uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES
ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND
POSDOCTORAL STUDIES

François Malric
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.C.S.
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Artificial Neural Network Based Optical Hand Posture Recognition Using a Color-coded Glove

TITRE DE LA THÈSE / TITLE OF THESIS

Nicolas Georganas
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Emil Petriu

Chris Joslin

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

ARTIFICIAL NEURAL NETWORK BASED OPTICAL HAND POSTURE RECOGNITION USING A COLOR-CODED GLOVE

by

François Malric

A thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

In partial fulfillment of the requirements for the

degree of

Master of Science

in

Computer Science

Ottawa-Carleton Institute for Computer Science

School of Information Technology and Engineering

Faculty of Engineering

University of Ottawa

Your file Votre référence
ISBN: 978-0-494-48478-4
Our file Notre référence
ISBN: 978-0-494-48478-4

Table of Contents

Abstract

Optical pose recognition of the hand is an extremely attractive method for user-computer interaction in many applications. The image of a hand in the frame of a video camera is processed and the pose it is making, its current finger configuration, is detected. Often combined with position tracking, it allows for a very natural way of giving commands. Furthermore, it alleviates the use of sometimes cumbersome pieces of hardware. Within immersive virtual reality systems, the liberty of movement of the commanding hand requires extra considerations not normally dealt with by typical optical hand posture recognition interfaces for desktop system applications. This research proposes an artificial neural network approach to the recognition of hand postures. The optical capture inside an immersive virtual reality workspace and the extraction of features of this hand are facilitated by the use of a specially coded color glove.

Acknowledgments

I would like to thank all the wonderful people I had the chance to work within the past twelve years at the DISCOVER laboratory. In particular, I thank my supervisor Prof. Nicolas Georganas.

Last, but not least, thanks to my wife Sylvie, my daughter Catherine and my son Nicolas for their love and support throughout the completion of this research.

List of Figures

List of Tables

List of Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
ANN	Artificial Neural Network
DIP	Distal Interphalangeal
DOF	Degrees of Freedom
MCP	Metacarpophalangeal
MLP	Multi-Layer Perceptron
PIP	Proximal Interphalangeal
VR	Virtual Reality

Chapter 1

Introduction

1.1 Motivation

It has become almost natural for today's human to interact with a computer by use of a mouse and a keyboard. With advances in the development of 3D display technologies, and as new applications evolve into sophisticate interactive virtual reality experiences, new methods for interacting with these systems must be explored. One method of interaction which people tend to use in day-to-day life, sometimes even unconsciously, is the hand gesture.

As opposed to traditional interaction methods using physical hardware devices, optical hand gesture recognition is much less intrusive and more convenient for exploring 3D virtual worlds. One or more cameras around a virtual reality (VR) display system capture the video of the user and detect his/her hand (Figure 1.1). The recognized posture of the hand can be interpreted as a control mode, and tracking of its dynamics used to infer the 3D manipulation required within the application [1, 2, 3].

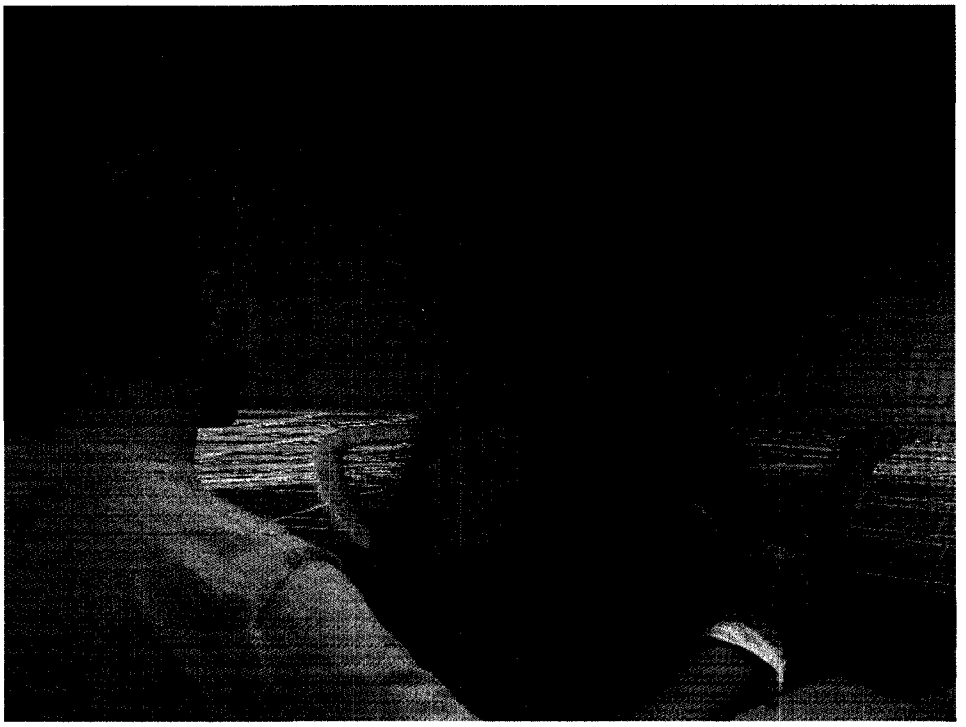

Figure 1.1 - Hand recognition used in an immersive VR application

Other applications could also benefit from the use of a flexible and real-time glove-based optical posture recognition system. Remote control, training (for example: piano playing, guitar playing [4]), interfacing with the hearing impaired, all could make use of such a system.

While much research has already tackled the problem of optical hand posture recognition, many restrictions usually apply, such as severe limits in detected hand orientation, restrictions in allowable background environment, or computationally intensive algorithms which restrain its use in real-time applications.

1.2 Problem Statement

We wish to enable the use of the user's hand as an interaction peripheral for immersive VR applications, using one (or more) simple fixed video camera(s) to capture the moving hand. Specifically, this user interface should work within an immersive VR display system where the movement of the hand is not restricted into staying in a plane parallel to the cameras' sensors. The hand of the user is free to move and rotate inside a volume that the user perceives as part of some virtual space. Within that space, the user's hand could take various postures to mean different intended interactions, such as pointing for moving in the direction of the index finger, closed fist for stopping, or making a C shape for grabbing, for example. A good number of hand postures should be recognizable. Also, the interactive nature of this user interface puts further constraints on the speed of all processing involved.

1.3 Proposed Solution

This research will present an artificial neural network technique for optical hand posture recognition using a color-coded glove. The proposed method allows for detecting, within a captured video frame, the posture a single gloved hand assumes, in a very wide range of view orientations and distances from a camera. An estimation of the hand orientation relative to the camera is also derived by this recognition system. We propose the use of a color-coded glove to aid in directly obtaining meaningful features and to ease the image

segmentation work required. A set of feed-forward artificial neural networks with voting and space partitioning is used for posture detection and hand orientation estimation from the extracted features. All proposed solutions are chosen to perform together at interactive rate.

1.4 Contributions

As a result of the work done as part of this thesis, we have:

- Developed the groundwork for a real-time optical hand posture recognition system, using a color-coded glove.
- Developed a method for specializing ANN systems to achieve better recognition and better training performance.
- Developed a 3D hand model that can be animated realistically, and a control framework to render various 2D views of this hand.
- Developed a real-time image segmentation algorithm for the purpose of detecting a number of specific uniformly-colored objects.

Future research, especially for achieving better segmentation of the color glove within the video frames, should allow us to complete this work into a usable system for VR interaction as envisioned by this thesis.

1.5 Thesis Outline

Following this introductory chapter, this thesis will be organized as follows:

- Chapter 2 reviews the main technologies used in this research, as well as some of the previous work done on this topic.
- Chapter 3 describes the various design issues and the options that this research had to consider, and the rationale behind its choices.
- Chapter 4 discusses the issues of proper and effective learning by the artificial neural networks and strategies employed to help this process.
- Chapter 5 provides an overview of the results achieved using the proposed methods and analyzes them.
- Chapter 6 concludes this thesis with a discussion on the results achieved and possible future research directions.

Chapter 2

Background

2.1 Immersive Virtual Reality

What makes someone perceive the three-dimensions of the real world is well summarized by Cruz-Neira et al [5]. It is the depth cues provided by:

«

1- Occlusion (hidden surface)

2- Perspective projection

3- Binocular disparity (stereo-glasses)

4- Motion parallax (head motion)

5- Convergence (amount eyes rotate toward center of interest, basically your optical range finder)

6- Accommodation (eye focus, like a single-lens reflex as range finder)

7- Atmospheric (fog)

8- Lighting and Shadows

»

All of 1, 2, 7 and 8 can easily be rendered with conventional computer graphics techniques. With the help of a head position and orientation tracking, 4 and 5 can be added. For binocular disparity (3) to be rendered on a projected surface requires clever physical manipulation of the imaging process. Typically, for quality rendering that does not affect the perception of colors, this can either be done by time-division or light-polarization multiplexing of the left-eye and right-eye images. Accommodation (6) cannot be represented in such a system, but "user learns to ignore the fact that everything is in focus" [5].

Immersive virtual reality systems could be described as systems that provide enough of the visual perception cues to make its users believe that the virtual material presented is part of their real spaces. Such systems for making users feel immersed in a virtual reality world have been around for sometime in various forms, the most popular and successful of which has been the CAVE system [5]. Until recently, because of the prohibitive cost of many required components, their accessibility has been limited to rich research facilities. This started to change recently with the advent of newer, more affordable computer and display technologies. The DIVINE system [6] is an example of a lower cost immersive VR system (designed by this author). Following are descriptions of these two systems.

2.1.1 CAVE

Immersive projection-based virtual reality setups like CAVE fill the field of view of the participants by projecting stereo imagery, rendered for one user's perspective, on large screens completely surrounding the users (Figure 2.1).

Figure 2.1 - A CAVE system

Screens are typically arranged at right angle to each other. A head tracking device provides position and orientation information of the user's head relative to the origin of the virtual world coordinate system. This information is used in the graphics rendering loop to give the sense of presence in the virtual world to the user. Stereo imaging is generally delivered to the user by help of some special glasses that filter out left and right eye views of the scene from the opposite eyes. Navigation and interaction within these systems is normally done with help of a special VR controller which is also tracked in space.

2.1.2 DIVINE

DIVINE is a personal desk-like virtual reality workspace that has "immersive" qualities similar to the afore-mentioned CAVE system. While its screens don't completely surround the participants in virtual imagery, it allows for collaborative manipulation and visualization of virtual objects that would normally be done in a desk-like setting in reality (Figure 2.2). Its operator is made to believe that the objects and scenes represented

in the virtual space are virtually present in the projected space. This is achieved by means of user-tracked stereo perspective rendering on orthogonally arranged projection screens, in the same way as in the CAVE. What sets DIVINE apart is mainly the increased

Figure 2.2 - DIVINE system in action

relative resolution of its screens since it is concentrated in this table workspace. At the same time, video cameras can be mounted near and at close to user's height for applications that makes use of video capture (i.e.: 3D video-conferencing, optical hand posture recognition, optical tracking, etc).

2.1.3 VR Controllers

While immersed in these systems, the users may want to control their virtual environment, such as navigating or modifying the objects within them. This is normally done by use of an electronic controller box, similar to a computer mouse, called a wand. This wand is tracked in 3D space. Some buttons on it allow for triggering interaction modes that are provided by the application such as moving around in the virtual world or interacting with objects.

It is this type of VR controller that a real-time optical hand posture recognition system such as the one envisioned in this thesis proposes to replace.

2.1.3.1 Instrumented Glove

Instrumented gloves are gloves with various types of sensors directly attached to them. They sense physical changes of some parts of interest on the hand. The information from the sensors is then interpreted to recover the current hand configuration.

Instrumented glove technologies have been available for some time. Nintendo released an instrumented glove for its gaming console back in 1989. This Power Glove, as they called it, provided basic 4-state bend detection for four of the fingers.

More recently, Immersion Corporation released a wireless version of its Cyberglove product [7]. This glove provides either 18 or 22 sensors with relatively good accuracy and fast data rate.

While the former is heavy, inflexible, cumbersome and no longer available, and the latter is very expensive, these and other similar instrumented glove devices could be used as interface for VR applications like the ones described in this research. Of course, other issues such as measurement precision and wiring constraints might also affect their effectiveness for this task, and in the end, a proper interpretation of the sensor data would still be necessary for posture recognition [8].

2.1.3.2 3D Tracking

Various form of 3D tracking sensors exist. Their purpose is to provide 3D information of the position and also usually the orientation of some point in reference to a base coordinate system. The device attached to the tracked point is called the tracker.

Inside Immersive VR systems, a tracker is attached to the head of the user so that the system can generate the graphical representation of the virtual world to the user's eyes' perspective. Another tracker is normally attached to a control interface device for navigation and interaction with the virtual environment.

Tracking can also be done by interpreting the position of some features inside video frames using one or more cameras. This is called optical tracking [9].

Optical position and orientation tracking of the color-glove used in this work, in addition to the proposed posture recognition, could also be possible, but is left for future research. We assume that the 3D position of the gloved hand would be tracked independently from our posture recognition system to complement the VR application control interface.

2.1.3.1 Multi-point Optical Hand Tracking

Multiple points can be tracked on a hand and then mapped, via specialized software, to a predefined 3D kinematic model. Commercial systems such as ones offered by Vicon [10] can do this with relatively high accuracy and at an interactive rate. They require using multiple specialized video cameras positioned appropriately. Special markers must be physically attached to the points on the hand to be tracked.

A pre-marked glove combined with such a system could be used as a VR controller like this research proposes, provided the posture recognition process is handled appropriately. This however might not prove to be a cost effective solution considering the relatively high price of these optical tracking systems.

2.2 Human Hand Physiology

The human hand is a complex system of five articulated fingers and a palm. It is composed of the skeletal structure pictured in Figure 2.3. This structure is highly malleable.

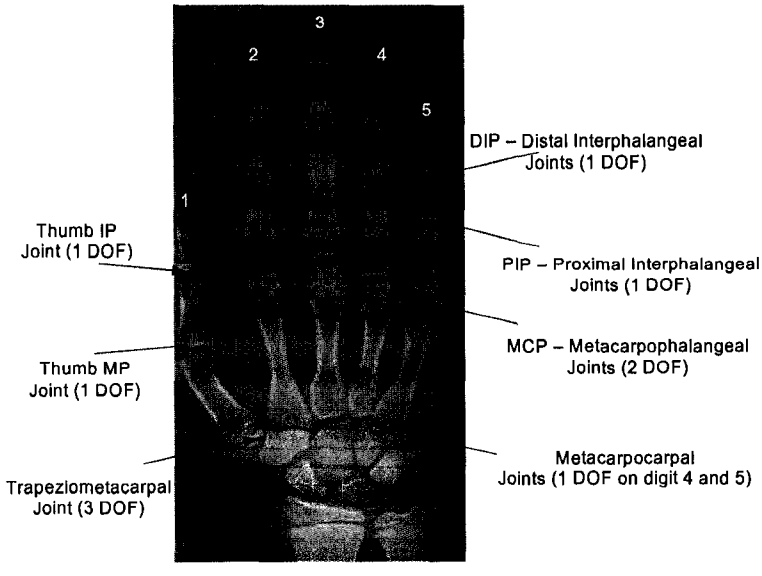

Figure 2.3 - Human Hand Skeletal Structure

Most hand joints can only provide rotation in one axis. This is the case for all Distal Interphalangeal (DIP) and Proximal Interphalangeal (PIP) joints, as well as for the Thumb Interphalangeal and Metacarpophalangeal joints and for the Metacarpocarpal joint of the ring and little finger. The Metacarpophalangeal (MCP) joints of all fingers (except the thumb which doesn't have such a joint) are articulated in 2 axes and the thumb Trapeziometacarpal joint is articulated in 3 axes. Overall, the hand can be said to have 23 articulations, or 23 degrees of freedom (DOF).

Further to the movements provided by this skeletal structure, muscles, skin and tendons also affect the visual appearance of the shape of the hand as it is animated.

2.3 Computer Animation

Computer animation software systems allow us to model objects and structures in some virtual space so that they can be rendered by the computer in various controllable ways (i.e. viewed through a virtual camera), and affected by diverse virtual environmental factors such as lights, fog, etc. They also permit to program how the objects, structures as well as the visualization parameters change or move over time in the scene, and how they interact with each other.

One popular software for 3D animation is Maya (by Autodesk). It provides capabilities for 3D modeling, keyframe animation, simulation of dynamic systems using skeletons, forward or reverse kinematics, deformations and rendering with various shading effect to name a few.

2.3.1 3D Modeling

3D modeling refers to the design of object representations in virtual 3D space. The geometry of the objects is described, as well as the material and properties of its parts. Several geometrical representations exist, the most popular one being the polygonal model. This model provides a good compromise between the quality of its representation, its scalability, flexibility, usability, and the computational complexity involved in using

and rendering them. Other models, such as NURBS provide smoother visual rendering, but usually at the cost of increased complexity.

2.3.2 Keyframe Animation

In computer animation, a frame refers to the representation of the models in the scene at a given time. When animating, we can simplify the description of the motion or change of shape of objects by only describing their position or shape at "key" frames. For the frames in-between keyframes, the objects are made to change position or shape according to mathematical functions that interpolate between these keyframes.

2.3.3 Dynamic Systems Animation

Objects can be defined to relate to other objects in ways that affect their movement and shape. For example, in a dynamic arm model, a hand object is related to a forearm object and will move in space in relation to the forearm's movements. The shape of objects can also change in relation to other objects. When the forearm is rotated around the elbow joint, the upper arm's shape changes as the bicep muscle contracts and itself changes shape. Several techniques for describing dynamic systems are usually available within computer animation software. Some of them, used in this research, are introduced in this section.

2.3.3.1 Skinning and Skeletons

Skinning is the process of defining a skeleton within a 3D object, or in other words, defining it as a skin attached to a bone structure. The surfaces and edges of the object are defined to be attached to the bones of this skeleton in some specific way. The skeleton

itself is defined as a set of bones and joints with relationships between each other. This skeleton does not necessarily have a visual representation for rendering purposes, but is used as a dynamic structure for the model's animation. The surfaces of the skinned object can be made to move or deform in various ways in relation to changes in its skeletal structure.

If we come back to our arm example of the previous section, the polygonal model of an arm (a single static object) can be skinned into a fully animation-enabled arm by defining a skeleton for it with a joint at its root (the shoulder), another one at the elbow and one at the wrist. We then can define how the skin bends at the elbow joint when it rotates, and how the forearm skin follows the movement of the bone it's attached to when this one rotates around the elbow.

2.3.3.2 **Forward and Inverse Kinematics**

In a linked system such as the skeleton described in the previous section, articulations and bones are defined to be in a parent-child tree relationship, starting at some root. Forward kinematics is the functional relationship that dictates how children nodes of such a linked system move in relation to their parents.

Inverse kinematics defines the opposite function, namely, how parent nodes move in relation to its children. This latter function is more complex as there can be multiple solutions or no solution at all for some children movement. These problems are usually solved by constraining the movements of the affected nodes.

These two functions on kinematic chains are normally available in computer animation software, and can greatly help the process of animation.

2.3.3.3 **Deformation**

Computer animation software normally provides the mechanism for animating dynamic deformation of objects. For instance, when our arm of the earlier example bends at the elbow, we might wish to have the upper arm change shape to simulate the effect of the bicep contracting.

Other type of deformation such as the effect of collision between objects, or the wiggling of skin in effect to gravity can sometimes also be simulated, depending on the software capabilities.

2.3.4 Rendering

Rendering, in computer animation, is the process of generating the 2D images representing a virtual camera's view of the modeled 3D scene at a given time. This rendering process can be affected by various environmental factors. Lights, material properties, camera properties, fog, artificial noise, and many other factors can all play a role in the rendering process, and affect the appearance of the objects in the scene.

2.4 Computer Vision

Computer vision is the field of computer science which is concerned with the conversion of digitized images into representations that the computer can use for obtaining information and for making autonomous decisions. One key process used in converting these images into higher-level representation is image segmentation.

2.4.1 Image Segmentation

Image segmentation is the process of separating parts of digital images into multiple regions for higher-level interpretation and analysis. Usual interpretation of segmented areas is that the pixels in a same segment are part of the same physical object, or of one of its visually notable component.

2.4.1.1 Threshold Color Filtering

If we know the approximate color values we expect for some uniformly-colored object we wish to locate, we can filter out all the pixels that are not close to these values (difference above the threshold), and obtain the location of pixels that should contain the object. This is the basic principle for threshold color filtering methods applied to segmentation. Different distance metrics can be used to determine this relative closeness of color pairs, the most common of which is the Euclidian distance of the color points in its color space. In this case, the segmentation process is described by:

$$T_k(i,j) = \begin{cases} 0, & \left|\vec{P}(i,j) - \vec{C}_k\right| > Threshold(\vec{C}_k) \\ 1, & \left|\vec{P}(i,j) - \vec{C}_k\right| \le Threshold(\vec{C}_k) \end{cases}$$

where $T_k(i,j)$ is the binary image resulting from the segmentation for the color class k, $\vec{P}(i,j)$ is the color vector at pixel location (i,j) in the image we are segmenting from, $\vec{C}_k$ is the color descriptor for class k.

Sometimes, changing the color space model of the image helps to better delimit the objects we look to segment. For example, using "rg-chromaticity" color space transformation (a.k.a. normalized RGB), the color values of the object's pixels can become less dependent on the lighting conditions.

rg-chromaticity's components are related to RGB by the following:

$$r = \frac{R}{(R+G+B)} \quad g = \frac{G}{(R+G+B)} \quad b = \frac{B}{(R+G+B)}$$

Since the b component value is equivalent to simply $1-(r+g)$, it is usually left out in this color representation.

There are other popular color spaces which propose similar independence from lighting conditions such as HSV or HSL. In these specific spaces, using Euclidian distance is troublesome since one component is cyclic. Other spaces, like CIE L*a*b*, have a more complex (and lengthy) conversion procedure from its RGB counterpart, and might not be appropriate for real-time computer vision processes.

A combination of color threshold filtering on multiple color spaces can be used to increase the precision of the segmentation, in detriment to an increased computational load.

2.5 Digital Camera Technologies

2.5.1 Image Sensor

Part of every digital camera, the image sensor converts light intensity at multiple points of the sensor (pixels) into an electrical signal which is ultimately interpreted by other electronic components into digital image information.

The most popular image sensor technologies (CMOS, CCD) used in common digital video camera don't detect color information directly, but instead rely on color filters aligned on top of the sensor to infer color. The usual approach is to use a Bayer filter.

2.5.2 Bayer Filter

This filter is composed of a mosaic of red, green and blue pixel size color filters, organized in a grid in a specific arrangement as pictured in Figure 2.4.

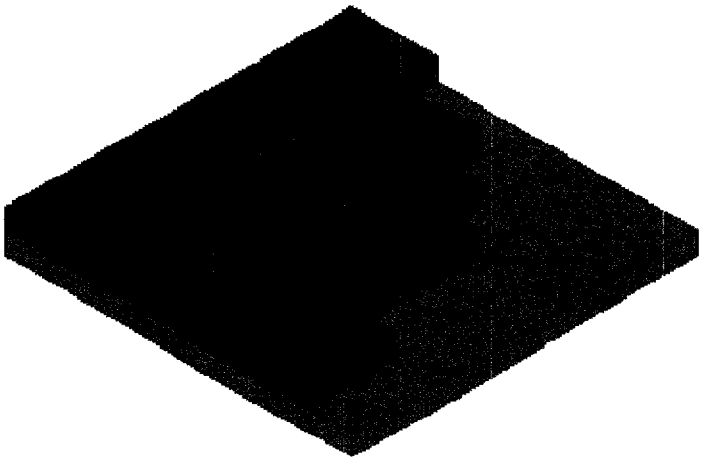

Figure 2.4 - Bayer filter's arrangement of individual color pixel filters over an image sensor array.

The number of green filter pixels is twice that of the other two individual color components (50% green, 25% red and 25% blue). This extra number of green pixels gives the underlying sensor a greater overall sensitivity to green light and therefore to luminance, like the human eye does.

The resulting raw image from a sensor using this Bayer filter must be processed by a so-called demosaicing algorithm. This algorithm interpolates the proper color of the image pixels from the raw color-filtered mosaic.

2.5.3 Lens

The lens of a camera is used to focus light coming from its field of view onto the image sensor. The image produced on the sensor is usually affected by some form of radial distortion caused by the shape of the lens. This distortion can be somewhat corrected by software given some pre-measurement of the lens' projective geometry.

2.6 Artificial Neural Networks

Artificial neural networks (ANNs) are computational models inspired from the biological neural networks of the human brain. It is composed of elementary computation units, named neurons (after their biological counterparts), which are interconnected in ways to try to approximate some useful function. Its computation is highly parallel, each element operating only on its own (local) inputs. Normally initialized at random, the parameters of its elements are tuned by use of special learning algorithms in order to teach the complete network to achieve its goals as best as it can.

There are two main families of ANNs, distinguished by their learning style. One family learns in a supervised way, by being shown input-output relationships. The other family learns unsupervised – it learns to organize its input space. The internal organization of the network is the ultimate interest in this family of ANNs.

In this research, we are only concerned with the first family of networks using supervised learning algorithms.

2.6.1 Artificial neurons

Individual neurons are modeled as pictured below (Figure 2.5):

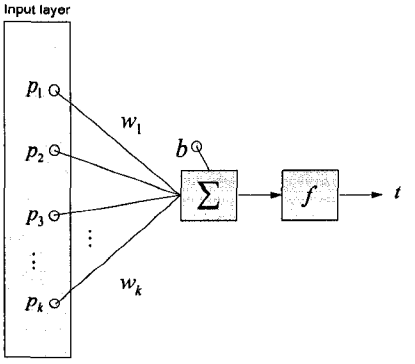

Figure 2.5 - Artificial neuron model

Its output t is computed by

$$t = f\left(W\vec{p} + b\right)$$

The activation function f can take many forms, as long as it is differentiable (this restriction comes from the learning algorithms used). Transfer functions used in this research (as defined by Matlab [11]) are the following:

purelin

The purelin function is computed by

$$f(x) = x$$

This transfer function (equivalent to not having a transfer function at all), is normally used in the output layer. It allows the network to generate continuous-valued outputs with no known bounds.

tansig

The tansig function is computed by

$$f(x) = \frac{2}{(1 + e^{-2x})} - 1$$

Its response curve is pictured below (Figure 2.6).

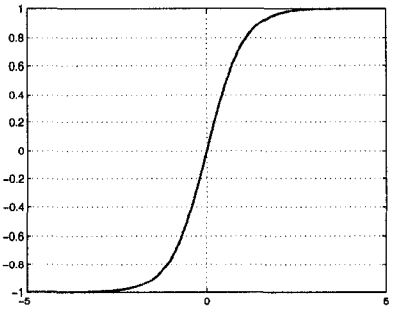

Figure 2.6 - Tansig activation function

It is said to be numerically equivalent to the hyperbolic tangent function *tanh*, but faster to compute.

2.6.2 Feed-forward Neural Networks

Feed-forward neural networks are a generic class of networks where neuron inter-connections don't form cycles.

2.6.3 Multi-Layer Perceptron

One of the most successful ANN architecture is the multi-layer perceptron (MLP). It is a feed-forward neural network composed of one input layer, one or more hidden layer(s) with sigmoid transfer functions (*tansig* in our case), and one output layer (possibly using a linear transfer function). Adjacent layers are fully inter-connected.

It is said that MLPs can learn to approximate any continuous function $f : R^n \rightarrow R^m$ with just one hidden layer using a sigmoid activation function. Although it might seem unnecessary, having more than one hidden layer can affect how fast the function can be learnt.

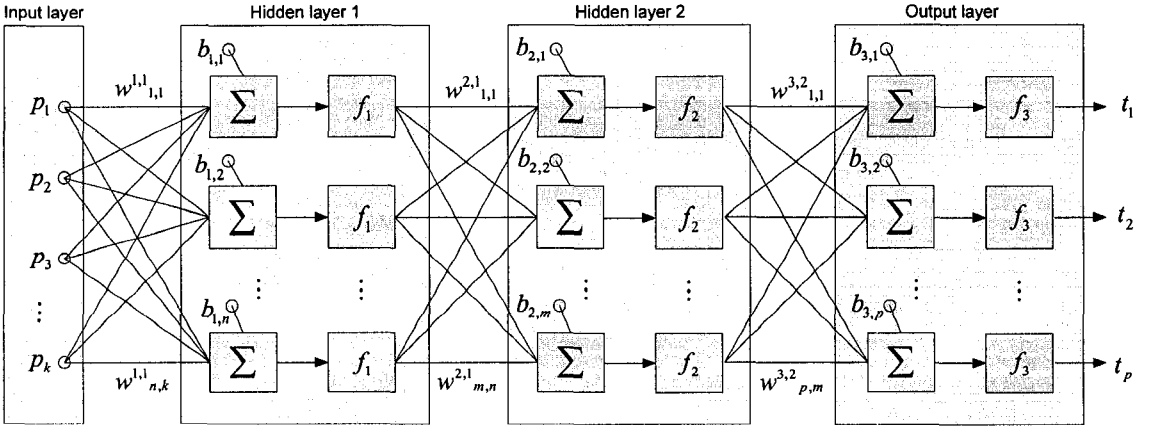

Figure 2.7 - Architecture of a two hidden-layers MLP network

For a two hidden layers network such as the one pictured in Figure 2.7, the outputs are computed by:

$$\vec{t} = f_3\left(W^{3,2} f_2\left(W^{2,1} f_1\left(W^{1,1}\vec{p} + \vec{b}_1\right) + \vec{b}_2\right) + \vec{b}_3\right)$$

The outputs of MLPs of other architectures are similarly derived.

2.6.4 Feed-forward Neural Networks Training

Training of a feed-forward neural network consists in iteratively tuning its weights and biases based on its response to the training samples. It is done by a process called back-propagation, which performs computation of the performance gradient (based on mean square error) by stepping backwards through the network. In this research, training of the ANNs was done using batch mode algorithms which require that the complete training set of samples be processed at each epoch (iteration of the training process). Specifically, we've used the MATLAB implementation of the Levenberg-Marquardt (trainlm) and the scaled conjugate gradient algorithms (trainscg).

2.7 Previous Work

There has been extensive research done on the topic of posture recognition. Many of them use an instrumented glove to acquire its data. Of the ones that propose to recognize postures using computer vision technology, few allow the degree of liberty in hand

orientation required by our application. Following is a brief overview of some related research that would offer the required hand orientation freedom.

2.7.1 Brief Overview of Related Research

In [12, 13, 14], a color-coded glove is used to track a hand's finger configuration and resorts to inter-frame tracking and model fitting to recover joint angles on a kinematic hand model. With this tracking strategy, an initialization step is required since it is assumed that the hand configuration from the previous frame is known and correct. The allowed orientation of the hand relative to the camera is usually flexible when using this approach. The higher level task of classification of the hand posture can be somewhat simplified by working directly with 3D hand features, but it is a process which must be handled separately.

Other approaches consider selecting closest match by database indexing [15, 16, 17], and some combining it with a form of temporal tracking assumption for speeding up the record retrieval process [18, 19], all of these using silhouette contour based features. When using this approach, features from the hand image to be recognized are extracted and then compared with those of each image in a set of reference (key) images, calculating distance is some metric and selecting the nearest one. This process is cascaded to finally decide on the closest matching hand configuration. With temporal information taken into consideration, it is initially assumed that the reference key of the first level indexing is the same one as for the previous frame, speeding up the recognition process if this assumption proves right.

Also using silhouette based features, and a maximum a posteriori framework to classify a set of 15 postures, multiple cameras are used in [20]. An active contour model based on level sets evolves the hand silhouette's curve. This curve is then used to derive a shape context descriptor which is compared to all possible cases in the synthesized dataset of detectable hand postures and evaluate their likelihood. They achieve 81.5% recognition in their best case when employing two cameras.

In [21, 22], moment invariants of the hand silhouettes are used as features in their specialized mappings architecture. Following this architecture, they segment the input feature space of the posture estimation framework by way of the expectation maximization algorithm. Artificial neural networks are then employed on the input space segments for posture estimation. The framework formulates multiple hypotheses. A map-back function verifies the likelihood of these hypotheses.

Chapter 3

Recognizing the Hand

3.1 Features for Recognition

From the direct observation of a hand in any posture, it appears that humans can make a pretty good guess of the configuration of all its fingers. Stereo-vision seems to have little effect on this capacity. We can distinguish the observable features, and make-up for missing visual information with our knowledge of the hand's morphology.

Directly assessable features for human recognition comprise:

- fingernails location
- skin creases location
- fingers length
- fingers bend
- global hand shape derived from detection of skin color

All these features are then matched to our prior knowledge of "possible" hand shapes for validation, and disambiguation [23].

In this research, the process of visual recognition – to be done by a computer, not a human - is somewhat simplified. The intention here is to provide visual features that directly relate to each finger individually. The higher-level task of the recognition of each finger then becomes unnecessary. This takes out a lot of the guesswork that humans can do, but proves difficult for computers.

Our strategy for helping this recognition consists in completely covering each finger and the palm by a unique uniform color (approach which we've later found was similarly used by Lamar [24] and also to some extent by Bebis et al [25]). These colors would be chosen as to leave the greatest possible distance between themselves in the RGB color space, excluding black and white. These colors form the set of the following RGB relative values: {(0,0,1),(0,1,0),(1,0,0),(1,0,1),(1,1,0),(0,1,1)}. This set corresponds to colors we would label as {"blue","green","red","magenta","yellow","cyan"} respectively.

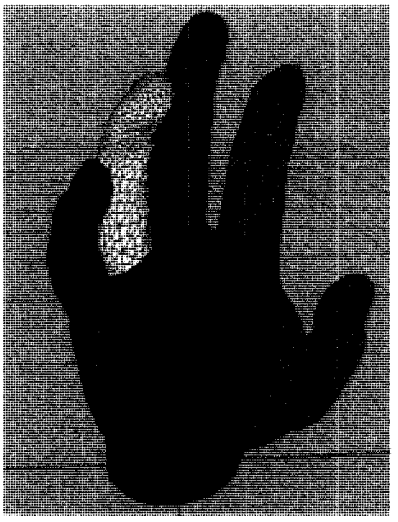

Figure 3.1 - Choice of color-coded visual features

A glove providing this color-coding is to be devised, and would be worn over the hand to be recognized by the system. Since digital cameras typically capture colors by using a RGB Bayer filter [26] over a monochromatic sensor array, using these colors on the fingers is believed to be the optimal way of separately detecting each finger and palm. Also, by using extreme color values, filtering out other objects and the background should also be simplified.

3.2 Recognition Process

Now that we have conceptualized this glove to help our recognition task, we need to decide what exactly we wish to recognize from the view of this gloved hand. One option is to find approximations for the rotation angles of all the articulations involved in the viewed posture of the hand. This corresponds to the regression task of finding the function approximation $f^* : R^n \rightarrow R^m$ which models the relationship, where n is the dimension of our input feature vector (extracted from the input image of the viewed hand-in-glove), and m is the DOF of our hand model +2 viewpoint orientation axis.

Another option is to categorize a set of postures that we wish to recognized, and classify the viewed hand as belonging to a specific class. This in turn corresponds to the task of finding the function $f^* : R^n \rightarrow \{a_1,...,a_m\}$ which models the classification relationship between the input feature vector and the set of m class labels.

We will discuss the two approaches in the following sections, and explain how we experimented with each one in this research.

The choice of using artificial neural networks for our machine learning method was determined mainly because of its proven ability to learn by example, to properly generalize solutions and for its simple and speedy computation once defined. Other machine learning techniques could be explored but is left as future research.

3.2.1 Regression

In this research, we jumped early-on into experimenting with this ANN regression problem without much consideration. We were quickly shown by the results that a more poised analysis and maybe a different approach (classification) might be in order.

For neural networks to be able to do appropriate regression on a function, enough examples must be provided, for every independent dimension of its outputs, sampled regularly over their complete ranges. Indeed, a neural network can only learn in the ranges that it is shown, and failure to cover the output range will lead to unpredictable results in the uncovered or under-sampled areas.

For our hand articulations angles estimation task, it means that we need examples of the hand viewed with its every articulations varied over their entire range of motion, for every configuration (i.e. for every set of other articulation values that are sampled), as well as for every viewpoint orientation. This is clearly an overwhelming number of samples to deal with.

If we let every 23 articulation angles (for our 23 DOF hand model) vary between 3 values: both the extreme of its range, and a middle value; the number of samples needed would be 3^{23}. Adding variations of the view orientation in steps of 15° around both the X and Y axes, the number of these samples then becomes approximately 2.4×10^{13}. And this

is only the minimum number of samples to ensure range coverage in the training set. We would need another validation set of examples of maybe one or two thirds this size to verify that the trained network did learn what was intended.

Simplification of the problem is possible. Indeed, we don't need to model the problem as a single function of 25 dimensions. Some segmentation of the function is possible. Furthermore a few finger articulations are somewhat dependent on others [27]. Also, some articulations have a very narrow range of rotational freedom and could be ignored.

In our case, we have tried to simply learn what could be learnt from a fixed length animation sequence of a hand dynamically changing posture. The postures and ranges of motion shown were somewhat arbitrary, but the animation was modeled to move in a natural way by mapping it to a real-life video capture of a hand changing posture (method to be described in more details in the following sections). This was done in an attempt to also represent the constraints of the motions of the human hand.

ANN learning issues of this approach will be further discussed in the next chapter, and its results in chapter 5.

However, for the purpose of providing an interface for immersive VR systems, it might be more appropriate to do classification of our visual features directly. Indeed, the posture's articulation angles that we would obtain by regression would still need to be classified for detecting commands.

3.2.2 Classification

The problem of classification is a bit more flexible in that not all articulation angles need to be exemplified. Difference between the classes is what is learnt by the ANN.

One issue with this approach is that the output space is divided by as many classes as there are to be classified. To teach a network to detect a specific posture, it must also be able to detect what is NOT this posture. This implies that the "non-posture" space must also be defined and exemplified in the training of the network. Luckily, with a large enough number of classes, we can simplify this non-posture space as the union of all other class spaces.

In this research, we've experimented with the classification of 20 different postures chosen more or less arbitrarily, but varied enough so that we can get an idea of the kind of differences that can be learnt (Figure 3.2).

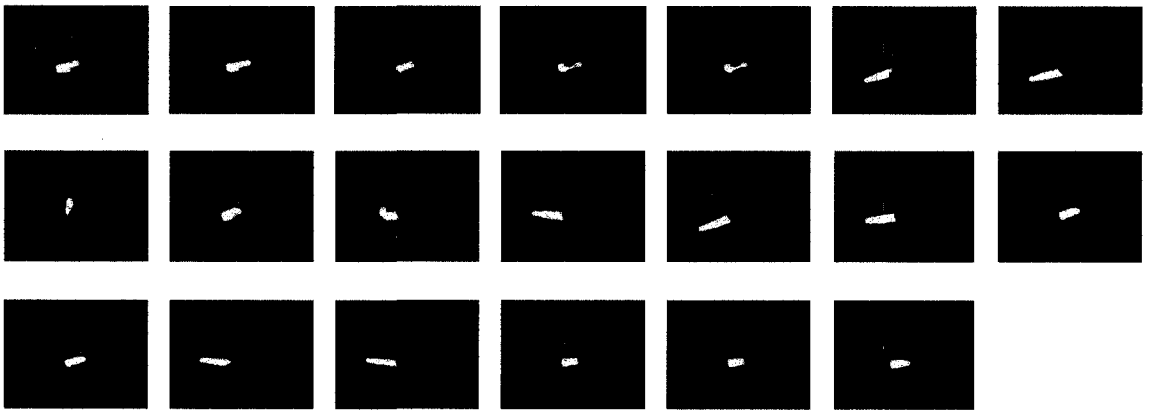

Figure 3.2 - The front view of then 20 hand postures chosen for classification

Other classification experiments were conducted using frames of a real segmented video sequence of a hand-in-color-glove, to classify the same 20 postures.

3.3 Providing Learning Material

In this research, in both cases of regression and classification, we wish to use a supervised artificial neural network machine learning approach which requires the use of many examples for teaching it our inputs-outputs relationships. We therefore need to find a way to generate the many examples.

One approach which was considered consisted in taking measurements from an instrumented glove such as the Cyberglove with a color glove fitted over it. Optical features could have been extracted in the same way as intended in the final application: by capturing using a camera, segmenting the color areas and measuring our set of features. Corresponding articulation angles would be available from the instrumented glove.

However, this approach was discarded for the following reasons:

- The Cyberglove is too bulky for our application. It affects the appearance of the hand wearing the color-glove.

- The Cyberglove is relatively imprecise and uses a simplified hand model (although the manufacturer claims high precision for their glove's sensors, we find that other factors, such as the rigidity of the material, leads to a much lesser precision).

- We would need both a color glove and the proper image segmentation routine before validating our regression or classification methods (which we want to validate first).

- It would be very time consuming (and hard to control) to generate a dataset for training and testing of our neural networks using this method.

Simulating views of a hand in glove using computer animation software was found to be a much more suitable method for our needs. The degree of control of all the posture parameters and the virtual camera is absolute, but all the factors affecting the hand's appearance must be properly modeled.

3.4 Hand Model and Animation

The anthropometric properties as well as the animation capabilities of the simulated hand representation should be carefully modeled to correspond closely to that of a real hand in order to get a realistic mapping of our input domain to the hand's appearance features. This section describes the techniques used for creating such a hand model.

3.4.1 3D Hand Model

A generic polygonal right hand model was extracted and imported from a license-free 3D polygonal model library of humans. The model was then modified to better correspond to the hand of the author (Figure 3.3), which would be used later on for real image testing purposes.

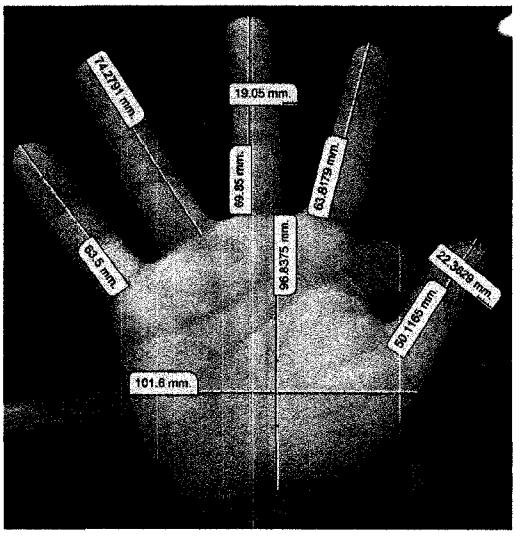

Figure 3.3 - Example of the anthropometric measurements of a hand

Also, the texture of the fingers and the palm had to be modified to represent the hand wearing a color glove. To do so, some edges of the polygonal model had to be changed or added to properly represent the structure of the glove, with the colored finger parts ending where the real glove's would (see Figure 3.4).

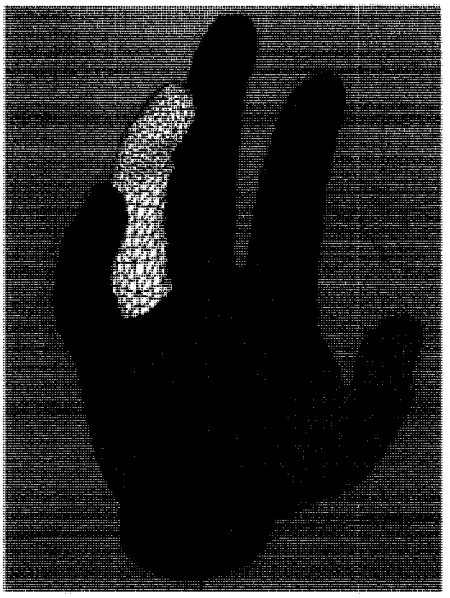

Figure 3.4 - 3D Polygonal Hand Model

Skinning was done to define an appropriate skeleton that would correspond to a real hand structure. Location of the articulations had to be carefully selected and their rotational axes and constraints properly set.

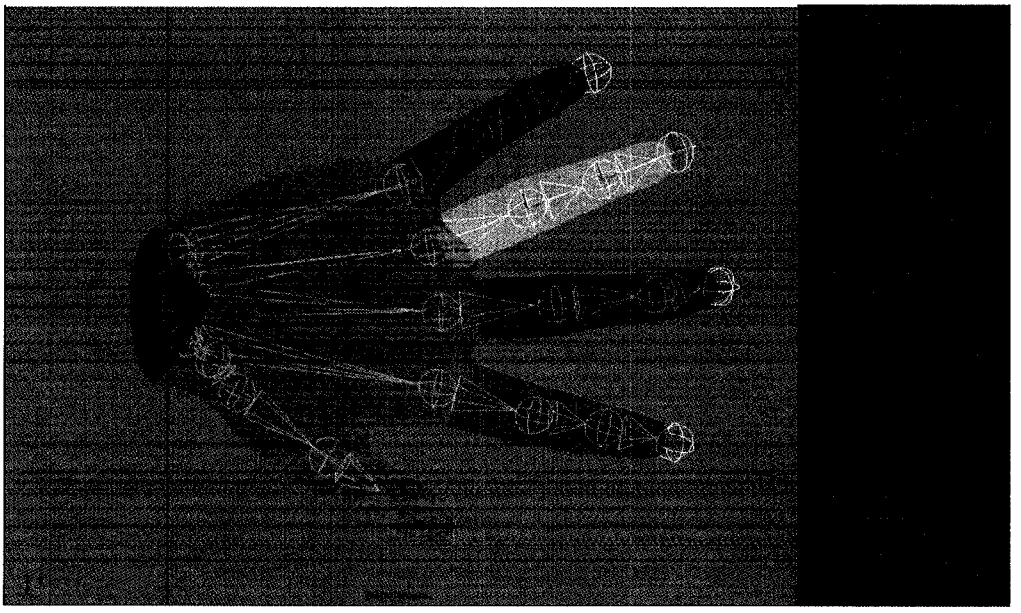

Figure 3.5 - Hand Model Skeletal Structure

Extra care was needed in the definition of how the skin is affected by the rotation of the joints. Influence weights were assigned to the appropriate vertices so as to properly represent the shape at the joints when bending fingers. Influence objects in the form of ovoids were added inside the model (Figure 3.6) and programmed to change size with respect to the rotation of some articulations. This in turn affects the shape of the hand's skin and serves as a good representation of the effect of muscles inside the hand.

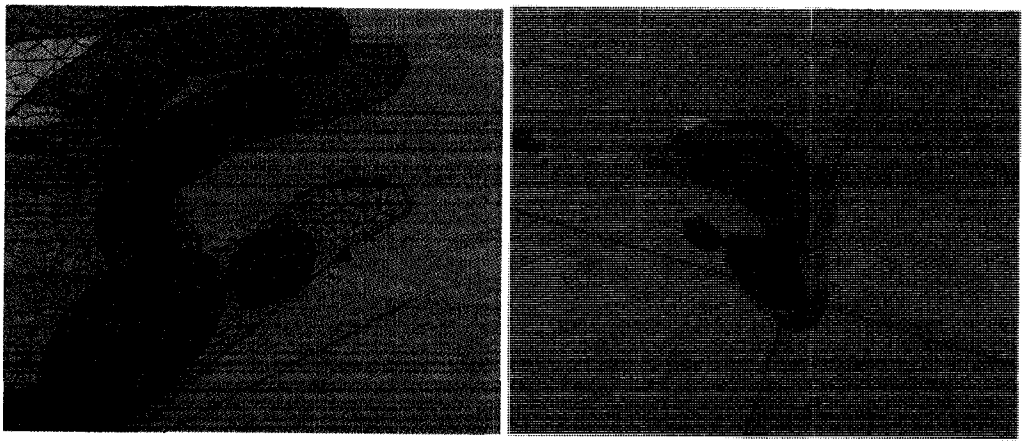

Figure 3.6 - Influence objects modeling muscle contraction

3.4.2 Animation of the Hand

Once the hand model is properly defined, it can be animated. In order to provide learning material for the regression task explained earlier, a video sequence of the author's right hand was captured by two synchronized cameras, positioned approximately perpendicular to each other, to the front and the right side of the hand (Figure 3.7).

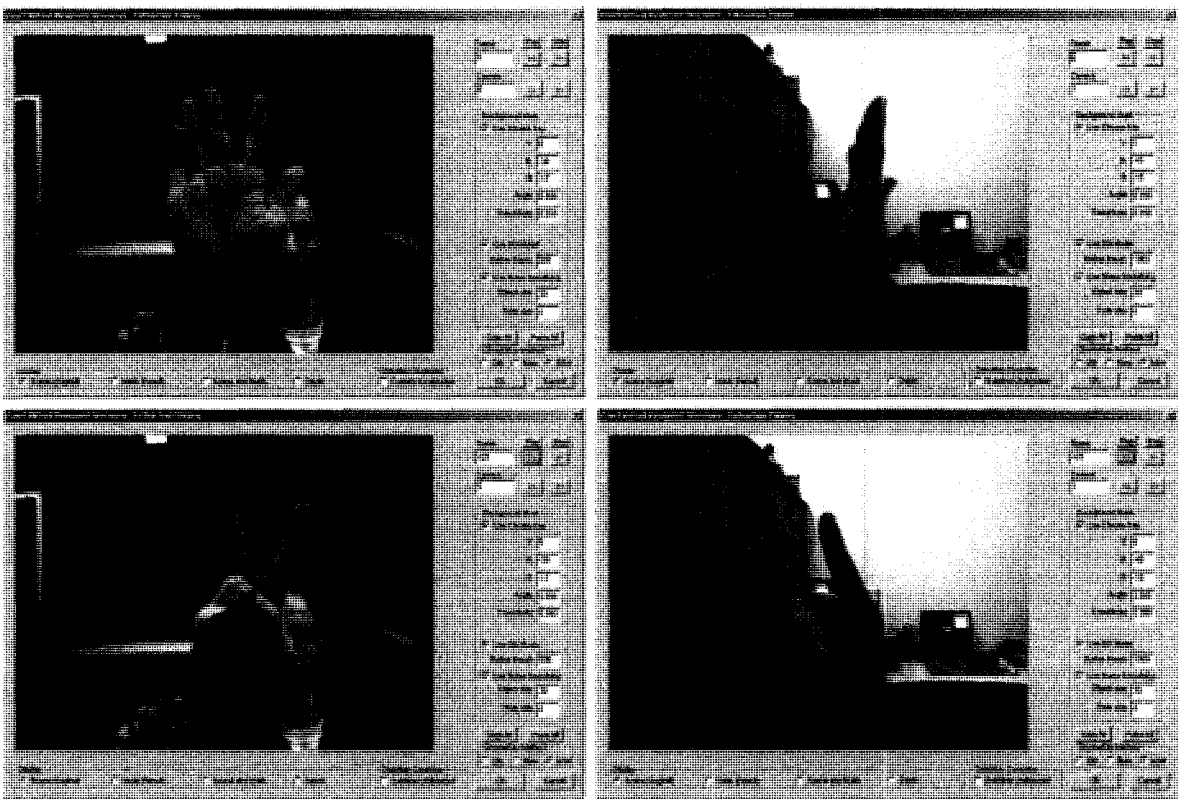

Figure 3.7 - Multi-view Hand Capture

Inside the 3D computer animation software's modeling space, two planes were defined with animated textures corresponding to the captured video frames of the synchronized cameras. These planes were positioned and scaled so that the hand model's view would map approximatly one-to-one with the hand pictured in the video planes in some virtual camera view. The textured frame sequence was programmed to follow the 3D animation frames sequence. The articulations of the hand model were changed to match the video at "key" frames, and the rest of the animation's frames were generated by interpolation, but monitored to closely follow the source video. When the interpolation wasn't following

the video properly, new keyframes were introduced. This process is pictured in Figure 3.8 for one of the keyframes.

Figure 3.8 - Hand animation overlaid on multi-view captured video planes

For describing the 20 different postures of our classification problem, some of the frames in this video sequence were used. For some other postures, the articulations where simply modified to set the hand in the required static posture without the visual aid of the synchronized videos, only relying on the hand model's interactive rendering views.

3.5 Color Glove

In the end, our system expects to do its optical posture recognition on a hand anthropometrically similar to the modeled hand. This hand will be wearing a specific glove, with a color-coding as previously described. Furthermore, this glove should be a tight fit for the hand to be recognized and should be flexible. Ideally, it should not introduce bulges and creases when worn on the moving hand, as these anomalies were not taken into consideration in the model animation used in the teaching of our system.

Other properties of this glove could be taken into consideration, such as the reflectance properties of the material and its thermal properties, but these were not considered crucial to mull over for our immediate needs.

For our testing purposes, we settled for a simple white cotton glove which we could color using fabric dyes. Of course the conception of this experimental color-glove was not without its difficulties in our inexperienced textile dyeing hands (color bleeding between regions, non-uniform coverage, non-optimal color choices and dye mixes, etc). However, by dyeing individual fingers and gluing them together, we arrived at a somewhat usable color-glove that we could employ for our real-image testing (Figure 3.9).

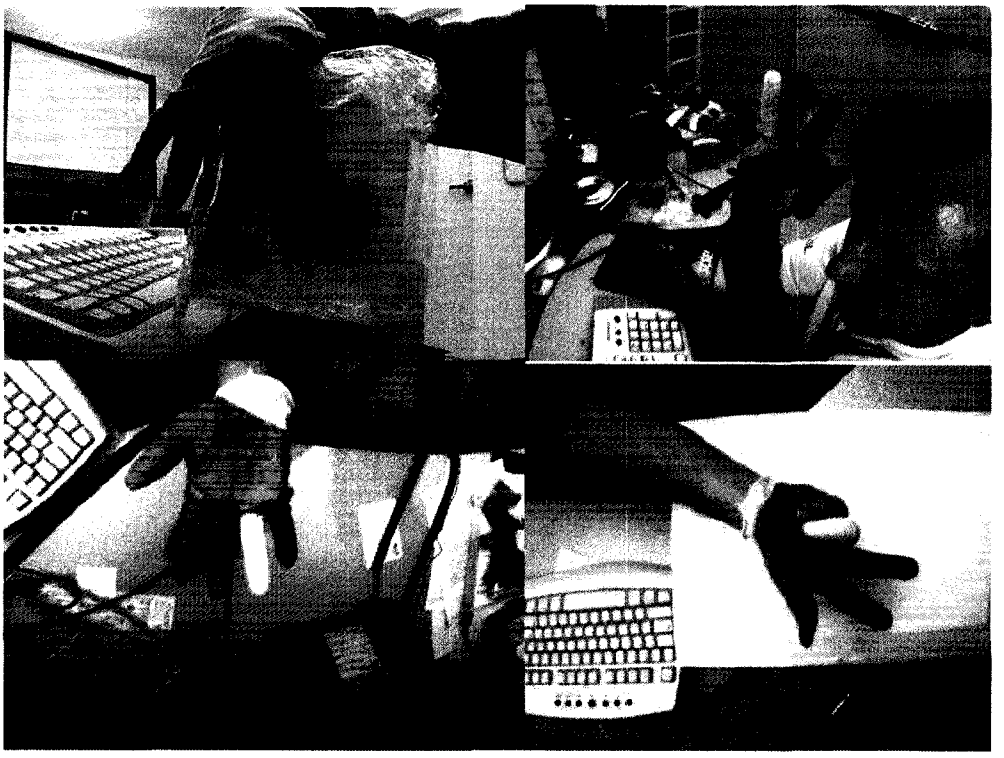

Figure 3.9 - Making of an experimental color glove

3.6 Image Segmentation

3.6.1 Color Difference Threshold Segmentation with Regional Boosting

Although this work does not focus its attention on the segmentation process itself, it is expected that by using color difference techniques [28] on a set of optimally separated colors in addition to background subtraction, we can obtain adequate and rapid segmentation of the hand and its fingers in arbitrary scenes. This might not always be the case. In most of our tests, we use a white background to ease this task. The colors used on our prototype glove are also not optimal, but sufficient for our demonstration needs. We increase the performance of the segmentation by using regional boosting, where an overlapping mosaic of pixel blocks provide information on regions of similar colors.

As exemplified in Figure 3.10, two matrices are computed for a target segmentation color, counting the number of pixels that would pass a narrow threshold filtering step in the immediate region.

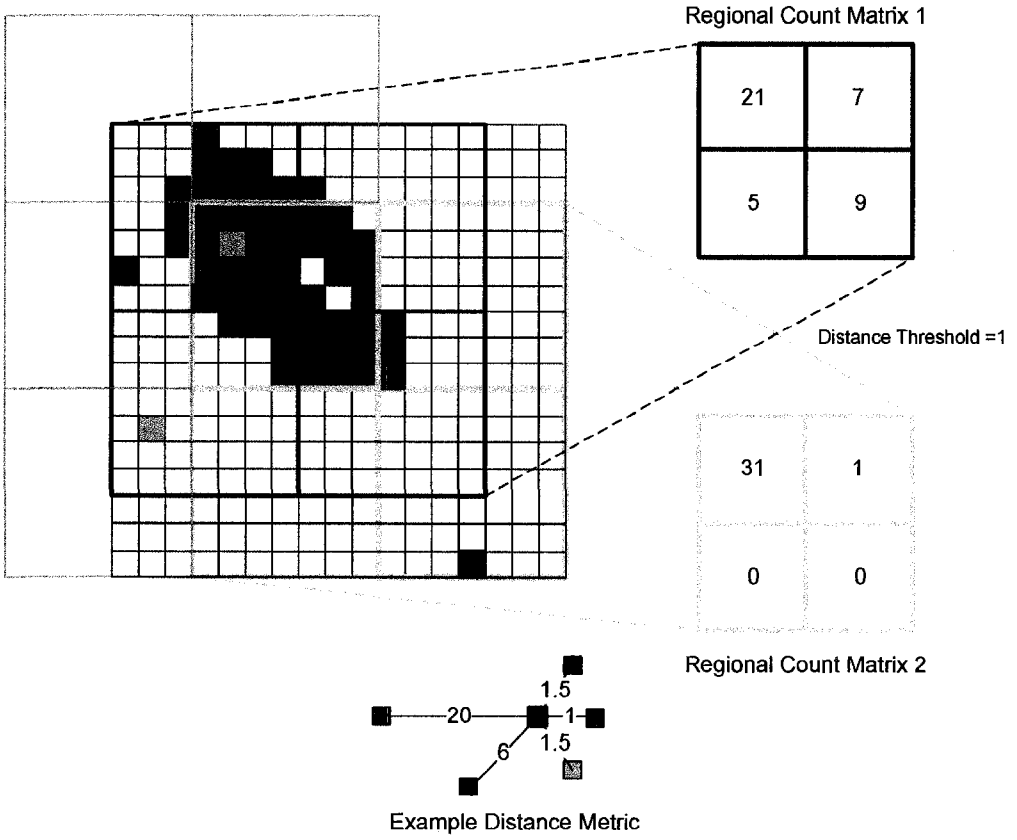

Figure 3.10 - Segmentation regional boosting process example (regional count)

A boosting factor is computed based on both regional count matrices for each intersection of the regions. In our experiments, we use the following boosting factor at pixel location (i, j):

$$boost(i,j) = \frac{regionCount_1(i,j)^2 + regionCount_2(i,j)^2}{\left\lceil \frac{regionSize}{2} \right\rceil^2}$$

Our final segmentation step is another color threshold filtering using a broader threshold value weighted by this boost factor (Figure 3.11).

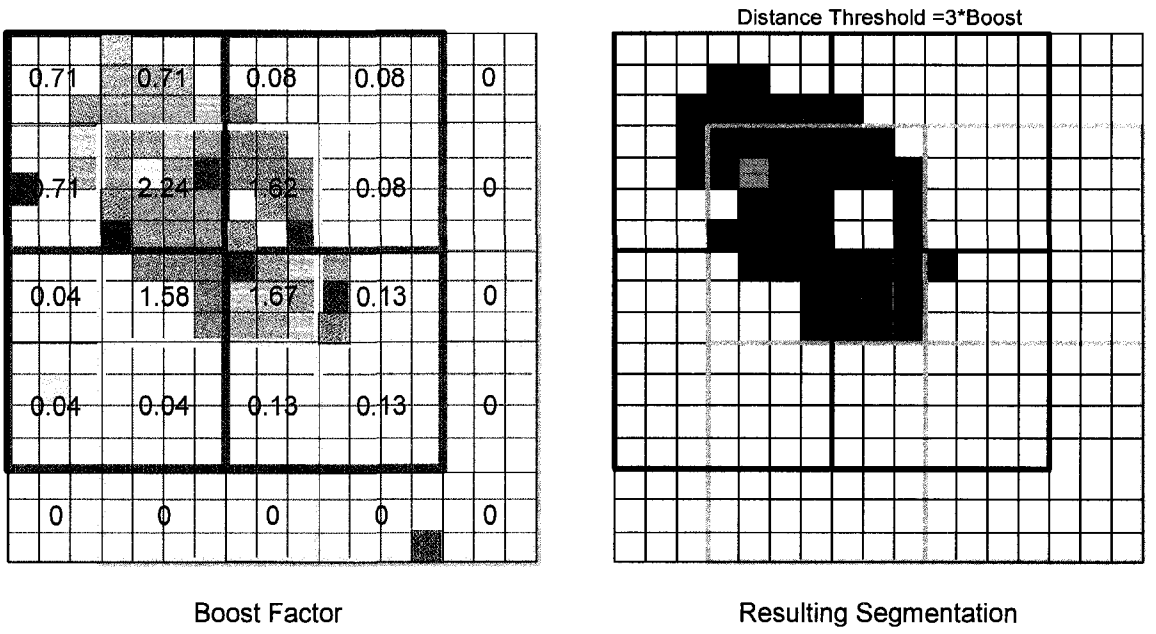

Figure 3.11 - Segmentation regional boosting process example (boosting and result)

Results of the segmentation process on a sample video frame when using this algorithm is shown below (Figure 3.12).

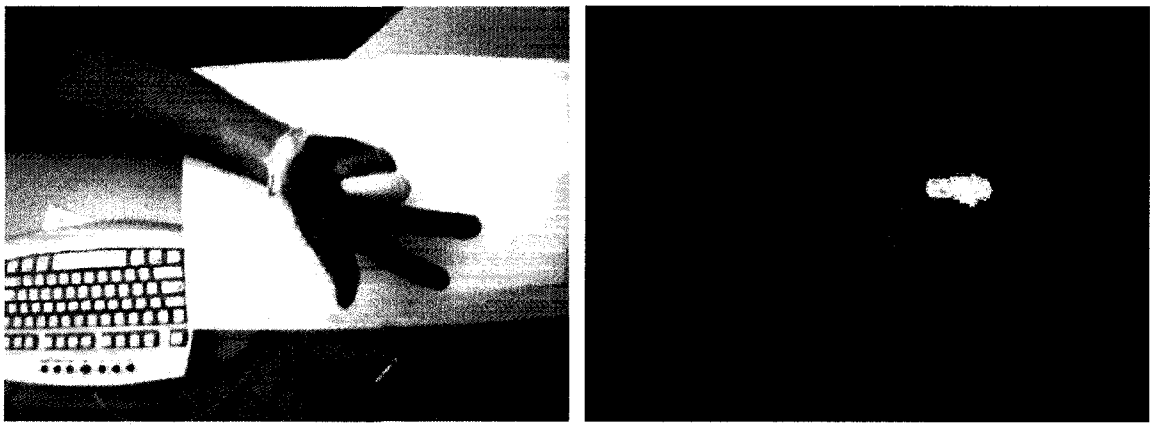

Figure 3.12 - Experimental color-coded glove in a video frame, and the corresponding segmentation

Distance metric used in our case is the number of standard deviation distance from the mean of a pre-sampled color region (for a specific segmentation class), for each color component of both RGB and rg-chromaticity color spaces. Threshold segmentation is computed for every color component separately and later combined by a logical AND operation.

More sophisticate segmentation techniques could also be employed, such as taking into consideration the spatial inter-relationship of the color patches, or by tracking the blobs across video frames and adapting color metrics, but these are left as future research.

3.6.2 Feature Vectors Extraction

When considering the features that can be used for our purpose, we quickly realize the need for them to be invariant to rotation in the plane of the video frame and to scaling. We use the following set of such features:

- Pixel area of each color blob, relative to the pixel area of the whole segmented hand object.
- The centroid location of each color blob, relative to the centroid of the whole hand object, with the hand's major axis realigned to be horizontal (major axis of the ellipse equivalent to its second moments), and all measurements scaled relative to its length.
- Orientation of the major axis of each blob, on the realigned hand object

These features are depicted in Figure 3.13 below. Extraction of these features is of $O(n)$ complexity for a n pixel segmented hand object and involve calculating image central moments of order up to 2.

Other feature data could be possible, and might be investigated in future research. Directly using the normalized central moments would be one valid option.

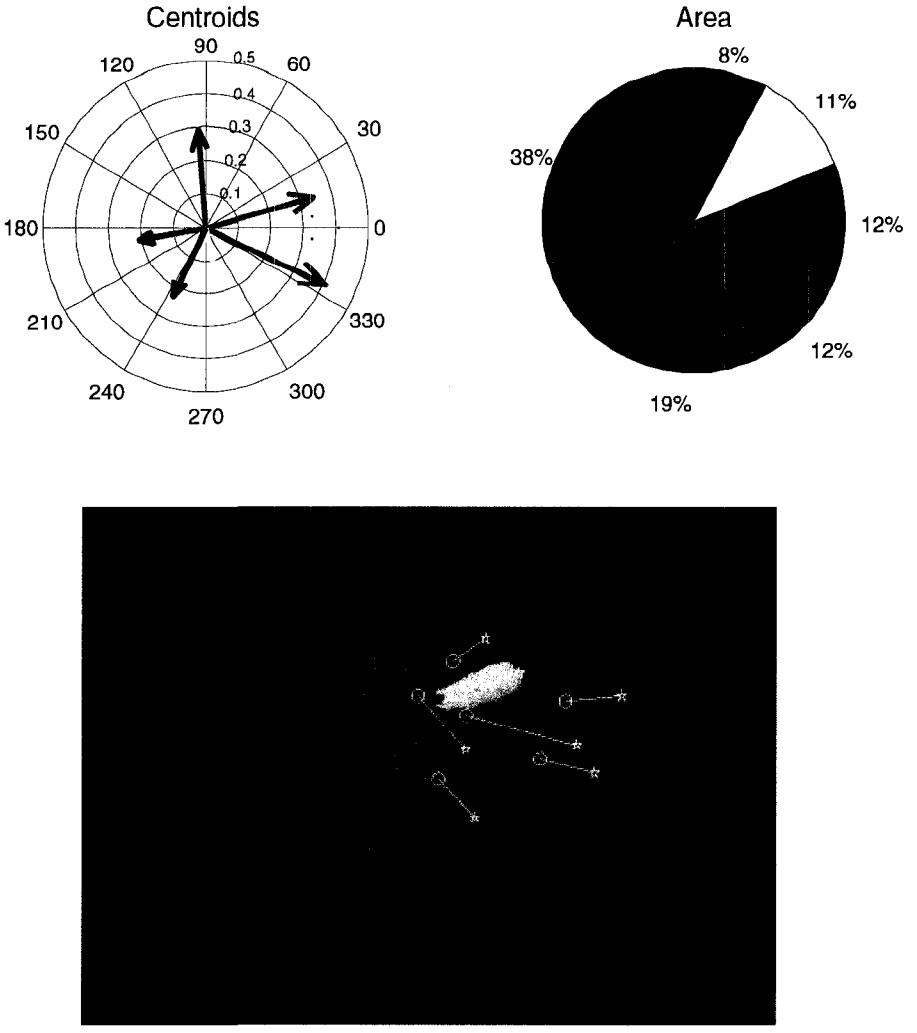

Figure 3.13 - Normalized centroids, area features and orientation (overlaid on image)

A key issue when considering features for using in ANN systems is that they should be constant in numbers. Lists of contour edges and such are therefore unusable, at least not in a straightforward way. Another consideration is that for continuous changes in the hand configuration or orientation, they must change in a continuous way for most of the output range. That is, the chosen visual features must somehow follow the dynamics of the 3D system they model. Furthermore, they should not allow many ambiguities, having the same features for different hand configurations.

Chapter 4

Artificial Neural Network Learning

4.1 Regression

In our experiments with learning the regression tasks described in the previous chapter, we have first tried to train one function approximation per hand articulation, and one per viewpoint orientation axis, using MLPs.

Our ANN training material consisted in the visual features extracted from the rendering, at different viewpoint orientation, of our hand-in-glove model, animated to follow a real video sequence of a moving hand as described in section 3.4.2. All articulation angles and viewpoint orientation parameters where extracted for every frame of the animation sequence and used as the corresponding outputs.

4.1.1 System Overview

The overall architecture of the system is depicted in Figure 4.1 below. A camera input will ultimately feed the posture recognition system. After initialization (color calibration and hand model selection), a segmented output gets extracted for each camera frame,

then processed to recover feature vector for recognition. For training of the system, the rendering of various views of an animated 3D hand model is used. The system outputs an estimate of the angle of rotation of every hand articulation, as well as the X and Y rotation angle of the whole hand (equivalent to the viewpoint).

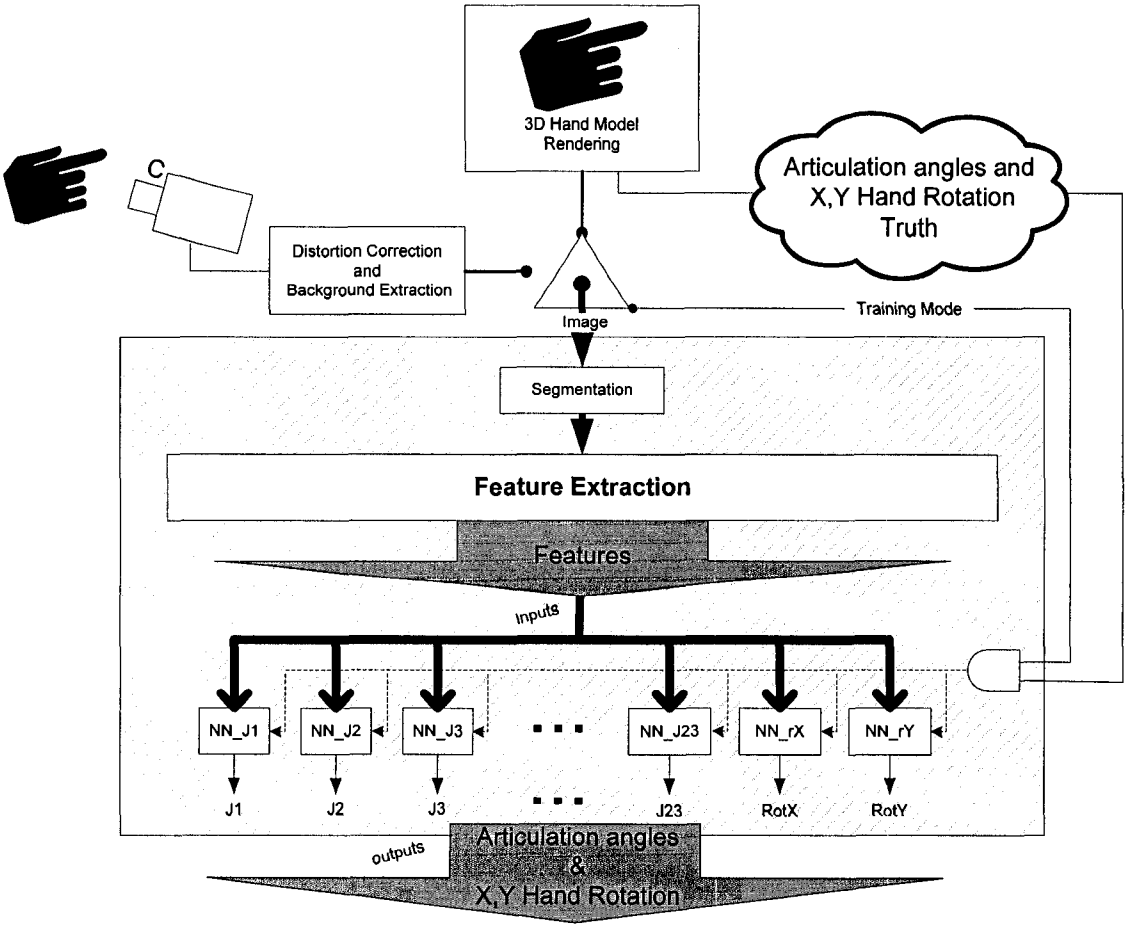

Figure 4.1 - Architecture of our ANN regression system

4.1.2 Training

The first obstacle faced was to properly choose the ANNs' architecture to solve our problem. After several attempts at manually specifying the architectures (number of layers and number of neurons per layer), we tried to automate this selection task by using the following workflow (Figure 4.2).

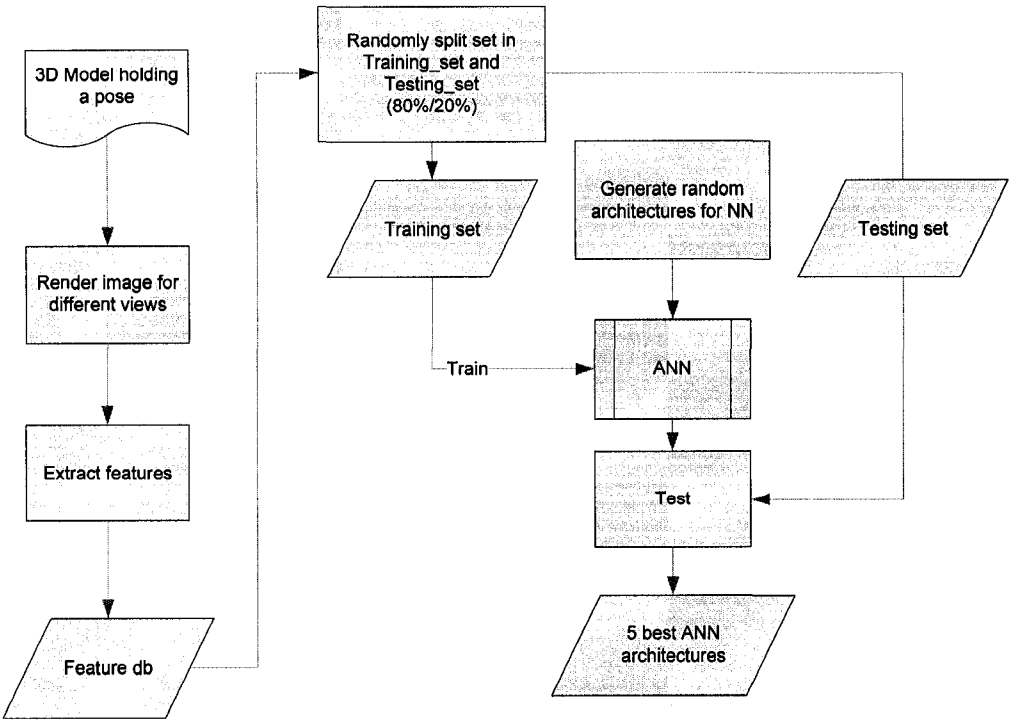

Figure 4.2 - ANN architecture generation workflow

The number of hidden layers was fixed to 2, and the number of neurons for each layer was randomly selected between 1 and 50. This process was iterated for a number of tries. Once the 5 best architectures where selected for the task, ANNs where randomly

generated following these architectures and training and testing was performed iteratively for 100 tries, keeping only the best resulting ANN (Figure 4.3).

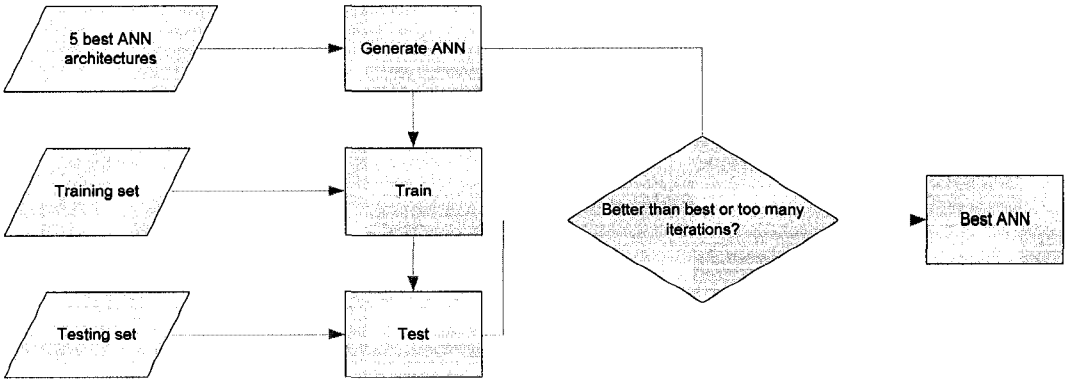

Figure 4.3 - Selecting best ANN

After obtaining limited success in training ANNs to generalize well over the complete range of data in this architecture, a second attempt was made at solving this regression problem by first segmenting our data into multiple sets, based on viewpoint orientation. First, classes had to be defined, and a classification ANN had to be trained to decide which regression system to use.

The new architecture classified postures based on their Y viewpoint orientation into 5 classes, defined to correspond roughly to Front views, Left views, Right views, and the intermediate Front Left views and Front Right views (Figure 4.4).

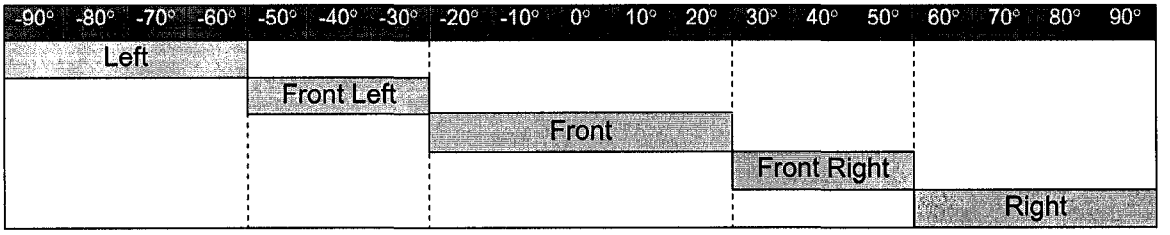

Figure 4.4 - Classification on Y viewpoint orientation

Figure 4.5 below shows the views of a sample posture at some limits of these ranges.

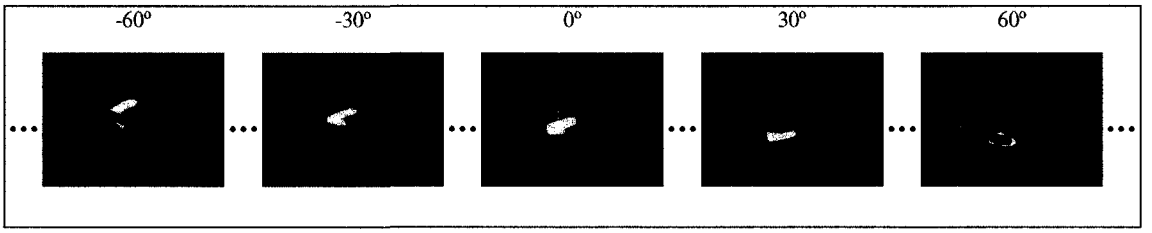

Figure 4.5 - Views of a posture of the hand varying in Y orientation (with X=0)

A single ANN was trained and tested to do this classification from the visual features with an acceptable recognition rate on test data (97.17%), which at least showed that such a classifier was possible. We therefore continued to try to solve our regression problem on only the Front class of viewpoints.

ANNs to find articulation angle estimates where then trained on this narrower set, following the same generative methodology previously described. Results of this experiment will be analyzed in the following chapter.

4.2 Classification

Having gained some experience from the earlier experiments, many of the previous techniques were adapted and improved for the new task of classification of hand postures.

4.2.1 System Overview

A camera input will ultimately feed the posture recognition system. After initialization (color calibration and hand model selection), a segmented output gets extracted for each camera frame, then processed to recover feature vector for recognition. Based on spatial partitioning of the output feature space, an artificial neural network (ANN) system interprets the feature vector and selects the appropriate ANN system for posture matching. This ANN system triggers possible matches to a set of trained postures. By using a number of these ANNs in parallel, voting is performed and a matched posture is proposed. The architecture of the system is depicted in Figure 4.6 below. To obtain the various sets of neural networks, a 3D hand-in-color-glove model is used. Frame rendering of this hand model, in the various postures to be recognized, and as seen from various point of views, are generated and processed to obtain the ANN training, testing and validation feature vectors.

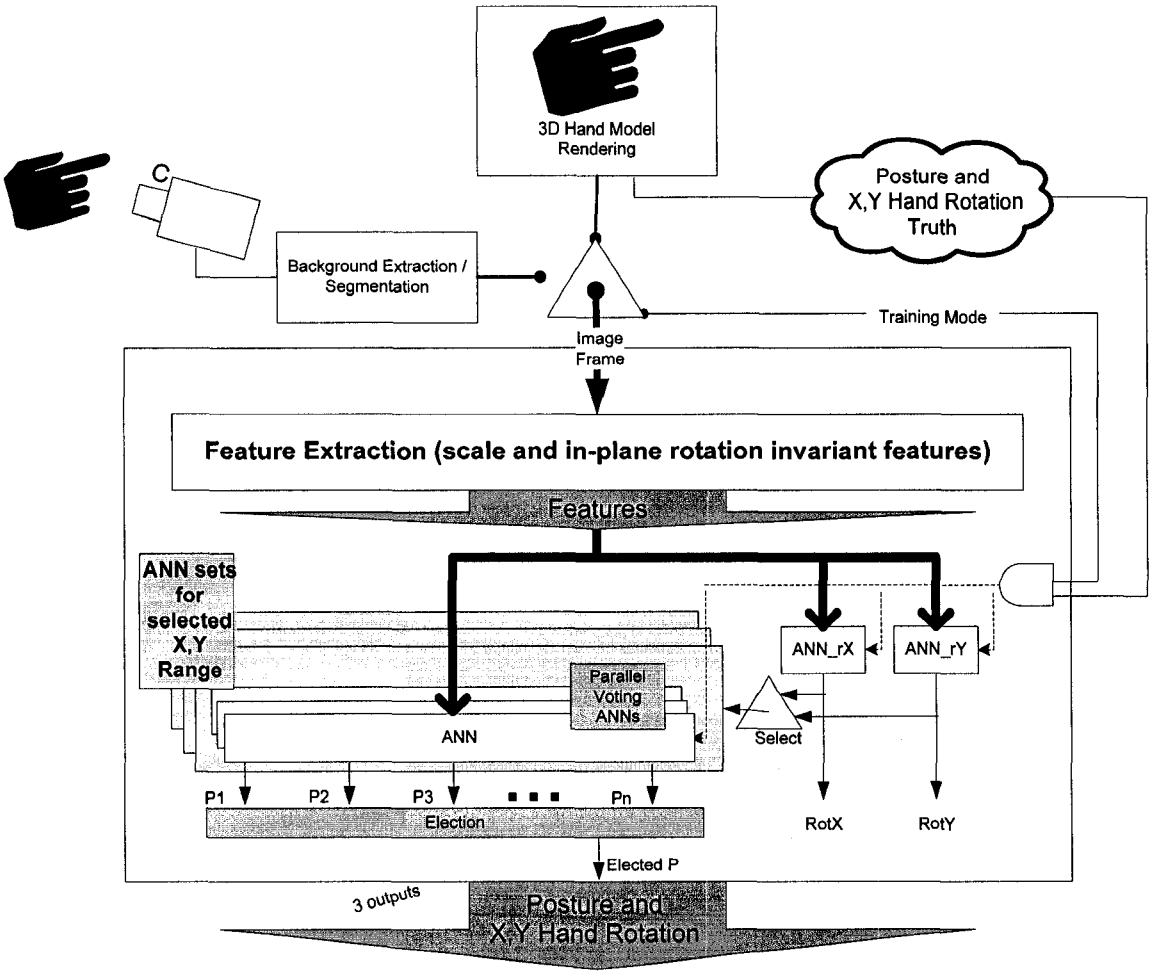

Figure 4.6 - Overall System Architecture

4.2.2 Artificial Neural Network Training

ANN systems can learn very well when modeling continuous functions. Discontinuities in a ANN modeled function tends to make its learning tedious and uncertain, especially since we must also find the proper network architecture in the first place, train long enough to get good results or give up trying. To simplify the task, we segment the output range of one of our continuous outputs (X or Y hand rotation) into overlapping sets of

ranges, using an intermediate mapping function that is more easily trained. This, it turns out, is similar in approach to what is done by Rosales et al [21, 22] in their specialized mapping architecture, or by Lamar [24] in his T-CombNet architecture, but in our case specialization is achieved through this intermediate approximation of some output (intermediate knowledge of the system), which we feel might be a better approach than their fragmentation of the input space. The comparison of these approaches could be the topic of future research.

Since typically, the ANN model of a function is imprecise at its range's edges, modeling of the functions in overlapped sub-ranges allows us to select more precise and more focused ANNs based on the estimated value of the range segmentation variables. The intermediate mapping function is another set of ANNs, separately trained and tested to allow us to estimate which ANN set to use.

For our recognition system's training, the 3D model of a hand wearing a color glove, fully reconfigurable, was developed using standard 3D animation software as described in chapter 3. Snapshots were rendered of a camera view centered and scaled appropriately, so as to fill the picture frame, but not exceed it when the hand is put in motion. It was set in 20 different postures, and rotated relative to its center, in steps of 5°, in both its tilt (X) and its roll (Y) axis (Figure 4.7). Frames of another ensemble of 20 postures considered in the same classes as in the first set, but with some variation in finger angles were also generated.

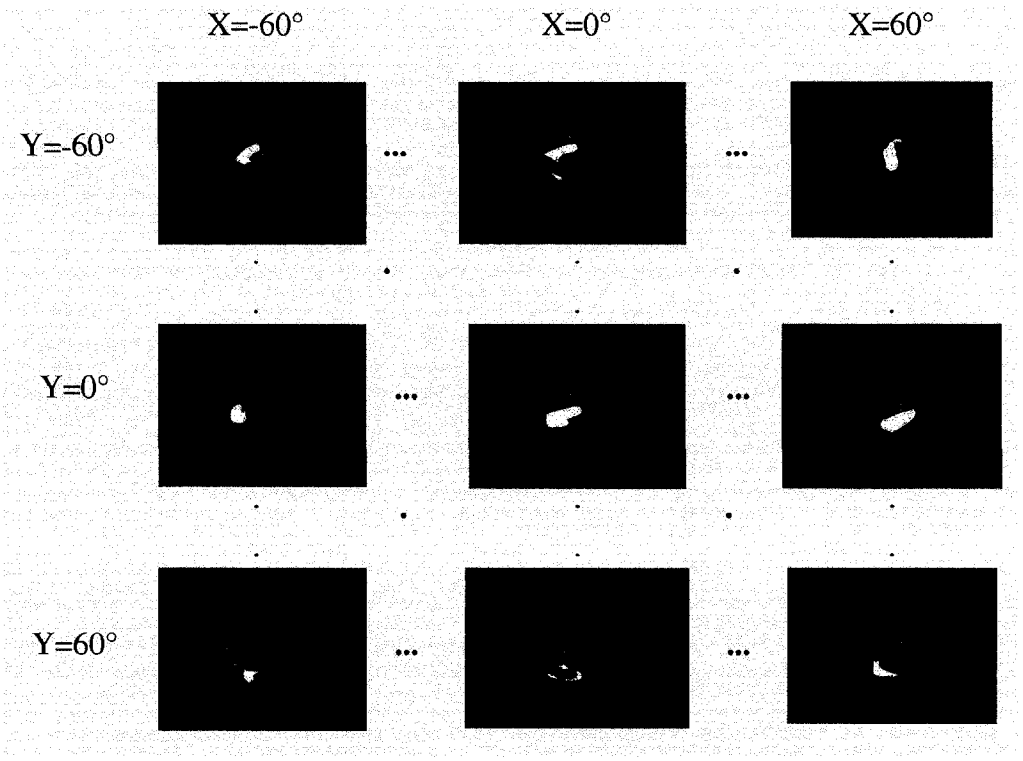

Figure 4.7 - Training set's X and Y rotation limits for hand posture #1

Feature vectors, as also described in the previous chapter, were extracted from all these pictures to produce our set of training and testing data.

For this classification application, we separate the X rotation input domain into ranges of 40°, overlapped as shown in Figure 4.8, going from -60° to 60° in both axis.

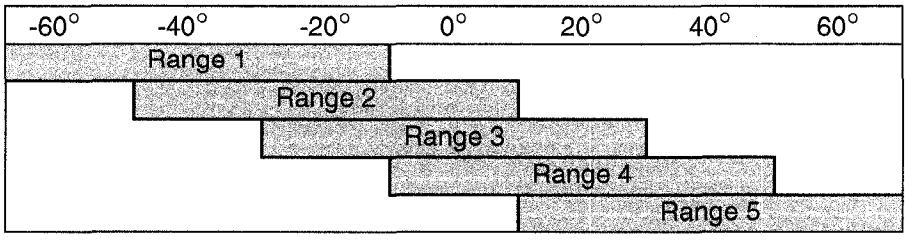

Figure 4.8 - Ranges of the X-rotation sub-division for ANN training

Architecture of the individual ANN is selected at random within a range of possible architectures. We allowed for either 2 or 3 hidden layers with between 5 to 55 neurons each, and trained for between 100 and 1000 epochs, on 60-90% of the data – the rest set aside for testing. We used the scaled conjugate gradient learning algorithm which provided a good compromise between memory use and learning time in our testing. This type of ANN generation and training was performed until we could find a set of 10 ANNs for each range that had over 95% recognition rate on the validation data (outside of the training set). We then use combined outputs of these 10 ANNs as a voting system to recognize the postures.

Separate ANNs are generated to estimate the X and Y rotations, but these obviously using the whole range of inputs for the training and testing data. The accuracy information of these networks on the testing sets is stored and used as a weighting factor in the mean calculation of the viewpoint estimation.

The process for generating the ANNs for the various ranges is depicted below (Figure 4.9).

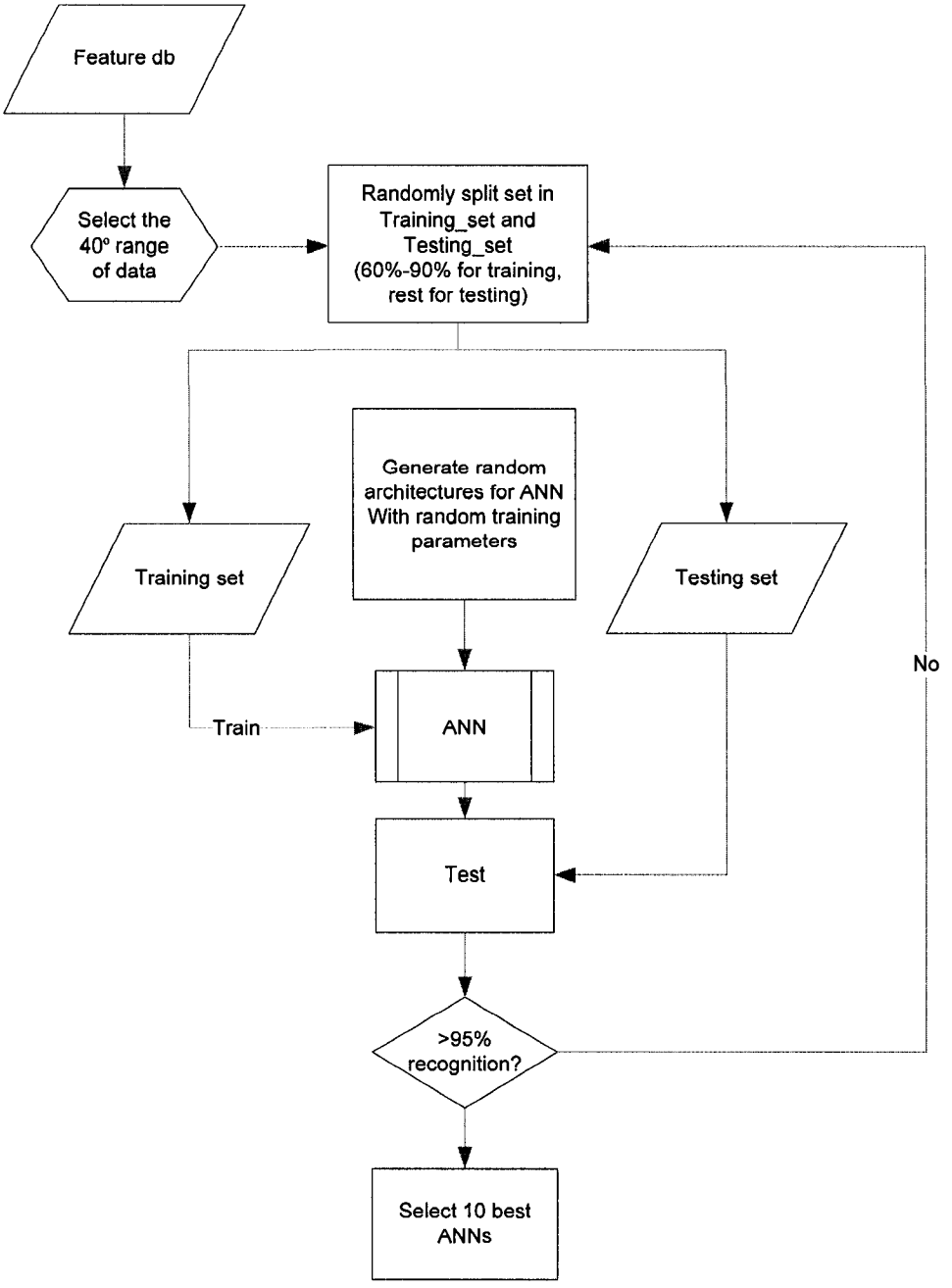

Figure 4.9 - Generative ANN selection for classification system

Chapter 5

Results and Analysis

5.1 Regression

For this task, a set of one ANN per articulation, the best that could be learnt as described earlier, was trained and tested for the Front data set (comprising the viewpoint orientations within $-20 \leq Y \leq 20$). Results are summarized in Table 5.1 below.

Results are the percentage of ANN estimation errors that are smaller than 10° from the truth for our training and testing set. We can note that better results appear to show for articulations which don't change much throughout our dataset.

We conclude that there is no evidence of learning of the generalized regression functions and that possibly a much larger set of training and testing data would be required for any form of true learning.

Given these discouraging results, the focus of our research now shifts toward the approach of posture classification.

Table 5.1 - Results for articulation angle regression task

Articulation	Training Set (Error < 10°)	Testing Set (Error < 10°)
1	90.13%	33.33%
2	87.34%	26.88%
3	93.78%	46.77%
4	93.35%	55.38%
5	89.91%	34.41%
6	84.33%	21.51%
7	97.42%	67.74%
8	94.21%	20.39%
9	76.61%	30.70%
10	96.14%	65.59%
11	84.33%	39.78%
12	76.18%	23.12%
13	84.55%	27.42%
14	97.85%	69.35%
15	97.21%	77.42%
16	86.05%	54.30%
17	80.69%	30.11%
18	97.64%	77.04%
19	94.42%	49.79%
20	92.49%	47.85%
21	82.83%	32.83%
22	82.83%	28.11%
23	86.48%	30.90%

5.2 Posture Classification

Our posture classification ANN system was trained to recognize the 20 postures of the hand pictured in Figure 5.1 (Set 1). A second set of postures, Set 2 (Figure 5.2), with variations in the articulation angles, but showing what we judge equivalent postures, where also used in the training in hopes to provide better generalization for our ANN system. Another third set of equivalent postures with small variations, Set 3 (Figure 5.3), was used for validation of our recognition system.

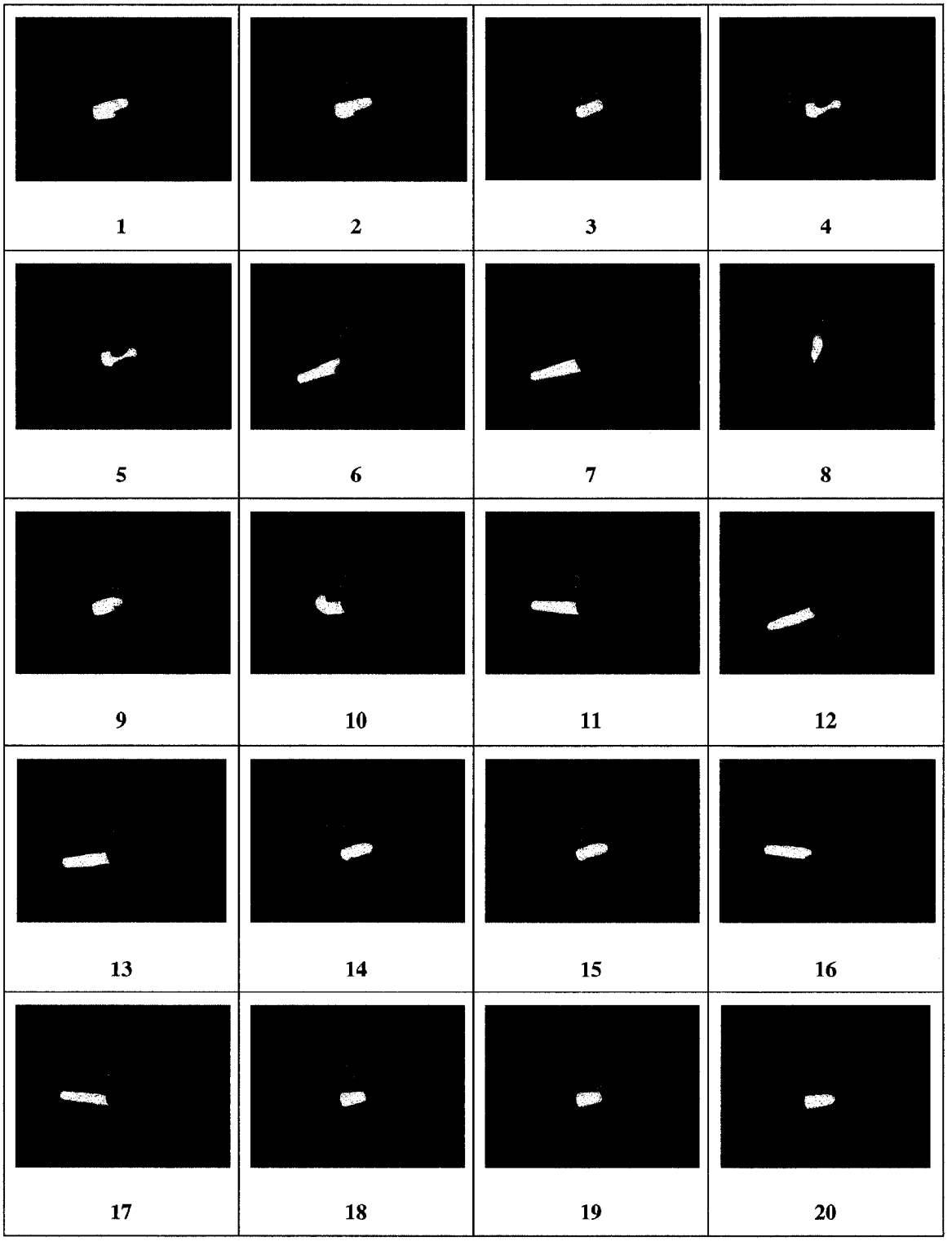

Figure 5.1 - Set 1: The 20 trained hand postures (view rotation X=0, Y=0)

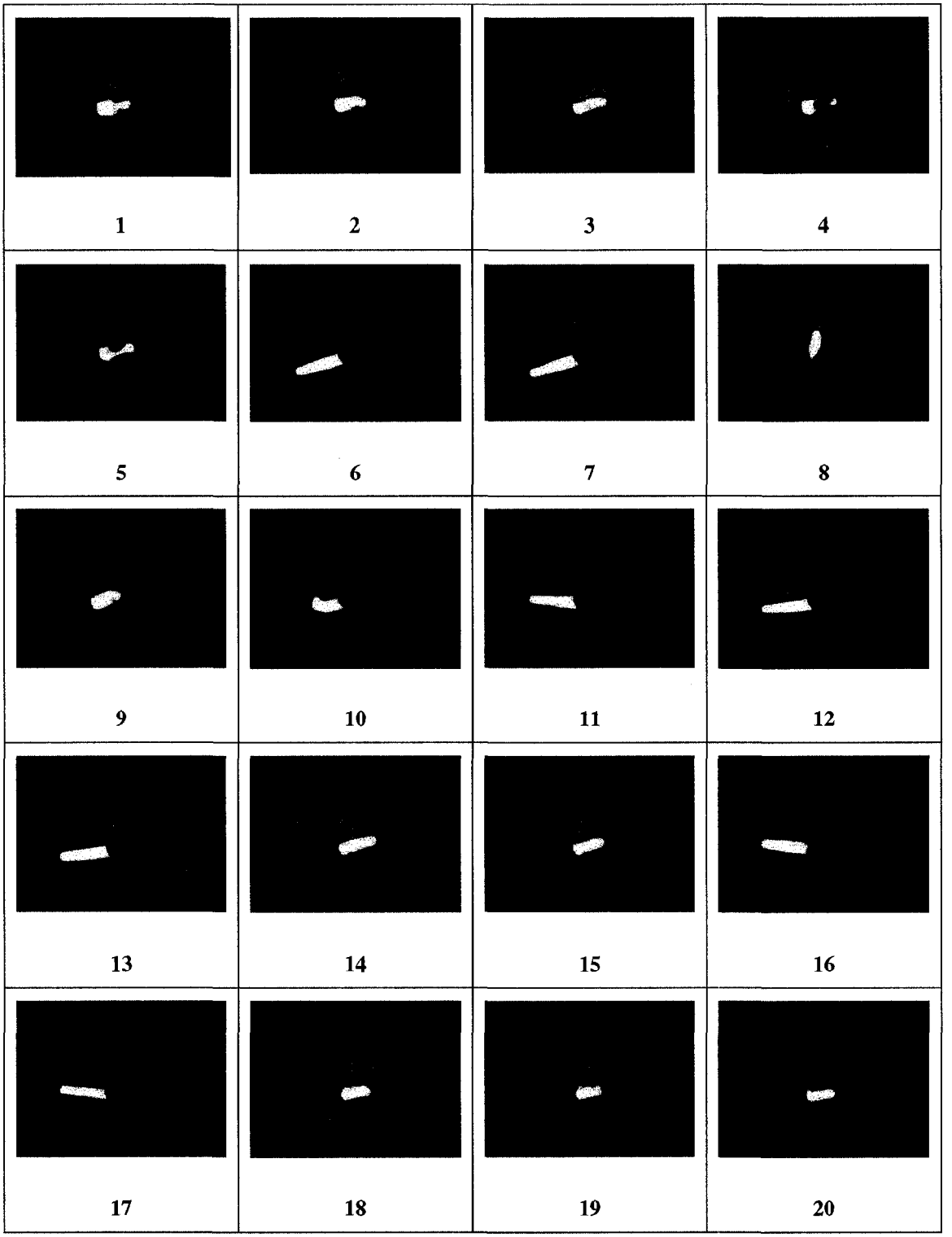

Figure 5.2 - Set 2: The same 20 hand postures with slight variations (view rotation X=0, Y=0)

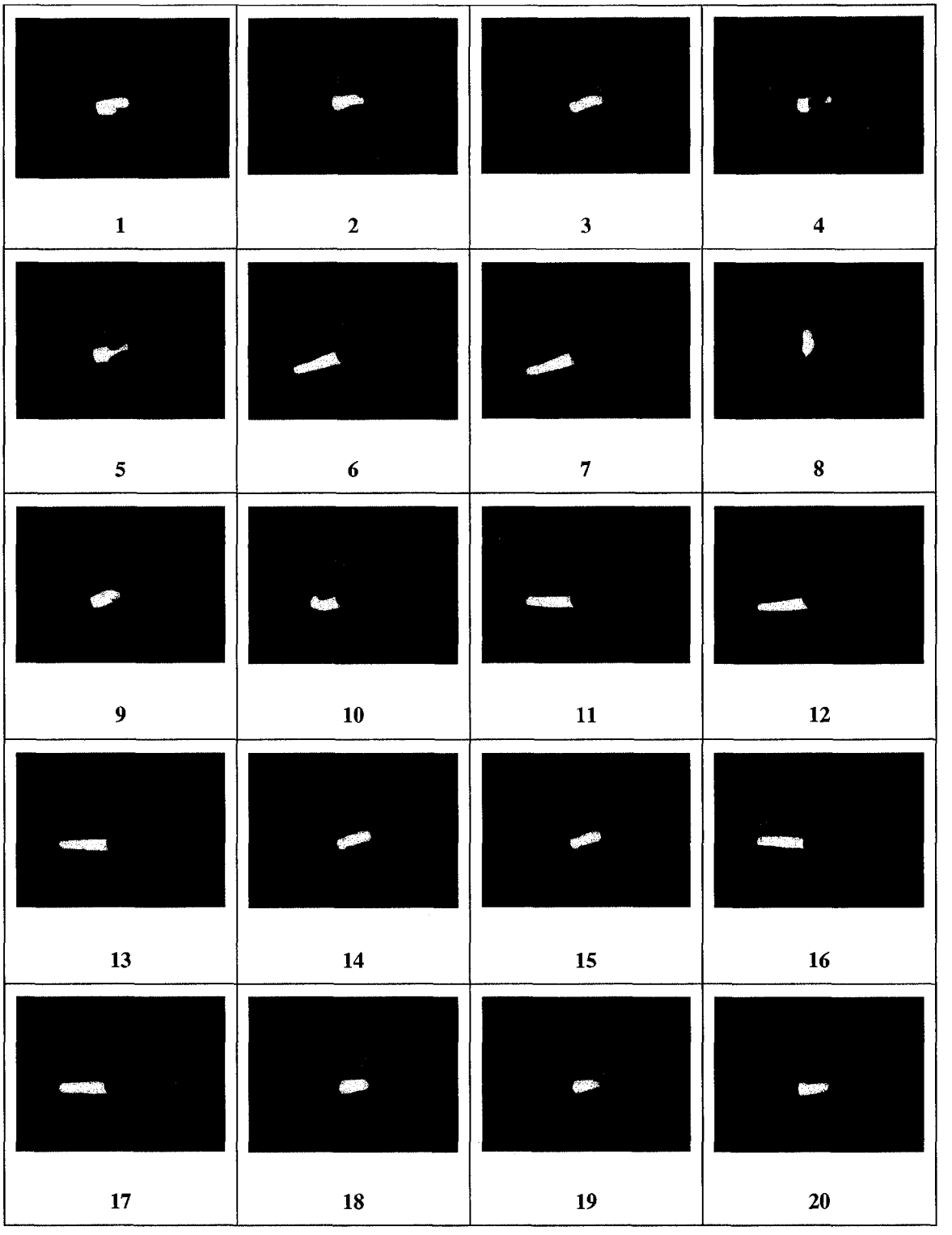

Figure 5.3 - Set 3: The same 20 hand postures with other slight variations (view rotation X=0, Y=0)

Results of posture recognition on various data sets are summarized in Table 5.2 below. Set 1 is the set of images of the 20 postures × 313 viewpoint orientations (each) used for the training and testing of the ANNs. Set 2 and Set 3 are sets of images of variation of these 20 postures × 169 viewpoint orientations each. Set 2 was partly used in the training and testing of the ANN system, while Set 3 was not.

Table 5.2 - Posture recognition performance summary for the different data sets

Posture	$-65° < X,Y < 65°$			$-45° < X,Y < 45°$		
	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3
1	99.68%	98.82%	73.37%	100.00%	100.00%	88.89%
2	99.68%	100.00%	99.41%	100.00%	100.00%	100.00%
3	98.40%	100.00%	92.90%	100.00%	100.00%	98.77%
4	100.00%	99.41%	99.41%	100.00%	100.00%	100.00%
5	98.40%	100.00%	82.84%	100.00%	100.00%	92.59%
6	99.36%	100.00%	76.33%	100.00%	100.00%	76.54%
7	100.00%	100.00%	99.41%	100.00%	100.00%	100.00%
8	97.12%	97.04%	98.22%	97.93%	100.00%	96.30%
9	100.00%	100.00%	99.41%	100.00%	100.00%	100.00%
10	99.68%	100.00%	94.08%	100.00%	100.00%	88.89%
11	100.00%	100.00%	98.22%	100.00%	100.00%	100.00%
12	100.00%	100.00%	98.22%	100.00%	100.00%	100.00%
13	98.08%	98.82%	94.67%	99.31%	100.00%	98.77%
14	96.81%	98.82%	98.82%	97.93%	100.00%	100.00%
15	96.49%	98.82%	95.27%	99.31%	100.00%	96.30%
16	98.72%	99.41%	98.82%	100.00%	100.00%	100.00%
17	99.68%	100.00%	95.27%	100.00%	100.00%	100.00%
18	98.08%	98.82%	91.12%	97.93%	100.00%	95.06%
19	86.90%	97.04%	79.88%	94.48%	100.00%	85.19%
20	99.68%	100.00%	98.82%	100.00%	100.00%	97.53%
AVG	**98.34%**	**99.35%**	**93.22%**	**99.34%**	**100.00%**	**95.74%**

Detailed results for posture recognition of Set 2 is shown below (Table 5.3), with mean difference in X and Y viewpoint orientation estimation (AVG(ΔX) and AVG(ΔY)) and their respective standard deviation (STD(ΔX) and STD(ΔY)).

Table 5.3 - Recognition results for posture Set 2 ($-65^{o}<X,Y<65^{o}$)

Posture	AVG(ΔX)	STD(ΔX)	AVG(ΔY)	STD(ΔY)	Recognition
1	10.6597	9.6155	12.1086	10.1483	98.82%
2	5.5761	4.3837	11.6004	7.8113	100.00%
3	4.5810	3.5846	4.8197	3.7013	100.00%
4	3.2185	2.4543	8.1997	5.8774	99.41%
5	2.1548	1.5520	3.3658	2.4893	100.00%
6	9.1748	7.3332	5.8017	4.6044	100.00%
7	6.3243	5.5295	3.6895	3.1938	100.00%
8	5.4388	4.0372	8.3851	6.4588	97.04%
9	4.4098	2.7419	5.6653	3.7827	100.00%
10	7.4718	5.7435	4.3136	2.8990	100.00%
11	4.3910	2.6983	5.0243	3.5391	100.00%
12	4.8090	4.4503	5.0262	4.2806	100.00%
13	3.8057	3.2486	4.7753	4.7117	98.82%
14	3.6811	3.2515	4.6232	3.7018	98.82%
15	4.3157	3.8938	8.6655	6.2242	98.82%
16	3.7665	3.1323	4.1053	3.6492	99.41%
17	3.1244	1.9970	4.2101	2.8844	100.00%
18	9.0614	5.9129	8.8023	6.7171	98.82%
19	6.4053	4.5202	6.6603	3.8612	97.04%
20	4.5953	3.4282	7.7890	5.9545	100.00%
AVG	5.3482	4.1754	6.3816	4.8245	99.35%

It is worthy of noting that the viewpoint orientation was trained and tested with part of Set 1 only. A new viewpoint orientation estimation system should ideally be trained and tested with some posture variations samples such as the ones in Set 2 or 3, and would probably yield better generalized results.

The validation set (Set 3), with posture variations previously unseen by the recognition system for its training and testing, shows good response in most cases. This indicates a good capacity to generalize the recognition process for most postures.

Table 5.4 - Recognition results for posture Set 3 (-65°<X,Y<65°)

Posture	AVG(ΔX)	STD(ΔX)	AVG(ΔY)	STD(ΔY)	Recognition
1	8.5389	6.0600	6.0109	4.8384	73.37%
2	5.3095	4.3178	8.9282	7.2049	99.41%
3	3.9542	3.0901	6.5309	4.9941	92.90%
4	3.1511	2.3496	8.0063	5.7704	99.41%
5	4.3553	4.0247	4.5325	3.5452	82.84%
6	10.0262	7.2000	16.1241	7.2332	76.33%
7	6.7770	5.8595	3.9933	2.9721	99.41%
8	5.6346	3.8753	11.0962	8.3001	98.22%
9	3.8046	3.0651	5.4703	4.5826	99.41%
10	5.2368	3.9342	6.1990	4.2661	94.08%
11	5.4420	3.2683	6.6940	4.1229	98.22%
12	7.2945	6.2128	6.3080	5.2883	98.22%
13	6.8711	6.4520	7.8998	7.2471	94.67%
14	3.6310	3.2198	4.8156	3.5485	98.82%
15	4.0974	3.8719	5.3948	4.2717	95.27%
16	3.5567	2.5116	7.3643	7.3724	98.82%
17	3.3227	2.1439	3.2910	2.8789	95.27%
18	12.5589	11.1232	13.2990	7.1657	91.12%
19	6.0751	4.6274	12.8721	10.7215	79.88%
20	7.6857	6.2308	9.7330	6.2892	98.82%
AVG	5.8662	4.6719	7.7282	5.6307	93.22%

Because of the inherent imprecision of the ANNs at their edges, in both X and Y viewpoint orientation ranges (need to extrapolate, which ANNs do not achieve well), it might be more appropriate to evaluate our recognition task on a narrower range of data like between $-45° < X, Y, < 45°$. Results for the posture recognition of Set 3, in this narrower range, are shown below (Table 5.5).

Table 5.5 - Recognition results for posture Set 3 (-45°<X,Y<45°)

Posture	AVG(ΔX)	STD(ΔX)	AVG(ΔY)	STD(ΔY)	Recognition
1	8.3218	4.9303	4.7026	4.0376	88.89%
2	3.6775	2.7865	7.9810	4.8424	100.00%
3	4.0286	2.5503	8.1089	5.4588	98.77%
4	3.6027	2.0630	7.8194	6.0435	100.00%
5	4.1767	4.1944	4.1895	2.6165	92.59%
6	9.6993	7.8434	13.5584	3.0129	76.54%
7	9.3582	6.6326	3.0363	1.9369	100.00%
8	6.0375	3.7416	12.1487	8.5472	96.30%
9	3.5167	3.4655	4.4596	3.2754	100.00%
10	4.9513	4.5914	5.9052	3.6913	88.89%
11	4.8551	2.1929	5.7512	2.9391	100.00%
12	4.7492	2.8569	4.5561	3.0265	100.00%
13	4.5664	4.1025	7.6326	5.0795	98.77%
14	4.8614	3.3798	3.5109	2.4145	100.00%
15	3.8679	4.3319	3.0361	2.3551	96.30%
16	3.0515	2.1251	3.8018	2.6528	100.00%
17	2.9017	1.9839	3.2882	2.5426	100.00%
18	15.9960	11.5522	17.5800	6.1477	95.06%
19	5.5866	4.6032	10.9014	8.4431	85.19%
20	9.7226	7.2900	8.9014	4.3580	97.53%
AVG	5.8764	4.3609	7.0434	4.1711	95.74%

We see a net improvement of the recognition rate when we avoid ANN extrapolation in this way. This is especially evident for some of the postures which are then recognized in all considered orientations. Recognition of posture #6, in this case, shows no improvement, which leads us to think that either the recognition system does not generalize this posture enough for variations, or the visual features used don't show enough differences between it and some other postures in some views.

Table 5.6 and Table 5.7 below show the X and Y orientation recovery and average recognition rate by viewpoint orientation for Set 3. These recognition rates are then plotted in Figure 5.4 below.

Table 5.6 - Recognition results by X viewpoint orientation for Set 3

X	AVG(ΔX)	STD(ΔX)	AVG(ΔY)	STD(ΔY)	Recognition
-60	6.5361	6.0451	10.0697	7.1990	94.23%
-50	5.6746	5.3846	8.1704	5.9937	95.00%
-40	5.0546	4.6220	7.2303	5.5034	95.77%
-30	4.9885	4.0397	6.7463	5.6447	93.85%
-20	5.0310	3.9793	6.8431	5.9187	95.77%
-10	5.4245	4.9722	6.7487	5.8850	95.77%
0	6.1587	6.2648	6.2339	5.5363	95.00%
10	6.4810	6.7852	6.3941	5.8291	93.85%
20	6.6152	6.5865	6.5756	5.6705	92.31%
30	6.2692	6.1258	6.9593	6.1157	92.69%
40	6.0544	5.8055	7.5958	6.6918	93.46%
50	5.9655	5.0923	8.6839	7.3671	88.85%
60	6.0069	6.4092	12.2151	10.7447	85.38%
AVG	**5.8662**	**5.5471**	**7.7282**	**6.4692**	**93.22%**

Table 5.7 - Recognition results by Y viewpoint orientation for Set 3

Y	AVG(ΔX)	STD(ΔX)	AVG(ΔY)	STD(ΔY)	Recognition
-60	6.1300	5.5167	7.2196	8.6689	93.85%
-50	5.7913	5.4383	6.8541	7.3623	96.54%
-40	5.4937	5.2322	6.8020	6.8536	96.92%
-30	5.5435	5.1272	6.9284	6.2068	97.31%
-20	5.5999	5.0509	7.1281	6.1510	95.77%
-10	5.7685	5.1034	7.4369	6.1483	95.00%
0	5.9800	5.5194	7.6484	6.2292	96.15%
10	6.2878	6.3682	8.5253	6.8800	93.08%
20	6.0303	6.5535	8.7779	7.0643	92.69%
30	5.9028	6.1771	9.3038	7.0971	92.69%
40	5.6818	5.5322	9.0714	6.4476	90.77%
50	5.6536	5.3894	7.8257	6.2888	86.92%
60	6.3969	6.0641	6.9443	6.0285	84.23%
AVG	**5.8662**	**5.6210**	**7.7282**	**6.7251**	**93.22%**

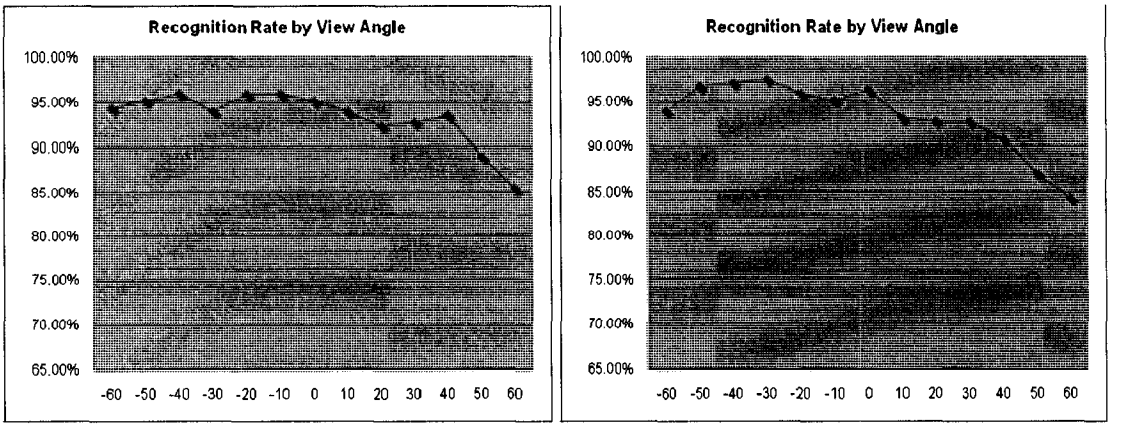

Figure 5.4 - Set 3 recognition rate by viewpoint orientation angle in X and Y

Performance for the recovery of the hand view rotation is adequate for our purpose of range estimation, as can be seen in Figure 5.5 and Figure 5.6. This rotation information can optionally be interpreted and used by the end application for other purposes.

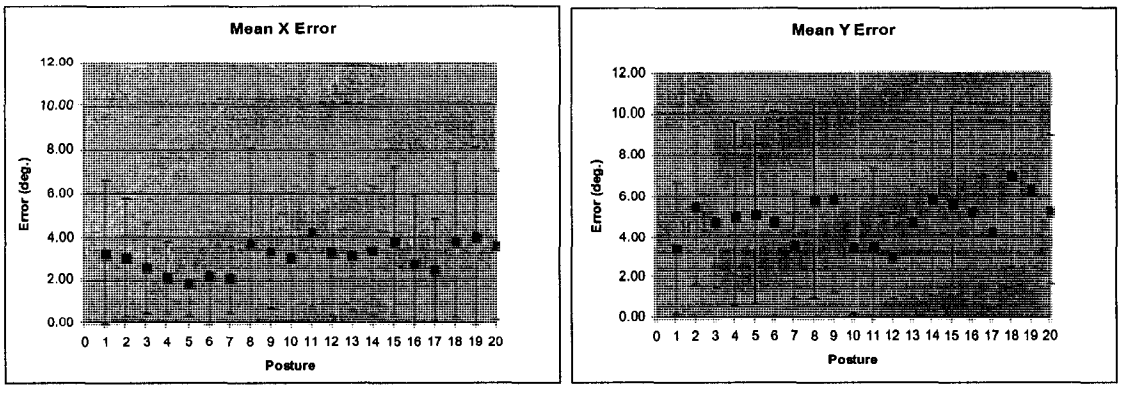

Figure 5.5 - X and Y rotation recovery mean error and standard deviation for Set 1

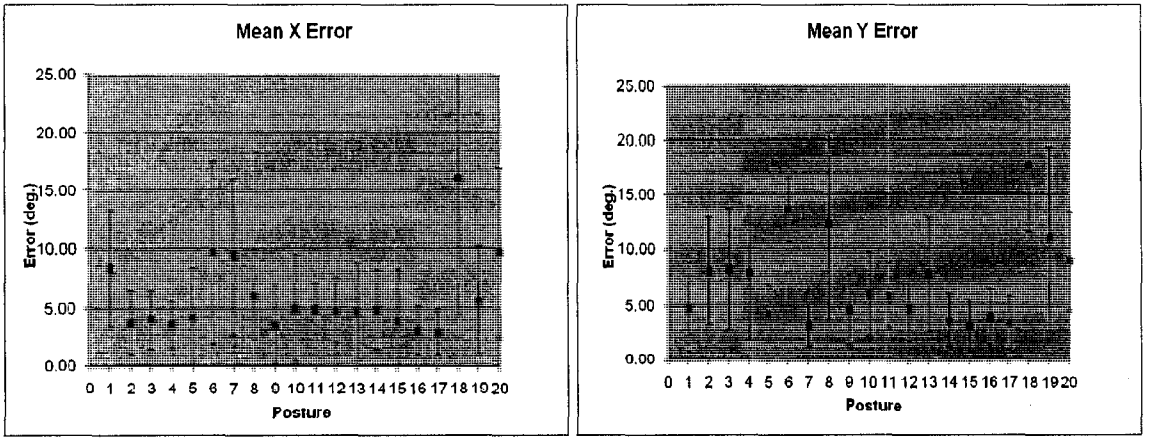

Figure 5.6 - X and Y rotation recovery mean error and standard deviation for Set 3 (-45°<X,Y<45°)

Overall, from our results, we see that some postures are better recognized and generalized than others. It would be best to avoid using the badly recognized postures at all in a control application. Alternatively, the networks could be trained more extensively on these postures, with possibly more variations to try to increase the ANNs recognition and generalization.

5.3 Real-Time Video Segmentation

Initial implementation of our segmentation method explained in section 3.4.1, using MATLAB, could hardly achieve more than 1 frame every 2 seconds on a 3GHz Xeon-based computer, with 320x240 pixels color frames. This processing speed would not really qualify for a real-time system. A later implementation using OpenCV [29] has been

tested to perform at higher than 30 frames per second on the same computer hardware, on the same frame sizes.

5.4 Real-Image Testing of Posture Classification

Some video sequences of a hand wearing our prototype glove, showing the various postures that we try to classify, were captured and stored for further processing. Segmentation of these video sequences was performed using the segmentation algorithm described in this thesis. The set of visual feature measurements was extracted for every frame of the video sequences and fed to our recognition system. Results were recorded for both the elected posture, and the X and Y viewpoint orientation at every frame.

Analysis of the performance of these tests is very subjective and we feel that their results are inconclusive. Although it seems that the recognition is working well in many cases, several issues remain.

First, the camera employed for capturing these sequences uses a wide-angle lens which distorts the images and ultimately affects the measurement of our visual features. Ideally, distortion correction should be performed prior to the feature extraction, or a different lens (or camera) should be employed. Also, the video segmentation proves problematic because of some of the colors used by our prototype glove. Lastly, since the trained system is somewhat limited in the allowed viewpoint orientation, restrictions should be put in the allowed rotation of the hand in the video sequences used for testing, or alternatively, a recognition system allowing all orientations should be built.

Combined with a more complete recognition system, a better color glove and a better real-image testing protocol should be elaborated, but that is left for future work.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

We have developed a system to classify postures of a hand wearing a color glove as viewed by a camera and the methods to generate, train and evaluate such a recognition system using artificial neural networks. We also have developed and implemented a real-time video segmentation algorithm that can be used to measure the visual features needed by this recognition system. We have obtained encouragingly high recognition rates for some of the postures in our evaluation when using our 3D hand model for validation. Given a close match of the real hand's morphology to our 3D hand model and a perfect video segmentation, there is no reason the system would not perform equally well on real camera images.

However, color segmentation has proven to be a very challenging process, even when using specific colors as in this research. Segmentation noise is bound to affect our results in live image applications if not correctly filtered. Also, our system does not properly

handle the detection of false positive recognition. This issue is quite limiting for a control application and should be handled properly for a complete posture recognition system. Furthermore, large anthropometric differences of the hand for different subjects must be handled by systems trained on appropriately different hand models.

6.2 Future Work

Our recognition system is currently restricted to operate only with viewpoints in the hemisphere to the front of the hand. This has more to do with the deadline for submission of this thesis than limitations of our approach. Viewpoints from the full sphere around the hand should ideally be considered for our application and is left as future work.

The use of different sets of visual features could be explored, including ones from images of bare hands, coupled with the appropriate training material. Features tolerant to a noisy and incomplete segmentation would be desirable.

Another aspect of this research which we would want to investigate further is the development of a method to automate the specialization of neural network systems in a way that makes use of intermediate knowledge, much like our use of the viewpoint orientation estimation. Example of other specialization could be the detection of inter-finger occlusions and estimation of the missing areas, or more simply the detection of completely extended or completely bent fingers.

A fully optical hand pose recognition system such as the one envisioned in this research would benefit from tracking of the hand, not only in the camera frames for segmentation

purposes, but also with reference to its 3D space, enabling even greater interaction possibilities within the context of VR applications. Optical tracking of a human hand, because of its deformable nature, poses many challenges. Possible tracking algorithms could take advantage of temporal inter-frame relationship to aid in the process.

As previously mentioned, the detection of false-positive recognition in our application, in order to ignore the posture commands that were recognized but not requested, should be investigated. Map-back validation methods such as done by Rosales et al [22] could be a solution to this problem.

Finally, our current implementation is relying heavily on the MATLAB software for feature extraction, normalization and computation of the ANNs results. However great this framework has been for our research work, licensing and performance issues prevents us from directly using this system in its intended environment. A real-time implementation of the complete optical posture recognition system, to work within our DIVINE immersive VR display, should therefore be completed outside of this MATLAB framework.

Bibliography

[1] Y. Wu and T. S. Huang, "Hand modeling analysis and recognition for vision-based human computer interaction", IEEE Signal Processing Magazine, Special Issue on Immersive Interactive Technology, vol. 18, no. 3, pp. 51–60, 2001.

[2] F. Bettio, A. Giachetti, E. Gobbetti, F. Marton, and G. Pintore, "A practical vision based approach to unencumbered direct spatial manipulation in virtual worlds", In Eurographics Italian Chapter Conference, Povo (Trento), Italy, February 2007.

[3] R. G. O'Hagan, "Vision-based Gesture Recognition as an Interface to Virtual Environments", Ph.D. Thesis, Australian National University, Canberra, Australia, 2002.

[4] A-M. Burns, "Computer Vision Methods for Guitarist Left-Hand Fingering Recognition", Master of Arts in Music Technology Thesis, McGill University, Montreal, QC, Canada, 2007.

[5] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, "Surround-screen projection-based virtual reality: the design and implementation of the CAVE", Proc. of the 20th annual conference on Computer graphics and interactive techniques, Anaheim, California, USA, pp 135-142, September 1993.

[6] K. Osman, F. Malric, S. Shirmohammadi, "A 3D Annotation Interface Using the DIVINE Visual Display", Proc. IEEE Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, ON, Canada, November 2006.

[7] "Immersion Corporation, Cyberglove II Datasheet." [Online], [cited 2008 May 15], Available at: http://www.immersion.com/3d/docs/cybergloveII_dec07v4-lr.pdf

[8] T. Metais, "A Dynamic Gesture Interface", Master of Science Thesis, University of Ottawa, Ottawa, ON, Canada, 2005.

[9] A. Yilmaz, O. Javed, M. Shah, "Object tracking: A survey", ACM Computing Surveys, Volume 38, Issue 4, Dec. 2006.

[10] "Vicon Systems." [Online], [cited 2008 Jun 20], Available at: http://www.vicon.com/products/systems.html

[11] H. Demuth, M. Beale, "Neural Network Toolbox User's Guide (Version 4) for use with MATLAB".

[12] B. Dorner, "Chasing the Colour Glove: Visual Hand Tracking", Master of Science Thesis, Simon Fraser University, Surrey, BC, Canada, 1994.

[13] A. El-Sawah, C. Joslin, N. D. Georganas, E. M. Petriu, "A Framework for 3D Hand Tracking and Gesture Recognition using Elements of Genetic Programming", Proc. VideoRec'07: Int. Workshop on Video Processing and Recognition, pp. 495-502, Montreal, QC, Canada, May 2007.

[14] J. Lee, T.L. Kunii, "Model-based analysis of hand posture", IEEE Computer Graphics and Applications, vol. 15, Issue 5, pp. 77-86, Sep 1995.

[15] V. Athitsos, S. Sclaroff, "Estimating 3D Hand Pose from a Cluttered Image", Conference on Computer Vision and Pattern Recognition (CVPR), Madison, Wisconsin, USA, pp. 432-439, June 2003.

[16] V. Athitsos and S. Sclaro, "Database indexing methods for 3D hand pose estimation", In Gesture Workshop, Genova, Italy, pp. 288-299. Springer-Verlag Heidelberg, 2003.

[17] H. Zhou, T. Huang, "Okapi-Chamfer Matching for Articulate Object Recognition", Tenth IEEE International Conference on Computer Vision (ICCV), Beijing, China, Vol. 2, pp. 1026-1033, 17-21 Oct. 2005.

[18] N. Shimada, K. Kimura, Y. Shirai, "Real-Time 3-D Hand Posture Estimation Based on 2-D Appearance Retrieval Using Monocular Camera", IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS'01), Vancouver, BC, Canada, pp. 23-30, 2001.

[19] A. Imai, N. Shimada, Y. Shirai, "3-D hand posture recognition by training contour variation", Proceedings. Sixth IEEE International Conference on Automatic Face and Gesture Recognition, Seoul, Korea, pp. 895-900, 17-19 May 2004.

[20] H. Guan, J. S. Chang, L. Chen, R. S. Feris, and M. Turk, "Multi-view Appearance-based 3D Hand Pose Estimation", In Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop, New York, NY, USA, June 17 - 22, 2006.

[21] R. Rosales, V. Athitsos, L. Sigal, S. Sclaroff, "3D Hand Pose Reconstruction Using Specialized Mappings", Eighth International Conference on Computer Vision (ICCV'01), Vancouver, BC, Canada, Volume 1, p. 378, 2001.

[22] R. Rosales and S. Sclaroff, "The Specialized Mappings Architecture", Department of Computer Science at Boston University (BUCS) Technical Report, Boston, MA, USA, March 28, 2003.

[23] E.B. Goldstein, "Sensation and Perception", second edition, Wadsworth Publishing Company, ISBN 0-534-03035-1.

[24] M.V. Lamar, "Hand Gesture Recognition using T-CombNET A Neural Network Model dedicated to Temporal Information Processing", Doctoral Thesis, Nagoya Institute of Technology, Nagoya, Japan, 2001.

[25] G. Bebis, F. Harris, A. Erol, "Development of a Nationally Competitive Program in Computer Vision Technologies for Effective Human-Computer Interaction in Virtual Environments", Progress report #1 submitted to NASA Space Grant/EPSCoR, December 2002.

[26] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, "Demosaicking: Color filter array interpolation in single chip digital cameras," IEEE Signal Processing Magazine (Special Issue on Color Image Processing), 2005.

[27] J. Lin, Y. Wu, and T.S. Huang, "Modeling the constraints of human hand motion", Proceedings of the Workshop on Human Motion (Humo'00), December 07 - 08, 2000.

[28] A. R. Smith and J. F. Blinn, "Blue screen matting", Proceedings of the 23rd Annual Conference on Computer Graphics and interactive Techniques SIGGRAPH '96. ACM, New York, NY, USA, 259-268.

[29] "Open Source Computer Vision Library." [Online], [cited 2008 May 15], Available at: http://www.intel.com/technology/computing/opencv/

[30] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, X. Twombly, "A Review on Vision-Based Full DOF Hand Motion Estimation", IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPRW'05) - Workshops, San Diego, CA, USA, p. 75, 2005.

[31] A. Hani. K. Almohair, Abd Rahman Ramli, Elsadig A.M. and B. Shaiful J. Hashim, "Skin Detection in Luminance Images using Threshold Technique", International Journal of The Computer, the Internet and Management, Vol. 15, no. 1, pp. 25 -32, January – April, 2007.

[32] N.D. Binh, E. Shuichi, T. Ejima, "A New Approach Dedicated to Real-Time Hand Gesture Recognition", The 2006 International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV'06), pp. 481-488, Las Vegas, USA, June 26-29 2006.

[33] L. Bretzner, I. Laptev, T. Lindeberg, "Hand Gesture Recognition using Multi-Scale Colour Features, Hierarchical Models and Particle Filtering", Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition, Washington, D.C., USA, pp. 405 – 410, 20-21 May 2002.

[34] H.Y. Guan, C.S. Chua, Y.K. Ho, "Hand Posture Estimation from 2D Monocular Image", Proceedings of the Second International Conference on 3-D Digital Imaging and Modeling, Ottawa, ON, Canada, pp. 424 – 429, 1999.

[35] J.M. Rehg, D.D. Morris, T. Kanade, "Ambiguities in Visual Tracking of Articulated Objects Using Two- and Three-Dimensional Models", The International Journal of Robotics Research, Vol. 22, No. 6, pp. 393-418, June 2003.

[36] Y. Wu and T. S. Huang, "Nonstationary Color Tracking for Vision-Based Human-Computer Interaction", IEEE Transactions on Neural Networks, vol. 13, no 4, pp. 948- 960, Jul 2002.

[37] Y. Beifang, F. C. Harris Jr., L. Wang, Y. Yan, "Real-Time Natural Hand Gestures", Computing in Science and Engineering, vol. 7, no. 3, pp. 92-96, c3, May/Jun, 2005.

[38] C. Nölker and H. Ritter. "GREFIT: Visual Recognition of Hand Postures", Proceedings of the International Gesture Workshop, Gif-sur-Yvette, France, pp. 61-72, 1999.

[39] X. Wang, X. Zhang, and G. Dai, "Tracking of deformable human hand in real time as continuous input for gesture-based interaction", Proceedings of the 12th international Conference on intelligent User interfaces, Honolulu, Hawaii, USA, January 28 - 31, 2007.

[40] J. Triesch, C. von der Malsburg, "A System for Person-Independent Hand Posture Recognition against Complex Backgrounds", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 12, pp. 1449-1453, Dec. 2001.

[41] D. S. Banarse, "Hand Posture Recognition with the Neocognitron Network", School of Electronic Engineering and Computer Systems, University College of NorthWales, Bangor, 1993.

[42] J. LaViola, "A Survey of Hand Posture and Gesture Recognition Techniques and Technology", Technical Report CS- 99-11, Department of Computer Science, Brown University, Providence, Rhode Island, 1999.

[43] M. B. Stinchcombe, "Neural network approximation of continuous functional and continuous functions on compactifications", Neural Networks, 12(3):467-477, 1999.

[44] R.Rojas, "Neural Networks - A Systematic Introduction", Springer-Verlag, Berlin, New-York, 1996.

[45] E. Rich and K. Knight, "Artificial Intelligence", second edition, McGraw-Hill, New York, NY, USA, ISBN 0-07-052263-4.