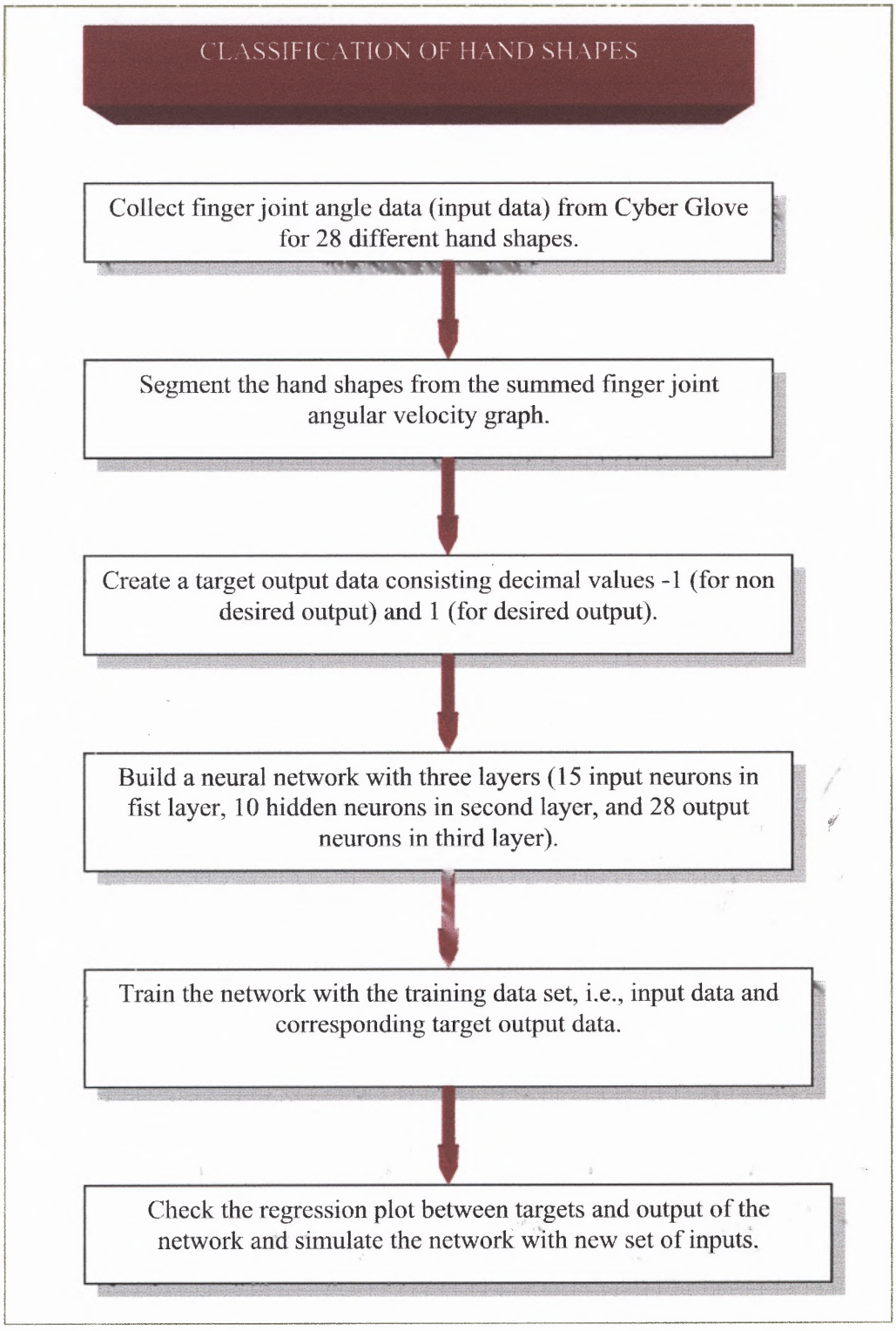


Figure 2.6 Step by step procedure for classification of hand shapes in continuous signing.

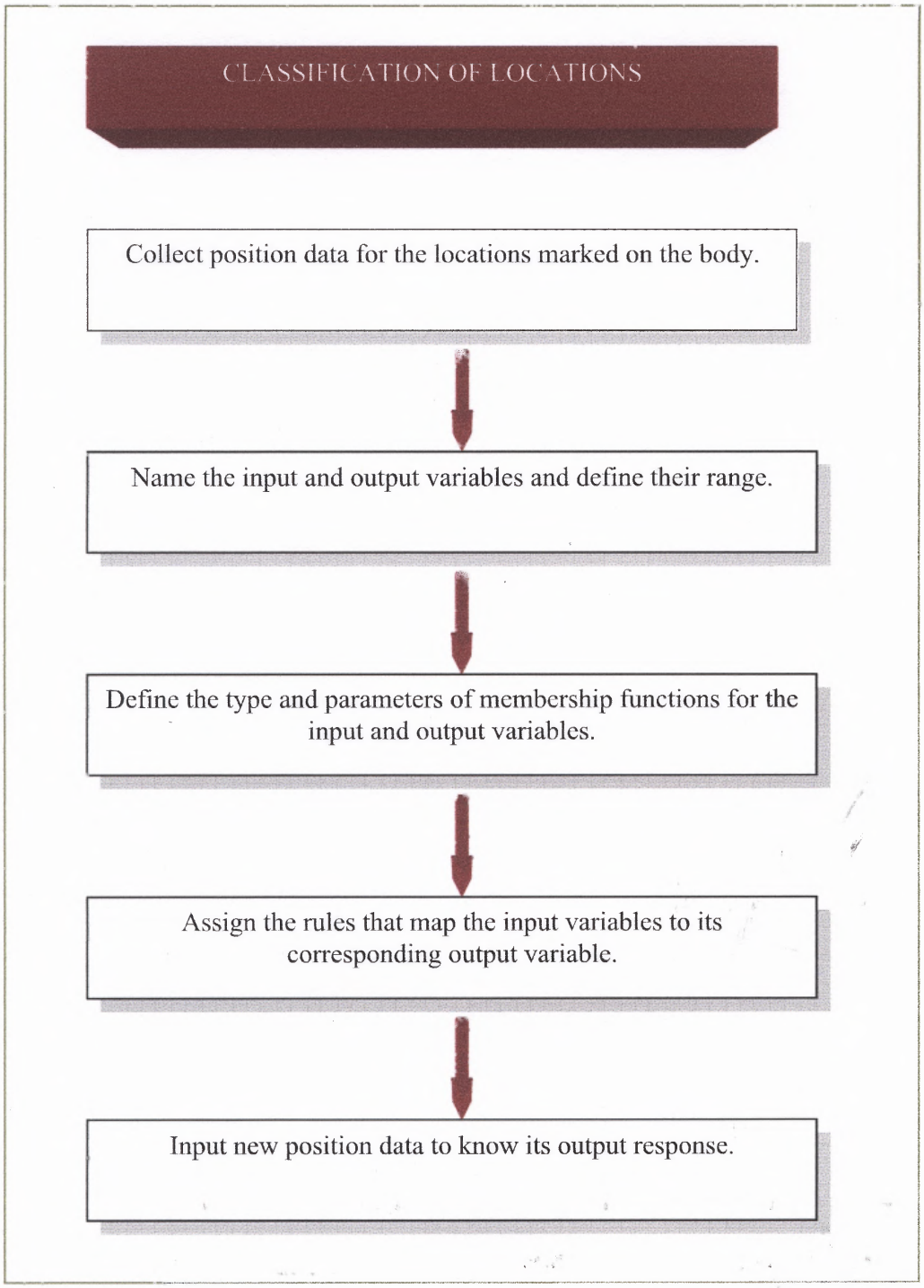


Figure 2.7 Step by step procedure for classification of locations in continuous signing.

2.2 Data Collection

2.2.1 Cyber Glove®

The two instruments that were used to collect data for classification of sign parameters were Cyber Glove® (CG) and Nest of Birds® (NOB).

Figure 2.8 Cyber Glove® system for recording finger joint angle data.
(Source: http"//infolab.usc.edu)

The Cyber Glove® (CG) system by Immersion Corporation (formerly Virtual Technology Inc.) detects the joint angles of the fingers through the piezoelectric bend and abduction sensors embedded inside the glove. Of the 18 flexible sensors, 11 bend sensors measure the metacarpophalangeal (MP) and interphalengeal (IP) joints, 4 abduction sensors measure the abduction of the fingers, and the remaining 3 sensors measure the wrist pitch, wrist yaw and arch of the hand. Although 18 joint angle data is available,

only 15 joint angles data were considered for hand shape classification. The description of each of these sensors is given in Ta le 2.1.

The joint angle data collected from the Cyber Gloves® comprised of digitized output ranging from 0 to 255 (counts), which serve as an alternative unit for angular units (radians or degrees). Since the conversion of counts into degrees involves approximation of joint angle measurement in relation to the initial conditions set during hand calibration using DCU software, counts were used as an alternative to degree or radians to avoid the approximation.

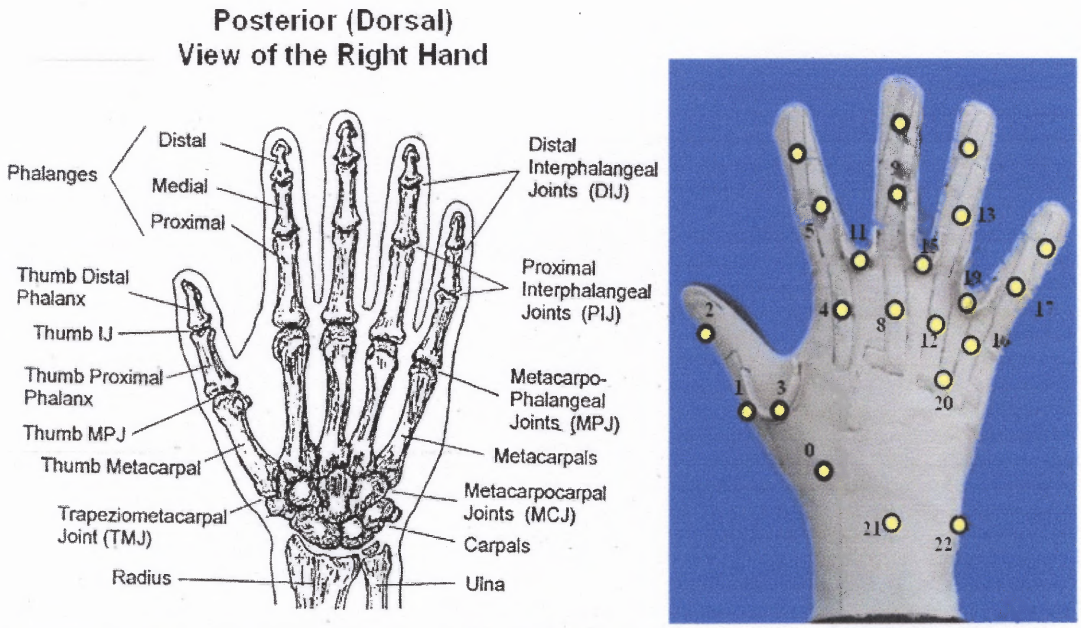


Figure 2.9 Posterior view of the hand (Left) displaying finger joints and Cyber Glove® (Right) displaying the sensor positions.
(Sources: Kapit W. & Elson L. M, 1993 and Virtual Technologies, 1998)

Table 2.1 CyberGlove® 15 Sensor Descriptions

CG Sensor	Finger and Joint Description
0	THUMB- Rotation
1	THUMB- Metacarpo- Phalangeal Joint
2	THUMB- Interphalangeal Joint
3	THUMB- Abduction
4	INDEX- Metacarpo- Phalangeal Joint
5	INDEX- Proximal Interphalangeal Joint
8	MIDDLE- Metacarpo- Phalangeal Joint
9	MIDDLE – Proximal Interphalangeal Joint
11	MIDDLE-INDEX- Abduction
12	RING- Metacarpo- Phalangeal Joint
13	RING- Proximal Interphalangeal Joint
15	RING-MIDDLE- Abduction
16	PINKIE- Metacarpo- Phalangeal Joint
17	PINKIE- Proximal Interphalangeal Joint
19	PINKIE-RING- Abduction

2.2.2 Nest of Birds®

Nest of Birds® (NOB) system by Ascension Technology is a six degrees-of-freedom measuring device that can be configured to simultaneously track position and orientation of up to four sensors by a transmitter, with each sensor capable of making from 20 to 120 measurements per second. The transmitter generates a pulsed DC magnetic field that is simultaneously measured by all the sensors by which the NOB determines position and orientation. The interface between the computer and NOB to command and receive data from this system is a full duplex RS-232C. This system is used in various applications like head/ hand/ body tracking for virtual reality, simulation and training, scientific visualization, entertainment, telerobotics/ telepresence, CAVE environments, virtual prototyping, biomechanical/ human factors analysis, rehabilitative feedback assessment, 3D graphics control and manipulation. The sensor was directly placed on the seven

locations marked on the body to record position data, which were used later to define the parameters of a membership function for the variables before building a fuzzy Logic Interface System. NOB with its transmitter and four sensors is shown in Figure 2.3 below.

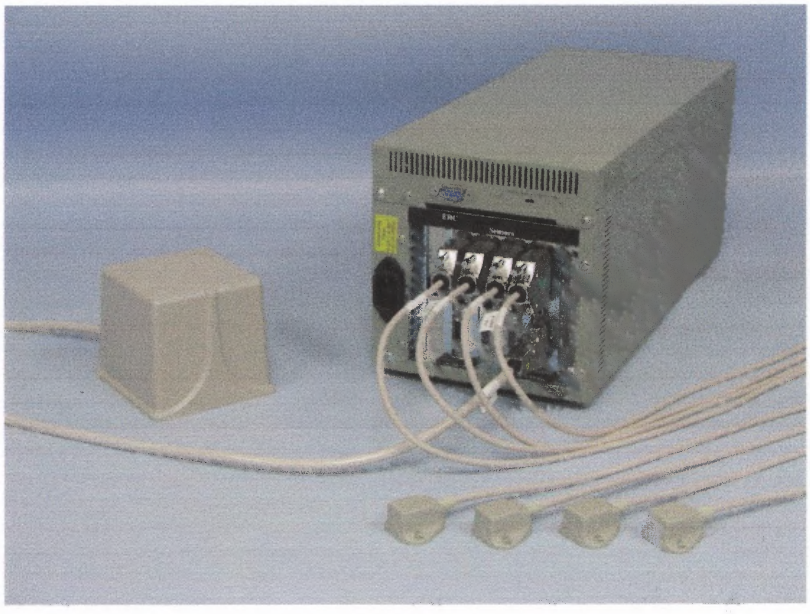


Figure 2.10 Nest of Birds® system for recording position data of index finger tip.
(Source: Nest of Birds manual, Virtual Technologies)

2.3 Calibration

Prior to the collection of data, the Glove has been calibrated to the user's hand shape. Calibration is necessary due to the variations in hand shapes and range of motion of the fingers. Glove calibration can be done using Immersion Technology's *Device Configuration Utility* (DCU) software. Figure 2.11, demonstrates the two-stage

calibration procedure, in which the hand was held still in two positions, and the DCU created and saved a unique calibration configuration file based on specific hand structure. Calibration of the measurement devices was necessary prior to the collection of data due to the range of variability in hand sizes and range of motion of the fingers.

The CyberGloves® were calibrated using Immersion Technology's *Device Configuration Utility* (DCU) software. Figure 2.11, demonstrates the two-stage calibration procedure, in which the hand was held still in two positions, and the DCU created and saved a unique calibration configuration file based on specific hand structure.

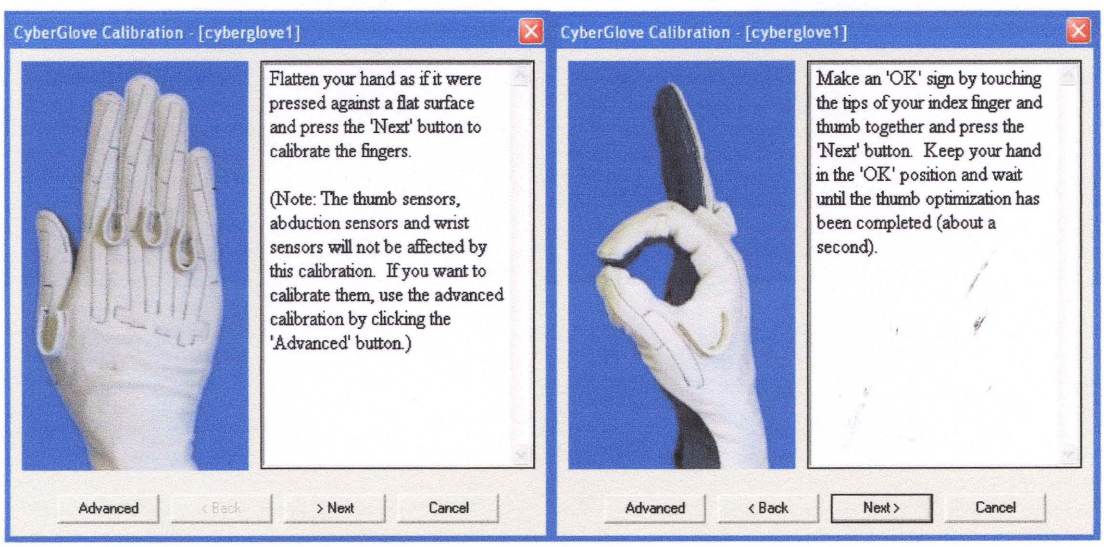


Figure 2.11 Calibration procedures for the CyberGloves®.
(Source: Virtual Technologies, 1998)

The following equation converted raw analog to digital sensor value into appropriate joint angles, where A/D is the analog to digital CG sensor value:

$$Angle = Gain * (A/D - Offset)$$

Gain affected the range of motion of the joint angles and the offset refers to the difference between the analog to digital values and the default hand-geometry position. An advanced calibration allowed the user to manipulate the gain and offset parameters for individual sensors.

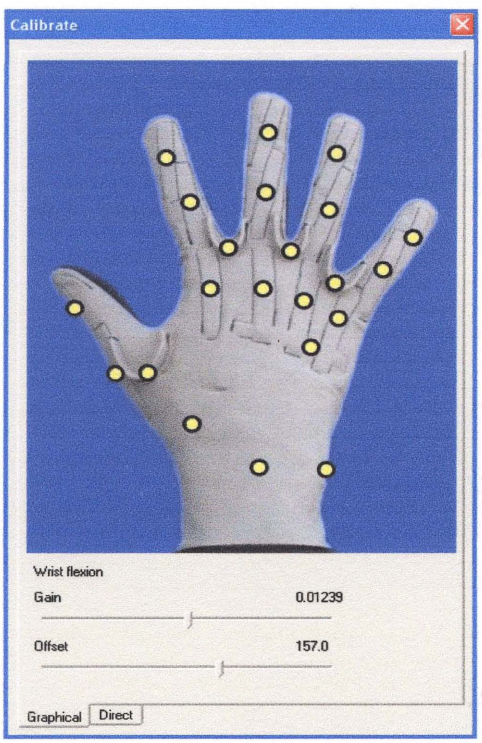


Figure 2.12 Advanced calibration of gain and offset.
(Source: Virtual Technologies, 1998)

Accuracy of the Nest of Birds system was checked by running the *WinBird* software from Ascension Technology and comparing the position and angle values against data collected from our Matlab programs.

2.4 Data Processing

Prior to the analysis of the raw data, some data pre-processing was required. A smoothing operation was required to increase the signal to noise ratio. Data from human movement experiments could contain additive noise from many sources, such as vibration, high frequencies, environmental interferences (60 Hz electrical lines), metal interference (NOB electromagnetic sensors) and error (human and measurement). Therefore, it is essential for the raw data to be smoothened and yet retain the useful information.

2.4.1 Filtering the Data

In biomechanical experiments, the most important frequencies are much lower than the sampling rate (f_s). Normal human movements have maximum frequencies in the range of 5 to 20 Hz. The raw, unprocessed sign language data were filtered with a 5^{th} order Butterworth filter, at a cut-off frequency (f_c) of 6 Hz. Winter (2005) suggests a low pass filter with f_c of 6 Hz for biomechanical data to attenuate all unwanted high frequencies. The 6 Hz f_c also followed Nyquist's Sampling Theorem as it is less than half of the sampling frequency. Signals above 12 Hz would be unreliable and would likely be the result of aliasing (distortion due to high frequency signals). A 5^{th} order, zero-lag filter smoothed the data and canceled the phase lead and lag that can occur in digital filtering.

Butterworth filters are digital, recursive filters that provide a superior linear representation of amplitudes of low frequencies. Matlab's function ***butter*** was used to compute the Butterworth coefficients; the filter order and the variable proportional to $2f_c/f_s$.

2.4.2 Velocity Derivation

Velocity was computed by using a central difference algorithm, which is a numerical differentiation of displacement (angular and X, Y and Z position) over time. Assuming the data are spaced equally and *i* represent the sample, the central difference equation is:

$$Central\ Difference = (Position_{(i+1)} - Position_{(i-1)}) / (Time_{(i+1)} - Time_{(i-1)})$$

This calculation of velocity represents the velocity at a point in time midway between two adjacent samples.

The X, Y and Z directional velocities of the NOB sensor were combined into one resultant known as the tangential velocity ($velocity_{tan}$). The tangential velocity combines three dimensions into one with direction tangent to the path of the sensor. It is computed by taking the square root of the sum of the squares:

$$velocity_{tan} = \sqrt{((velocity_X)^2 + (velocity_Y)^2 + (velocity_Z)^2)}$$

The units of the $velocity_{tan}$ of the NOB sensors were in millimeters per second and the angular velocities of the finger joints were in counts per second.

CHAPTER 3

CLASSIFICATION OF HAND SHAPES

3.1 Neural Networks

Neural networks simulate network of biological neurons in solving large computational problems in various fields like pattern recognition, identification, classification, speech, vision, control systems, etc. They attempt to exploit the way the information is processed by the human brain to deal with the kinds of problems that require a generalized solution, and try to adapt themselves even to a noisy environment. A network can be trained with certain inputs and target outputs to perform a specific task. Typically many such input-target pairs are required in supervised learning for the network to perform well. The function of an artificial neuron with respect to a biological neuron is explained in sections 3.1.1 and 3.1.2.

3.1.1 Function of a Biological Neuron

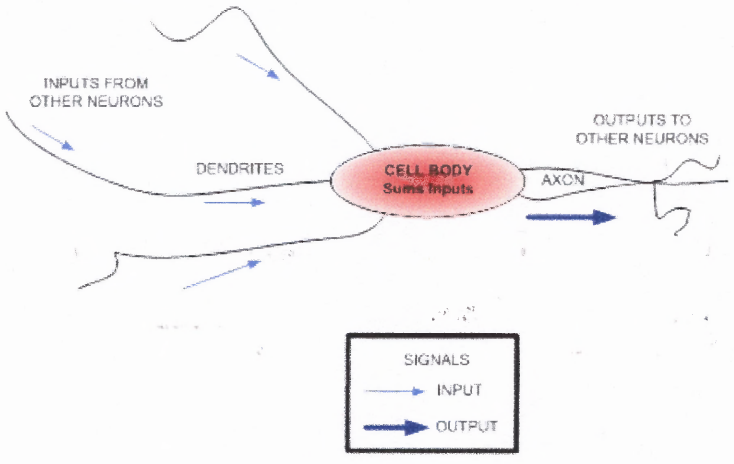


Figure 3.1 A simple biological neuron.
(Source: www.usingneuralnetworks.com)

A biological neuron, which is the most basic functional units of a nervous system exchanges information with other neurons in a parallel fashion. It is composed of mainly four parts: 1) Dendrites- that receive input from adjacent neurons. 2) Cell body- where all the inputs are summed up and are processed to axons. 3) Axons- turn the processed inputs into outputs, and 4) Synapses- the electro chemical contact between the adjacent neurons, where the outputs are transmitted from one neuron to another. Figure 3.1 demonstrates the basic mechanism of a biological neuron. The arrow marks indicate the flow of signal.

3.1.2 Function of an Artificial Neuron

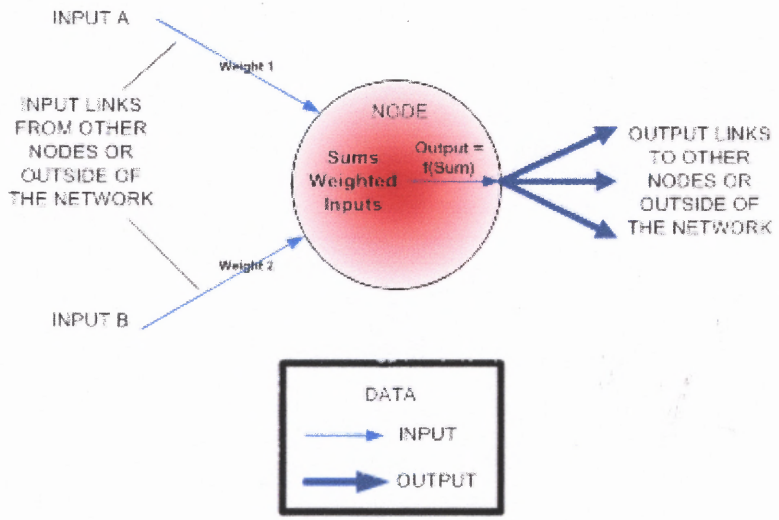


Figure 3.2 A simple artificial neuron.
(Source: www.usingneuralnetworks.com)

Just as a biological neuron receives input signals through dendrites and sends output signals through axons, the artificial neurons receives summed weighted input signals from the prior neuron and sends output signals to the successive neurons. The synapses of the neuron are modeled as weights, which determine the strength of the connections

between them. Negative weight values imply inhibitory connections and positive values imply excitatory connections. An artificial neuron may or may not have a bias input, which possess a constant value, usually one. Figure 3.2 demonstrates the basic mechanism of an artificial neuron. A node is a point where the received weighted inputs are summed.

Artificial neural networks are composed of group of such neurons (described in figure 3.2) arranged in layers and the connections (called weights) between them largely determine the network function. The order of the neurons is irrelevant as they operate in parallel. Figure 3.3 demonstrates the basic learning mechanism of a NN, in which output from the network are compared against target/desired outputs and the error is fed back to the network, where the weights are adjusted accordingly until the error is minimized, or in other words, until output of the system matches target outputs.

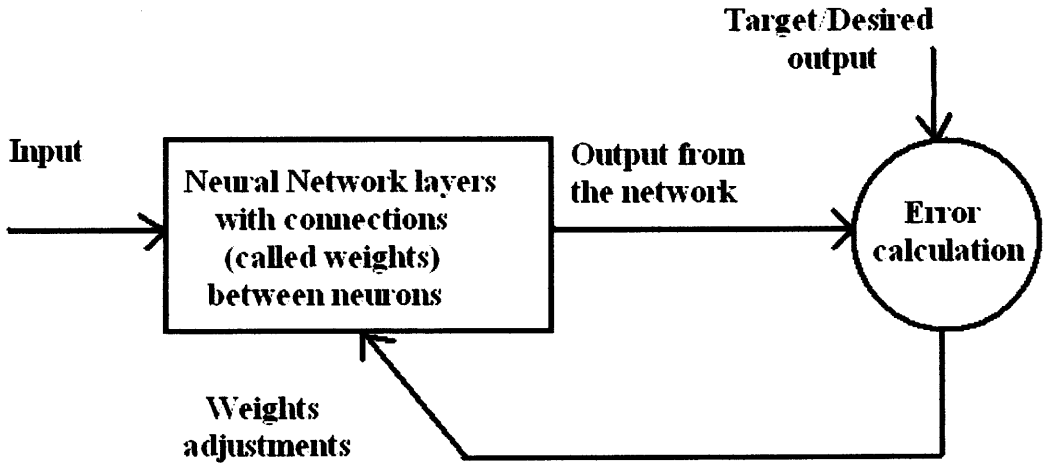


Figure 3.3 Basic mechanism of a neural network.
(Source: Neural network user guide, Matlab Technologies)

3.1.3 Network Architecture

Using neural networks toolbox in Matlab, a feedforward back propagation network is built, which has three layers:

- **Input layer-** consists of 15 input neurons, one for each of the finger joint angle values collected from CG.
- **Hidden layer-** receives data from the input layer and sends data to the output layer. Number of hidden neurons for this layer is 10, which was determined by trial and error method.
- **Output layer-** receives input from the hidden layer and sends data out of the network. Number of output neurons is set to 28, each representing different hand shape (manual alphabet and first ten digits). These outputs are compared against target outputs and the difference is fed back to the network, where the weights are updated accordingly.

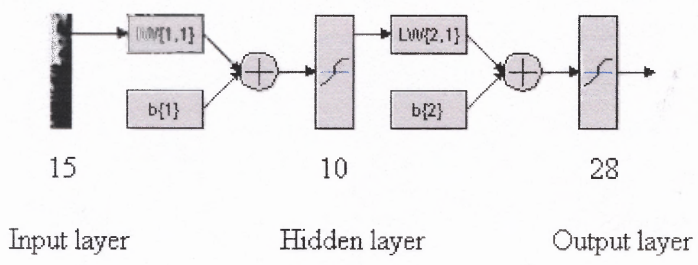


Figure 3.4 Network Architecture.

The network's training data, i.e., the input data and desired output data consists of decimal numbers ranging from -1 to 1, which is determined by the transfer function. A non linear transfer function called tan-sigmoid is chosen for the hidden layer and the output layer. The network is trained using least mean square or Widrow Hoffs learning

algorithm. Under this supervised learning, the neural network learns to recognize the relationship between input and output data. Figure 3.4 shows the outline of the network architecture.

3.1.4 Transfer Function

A transfer function is a function that maps a layer's net output 'n' to its actual output. The backpropagation network's training data, i.e., the input data and desired output data, usually consists of decimal numbers in the range 0 to 1 or sometimes from -1 to 1. This range is determined by the transfer function used in the network. The transfer function for the hidden layer and output layer of our network is a non linear function called as tan sigmoid transfer function, with the range from -1 to 1. It calculates the layer's output from its net input. Figure 3.4 shows the curve of a tan-sigmoid transfer function.

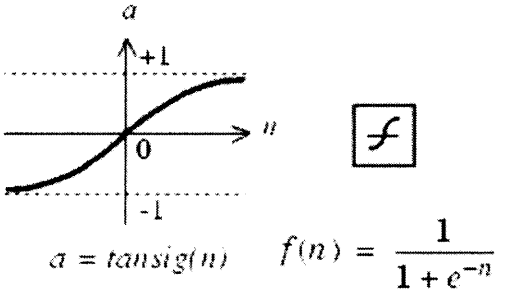


Figure 3.5 Tan sigmoid transfer function ranging from -1 to 1.

3.1.5 Learning Rule

The learning rule used for training the network is Widrow-Hoffs rule or Delta rule, which states that if two layers, say Input layer (i) and Hidden layer (h), are active simultaneously, the interconnections between them, i.e., weights (w) must be

strengthened. For a given input vector x (n), if d (n) is the desired output of the neuron and y (n) is the calculated outputs from the network, then the error is given by:

$$Error,\ e\ (n) = d\ (n) - y\ (n)$$

The synaptic weights during the learning process are adjusted according to the following learning rule:

$$Synaptic\ adjustment,\ \Delta w_{(i)} = \eta\ e\ (n) * x_{(i)}\ (n)$$

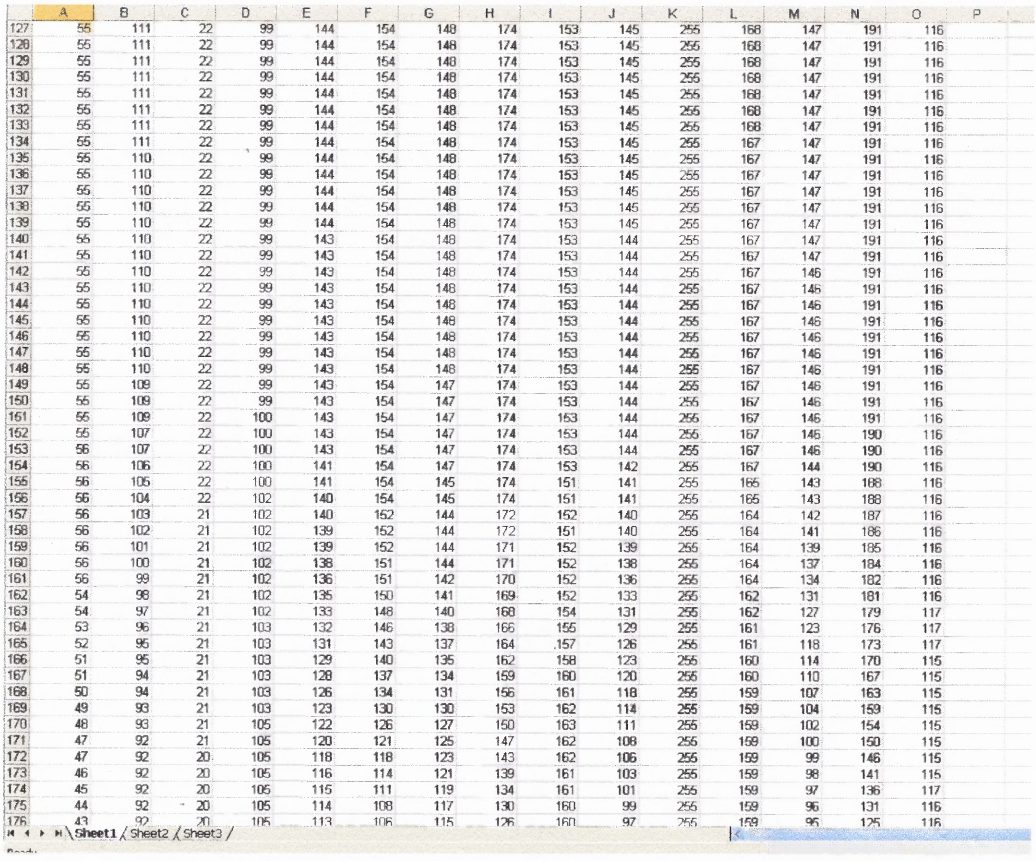
$$Updated\ synaptic\ weight,\ w_{(i)}\ (n+1) = w_{(i)}\ (n)$$

Where, eta is the learning rate parameter.

3.1.6 Steps in creating the network:

STEP 1: Assemble the training data:

The training data includes training inputs and target/desired outputs. The input set in out case is the finger joint angle data collected from CG and the target outputs are decimal numbers ranging from -1 to 1. The outputs are arranged such that for each combination of 15 finger joint angle data, the output shows 1 for the desired hand shape and -1 for other hand shapes. The sample training inputs and target outputs is shown in figures 3.6 and 3.8, respectively.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
127	55	111	22	99	144	154	148	174	153	145	255	168	147	191	116	
128	55	111	22	99	144	154	148	174	153	145	255	168	147	191	116	
129	55	111	22	99	144	154	148	174	153	145	255	168	147	191	116	
130	55	111	22	99	144	154	148	174	153	145	255	168	147	191	116	
131	55	111	22	99	144	154	148	174	153	145	255	168	147	191	116	
132	55	111	22	99	144	154	148	174	153	145	255	168	147	191	116	
133	55	111	22	99	144	154	148	174	153	145	255	168	147	191	116	
134	55	111	22	99	144	154	148	174	153	145	255	167	147	191	116	
135	55	110	22	99	144	154	148	174	153	145	255	167	147	191	116	
136	55	110	22	99	144	154	148	174	153	145	255	167	147	191	116	
137	55	110	22	99	144	154	148	174	153	145	255	167	147	191	116	
138	55	110	22	99	144	154	148	174	153	145	255	167	147	191	116	
139	55	110	22	99	144	154	148	174	153	145	255	167	147	191	116	
140	55	110	22	99	143	154	148	174	153	144	255	167	147	191	116	
141	55	110	22	99	143	154	148	174	153	144	255	167	147	191	116	
142	55	110	22	99	143	154	148	174	153	144	255	167	146	191	116	
143	55	110	22	99	143	154	148	174	153	144	255	167	146	191	116	
144	55	110	22	99	143	154	148	174	153	144	255	167	146	191	116	
145	55	110	22	99	143	154	148	174	153	144	255	167	146	191	116	
146	55	110	22	99	143	154	148	174	153	144	255	167	146	191	116	
147	55	110	22	99	143	154	148	174	153	144	255	167	146	191	116	
148	55	110	22	99	143	154	148	174	153	144	255	167	146	191	116	
149	55	109	22	99	143	154	147	174	153	144	255	167	146	191	116	
150	55	109	22	99	143	154	147	174	153	144	255	167	146	191	116	
151	55	109	22	100	143	154	147	174	153	144	255	167	146	191	116	
152	55	107	22	100	143	154	147	174	153	144	255	167	146	190	116	
153	56	107	22	100	143	154	147	174	153	144	255	167	146	190	116	
154	56	106	22	100	141	154	147	174	153	142	255	167	144	190	116	
155	56	105	22	100	141	154	145	174	151	141	255	166	143	188	116	
156	56	104	22	102	140	154	145	174	151	141	255	165	143	188	116	
157	56	103	21	102	140	152	144	172	152	140	255	164	142	187	116	
158	56	102	21	102	139	152	144	172	151	140	255	164	141	186	116	
159	56	101	21	102	139	152	144	171	152	139	255	164	139	185	116	
160	56	100	21	102	138	151	144	171	152	138	255	164	137	184	116	
161	56	99	21	102	136	151	142	170	152	136	255	164	134	182	116	
162	54	98	21	102	135	150	141	169	152	133	255	162	131	181	116	
163	54	97	21	102	133	148	140	168	154	131	255	162	127	179	117	
164	53	96	21	103	132	146	138	166	155	129	255	161	123	176	117	
165	52	95	21	103	131	143	137	164	157	126	255	161	118	173	117	
166	51	95	21	103	129	140	135	162	158	123	255	160	114	170	115	
167	51	94	21	103	128	137	134	159	160	120	255	160	110	167	115	
168	50	94	21	103	126	134	131	156	161	118	255	159	107	163	115	
169	49	93	21	103	123	130	130	153	162	114	255	159	104	159	115	
170	48	93	21	105	122	126	127	150	163	111	255	159	102	154	115	
171	47	92	21	105	120	121	125	147	162	108	255	159	100	150	115	
172	47	92	20	105	118	118	123	143	162	106	255	159	99	146	115	
173	46	92	20	105	116	114	121	139	161	103	255	159	98	141	115	
174	45	92	20	105	115	111	119	134	161	101	255	159	97	136	117	
175	44	92	20	105	114	108	117	130	160	99	255	159	96	131	116	
176	43	92	20	105	113	106	115	126	160	97	255	159	95	125	116	

Sheet1 / Sheet2 / Sheet3 /

Figure 3.6 Sample of finger joint angle (input) data collected from CG.

Before creating target output data, we have to classify each hand shape based on the velocity plot. Figure 3.7 shows two subplots of angular velocities. The first subplot shows the joint angular velocities of all the 15 angles. The second subplot shows the absolute summed velocities of the finger joint angles. 28 velocity minima correspond to the 28 hand shapes in the training data. The data points (red) are marked where there is an abrupt change in hand shape and accordingly the target output data are created with decimal numbers 1 (for the desired output hand shape) and -1 (for other hand shapes).

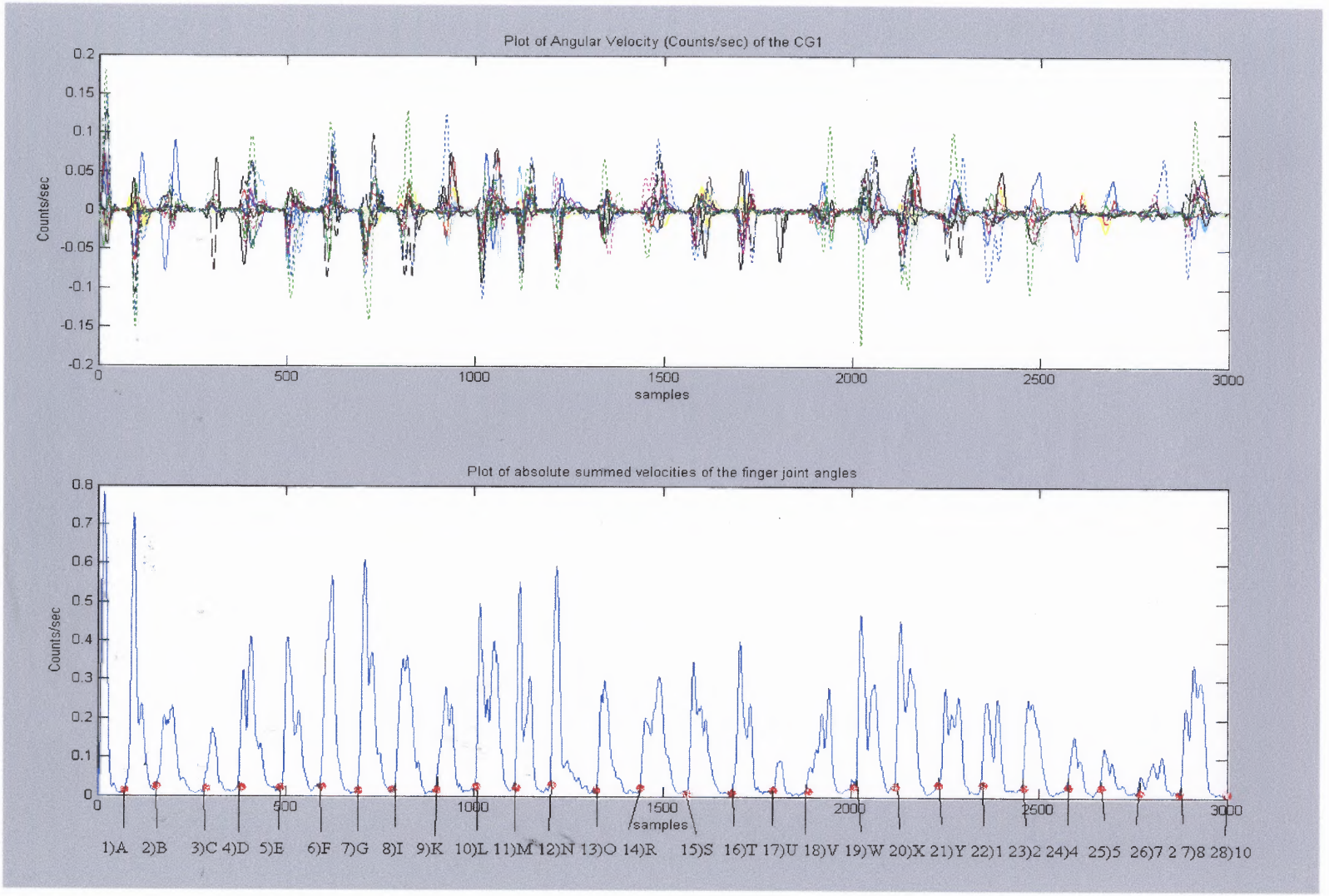


Figure 3.7 Figure showing angular velocities of 15 finger joint angles (first subplot) and summed velocities (second subplot).

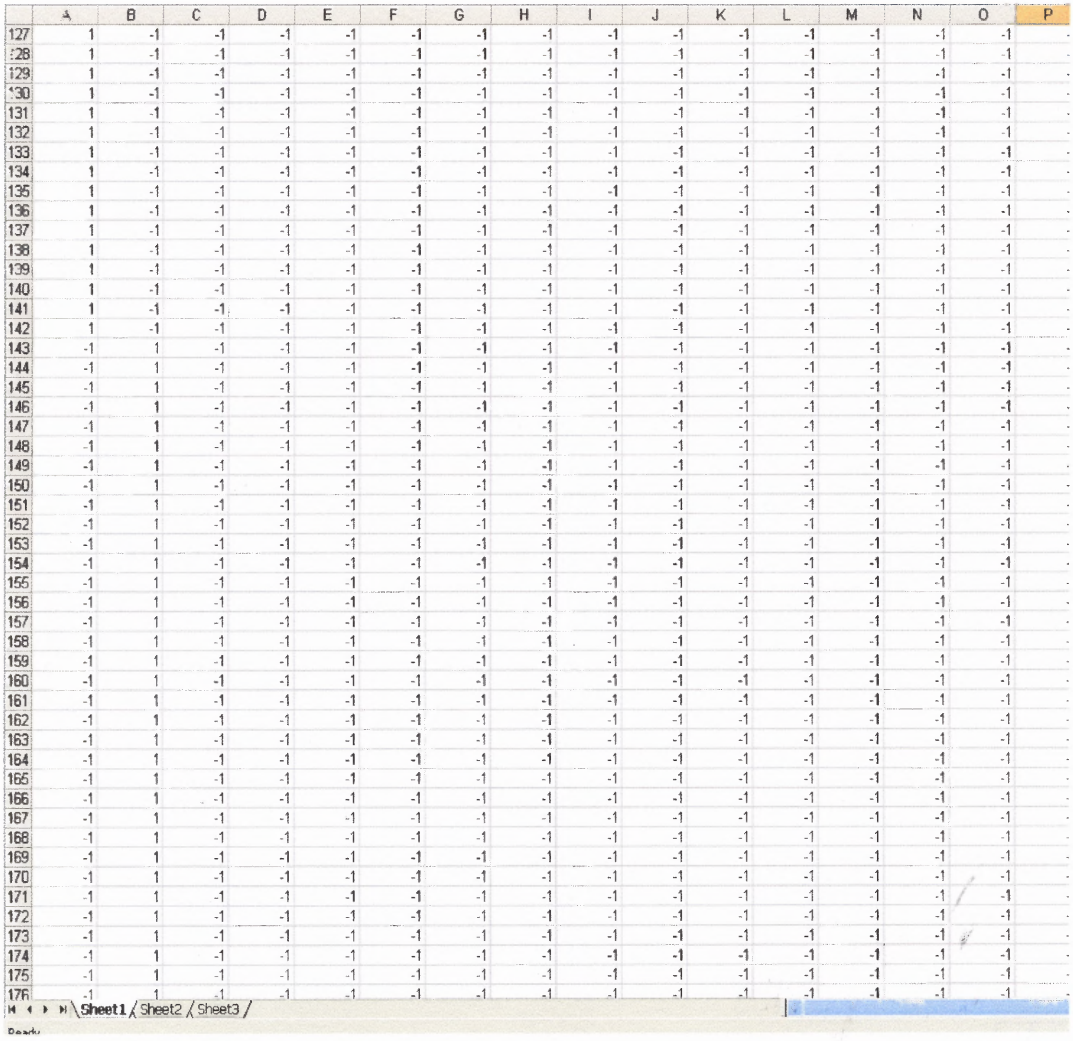


Figure 3.8 Sample of target output sets for training purpose.

STEP 2: Create the network object-

A three layered feed-forward network, with 15 neurons (for 15 finger joint angles) in the Input layer (the first layer), 10 neurons in the Hidden layer (second layer) and 28 neurons (for 28 different hand shapes) in the Output layer (third layer), is created. The number of neurons in the hidden layer is determined by trial and error method.

Matlab command 'newff' creates a feedforward network, in which the first input is an R by 2 matrix of minimum and maximum values for each of the R elements of the

input vector. The second input is an array containing the sizes of each layer. The third input is a cell array containing the names of the transfer functions to be used in each layer. The final input contains the name of the training function to be used. The default training function is usually 'trainlm'.

net = newff (minmax (TI), [10 size (TO,1)], {'tansig' 'tansig'})

This creates the network object and also initializes the weights and biases of the network. Now the network is ready for training.

STEP 3: Train the network-

The Training Inputs and Outputs are loaded into the program and the network is trained. The training data is divided into training set (60%), validation set (20%) and testing set (20%). The training set can be decreased to improve generalization of the network. Matlab command 'dividevec' is used to separate a set of input and target data into groups of vectors for training, validating network performance during training so that training stops early if it attempts to overfit the training data, and test data is used for an independent measure of how the network might be expected to perform on data it was not trained on.

dividevec (P, T, 20%, 50%) takes the following input,

- R x Q matrix of inputs.
- S x Q matrix of targets.

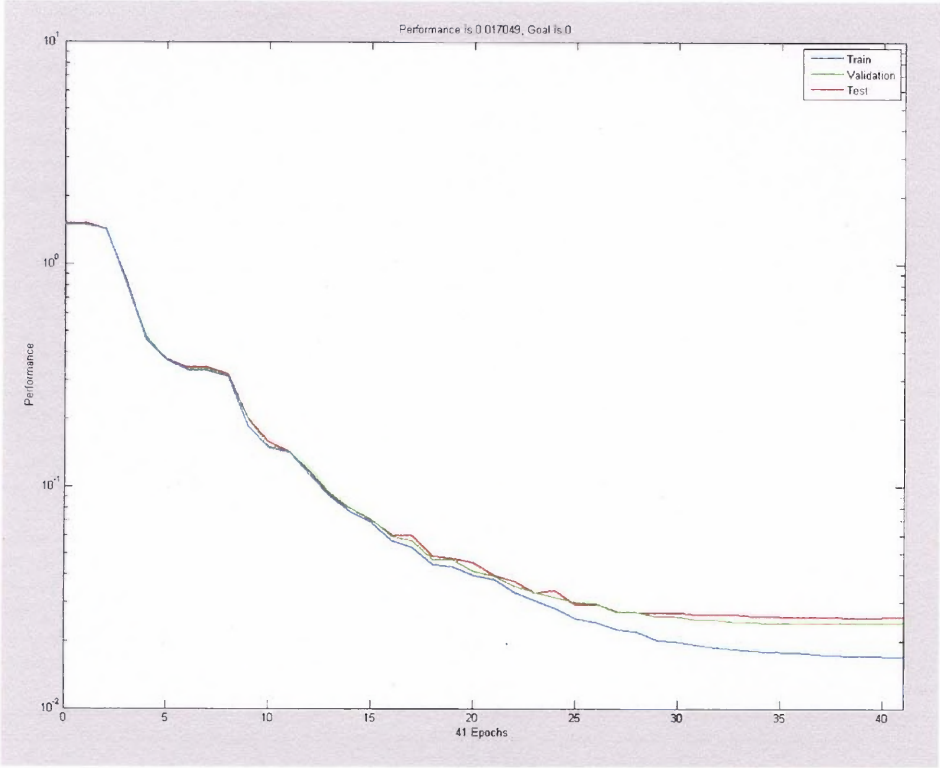


Figure 3.9 Figure showing the error after each epoch while training with TRAINLM.

An epoch is presentation of the set of training (input and/or target) vectors to a network and the calculation of new weights and biases. Training stops when the goal (for error) is reached, or when the epoch reaches the maximum value. Figure # shows the performance of the network with the training, validation and testing data sets. Initially the error is high since the initial weights of the network are chosen at random. The error gradually gets reduced as the weights get updated after every epoch. From the figure, we can see that the error reached zero at 42nd epoch and hence the training has stopped. Training can be stopped early before the goal is reached in order to improve generalization of the network.

STEP 4: Simulate the network response to new inputs-

The regression plot of the desired outputs against the obtained outputs determines whether the network is ready for classification of hand shapes on new inputs. Figure 3.10 shows the regression plot between the target, T and output of the network, A. The black circles refer to data points and the red line is the best fitting line between the two data sets. The regression value is R = 0.98, which indicate a good linear fit and that the network is now ready for classifying new set of inputs. Matlab function 'sim' simulates a network to a new set of inputs. It takes the network input, TI and network object, 'net' and returns the network output A.

A = sim (net, TI)

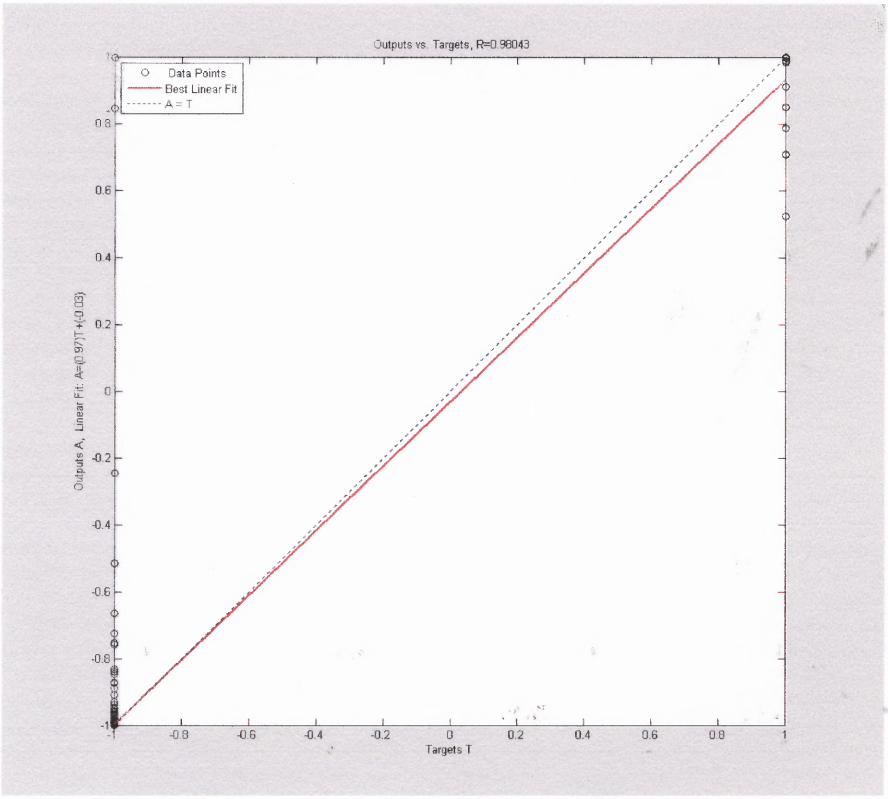


Figure 3.10 Regression plot between targets T and network output, A.

3.2 Results and Discussion

After training the network with trial 01 data set, the outputs were plotted for the test set (for about 600 samples). As discussed earlier, the input data was divided into three sets, training data, validation and testing in a 60:20:20 ratios. That means if there are 100 samples for the first hand shape, 60 samples are taken for training, 20 for validating and 20 for testing on the network. These testing samples are the unknown values for the network since they were not included in training. Figure 3.11, 3.12 and 3.13 show the plot of output values for each sample. X-axis numbers from 1 to 28 indicate the 28 hand shapes in the following sequence: 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'i', 'k', 'l', 'm', 'n', 'o', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', '1', '3', '4', '5', '7', '8', '10' and Y-axis indicate the output values from the network (ranging from -1 to 1). It was observed that the network was able to classify all the samples accurately.

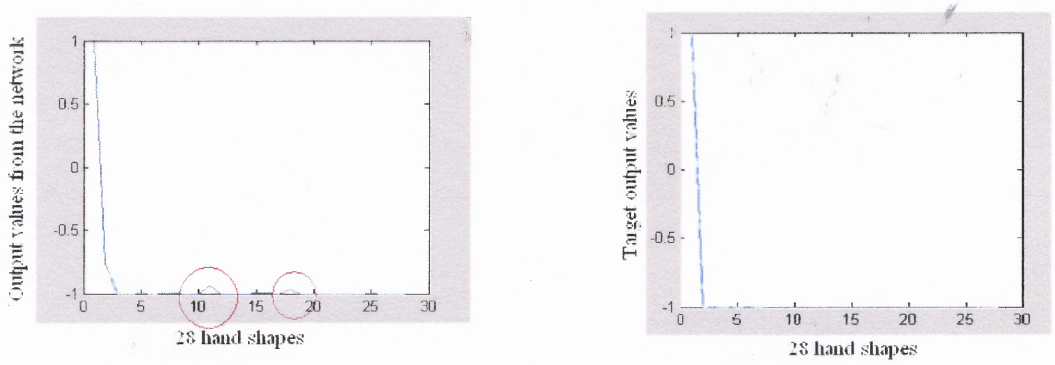


Figure 3.11 Figure comparing the output response of the network with the target output for sample 1.

Output plot for the sample 1 (figure 3.11) shows that the highest value is seen for the alphabet 'a' and little value is shared for the alphabets 'm' (first red circle in the left

plot) and 'v' (second red circle in the left plot). Since the variation in hand configurations for those alphabets is just by one sensor.

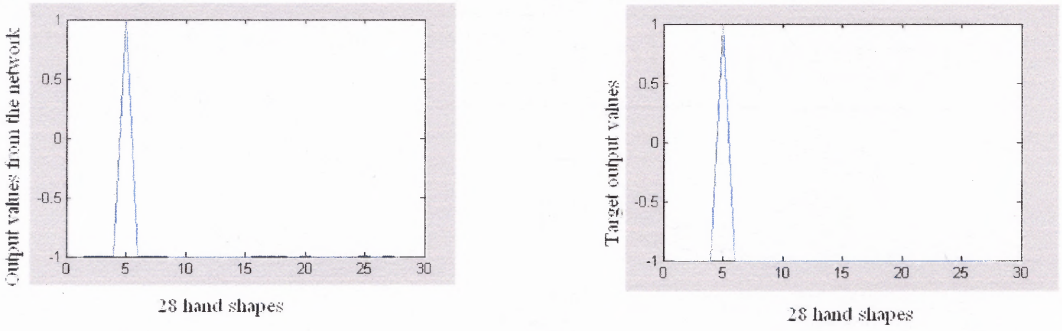


Figure 3.12 Figures comparing the output response of the network with the target output for sample 69.

Output plot for the sample 69 shows the highest value is given for the alphabet 'e', which is same as that of the desired output.

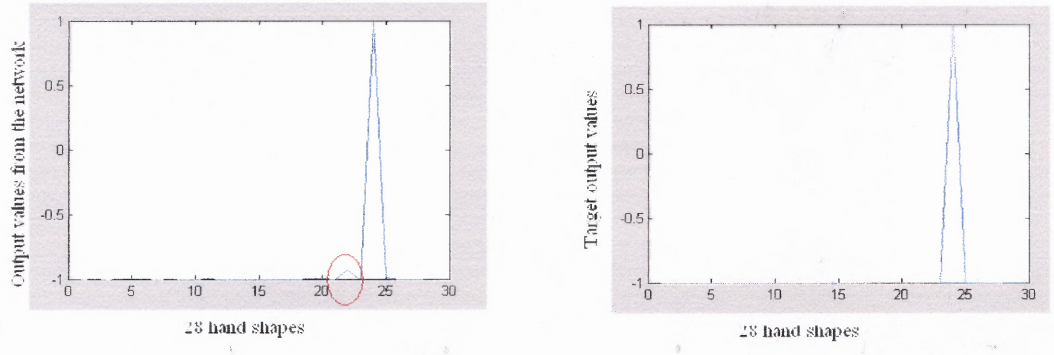


Figure 3.13 Figures comparing the output response of the network with the target output for sample 504.

The small peak in the left plot (figure 3.13) indicate the transition of hand shape '3' to '4' (sign language numbers) as the data that was given for training is collected in a sequence that had '4' followed by '3'.

The above results confirm that the network was able to recognize all the trained alphabets perfectly and is ready to perform on new input data. The network was simulated for the remaining trials as well and it was observed that the network was able to identify 50% of the hand shapes accurately without any ambiguity. The following figures demonstrate the problems occurred while classifying the new hand shape input data.

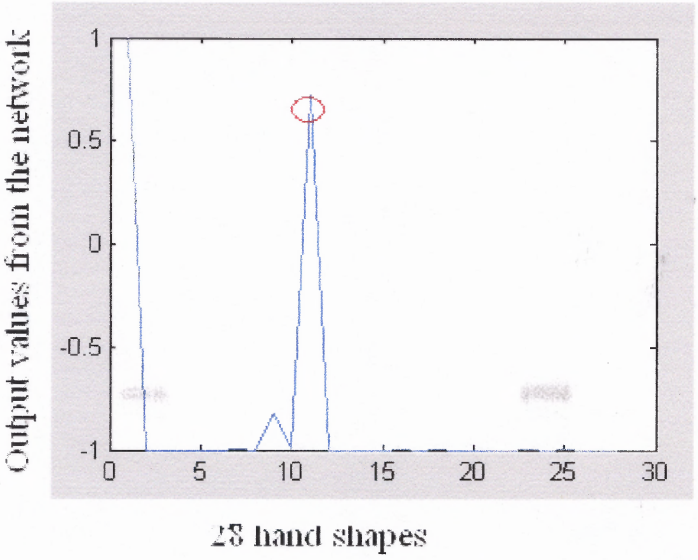


Figure 3.14 Plot of output of the network for sample 1 for new set of inputs.

From the plot it is seen that maximum value (two peaks) is shown for the number 1 and 11 corresponding to alphabets 'a', and 'm', respectively. The reason for this ambiguity could be the absence of the sensor (bend sensor) present for the

trapeziometacarpal joint at the thumb that can detect angles when the thumb is bent down to touch ring finger while signing the alphabet 'm'. The data collected with this variation in the glove can be given for the training data to improvise the network.

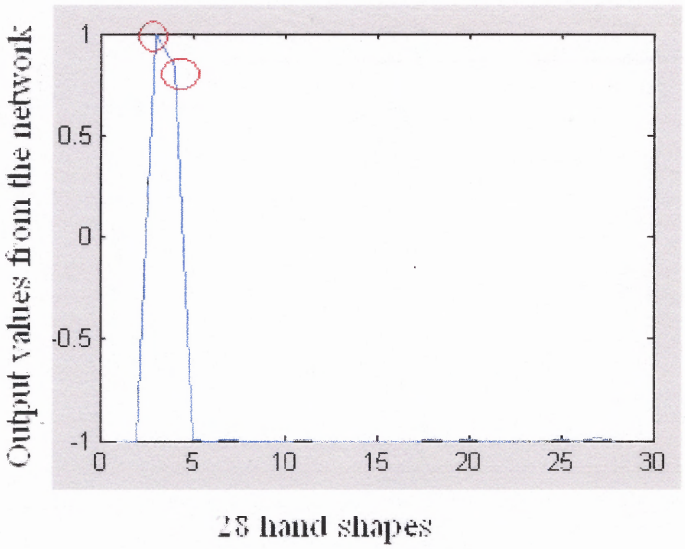


Figure 3.15 Plot of output of the network for sample 156 for new set of inputs.

From the plot it is seen that the output response is ambiguous with two peaks sharing the highest value. This is because of training the network with transition values for each alphabet. One such case is demonstrated in figure 3.16, which shows the transition of alphabet from 'c' to 'd'.

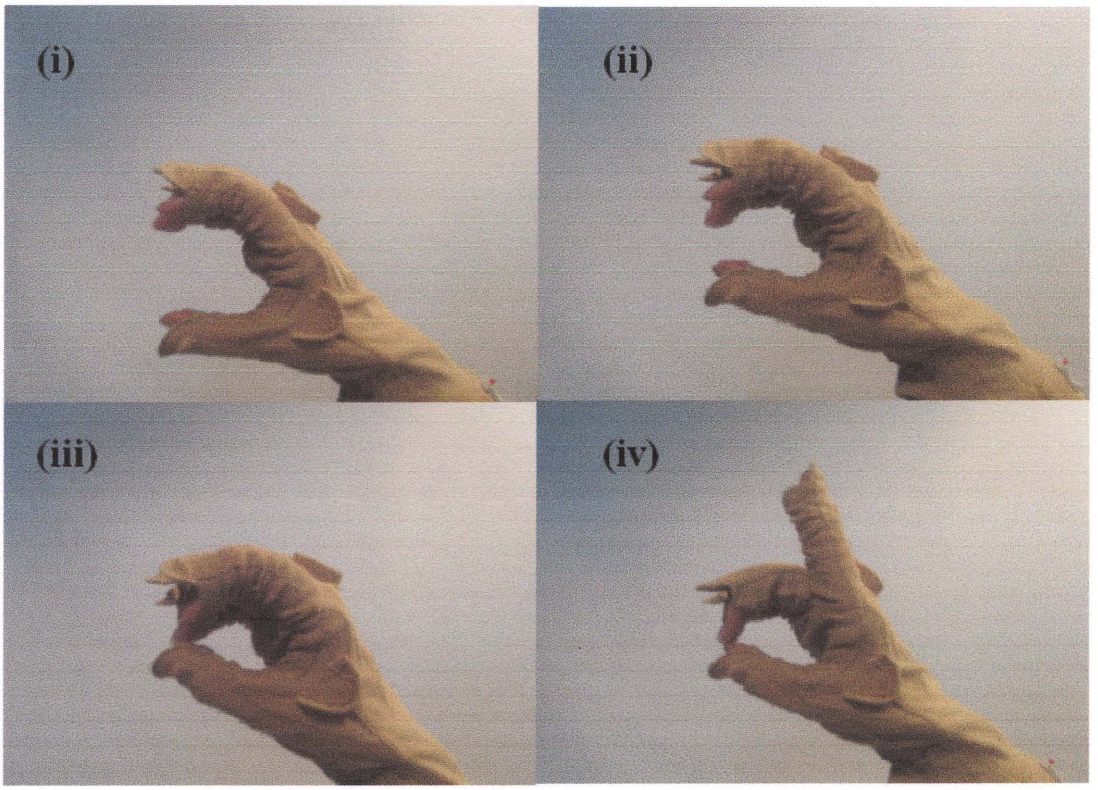


Figure 3.16 Figure showing the transition of hand shape instances when signing 'c' to 'd'
(i) Alphabet 'c', (ii) transition, (iii) transition (looks like alphabet 'o')
(iv) Alphabet 'd'

This ambiguity in the output can be eliminated either by increasing the weights of the network while training or by excluding the sensor input that is less significant.

CHAPTER 4

CLASSIFICATION OF LOCATIONS

4.1 Fuzzy Logic

Fuzzy logic is a problem-solving methodology applied in the areas where precision plays a trivial role. As the name suggests, the concepts of fuzzy logic are based on human reasoning which are nearly accurate rather than precise. It is similar to fuzzy set theory where classes of objects have unsharp boundaries and whose membership is a matter of degree. We use fuzzy logic to map an input space to an output space by defining a set of if-then statements called rules. Once the input and output variables are defined, the rules that interpret these input-output terms are determined. The rules are evaluated in parallel and the order is irrelevant. Figure 4.1 shows the basic mechanism of fuzzy logic for a general case.

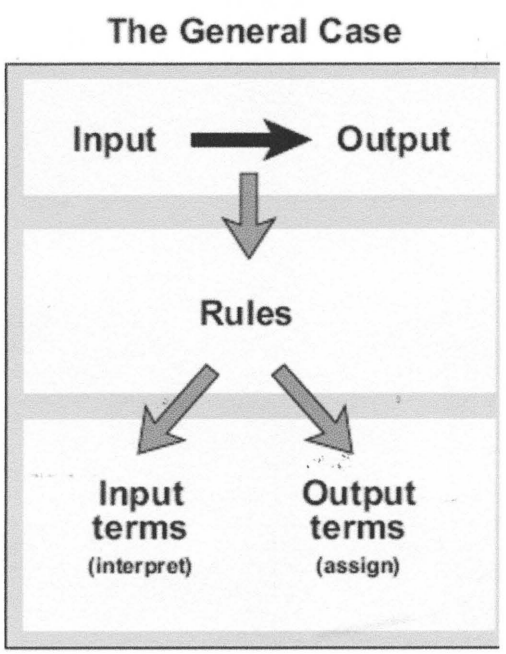


Figure 4.1 The basic mechanism of Fuzzy Logic.

Fuzzy logic is based on natural language. It is flexible, faster, cheaper, can be blended with conventional methods, and is a powerful technique which can deal inadequately with imprecision and nonlinearity. Some ingenious people have stated in the past:

"Precision is not truth."— Henri Matisse

"Sometimes the more measurable drives out the most important."
— René Dubos

"Vagueness is no more to be done away with in the world of logic than friction in mechanics." — Charles Sanders Peirce

"So far as the laws of mathematics refer to reality, they are not certain. And so far as they are certain, they do not refer to reality."
— Albert Einstein

"As complexity rises, precise statements lose meaning and meaningful statements lose precision."— Lotfi Zadeh

Hence in fuzzy logic the truth of any statement becomes a matter of degree.

4.2 Fuzzy Logic Interface Structure

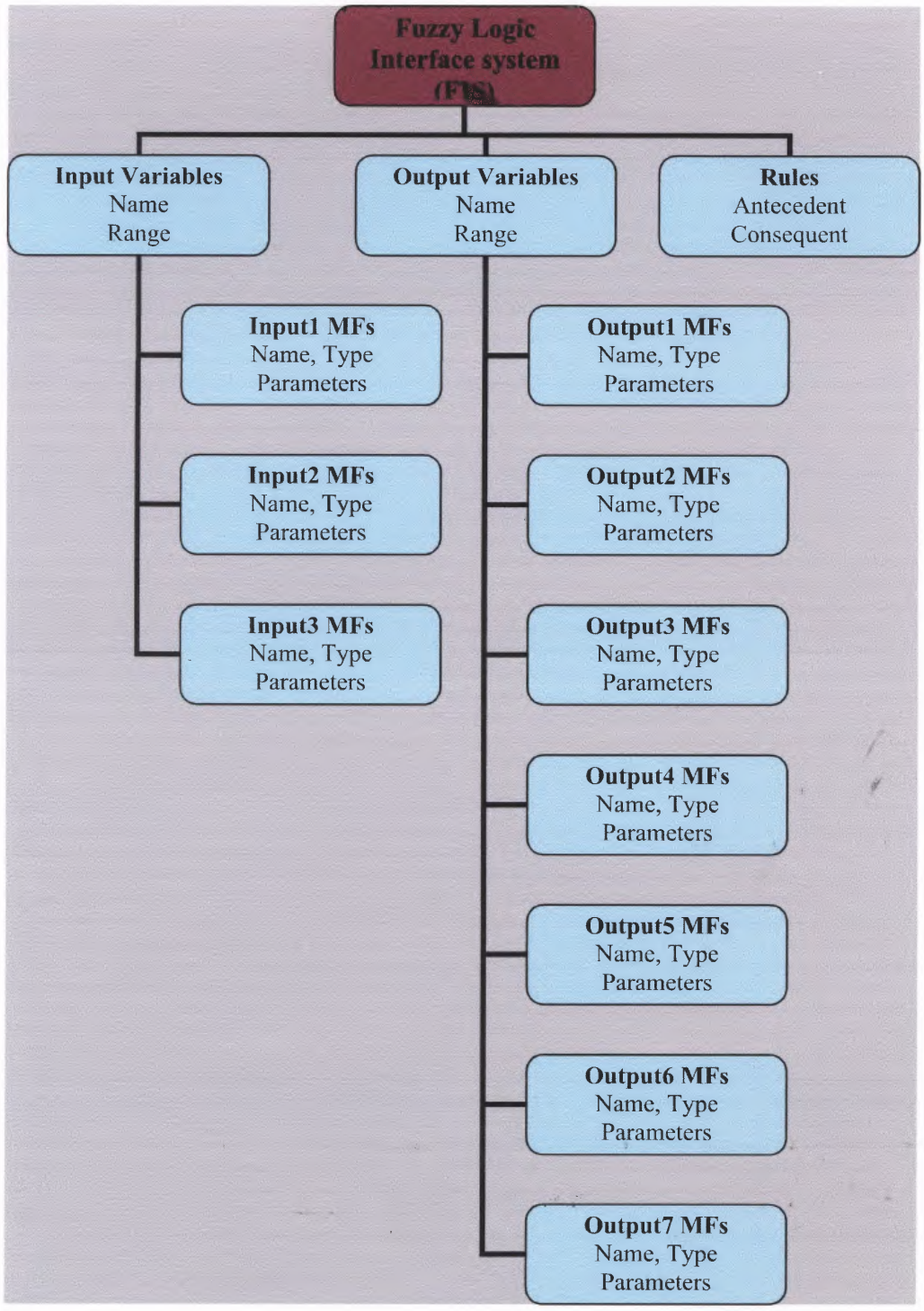


Figure 4.2 Structure of FIS for location classification in sign language.

Fuzzy Logic is a method that interprets the values in the input vector and, based on some set of rules, assigns values to the output vector. The first step in building such system is to define the input-output variables and designate a set of rules that interpret these variables. The basic structure of FIS is shown in figure 4.2, and its parameters are described as follows:

- Name: 'handlocations7'
- Type: 'Mamdani'.
- Membership function Type: 'Gaussian'
- AND Method: 'Minimum'
- OR Method: 'Maximum'
- Implication Method: 'min'
- Aggregation Method: 'max'

Mamdani type- There are two types of fuzzy interface methods, Mamdami and Sugeno or Takagi-Sugeno-Kang. The main difference between these two is that the sugeno output membership functions are either linear or constant. Since our output membership function is non-linear we prefer Mamdani type.

Logical operators- The Fuzzy Logic reasoning is a superset of Boolean logic. Figure 4.3 shows the difference between the general two-valued logic/ Boolean logic and multivalued logic/ Fuzzy logic. Fuzzy Logic 'AND' operator takes the minimum value, among the input membership function values, A and B, min (A, B), while 'OR' operator takes the maximum value, max (A, B). And NOT operator is a negation, (1-A).

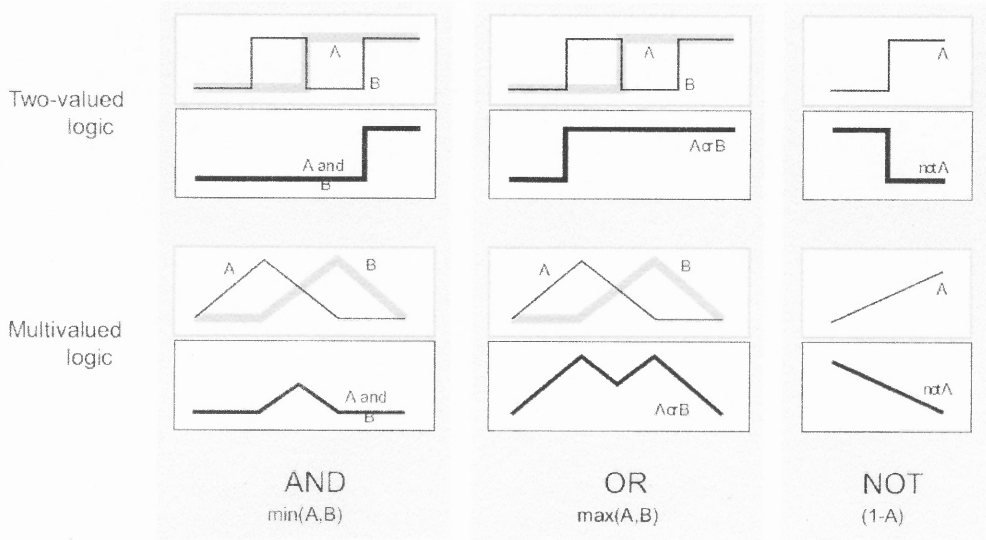


Figure 4.3 Figure showing the difference between Boolean and Fuzzy Logic.

Membership function (MF): A membership function (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The function itself can be an arbitrary curve whose shape we can define as a function that suits us from the point of view of simplicity, convenience, speed, and efficiency. If X is the universe of discourse and its elements are denoted by x, then a fuzzy set A in X is defined as a set of ordered pairs.

$$A = \{x, \mu_A(x) \mid x \in X\}$$

$\mu_A(x)$ is called the membership function (or MF) of x in A. The membership function maps each element of X to a membership value between 0 and 1. Figure 4.4 shows the Gaussian curve used as a membership function for our system, where P takes two parameters, 2 and 5.

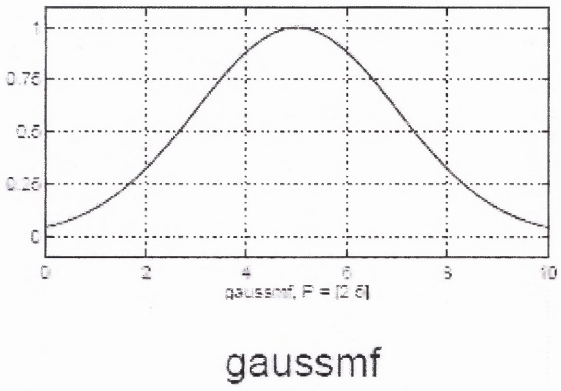


Figure 4.4 Gaussian curve.

Implication: Implication is the consequent or the then-part of the rule represented by a membership function. The input for the implication process is a single number given by the antecedent, and the output is a fuzzy set. Implication is implemented for each rule. There are two types for implication process, 'min' (minimum), which truncates the output fuzzy set and prod' (product), which scales the output fuzzy set.

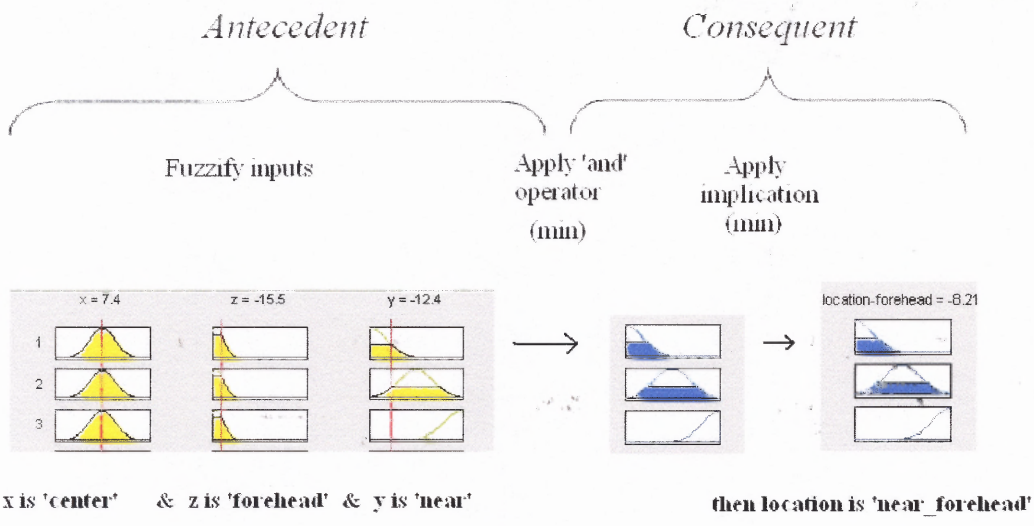


Figure 4.5 Figure illustrating the 'and' operator and 'implication' method in Fuzzy Logic.

4.3 Fuzzy Logic Interface System

A Fuzzy Logic Interface system is build with the help of fuzzy logic toolbox in Matlab, which can classify seven major locations in Sign language. Figure 4.6 shows an overview of this system.

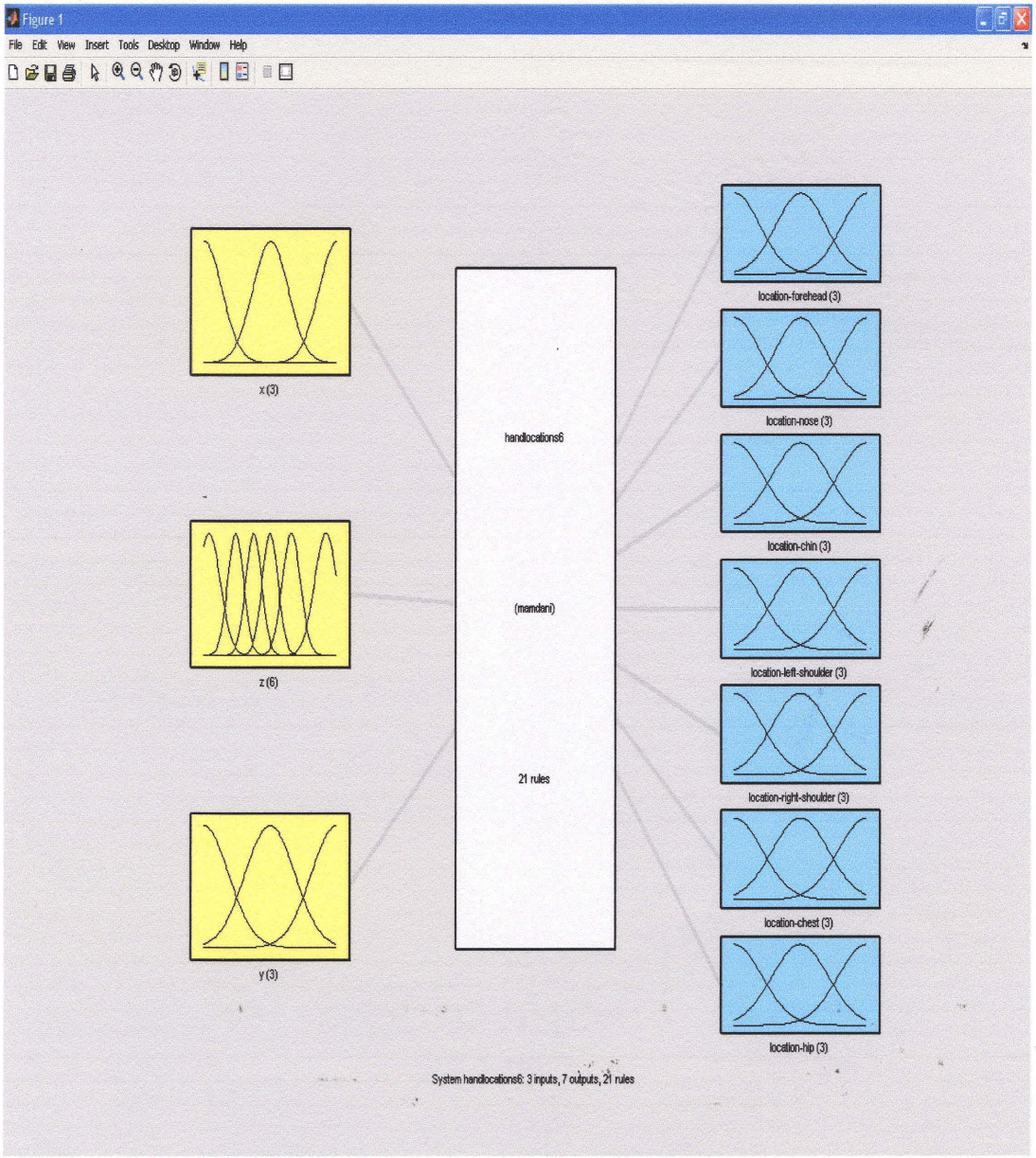


Figure 4.6 Overview of Fuzzy Logic Interface system built in Matlab.

The steps in building the interface system is as follows:

STEP 1: Define input variables:

Positions X, Z and Y are our input variables.

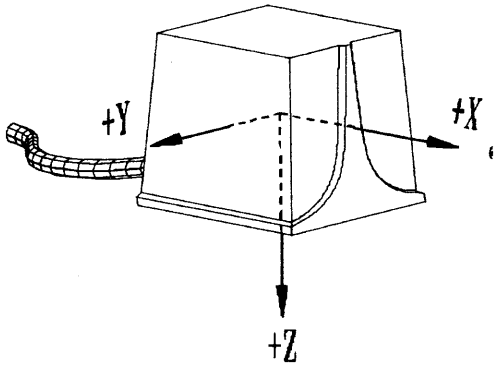


Figure 4.7 Figure showing the X, Y, Z directions with respect to the transmitter of NOB.

STEP 2: Set the range for each input variable

- X- From right to left [0 15].
- Z- From top to bottom [-18 7].
- Y- From near the body to away from the body [-18 6].

STEP 3: Define the number and type of membership functions for each of these variables and set the parameters. The type of membership function used is a 'Gaussian' function. Figures 4.7 shows the membership functions for the input variables X, Z and Y, respectively.

- X- 'left', 'center', 'right'.
- Z- 'forehead', 'nose', 'chin', 'shoulder', 'chest', 'hip'.
- Y- 'near', 'in front of', 'far'

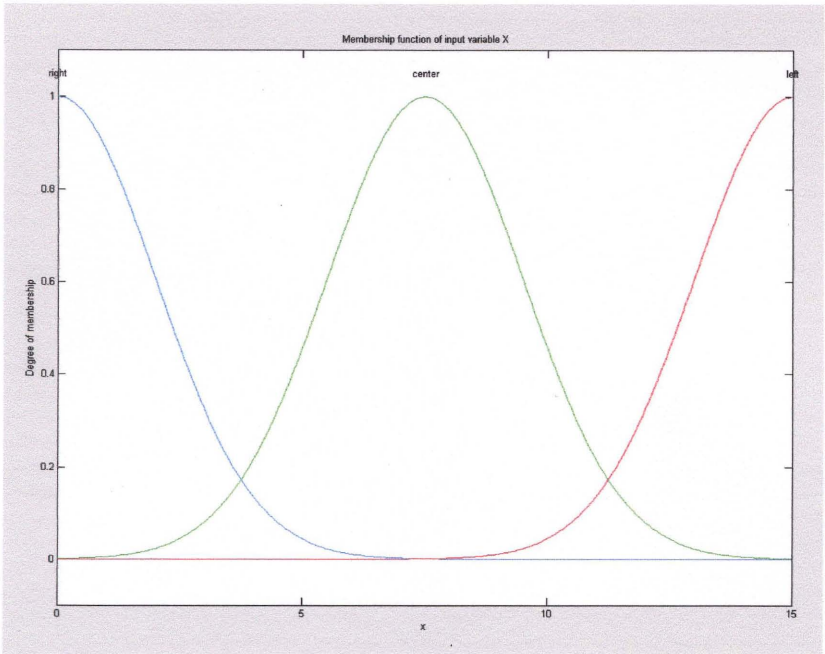


Figure 4.8 Membership functions for the input variable X.

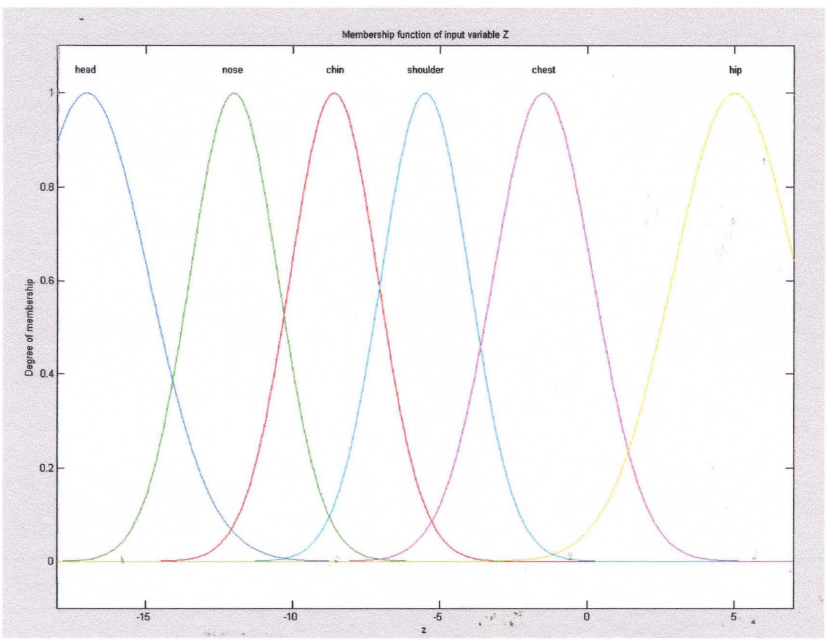


Figure 4.9 Membership function for the input variable Z.

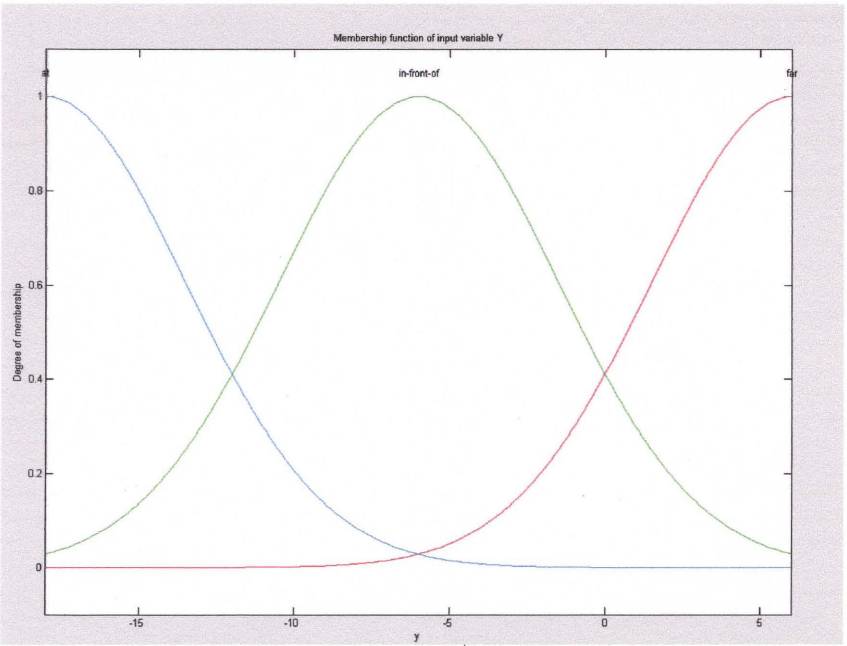


Figure 4.10 Membership function for the input variable Y.

STEP 4: Define the output variables:

The output variables are locations in sign language, i.e., Forehead, Nose, Chin, Left shoulder, Right shoulder Chest, and Hip.

STEP 5: Specify the range for the output variables

- Forehead- [-18 6].
- Nose- [-18 6].
- Chin- [-18 6].
- Left Shoulder- [-18 6]
- Right Shoulder- [-18 6]
- Chest- [-18 6]
- Hip- [-18 6]

STEP 6: Define the membership functions for each of the output variable.

- Forehead- 'near_head', 'in_front_of_head', 'far_head'.
- Nose- 'near_nose', 'in_front_of_nose', 'far_nose'.
- Chin- 'near_chin', 'in_front_of_chin', 'far_chin'.
- Left Shoulder- 'near_leftshoulder', 'in_front_of_leftshoulder', 'far_leftshoulder'.
- RightShoulder- 'near_rightshoulder', 'in_front_of_rightshoulder', 'far_rshoulder'.
- Chest- 'near_chest', 'in_front_of_chest', 'far_chest'.
- Hip- 'near_hip', 'in_front_of_hip', 'far_hip'.

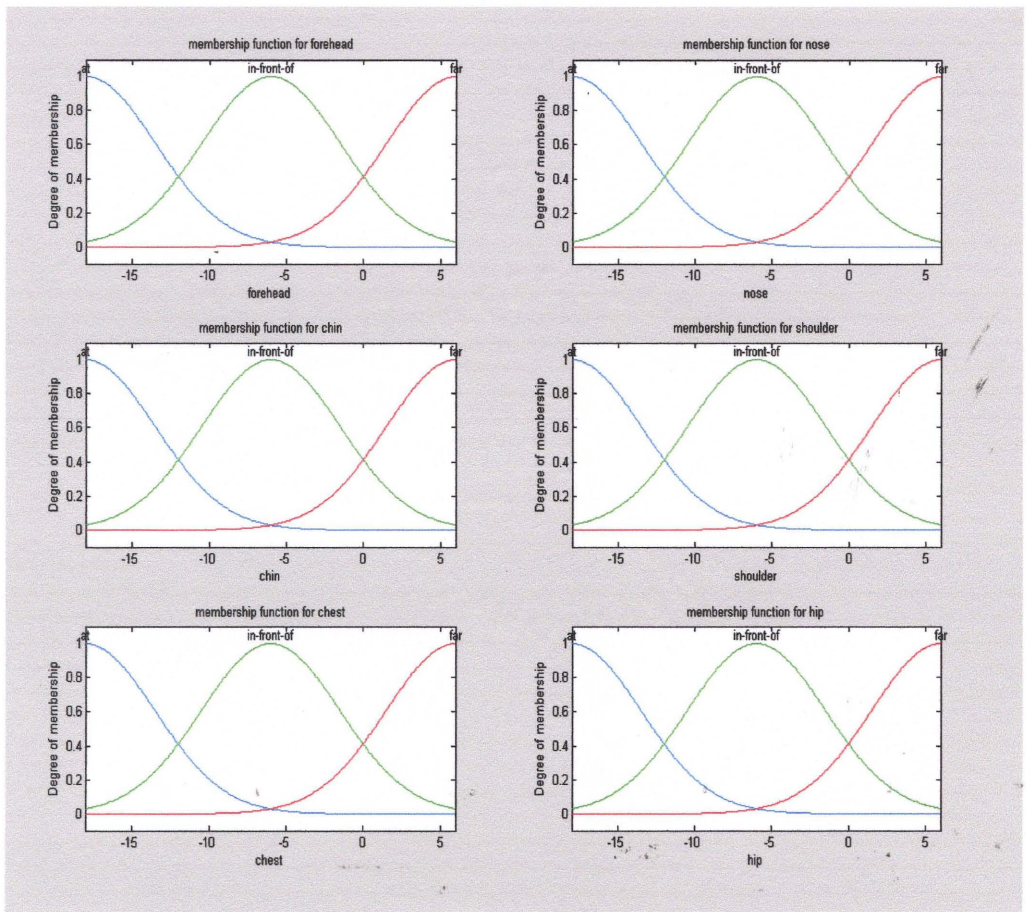


Figure 4.11 Membership functions for the output variables, Forehead, Nose, Chin, Shoulder, Chest, and Hip.

STEP 7: Finally assign the rules using one of the logical operators- AND/ OR/ NOT.

Figure 4.12 shows the rules assigned for input space to output space using an AND operator.

Figure 4.12 Figure showing edition of rules.

Fuzzy sets and fuzzy operators are the subjects and verbs of the fuzzy logic. These if-then rule statements are used to formulate the conditional statements that comprise fuzzy logic. A single fuzzy if-then rule assumes the form-

If x is 'left' and z is 'shoulder' and y is 'in front of', then location is 'in-front-of shoulder'.

Where 'left', 'shoulder', 'in front of', 'in front of shoulder' are linguistic values defined by the fuzzy sets on the ranges (university of discourse) x, z, y and location respectively. Table 4.1 shows all the rules defined in FIS. Column one shows the rule number. Columns from two to six represent the antecedent part of the rule and the last two columns represent the consequent part of the rule.

Table 4.1 Table Defining Rules.

#		X		Z		Y		Location
1	If	Center	&	Forehead	&	Near	Then	Near-forehead
2	If	Center	&	Forehead	&	In-front-of	Then	In-front-of-forehead
3	If	Center	&	Forehead	&	Far	Then	Far-forehead
4	If	Center	&	Nose	&	Near	Then	Near-nose
5	If	Center	&	Nose	&	In-front-of	Then	In-front-of-nose
6	If	Center	&	Nose	&	Far	Then	Far-nose
7	If	Center	&	Chin	&	Near	Then	Near-chin
8	If	Center	&	Chin	&	In-front-of	Then	In-front-of-chin
9	If	Center	&	Chin	&	Far	Then	Far-chin
10	If	Left	&	Shoulder	&	Near	Then	Near-left-shoulder
11	If	Left	&	Shoulder	&	In-front-of	Then	In-front-of-left-shoulder
12	If	Left	&	Shoulder	&	Far	Then	Far-left-shoulder
13	If	Right	&	Shoulder	&	Near	Then	Near-right-shoulder
14	If	Right	&	Shoulder	&	In-front-of	Then	In-front-of-right-shoulder
15	If	Right	&	Shoulder	&	Far	Then	Far-right-shoulder
16	If	Center	&	Chest	&	Near	Then	Near-chest
17	If	Center	&	Chest	&	In-front-of	Then	In-front-of-chest
18	If	Center	&	Chest	&	Far	Then	Far-chest
19	If	Center	&	Hip	&	Near	Then	Near-hip
20	If	Center	&	Hip	&	In-front-of	Then	In-front-of-hip
21	If	Center	&	Hip	&	Far	Then	Far-hip

4.4 Results and Discussion

4.5.1 Rule viewer

The rule viewer displays the roadmap of the whole fuzzy interface process. It is based on the fuzzy interface diagram described in figure 4.2. Figure 4.13 shows 210 nested small plots, where 10 small plots across the top represent the antecedent and consequent of the first rule. Each rule is a row of plots, and each column is a variable. The first three columns of the plots (the 63 yellow plots) show the membership functions referenced by the antecedent, or the if-part of each rule. Columns from four to ten of the plots (the 21 blue plots) shows the membership functions referenced by the consequent, or the then part of each rule. The red lines on the input plots can be slided to change the input values to generate a new output response. The blank plot in the output side corresponds to the non locations for all other combinations of the input variables. The twenty second plot under output plots represents the aggregate weighted decision for the interface system. The decision will depend on the input values of the system. The lower right area (text box) allows us to enter a specific input values to check the corresponding output response.

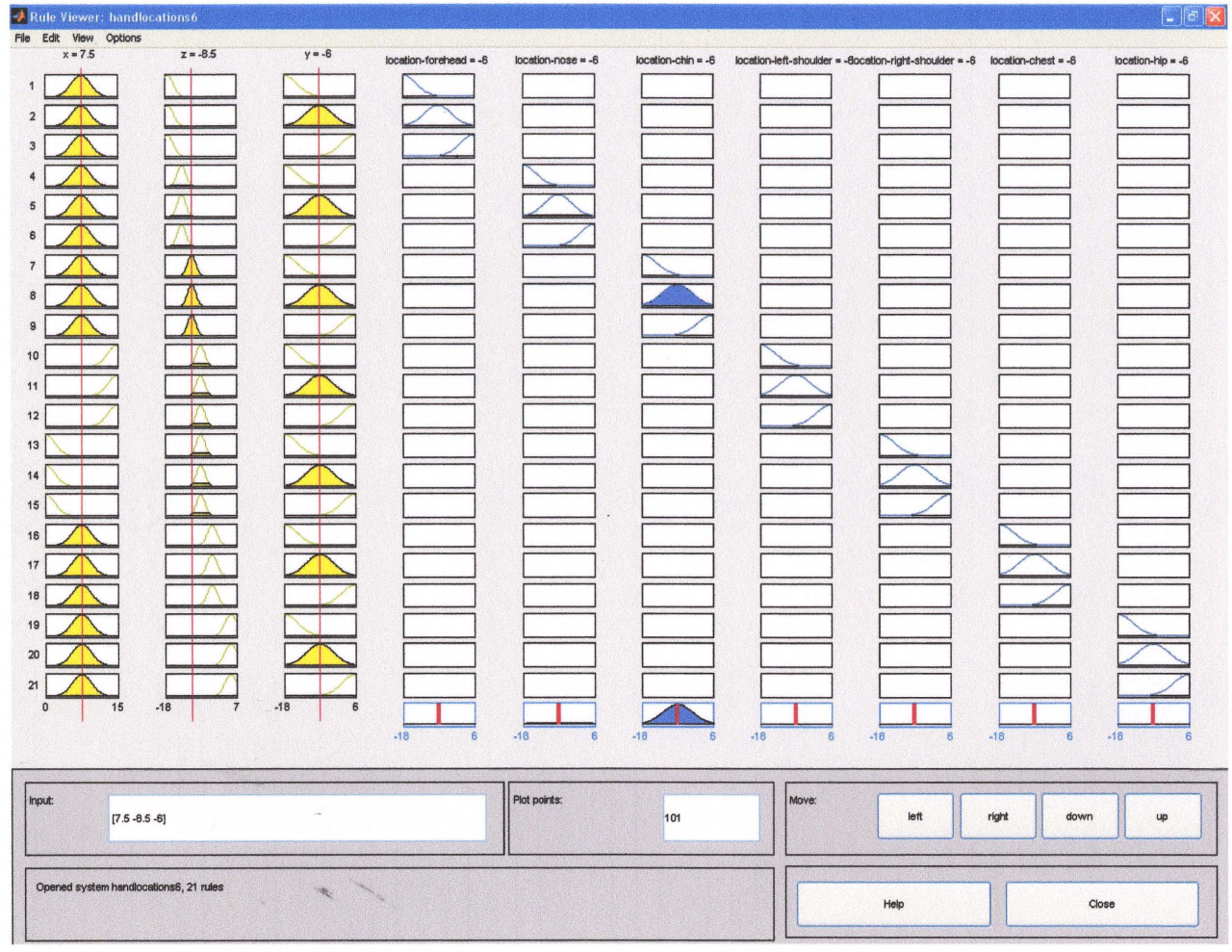


Figure 4.13 Rule viewer.

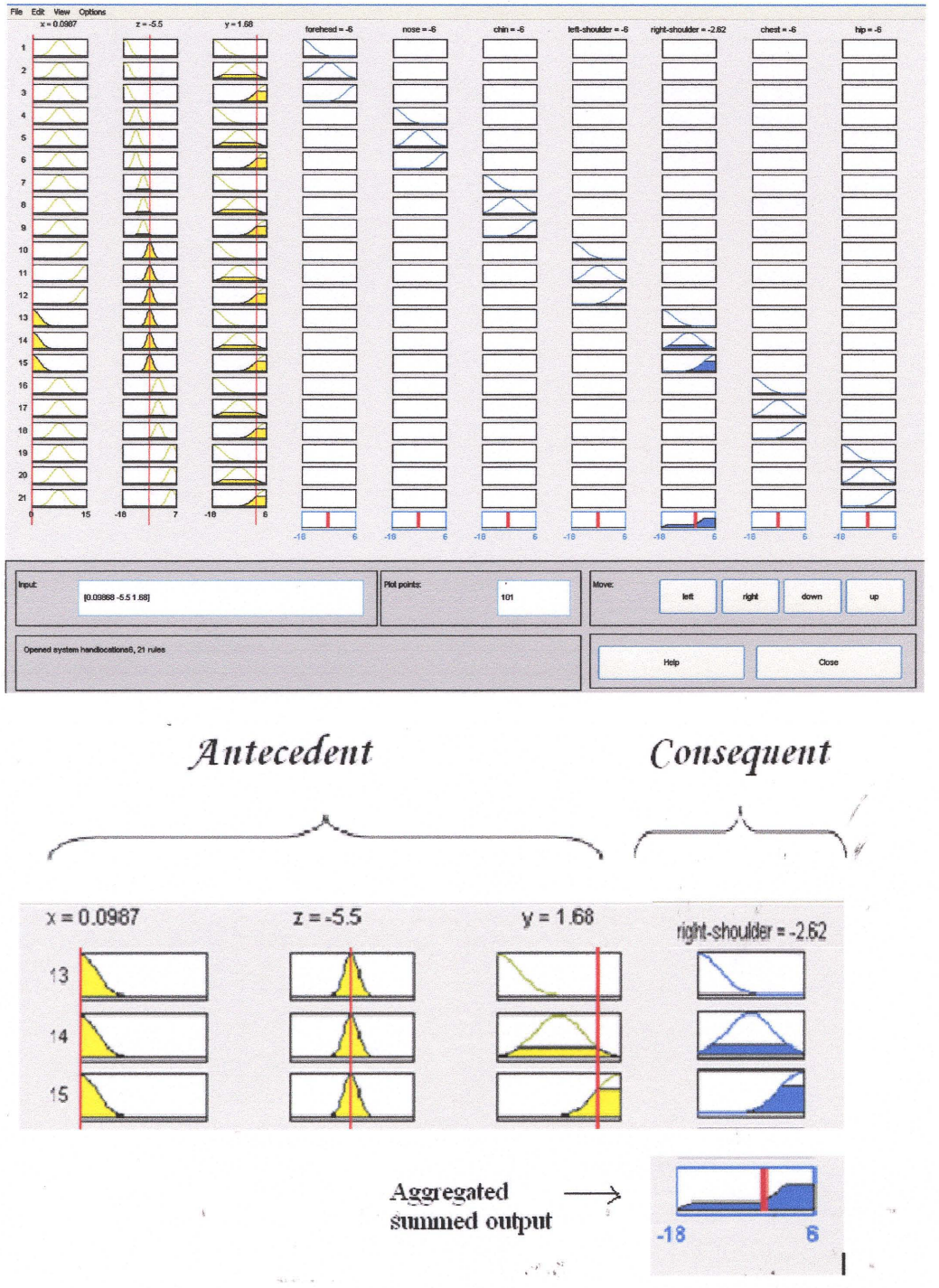


Figure 4.14 Figure on the top shows the rule viewer for the input X is 'right', Y is 'shoulder' and Z is 'far'. The figure below explains the rule.

The statement in figure 4.14 is if X is at the 'right', Z is at 'shoulder', and Y is at 'far', then output location is at 'far-right-shoulder'. It is also seen that input Y value shares certain membership function with the 'in-front-of', but the degree of membership function is higher for 'far' according to which the output membership functions are varied (blue plot for rules 14 and 15). Hence the fifteenth rule is considered, whose output membership function take the value equal to minimum value among the input membership functions (since AND operator). The output decision is based by comparing the aggregated output plots (last row). It can be seen from the figure that the aggregated output value which is more for location-in-front-of- than location-forehead, and hence the final output location is at 'far-right-shoulder'.

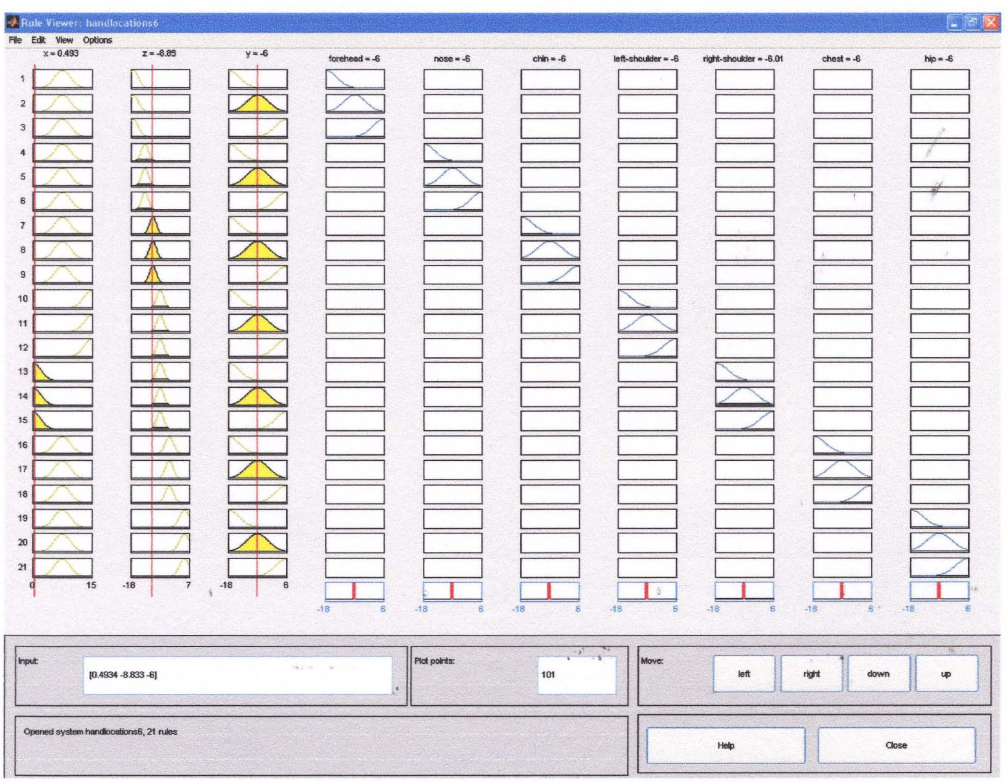


Figure 4.15 Rule viewer showing a case for non location.

The statement in figure 4.15 is if X is 'left', Z is 'chin', and 'Y is 'in-front-of', then location is no location, since no rule is defined for this combination of inputs and hence the FIS identifies it at a non location (empty plots).

CHAPTER 5

APPLICATIONS AND FUTURE WORK

Current techniques of translating spoken language into sign language include sequenced video clips, video animation and sign synthesis using parametrically driven animation. So far progress in sign language recognition has been limited only to static signs. This thesis demonstrates the possibility of recognition of two of the main parameters of sign language, the hand shape and location in dynamic signing. Future works will be carried on identification of other parameters of sign, like orientation and movement, which can form a main basis for sign segmentation.

APPENDIX A

MATLAB CODE FOR CLASSIFICATION OF HAND SHAPES

```
% Filtering the data

dataCG=xlsread('28 hand shapes aug 21 trial 02 mod');
L=length(dataCG(:,1));
time=0.01:0.01:(L/100);
co=6;  %Cut-off frequency
samp=100;   %Sampling rate
C=(co/(0.5*samp)); %10 hz cutofffreq * 2/ samplingfreq
[B,A]=butter(5,C);

fCG1=filtfilt(B,A,dataCG(:,1));
fCG2=filtfilt(B,A,dataCG(:,2));
fCG3=filtfilt(B,A,dataCG(:,3));
fCG4=filtfilt(B,A,dataCG(:,4));
fCG5=filtfilt(B,A,dataCG(:,5));
fCG6=filtfilt(B,A,dataCG(:,6));
fCG7=filtfilt(B,A,dataCG(:,7));
fCG8=filtfilt(B,A,dataCG(:,8));
fCG9=filtfilt(B,A,dataCG(:,9));
fCG10=filtfilt(B,A,dataCG(:,10));
fCG11=filtfilt(B,A,dataCG(:,11));
fCG12=filtfilt(B,A,dataCG(:,12));
fCG13=filtfilt(B,A,dataCG(:,13));
fCG14=filtfilt(B,A,dataCG(:,14));
fCG15=filtfilt(B,A,dataCG(:,15));

% Angular veloctity of CG

vCG1=CentralDiff(time,fCG1)';
vCG2=CentralDiff(time,fCG2)';
vCG3=CentralDiff(time,fCG3)';
vCG4=CentralDiff(time,fCG4)';
vCG5=CentralDiff(time,fCG5)';
vCG6=CentralDiff(time,fCG6)';
vCG7=CentralDiff(time,fCG7)';
vCG8=CentralDiff(time,fCG8)';
vCG9=CentralDiff(time,fCG9)';
vCG10=CentralDiff(time,fCG10)';
vCG11=CentralDiff(time,fCG11)';
vCG12=CentralDiff(time,fCG12)';
vCG13=CentralDiff(time,fCG13)';
```

```
vCG14=CentralDiff(time,fCG14)':
vCG15=CentralDiff(time,fCG15)';


L=length(vCG5);
subplot(2,1,1)
plot(vCG1,'DisplayName', 'sensor0')
hold on
plot(vCG2,'r','DisplayName', 'sensor1')
hold on
plot(vCG3,'--c','DisplayName', 'sensor2')
hold on
plot(vCG4,'y','DisplayName', 'sensor3')
hold on
plot(vCG5,'c','DisplayName', 'sensor4')
hold on
plot(vCG6,'--k','DisplayName', 'sensor5')
hold on
plot(vCG7,'m','DisplayName', 'sensor8')
hold on
plot(vCG8,':b','DisplayName', 'sensor9')
hold on
plot(vCG9,'k','DisplayName', 'sensor11')
hold on
plot(vCG10,':r','DisplayName', 'sensor12')
hold on
plot(vCG11,':c','DisplayName', 'sensor13')
hold on
plot(vCG12,':m','DisplayName', 'sensor15')
hold on
plot(vCG13,'--','DisplayName', 'sensor16')
hold on
plot(vCG14,':g','DisplayName', 'sensor17')
hold on
plot(vCG15,'g','DisplayName', 'sensor19')

title 'Plot of Angular Velocity (Counts/sec) of the CG1'
ylabel 'Counts/sec'
xlabel 'samples'

%Plot of summed absolute velocities of finger joint angles

VCG = [vCG1 vCG2 vCG3 vCG4 vCG5 vCG6 vCG7 vCG8 vCG9 vCG10 vCG11
vCG12 vCG13 vCG14 vCG15];
absVCG = abs(VCG);
```

```
for i=1:L
   abssumVCG(i,1) = sum(absVCG(i,:));
   i=i+1;
end

subplot(2,1,2)
plot(abssumVCG);
title 'Plot of absolute summed velocities of the finger joint angles'
ylabel 'Counts/sec'
xlabel 'samples'

% CREATING NETWORK FOR TRAINING THE INPUT DATA

clear all
clc

%4000 samples of 15-element input and 28-element target vectors are created

TI=xlsread('28 hand shapes trial 01')';
TO=xlsread('28 hand shapes trial 01 ops')';


%Here they are divided into training, validation, and test sets. Validation and test sets
contain 20% of the vectors each, leaving 60% of the vectors for training.

[trainV,valV,testV] = dividevec(TI,TO,0.20,0.20);


%A network is created and trained with the data.

net = newff(minmax(TI),[10 size(TO,1)]);


% The network parameters are set here

net.trainParam.show = 10;
net.layers{1}.transferFcn = 'tansig'
net.layers{2}.transferFcn = 'tansig'
net.performFcn = 'msereg';
net.performParam.ratio = 0.5;
net.trainParam.show = 5;
net.trainParam.epochs = 100;
net.trainParam.goal = 1e-5;
net.trainParam.min_grad = 0;
```

```
% Simulate the network to new inputs
Y1=sim(net,trainV.P);
Y2=sim(net,TI);
Y3=sim(net,testV.P);
```

APPENDIX B

MATLAB CODE FOR CLASSIFICATION OF LOCATIONS

```
% This program creates a Fuzzy Interface System (FIS) which maps the input position
% data to target locations on the body reached by the hand. The inputs are
%'x'(right to left), 'z'(above to below) and 'y'(from-the-body to far-away). The outputs
% (locations on the body) are 'forehead', 'nose', 'chin', 'shoulder', 'chest', 'hip'.


%creates a new FIS namely '6handlocations'
a = newfis ('6handlocations');

%adds an input variable 'x' whose range is [0 15].
a = addvar (a,'input','x',[0 15]);
%adds a gaussian membership function for the input variable 'x'.
a = addmf (a,'input',1,'right','gaussmf',[2 0]);
a = addmf (a,'input',1,'center','gaussmf',[2 7.5]);
a = addmf (a,'input',1,'left','gaussmf',[2 15]);
figure(1)
plotmf(a,'input',1)

%adds an input variable 'z', whose range is [-18 7]
a = addvar (a,'input','z',[-18 7]);
%adds a gaussian membership function for the input variable 'z'.
a = addmf (a,'input',2,'head','gaussmf',[2.123 -17]);
a = addmf (a,'input',2,'nose','gaussmf',[1.5 -12]);
a = addmf (a,'input',2,'chin','gaussmf',[1.5 -8.6]);
a = addmf (a,'input',2,'shoulder','gaussmf',[1.5 -5.5]);
a = addmf (a,'input',2,'chest','gaussmf',[1.7 -1.5]);
a = addmf (a,'input',2,'hip','gaussmf',[2.123 5]);
figure(2)
plotmf(a,'input',2)

%adds an input variable 'y', whose range is [-18 6]
a = addvar (a,'input','y',[-18 6]);
%adds a gaussian membership function for the input variable 'y'.
a = addmf (a,'input',3,'at','gaussmf',[4.5 -18]);
a = addmf (a,'input',3,'in-front-of','gaussmf',[4.5 -6]);
a = addmf (a,'input',3,'far','gaussmf',[4.5 6]);
figure(3)
plotmf(a,'input',3)

%adds an output variable 'forehead'
a = addvar (a,'output','forehead',[-18 6]);
```

```
a = addvar (a,'output','forehead',[-18 6]);
%adds a gaussian membership function for the output variable 'forehead'.
a = addmf (a,'output',1,'at','gaussmf',[4.5 -18]);
a = addmf (a,'output',1,'in-front-of','gaussmf',[4.5 -6]);
a = addmf (a,'output',1,'far','gaussmf',[4.5 6]);
figure(4)
plotmf(a,'output',1)

%adds an output variable 'nose'
a = addvar (a,'output','nose',[-18 6]);
%adds a gaussian membership function for the output variable 'nose'.
a = addmf (a,'output',2,'at','gaussmf',[4.5 -18]);
a = addmf (a,'output',2,'in-front-of','gaussmf',[4.5 -6]);
a = addmf (a,'output',2,'far','gaussmf',[4.5 6]);
figure(5)
plotmf(a,'output',2)

%adds an output variable 'chin'
a = addvar (a,'output','chin',[-18 6]);
%adds a gaussian membership function for the output variable 'chin'.
a = addmf (a,'output',3,'at','gaussmf',[4.5 -18]);
a = addmf (a,'output',3,'in-front-of','gaussmf',[4.5 -6]);
a = addmf (a,'output',3,'far','gaussmf',[4.5 6]);
figure(6)
plotmf(a,'output',3)

%adds an output variable 'shoulder'
a = addvar (a,'output','shoulder',[-18 6]);
%adds a gaussian membership function for the output variable 'shoulder'.
a = addmf (a,'output',4,'at','gaussmf',[4.5 -18]);
a = addmf (a,'output',4,'in-front-of','gaussmf',[4.5 -6]);
a = addmf (a,'output',4,'far','gaussmf',[4.5 6]);
figure(7)
plotmf(a,'output',4)

%adds an output variable 'chest'
a = addvar (a,'output','chest',[-18 6]);
%adds a gaussian membership function for the output variable 'chest'.
a = addmf (a,'output',5,'at','gaussmf',[4.5 -18]);
a = addmf (a,'output',5,'in-front-of','gaussmf',[4.5 -6]);
a = addmf (a,'output',5,'far','gaussmf',[4.5 6]);
figure(8)
plotmf(a,'output',5)

%adds an output variable 'hip'
a = addvar (a,'output','hip',[-18 6]);
```

```
%adds a gaussian membership function for the output 'hip'.
a = addmf (a,'output',6,'at','gaussmf',[4.5 -18]);
a = addmf (a,'output',6,'in-front-of','gaussmf',[4.5 -6]);
a = addmf (a,'output',6,'far','gaussmf',[4.5 6]);
figure(9)
plotmf(a,'output',6)

%here the rules are listed. The first 3 columns are inputs and the next 6
%columns are outputs. last but one column represents the weight for each rule
%and the last column represents 'AND'(1) or 'OR'(2) logical operator.


ruleList=[2 1 1 1 0 0 0 0 0 1 1
2 1 2 2 0 0 0 0 0 1 1
2 1 3 3 0 0 0 0 0 1 1
2 2 1 0 1 0 0 0 0 1 1
2 2 2 0 2 0 0 0 0 1 1
2 2 3 0 3 0 0 0 0 1 1
2 3 1 0 0 1 0 0 0 1 1
2 3 2 0 0 2 0 0 0 1 1
2 3 3 0 0 3 0 0 0 1 1
3 4 1 0 0 0 1 0 0 1 1
3 4 2 0 0 0 2 0 0 1 1
3 4 3 0 0 0 3 0 0 1 1
1 4 1 0 0 0 1 0 0 1 1
1 4 2 0 0 0 2 0 0 1 1
1 4 3 0 0 0 3 0 0 1 1
2 5 1 0 0 0 0 1 0 1 1
2 5 2 0 0 0 0 2 0 1 1
2 5 3 0 0 0 0 3 0 1 1
2 6 1 0 0 0 0 0 1 1 1
2 6 2 0 0 0 0 0 2 1 1
2 6 3 0 0 0 0 0 3 1 1];
a = addrule(a,ruleList);
rules = showrule(a,1:21);

fuzzy(a)
surfview(a)

X=input('enter the value of x')
Z=input('enter the value of Z')
Y=input('enter the value of Y')
% a=readfis('6handlocations')
evalfis([X, Z, Y],a)
ruleview(a)
plotfis(a)
```