Ankit Chaudhary

Robust Hand Gesture Recognition for Robotic Hand Control

Springer

Robust Hand Gesture Recognition for Robotic Hand Control

Ankit Chaudhary

Robust Hand Gesture Recognition for Robotic Hand Control

Springer

Ankit Chaudhary
Department of Computer Science
Northwest Missouri State University
Maryville, MO
USA

ISBN 978-981-10-4797-8 ISBN 978-981-10-4798-5 (eBook)
DOI 10.1007/978-981-10-4798-5

Library of Congress Control Number: 2017940231

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

"If we knew what it was we were doing, it would not be called research, would it?"

—Albert Einstein

Preface

In the past few decades, hand gesture recognition has been considered to be an easy and natural technique for human–machine interaction. Many applications have been developed and enhanced based on hand gesture recognition. These applications range from mobile phones to advanced robotics and from gaming to medical science. In most of the existing commercial and research applications, recognition of hand gestures has been performed by employing sensor-based wired embedded gloves or by using vision-based techniques where colors, chemicals, or paperclips are used on the hand. However, it is desirable to have hand gesture recognition techniques that are applicable to a natural and bare hand, which is normally used for depicting gestures in verbal communication. Another important issue involved in vision-based techniques is their variance to light conditions. As the light conditions change, the threshold used for the segmentation also has to be changed. Bare hand gesture-based applications where no external device or color is used, do not work for different light intensities. In the case of human skin, different user's skin color appears different in the same light intensity while same user's skin color varies in different light conditions.

The prime aim of this book is to enhance the current state of the art of bare hand gesture recognition and present a model for controlling an electromechanical robotic hand. The focus is on the development of intelligent techniques applicable to the spatial domain for processing where the user has no limitation on hand direction and will not use any extra material. The light and direction invariant methods for hand gesture recognition were investigated, which provide natural comfort to the user. Also, the controlling of robotic hand using natural hand gives a feeling of virtual hand to the user and it is much better way than entering the values of finger bending angles. Such a robotic hand has real-life applications in commercial, military, or emergency operations, where human life cannot be risked.

The research problems are discussed in Chap. 1. It gives a brief about the gesture recognition process and its effectiveness with real-time constraints. The goals are defined in Chap. 2. Current state of art in the context of natural computing is reviewed in Chap. 3, where different intelligent and soft computing-based HGR techniques are described. The latest results in real-time performance are also given.

It is also shown that current method detects only open fingers efficiently while bent fingers are either not counted or methods ignore them.

The preprocessing is discussed in Chap. 4, where ROI is extracted from the image frame and image would be cropped. The reduced size image would make the further process faster than before. ROI segmentation is also shown using the specialized device MS KINECT®, where depth information is used for gesture detection. The hand gesture recognition is explained in Chap. 5, which demonstrates light-invariant gesture recognition. Few gestures were already selected for the system and their OH was compared with the test image ROI to classify the gesture. The results are encouraging as in two very different lighting conditions, gestures were identified correctly. Gesture classification was perfomed using Euclidean distance and using ANN.

Chapter 6 explains the process of HGP detections using webcam and KINECT. The fingertip detection is direction invariant in both conditions, the user is free to show the hand in any direction. Chapter 7 describes the calculation of bent fingers' angle first using a geometrical method and later using ANN implementation. The performance analysis is also given for both approaches. Chapter 8 discusses what if both hands are shown by the user. A new concurrent fingertip method was applied to reduce computational time.

Further investigations in this area would include closed finger detections and motion-invariant methods. Although, KINECT is able to detect closed finger positions with few constraints using simple devices it can still be achieved. This work is focused on spatial domain analysis, someone can also investigate these issues in frequency domain- or model-based approaches. The performance on real-time embedded system is still a big issue as the cameras are going to have high resolution, more pixels are needed to process in a spatial domain. The processing power and memory is increasing with camera resolution, but methods to minimize these latencies should be investigated.

New Delhi, India Ankit Chaudhary

Contents

About the Author

Ankit Chaudhary is Ph.D. in Computer Engineering and currently Professor of Computer Science at Northwest Missouri State University, MO, USA. He is an Associate Editor of Computers and Electrical Engineering. His areas of research interest are Computer Vision, Artificial Intelligence, and Machine Learning. He has written more than 70 research papers and edited one book. He has been the Lead Guest Editor for different journals in the area of computer vision.

Acronyms

ANN	Artificial Neural Network
BLOB	Biggest Linked Objects
CCF	Concentric Circular Filter
COP	Centre of Palm
CSF	Circular Separability Filter
DOF	Degree of Freedom
HCI	Human–Computer Interface
HG	Hand Gesture
HGP	Hand Geometry Parameters
HGR	Hand Gesture Recognition
HMI	Human–Machine Interface
OH	Orientation Histogram
ROI	Region of Interest

List of Figures

List of Tables

Chapter 1
Introduction

> Vision is the best gift given to us by the creator out of all our strengths and features. The way of seeing an object and recognizing it seems very easy from human point of view but if you want a machine to do the same then it is probably the most complex task on the earth. Interested researchers are trying to do the same since 40 years but still no machine exists which can recognize any object without ambiguity (Graham 1991).

As computers are becoming increasingly pervasive, there is a growing interest in the development of new approaches and technologies for bridging the human–computer barrier. The aim is to bring human–computer interaction (HCI) to a regime where the interactions with computers become as natural as the interactions between humans. As a part of HCI, incorporating hand gestures into communication methods is an important research area. Hand gesture recognition (HGR) is the natural way of human–machine interaction. Today many researchers in the academia and industry are studying different techniques that make such interactions easier, natural, and convenient without the requirement for any additional devices.

Gesture recognition systems offer to the machine the ability to identify, recognize, and interpret human emotions through gestures. Gesture identification is a natural way to pass emotional signals to the machine, as a human expresses his/her feelings most often through gestures. Generally defined as any meaningful body motion, gestures play a central role in everyday communication and often convey emotional information about the gesticulating person.

This book is particularly focused on identifying hand gestures in natural condition. Techniques for preprocessing and segmentation of the original image, and hand image interpretation are also considered in this book.

A. Chaudhary, *Robust Hand Gesture Recognition for Robotic Hand Control*,
DOI 10.1007/978-981-10-4798-5_1

1.1 Hand Gesture Recognition

Though the research has presented advancements in gesture recognition, further advancements are required for making the gesture recognition process more accurate. The existing research on gesture recognition can be classified into three categories. The first category is the glove-based analysis. This employs sensors attached to a glove that transduce finger flexions into electrical signals for determining the hand posture. This system requires the user to wear a cumbersome device and carry a load of cables that connect the device to a computer. This hinders the ease and naturalness with which a user can interact with a computer-controlled environment. Potentially, the awkwardness in using gloves and other devices can be overcome by video-based noncontact interaction techniques identifying gestures.

The second category comprises the models based on gesture images. Many statistical, probabilistic, and other models have been proposed. One of the techniques in this approach involves building a three-dimensional model of a hand. This model is matched to the hand by one or more cameras and parameters corresponding to the palm orientation and joint angles are estimated. These parameters are used to perform corresponding gesture classification. Additionally, in order to initialize the parameters the user places the hand in a specified position. The parameters can also be manually instantiated.

The third category involves the analysis of drawing gestures. This method usually involves stylus, which is used as an input device. This method also leads to the recognition of written text. The majority of the work on HGR has employed mechanical sensing, most often direct manipulation for the virtual environment and occasionally for symbolic communication. Sensing the hand posture mechanically involves a range of problems including reliability, accuracy, and electromagnetic noise. Visual sensing has the potential to make gestural interaction more practical, but potentially embodies some of the most difficult problems in machine vision. This is because the hand is a nonrigid object and even worse self-occlusion is commonly encountered.

1.1.1 Gesture Recognition Process

The live input of images can be taken using any normal web camera. As a video is a sequence of images, the image sequence would be processed one by one. An image frame may contain many undesired objects in addition to the hand gesture under consideration. To identify the gesture correctly, the part of the image depicting the hand needs to be separated from the part composed of the undesired objects. To separate the skin from the image, a commonly used method involves applying a suitable skin filter with an experimentally obtained threshold. This results in the separation of the skin from the image while the undesired parts of the image can be removed. In image processing, like other fields, the results are not extremely sharp

when preprocessing is performed. There can be noise in the image as the falsely detected skin pixels and also the detected portion can have jagged edges. Noise removal and smoothing operations need to be applied to the image in order to get only the region of interest (ROI) with smooth edges in the resultant images.

After obtaining ROI, the gesture can be classified on the basis of many known techniques. If there is a known set of gestures which could occur in the image, then ROI has to be compared with the collected data. This can be done by storing the gestures and extracting features from them into a database. Consequently, the test image feature vector would be compared against it. Many researchers have also used soft computing techniques for the same purpose.

Many vision-based applications have used fingertips to track or manipulate gestures. This will help in understanding the hand position in the image. Another hand geometry parameter is the center of palm (COP). If the COP and the fingertips are known, the clarity of the hand gesture in the image can be significantly enhanced.

1.1.2 Issues

Hand gestures can be classified into two categories: static and dynamic. A static gesture is a particular hand configuration or pose that is represented by a single image, while a dynamic gesture can be considered as a continuous motion of the hand. The static gesture has only one type of a gesture, while the dynamic gesture can contain more than one gesture during a particular period of time. Gesture recognition from video sequences or an interactive input is one of the most important challenges in real time for researchers working in the area of computer vision and image understanding. It is very complex to perform a required operation on an arbitrary image. One needs to extract the ROI, which makes the work faster and reduces the computational complexity. However, image processing algorithms involve an old and known problem of incremental time. A technique that involves a smaller number of computations is required in order to reduce the time involved in the initial processing.

The skin color in the images varies in different light intensities. The same skin color may seem different when the light intensity changes. Moreover, if different users employ the system, then their skin colors are quite likely to be different. The system needs to detect the ROI for all types of skin colors. This is still a major issue when it comes to installing applications in the public domain. Open finger detection is a known process in vision techniques, which provides accurate results. However, closed finger detection does not provide accurate results. Many methods do not consider the closed finger in the gesture or they count it wrongly. Limited research has been performed in the area of bent fingers' angle calculation using vision-based methods. This area helps mimic the hand operations virtually or on hardware devices.

1.1.3 Applications

The geometry of the hand is complex and it is hence hard to construct it in a virtual environment. However, the functionality and the degree of freedom of the hand encourage researchers to develop a hand-like instrument. During the last few decades, researchers have been interested in automatically recognizing human gestures for several applications, namely sign language recognition, socially assistive robotics, directional indication through pointing, control through gestures, alternative computer interfaces, immersive game technology, virtual controllers, affective computing, and remote-controlled robots. Mobile companies are trying to make handsets which can recognize gestures and hence operate over short distances (Kroeker 2010; Tarrataca et al. 2009). Pickering (Pickering 2005) wrote that initially touch-based gesture interfaces would be popular, while noncontact gesture recognition technologies would be more attractive finally. This is absolutely true today. Recently human gesture recognition catches the peak attention of the research in both software and hardware environments. Hand gesture with a single hand can be very worthy in the case of giving the command to the computer or to a robotic system (Shimizu et al. 2006). However, in most practical cases, both hands are involved. Therefore, taking into account both hands is also a challenge.

1.2 Book Organization

This book targets researchers and engineers to change the current way of developing vision-based applications by bringing the natural computing in the interaction. The user should be able to naturally use vision-based systems as they behave with other humans. Chapter 2 states the goals which have been discussed in this book. Chapter 3 presents the current state of the art with a focus on artificial intelligence (AI) and human-related computing. Chapter 4 discusses the preprocessing of captured images using cameras and sensors. Consequently, the ROI would be obtained. The HGR and light invariant results are discussed in Chap. 5 while different hand geometry parameters and their detection are presented in Chap. 6. Chapter 7 presents the single hand fingers' angle calculation methods using trigonometry and AI, while Chap. 8 presents the angle calculations if both hands are employed for depicting gestures.

References

W. Graham, *Artificial Intelligence in Engineering* (Wiley, West Sussex, 1991)
K.L. Kroeker, Alternate interface technologies emerge. Commun. ACM **53**(2), 13–15 (2010)
C.A. Pickering, in The search for a safer driver interface: a review of gesture recognition human machine interface. IEE Comput. Control Eng. 34–40 (2005)

L. Tarrataca, A.C. Santos, J.M.P. Cardoso, in *The Current Feasibility of Gesture Recognition for a Smartphone Using J2ME.* Proceedings of the ACM Symposium on Applied Computing, pp. 1642–1649 (2009)

M. Shimizu, T. Yoshizuka, H. Miyamoto, in *A Gesture Recognition System Using Stereo Vision and Arm Model Fitting*. Proceedings of Third International Conference on Brain-Inspired Information Technology, Kitakyushu, Japan, pp. 89–92, 27–29 Sept 2006

Chapter 2
Scientific Goals

The following issues in the context of hand gesture based robot control have been covered in this book:

- Improve the state of the art of hand gesture recognition using enhanced intelligent methods
- Investigate a faster method to reduce image preprocessing algorithm time
- Investigate a direction invariant method to detect hand gesture in real time
- Investigate a light intensity invariant method to detect hand gesture in real time
- Explore automated techniques to calculate angles of bending fingers from hand gesture
- Develop an automatic system to detect angles of fingers in motion

These issues are discussed in the implementation sequence to enhance the understanding of the user.

A. Chaudhary, *Robust Hand Gesture Recognition for Robotic Hand Control*,
DOI 10.1007/978-981-10-4798-5_2

Chapter 3
State of the Art

This chapter provides a brief overview of the state-of-the-art methods and results related to hand gesture recognition techniques. These methods are used for comparison and as a stepping stone for next chapters. Here different methods for hand gesture recognition are presented and discuss the major problems in this area which are addressed in this book.

Hand gestures (HG) recognition is one of the major areas of research for the engineers, scientists, and bioinformatics to interpret the human behavior in different scenarios. HG-based interaction with machines is a natural technique for communication which people follow in their general lives. Today many researchers in the academia and industry are studying various applications for making HG-based interactions more easy, natural, and convenient without wearing any extra devices. HG has applications that range from games control to computer vision-enabled robot control, from virtual reality to smart home systems, from security systems to training systems, and into many other areas.

Gesture recognition is a subfield of vision and image understanding that helps in interpreting an image or sequence of images, i.e., video into a meaningful description. In this chapter the state of art in the area of hand gesture recognition is discussed with a focus on the intelligent approaches including soft computing-based methods like artificial neural network, fuzzy logic, genetic algorithms, etc. The methods involved in the preprocessing of an image for segmentation of a region of interest and hand image construction are also studied. Most researchers have used fingertips for hand detection in appearance-based modeling. In this chapter, various applications are described which are focused on fingertip detection based methods. Finally, a comparative study of results presented in various research studies is presented.

A. Chaudhary, *Robust Hand Gesture Recognition for Robotic Hand Control*,
DOI 10.1007/978-981-10-4798-5_3

3.1 Natural Hand Gesture Recognition

The natural gesture research community has identified the fundamental types of natural HG recognition techniques. Natural HG communication is a very active research area in the field of computer vision. Such a communication technique provides the ease to interact with machines without the requirement of any extra device. Moreover, if the users do not have sufficient technical knowledge about the system, they are still able to use the system with their bare hands. When communicating, humans often express their statements through gestures. The gestures are depicted naturally with the words and the gestures help enhance the understanding (of the communication) of the listener. Gestures allow individuals to communicate feelings and thoughts with different emotions with words or without words (Kendon 2004). A human gesture can be any but few have a special meaning.

A human hand can move in any direction and can bend at any angle in all available vector spaces. Chinese sign language as shown in Fig. 3.1 uses HG to represent digits as well as alphabets. In the past, many researchers (Sturman and Zeltzer 1994; Pavlovic et al. 1997; Cho et al. 1999; Huang and Pavlovic 1995; Quek 1994; Wang and Cannon 1993) have tried with different instruments and equipment to measure hand movements like gloves, sensors, or wires. However, in these techniques, a user has to wear the device, which is highly impractical. This deficiency in previous studies motivates the investigation of a technique involving contactless gesture recognition. Such a technique, which provides a natural human-to-human interaction, provides a new research dimension in the area of computer vision.

According to Mitra (Mitra and Acharya 2007), gesture recognition is a process where a user makes a gesture and a receiver recognizes it. Using such a technique, an easy interact with machines is possible and can convey to them a particular message based on the environment and application syntax. Even people who cannot communicate orally (i.e., sick, old, or young child) would also benefit from this technology. Researchers are working to develop such HG-based systems for handicapped people. On the commercial side, mobile companies are trying to make

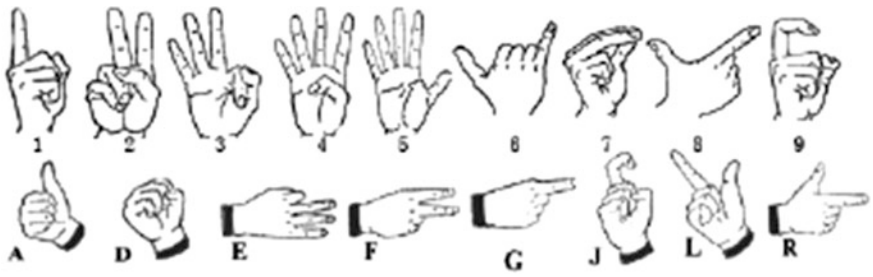

Fig. 3.1 Chinese sign language (Zhang et al. 2009)

handsets which are able to recognize gestures naturally and could operate over small distances (Kroeker 2010; Tarrataca et al. 2009). The focus of this chapter is on human-to-machine interaction (HMI), in which a machine is able to recognize the gesture depicted by a human. There are two types of approaches for HG recognition that are found in the literature. These are as follows:

(a) Appearance-based approaches in which a hand image is reconstructed using the image properties and extraction.
(b) Model-based approaches in which different models for the images are employed and consequently the images are represented on the computer.

Here the approaches are classified based on the method used for HGR. Many approaches have been developed to interact with machines using glove-based systems (Baudel et al. 1993; Sturman and Zeltzer 1994) to neural networks (Nolker and Ritter 2000). Users always prefer an easily implementable and natural technology in HMI and hence prefer technologies that interpret visual inputs (Pavlovic et al. 1997). Fels (Fels and Hinton 1993) implemented a multilayer neural network to transform HG into audio output using gloves. Pickering (Pickering 2005) stated that even though initially touch-based gesture interfaces would be popular, ultimately, noncontact gesture recognition technologies would be in demand. The input to a machine using a gesture is simple and convenient. However, the implementation of such a system involves a number of difficulties as Xu (Xu and Zhu 2009) states "The human hand is a complex deformable object and gesture itself has many characteristics, such as diversities, ambiguities, temporal and spatial differences and human vision itself is an ill-posed problem."

Pickering (Pickering 2005) describes a real-time gesture-based driving system simulator developed at Carnegie Mellon University with the help of General Motors. Many researchers (Do et al. 2006; Premaratne and Nguyen 2007; Kohler 1996; Bretzner et al. 2001; Sawah et al. 2007; Kim and Fellner 2004) have also used a color strip or a full-sleeved shirt to detect a hand image in captured images. A detailed survey on gesture recognition is presented in Mitra and Acharya (2007), Pavlovic et al. (1997), and Ong and Ranganath (2005). Gesture segmentation is a part of the gesture recognition process, which has been reviewed in Xu and Zhu (2009) and Mahmoudi and Parviz (2006) based on color spaces.

Choi (Choi et al. 2001) brings the attention of researchers to an old problem, namely the incrementing processing time of algorithm's complexity. Choi says that "the most important issue in the field of the gesture recognition is the simplification of the algorithm and the reduction of processing time." He uses a morphological operation to implement his system using the center points extracted from primitive elements by morphological shape decomposition. Lu (Lu et al. 2009), Gastaldi (Gastaldi et al. (2005), and Ozer (Ozer et al. (2005) use a parallel approach in the design and implementation of their system. Different threads are implemented in such a manner that they can run in parallel and can compute faster.

Shin (Shin et al. 2004) presented a 3D HG-based system that has application in fruit fly chromosomes based on 2D slices of CT scan images. Lee (Lee et al. 2004)

shows that a system that he developed for remote control systems is also implementable in motion recognition. He uses 3D systems with two or more cameras to detect commands issued by a hand. Villani (Villani et al. 2007) has attempted to develop a system for teaching mathematics to the deaf with an easy user interface. Morimoto (Morimoto et al. 2007) presents an interesting virtual system in which he pushes virtual buttons using fingers in the air and recognizes it by employing 3D sensors.

3.2 Hand Detection Approaches

A number of techniques are presented in the literature for detecting a hand gesture in the acquired image after preprocessing. As discussed previously, these approaches are classified into two categories, namely the appearance-based approaches and the model-based approaches. In this section, a detailed discussion and literature review of these two approaches is presented.

3.2.1 Appearance-Based Approaches

These are the approaches in which the image is considered as a scene in the processing. In these approaches, different types of noise could affect the preprocessing result. In the spatial domain, the light intensity and background of an object play an important role. Many applications are (Nolker and Ritter 2002; Verma and Dev 2009; Nguyen et al. 2009; Shin et al. 2004; Lee and Park 2009; Zhou and Ruan 2006; Gastaldi et al. 2005; El-Sawah et al. 2007; Kim and Fellner 2004; Lee and Chun 2009; Lien and Huang 1998) based on fingertip detection for the hand image construction. In the spatial domain, which is the focus of this research, the fingertip is detected and consequently the hand position is defined.

Nolker (Nolker and Ritter 2002) focuses on a large number of 3D hand postures in her system called GREFIT. She uses fingertips in the hands as the natural determinant of a hand posture to reconstruct the image. A grayscale image of size 192×144 is processed. In her system, she suggests a few approaches to locate the fingertip in a hand, namely

1. Color fingertips and make a histogram, and
2. Use different templates or images of a prototype.

Verma and Dev (2009) has extracted the features from the image that comprised fingertips, edges, and vectors for 2D modeling. He uses the Harris corner detector to extract fingertips corresponding to corners. Nguyen (Nguyen et al. 2009) uses grayscale morphology and geometric calculations to relocate the fingertip locations using learning-based model on a 640×480 pixel size frame. Nguyen uses an approach that is similar to that used by Shin (Shin et al. 2004) to detect both hands

by considering the skin color. To identify the hands, Nguyen (Nguyen et al. 2009) uses a skin segmentation technique based on the Gaussian model. The density function of the skin color distribution is given by

$$\text{Prob}(c \mid \text{skin}) = \sum_{i=1}^{k} \pi_i p_i(c \mid \text{skin}), \tag{3.1}$$

where k is the number of components and π_i is the weight factor of the ith component. He uses CIELUV color space to represent skin and interestingly he uses the palm-to-finger length ratio to construct the hand image. Zhou (Zhou and Ruan 2006) works with 320 × 240 size 24-bit image frames. Zhou uses the Markov random field to remove the noise component in the processed image.

Gastaldi (Gastaldi et al. 2005) obtains the hand perimeter using Gaussian filters and Freeman's algorithm (Freeman 1961) to localize fingertips in the image for 3D detection. Kim (Kim and Fellner 2004) tries to recognize gesture in a dark room on black projection for his system. Although the system is vision based, he used fluorescent white paper to mark fingertips in the captured image. This is not practical for generic purposes since the user has to wear white florescent strips.

Kim uses the Kalman filter for finding fingertips and their correct positions in a recursive manner. Stefan (Stefan et al. 2008) implements a system which can detect the motion of fingers in the air visually. Such a system is used to recognize the numbers from 0 to 9 for command transfer. Ng (Ng and Ranganath 2002) develops a system to recognize the fourteen predefined gestures in 320 × 240 pixel sizes in 24-bit color, where hands are moving and the system is able to work with one or both hands. Ng performs a wrist cutting operation on hand images to make both images invariable.

3.2.2 *Model-Based Approaches*

In these approaches, different models can be constructed to get a result in other forms which can be used to get the relevant image part later. These models are inspired by machine learning or the frequency domain. El-Sawah (El-Sawah et al. 2007) uses histograms for calculating the probability for skin color observation. Hu (Hu et al. 2000) takes the Gaussian distribution for background pixels marking. He then subtracts the pixels from the new image to acquire the gesture image as defined in (3.2). Lee (Lee et al. 2004) uses the same technique to get the hand gesture image:

$$\Delta = |I_n - B|. \tag{3.2}$$

In the modeling of his application of human activity monitoring, Hu (Hu et al. 2000) applies genetic algorithm (GA) to chromosome pool with P_{c0} and P_{m0} as crossover and mutation rate, respectively, which he found using different statistic attributes. Crossover creates new chromosomes while in this case, the mutation introduces new genes into the chromosome. Lee (Lee and Chun 2009) uses the

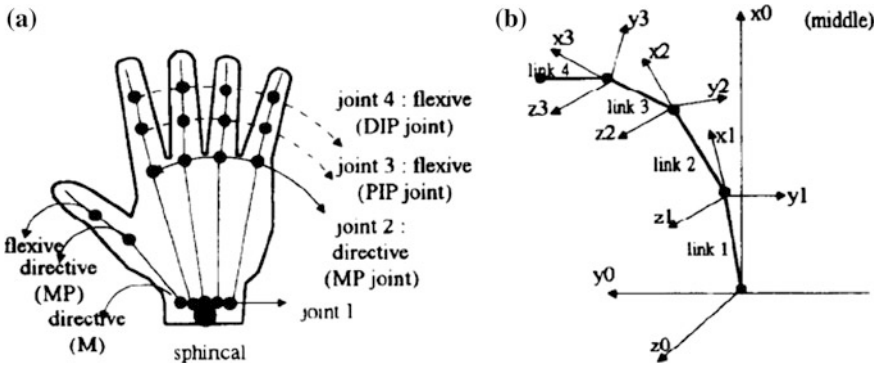

Fig. 3.2 Hand geometry mapping **a** hand model **b** local coordinate frames on the joint position for middle finger (Lien and Huang 1998)

YC_bC_r skin color model to detect the hand region and then he applies the distance transform. Tarrataca (Tarrataca et al. 2009) uses the RGB and HSI color space model based algorithm for skin detection. Malassiotis (Malassiotis and Strintzis 2008) develops a system to recognize real-time HG in German sign language in 3D using a sensor-enabled camera which can recognize the pattern based on illumination and compute the 3D coordinates of each point on the surface. The details about the pattern finding 3D coordinates are presented in his other publications (Tsalakanidou et al. 2005).

Lien (Lien and Huang 1998) presents a model-based system for HGR where the joints in fingers have one degree of freedom (DOF), effective joins have two DOFs and spherical joints have three DOFs. Hence, all fingers have four DOFs, while the thumb has five. He then defines the local coordinate system with the origin located at the joints as shown in Fig. 3.2. This system is interdependent on the fingers movement. He uses the fast fitting method to determine the angle imposed by each finger.

3.3 Soft Computing Approaches

Under the umbrella of soft computing, some of the principal constituents are neural networks, fuzzy systems, machine learning, evolutionary computation, probabilistic reasoning, and their hybrid approaches. In this section, the focus is mainly on three components, which are applied to a wide range of applications. These are as follows:

(a) Artificial neural networks,
(b) Fuzzy logic, and
(c) Genetic algorithm.

3.3.1 Artificial Neural Network

An artificial neural network (ANN) is composed of a number of highly interconnected processing elements (also called neurons) which operate together to solve specific problems (Sivanandam and Deepa 2007). ANN can be configured for solving problems like pattern recognition or data mining through learning-based models. ANN also has capabilities like adaptive learning, self-organizing, and real-time operations using special hardware.

Nolker (Nolker and Ritter 2002) uses an ANN-based layered approach to detect fingertips. The fingertip vectors are obtained and are transformed into finger joint angles to an articulated hand model. For each finger, a separate network is trained on the same feature vectors. The input to each network is a vector of size 35, while the output is only two dimensional. Lee (Lee and Park 2009) uses the hidden Markov model (HMM) for gesture recognition using shape features. Gesture state is determined after stabilizing the image component as open fingers in consecutive frames. He also used the maxima and minima approaches like Raheja (Raheja et al. 2010) for constructing the hand image and FSM like Verma (Verma and Dev 2009) for gesture finalization.

Wang (Wang and Mori 2009) proposed an optical flow based powerful approach for human action recognition using learning models. In this optical flow approach, the hidden parts of the image are also labeled. This max-margin based algorithm can be applied to gesture recognition. In his system Kim (Kim and Fellner 2004) uses a learning model for dynamic gesture recognition. Huang (Huang et al. 2010) uses HMM and RNNs for gesture classification from the collected vectors of hand pose frames. Outputs of both the classifiers are combined to get better results and it was input to the developed GUI. He used Fourier descriptors to represent the boundary of extracted binary hand and trained radial basis function consisted of 56 input nodes, 38 hidden layer, and 5 output nodes. The activation function of the jth hidden node is given by (3.3)

$$\varphi_j(x) = \exp\left(-\frac{||x - c_j||^2}{2\sigma_j^2}\right), \tag{3.3}$$

where x is the input vector, c_j is the center, and σ_j is the spread of $\varphi_j(x)$. Just (Just and Marcel 2009) presents a comparative study of HMM and IOHMM HG recognition techniques on the two openly accessible databases and it is concluded that HMM is a better choice for HG recognition modeling.

Stergiopoulou (Stergiopoulou and Papamarkos 2009) uses an unsupervised self-growing and self-organized neural gas network for 31 prespecified gestures. Although he makes several assumptions, namely the arm should be vertical and the user employs only his right hand while the system is not applicable to left-handed users, the raised fingers detection in the hand is performed by finding the fingertip neuron, which is followed by the other neurons chain as shown in Fig. 3.3.

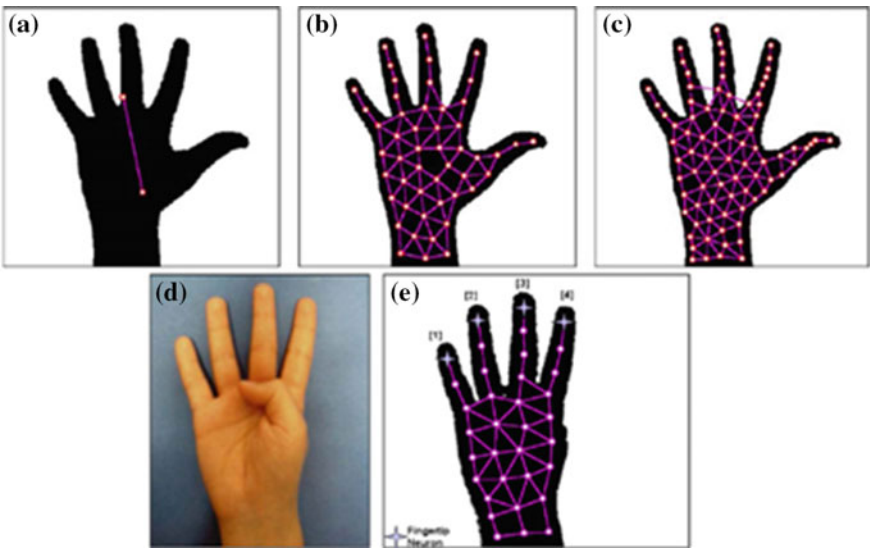

Fig. 3.3 SGONG network working **a** start with two points **b** growing stage with 45 neurons **c** output with 83 neurons **d** hand gesture **e** only raised fingers would be counted (Stergiopoulou and Papamarkos 2009)

The center of the palm can be calculated by employing the gravity method from the neuron only in the palm area and the distance from the fingertips to the palm center is calculated. However, the main problem in gesture recognition is that only the raised fingers can be counted in the presented algorithm (as shown in Fig. 3.3e) and gesture is recognized accordingly. Then he applied a likelihood classification to obtain the gestures that are predefined based on the raised fingers.

3.3.2 Fuzzy Logic-Based Approaches

In the 60s, Lotfi Zadeh (Zadeh 1965), a Professor at UCB USA, presented fuzzy logic in an innovative way. His view was that for precise processing, accurate information is not necessary, and it can even be performed if imprecise data is available. It is more realistic than a computer obtained a binary result. As described in Sivanandam and Deepa (2007) "Fuzzy logic is a multi-valued logic that allows intermediate values to be defined between conventional evaluations." Verma (Verma and Dev 2009) uses c-mean fuzzy clustering based finite-state machines (FSM) to recognize HG. The formula for centroid calculation of fuzzy c-means clusters states that the centroid is the mean of all points weighted by their degree of belongings to the cluster center. Hence, for each point x and a coefficient $u_k(x)$ that gives the degree in the kth cluster, the mean (i.e., center$_k$) is given by (WIKIa)

$$\text{center}_k = \frac{\sum_x u_x(x)^m x}{\sum_x u_x(x)^m}, \tag{3.4}$$

where x_k is the kth trajectory point.

In second phase these clusters map onto FSM states and the final state shows gesture recognition. However, Verma (Verma and Dev 2009) did not implement it. Schlomer (Schlomer et al. 2008) uses the k-mean algorithm on clusters and then applies HMM and Bayes classifier on vector data. Trivino (Trivino and Bailador 2007) tried to make a more descriptive system which can convert human gesture positions into a linguistic description using fuzzy logic. He related it to natural language processing (NLP). He used sensors and took only a few positions into consideration, namely the sitting and standing positions.

3.3.3 Genetic Algorithm Based Approaches

Genetic algorithm originates from the field of biology, but it also has wide applications in optimization performed in computational sciences. This method is very effective when obtaining optimal or suboptimal solutions to problems as it has only a few constraints (Sivanandam and Deepa 2007). It uses to generate and test mechanism over a set of probable solutions (called as the population in GA) and provides an optimal and acceptable solution. It executes its three basic operations (i.e., reproduction, crossover, and mutation) iteratively on the population.

El-Sawah (El-Sawah et al. 2007) focuses on a very generic scenario where he uses a generic nonrestricted environment, generic nonspecific application for gesture recognition using genetic programming. He uses crossover for noise removal in gesture recognition, dynamic Bayesian network (DBN) for gesture segmentation and performs gesture recognition with fuzzification. Hu (Hu et al. 2000) applies the GA on his system to make a 2D parametric model with the human silhouette in his application of human activity monitoring. The best quality of GA is that it operates in parallel at different points, hence performing faster computations.

3.3.4 Other Approaches

There are many other intelligent approaches for hand gesture recognition. Raheja (Raheja et al. 2010) proposes a new methodology for real-time robot control using principal component analysis (PCA) for gesture extraction and pattern recognition with saved images of 60 × 80 image pixels formats in a database. He uses the syntax of a few gestures and determines the corresponding actions of the robot. He claims that the PCA method is significantly faster than the neural network based methods which require training databases and a greater computational power.

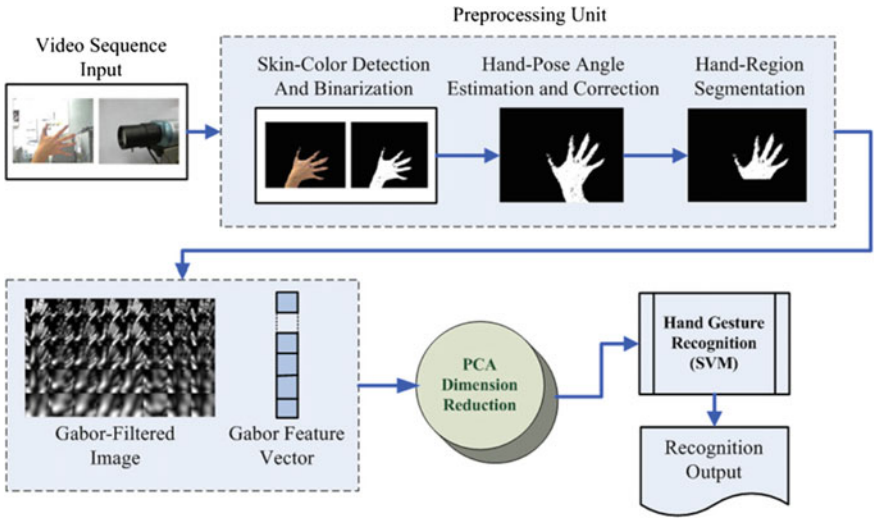

Fig. 3.4 Hand gesture recognition process from the video (Huang et al. 2010)

Huang (Huang et al. 2010) uses PCA for dimensionality reduction and SVM for gesture classification in using skin color model switching for varying illumination environment. In Huang's approach (see Fig. 3.4) image sequences are sent for skin color detection, hand-pose angle estimation, and hand region segmentation. Then the resultant images are divided into 40 20 $\times$ 20 pixels and are run through the Gabor filter. Morimoto (Morimoto et al. 2007) also uses PCA and maxima methods. Gastaldi (Gastaldi et al. 2005) uses PCA to compress five image sequences into one and get eigenvectors and eigenvalues for each gesture. He uses statistical HMM model for gesture recognition. Lee (Lee and Kim 1999) also used HMM with threshold model for predefined gesture recognition.

Zaki (Zaki and Shaheen 2011) uses PCA where the hand representation is transformed from the image coordinates to eigenvector space. After vector rotation, the largest eigenvector is aligned with the mid of data as shown in Fig. 3.5. He used three HMMs for every sign, one for each feature: PCA Sequence, Kurt Pos Sequence, and MCC Sequence. Shin (Shin et al. 2006) presents gesture extraction and recognition using entropy analysis and low-level image processing functions. Lee (Lee et al. 2004) also uses entropy to obtain color information. He uses PIM to quantify the entropy of the image using Eq (3.5):

$$\text{PIM} = \sum_{i=0}^{L-1} h(i) - \text{Max}_j h(i), \tag{3.5}$$

where $h(i)$ is the ith histogram value of each image or block. To acquire the PIM value, all pixels in each block are subtracted from the maximum frequency in the histogram model.

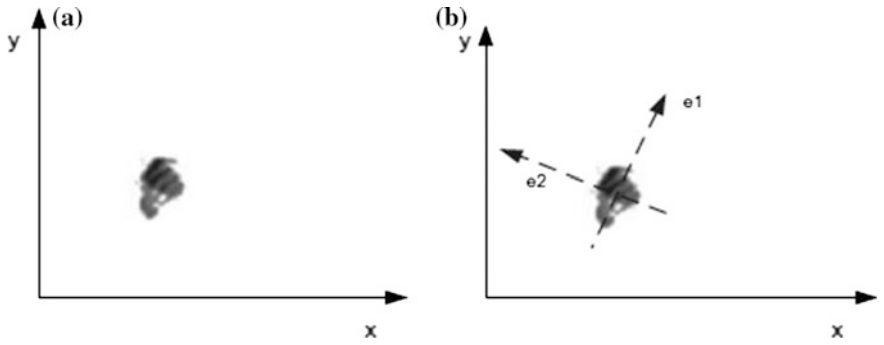

Fig. 3.5 Transformation from hand to eigenspace **a** coordinates and **b** eigenvectors (Zaki and Shaheen 2011)

Lu (Lu et al. 2009) implements a system for 3D gesture recognition where he fuses different positions of a gesture using coordinate transformations and then uses stored prespecified gestures for gesture recognition. Stefan (Stefan et al. 2008) uses dynamic space-time warping (DSTW) (Alon et al. 2005) to recognize a set of gestures. This technique does not require hands to be correctly identified in each frame.

Zou (Zou et al. 2009) uses deterministic finite-state machine (DFSM) to detect hand motion and then applies rule-based techniques for gesture recognition. He defines the gesture into two categories based on motion. These categories include linear and arc-shaped gestures. Tarrataca (Tarrataca et al. 2009) uses convex hull method based clustering algorithm for posture recognition. Chang (Chang et al. 2008) uses a feature alignment approach based on curvature scale space to recognize hand posture.

3.4 Implementation Tools

Most researchers who study image processing use MATLAB® with the image processing toolbox. A few also use C++. Lu (Lu et al. 2009), Lee (Lee and Chun 2009), and Zou (Zou et al. 2009) use C++ for implementation on Windows XP®. Lu (Lu et al. 2009) and Lee (Lee and Chun 2009) use Microsoft® Foundation Classes (MFC) to build user interface and control. Intel OpenCV library is also popular in vision application processing. Even OpenCV has been used with MATLAB® to implement few systems (Suk et al. 2010). Stergiopoulou (Stergiopoulou and Papamarkos 2009) uses Delphi to implement the HGR system using SGONG network.

3.5 Accuracy

Different systems have been discussed in this chapter as well as the approach used to implement them. The systems' accuracy w.r.t. hand gesture recognitions is discussed. GREFIT (Nolker and Ritter 2002) system is able to detect fingertips even when it is located in front of the palm. It is able to construct a 3D image of the hand that is visually comparable. Nguyen (Nguyen et al. 2009) claims that the results are 90–95% accurate for open fingers, but only 10–20% for closed fingers. Figure 3.6 presents a scenario in which closed or bent fingers are located in front of the palm. Hence, the skin color detection cannot differentiate between the palm and the finger. According to Nguyen the image quality and morphology operator are the main reasons for low accuracy in detection.

Raheja (Raheja et al. 2010) claims approximately 90% accuracy in his results if the lighting conditions are good. Hu (Hu et al. 2000) uses six different parameters to control the performance of the system. For scenarios involving a significant amount of noise, the noise is controlled by employing two parameters. Stergiopoulou (Stergiopoulou and Papamarkos 2009) claims about 90.45% accuracy, though the hidden finger is not detected in his approach. Morimoto (Morimoto et al. 2007) claims for his system approximately 91% accuracy after applying normalization. Ng (Ng and Ranganath 2002) claims 91.9% correct results for his approach that involves a combination of HMM and RNNs. Huang (Huang et al. 2010) claims 93.7% recognition results using Gabor filters.

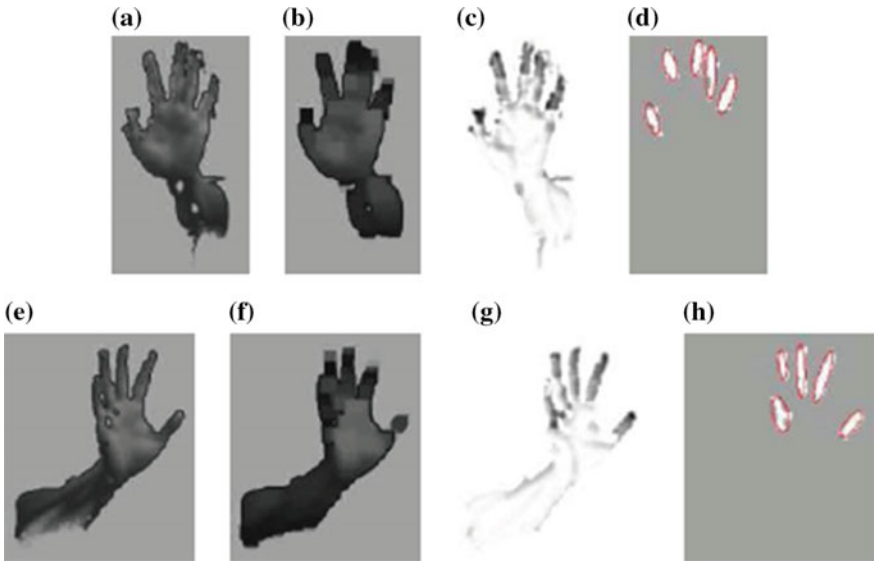

Fig. 3.6 Result of finger extraction using grayscale morphology operators and object analysis (Nguyen et al. 2009) which work for bent finger also, but with a lower accuracy, i.e., 10–20%

Lee (Lee et al. 2004) shows results for six kinds of gestures with a recognition rate of more than 95%. The focus of the research is only on whether the finger is bent and not on the degree of bending. Stefan (Stefan et al. 2008) achieves 96% accuracy over 300 tests. He also states that parallel image processing and pattern matching operations are not real-time compatible in MATLAB®, but can be faster if implemented in C++. For running gestures like *Bye* and *showing a direction*, Suk (Suk et al. 2010) claims 80.77% accurate results.

3.6 Conclusion

It is discussed in this chapter that a wide range of literature is available on open hand gesture recognition. However, little research work has been performed in the area of static closed fingers identification. Moreover, no previous study has considered the bent fingers' angle calculation in 2D. Nolker has conducted such a modeling only in 3D space.

In order to recognize the hand gesture, one needs to extract the required information from the video and for that the live image sequence needs to be segmented. The preprocessing of input images is required for segmentation, noise removal, and smoothen them. In the next chapter, the preprocessing is discussed for hand segmentation and noise removal to obtain the region of interest when the input is a live stream of hand gesture images, and a novel faster algorithm is also presented to reduce processing time.

References

A. Kendon, *Gesture: Visible Action as Utterance* (Cambridge University Press, UK, 2004)

J. Zhang, H. Lin, M. Zhao, in *A Fast Algorithm for Hand Gesture Recognition Using Relief*. Proceedings of Sixth International Conference on Fuzzy Systems and Knowledge Discovery, Tinajin, China, pp. 8–12, 14–16 Aug 2009

D. Sturman, D. Zeltzer, A survey of glove-based input. IEEE Trans. Comput. Graph. Appl. **14**(1), 30–39 (1994)

V.I. Pavlovic, R. Sharma, T.S. Huang, Visual interpretation of hand gestures for human-computer interaction: a review. IEEE Trans. Pattern Anal. Mach. Intell. **19**, 677–695 (1997)

O.Y. Cho, H.G. Kim, S.J. Ko, S.C. Ahn, A hand gesture recognition system for interactive virtual environment. IEEK **36-s**(4), 70–82 (1999)

S. Mitra, T. Acharya, Gesture recognition: a survey. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **37**(3), 2127–2130 (2007)

K.L. Kroeker, Alternate interface technologies emerge. Commun. ACM **53**(2), 13–15 (2010)

L. Tarrataca, A.C. Santos, J.M.P. Cardoso, in *The Current Feasibility of Gesture Recognition for a Smartphone Using J2ME*. Proceedings of the ACM Symposium on Applied Computing, pp. 1642–1649 (2009)

T. Baudel, M.B. Lafon, Charade: remote control of objects using free-hand gestures. Commun. ACM **36**(7), 28–35 (1993)

C. Nolker, H. Ritter, in *Parameterized SOMs for Hand Posture Reconstruction*. Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, Portland, USA, vol. 4, pp. 139–144, Nov 2000

S.S. Fels, G.E. Hinton, Glove-talk: a neural network interface between a data-glove and a speech synthesizer. IEEE Trans. Neural Netw. **4**, 2–8 (1993)

C.A. Pickering, The search for a safer driver interface: a review of gesture recognition human machine interface. IEE Comput. Control Eng. 34–40 (2005)

Z. Xu, H. Zhu, in *Vision-Based Detection of Dynamic Gesture*. Proceedings of the International Conference on Test and Measurement, pp. 223–226, 5–6 Dec 2009

J.H. Do, J.W. Jung, S. Jung, H. Jang, Z. Bien, in *Advanced Soft Remote Control System Using Hand Gestures*. MICAI (Advances in Artificial Intelligence), LNAI, vol. 4293, pp. 745–755 (2006)

P. Premaratne, Q. Nguyen, Consumer electronics control system based on hand gesture moment invariants. IET Comput. Vision **1**(1), 35–41 (2007)

M. Kohler, in *Vision Based Remote Control in Intelligent Home Environments*. 3D Image Analysis and Synthesis, pp. 147–154 (1996)

L. Bretzner, I. Laptev, T. Lindeberg, S. Lenman, Y. Sundblad, A prototype system for computer vision based human computer interaction. Technical report ISRN KTH/NA/P-01/09-SE (2001)

A. El-Sawah, C. Joslin, N.D. Georganas, E.M. Petriu, in *A Framework for 3D Hand Tracking and Gesture Recognition Using Elements of Genetic Programming*. Proceedings of the Fourth Canadian Conference on Computer and Robot Vision, pp. 495–502, Montreal, Canada, 28–30 May 2007

H. Kim, D.W. Fellner, in *Interaction with Hand Gesture for a Back-Projection Wall*. Proceedings of Computer Graphics International, pp. 395–402, 19 June 2004

S.C.W. Ong, S. Ranganath, Automatic sign language analysis: a Survey and the future beyond lexical meaning. IEEE Trans. Pattern Anal. Mac. Intell. **27**(6) (2005)

F. Mahmoudi, M. Parviz, in *Visual Hand Tracking Algorithms*. Geometric Modeling and Imaging-New Trends, pp. 228–232, 16–18 Aug 2006

J. Choi, N. Ko, D. Ko, in *Morphological Gesture Recognition Algorithm*. Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology, Coimbra, Portugal, pp. 291–296, 19–22 Aug 2001

G. Lu, L.-K. Shark, G. Hall, U. Zeshan, in *Dynamic Hand Gesture Tracking and Recognition for Real Time Immersive Virtual Object Manipulation*. Proceedings of the International Conference on Cyber Worlds, pp. 29–35, 7–11 Sept 2009

G. Gastaldi, A. Pareschi, S.P. Sabatini, F. Solari, G.M. Bisio, in *A Man-Machine Communication System Based on the Visual Analysis of Dynamic Gestures*. Proceedings of IEEE International Conference on Image Processing, Genoa, Italy, vol. 3, pp. III, 397–400, 11–14 Sept 2005

I.B. Ozer, T. Lu, W. Wolf, Design of a real time gesture recognition system: high Performance through algorithms and software. IEEE Sig. Process. Mag. 57–64 (2005)

M.C. Shin, L.V. Tsap, D.B. Goldgof, Gesture recognition using Bezier curves for visualization navigation from registered 3-D data. Pattern Recogn. **37**(5), 1011–1024 (2004)

J. Lee, Y. Lee, E. Lee, S. Hong, in *Hand Region Extraction and Gesture Recognition from Video Stream with Complex Background Through Entropy Analysis*. Proceedings of the Twenty Sixth Annual International Conference of the IEEE Engineering in Medicine and Biology Society, San Francisco, CA, USA, pp. 1513–1516, 1–5 Sept 2004

N.A. Villani, J. Heisler, L. Arns, in *Two Gesture Recognition Systems for Immersive Math Education of the Deaf*. Proceedings of the First International Conference on Immersive Telecommunications, Bussolengo, Verona, Italy, Oct 2007

K. Morimoto, C. Miyajima, N. Kitaoka, K. Itou, K. Takeda, in *Statistical Segmentation and Recognition of Fingertip Trajectories for a Gesture Interface*. Proceedings of the Ninth International Conference on Multimodal Interfaces, Nagoya, Aichi, Japan, pp. 54–57, 12–15 Nov 2007

C. Nolker, H. Ritter, Visual recognition of continuous hand postures. IEEE Trans. Neural Netw. **13** (4), 983–994 (2002)

R. Verma, A. Dev, in *Vision-Based Hand Gesture Recognition Using Finite State Machines and Fuzzy Logic*. Proceedings of the International Conference on Ultra-Modern Telecommunications & Workshops, pp. 1–6, 12–14 Oct 2009

D.D. Nguyen, T.C. Pham, J.W. Jeon, in *Fingertip Detection with Morphology and Geometric Calculation*. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, USA, pp. 1460–1465, 11–15 Oct 2009

D. Lee, Y. Park, Vision-based remote control system by motion detection and open finger counting. IEEE Trans. Consum. Electron. **55**(4), 2308–2313 (2009)

H. Zhou, Q. Ruan, in *A Real-Time Gesture Recognition Algorithm on Video Surveillance*. Proceedings of Eighth International Conference on Signal Processing, Beijing, China, pp. 1754–1757, Nov 2006

B. Lee, J. Chun, in Manipulation of Virtual Objects in Marker-Less AR System by Fingertip Tracking and Hand Gesture Recognition. Proceedings of the Second International Conference on Interaction Science: Information Technology, Culture and Human, Seoul, Korea, pp. 1110–1115 (2009)

C.C. Lien, C. Huang, Model-based articulated hand motion tracking for gesture recognition. Image Vision Comput. **16**(2), 121–134 (1998)

H. Freeman, On the encoding of arbitrary geometric configurations. IRE Trans. Elect. Comput. **EC-10**(2), 260–268 (1961)

A. Stefan, V. Athitsos, J. Alon, S. Sclaroff, in *Translation and Scale Invariant Gesture Recognition in Complex Scenes*. Proceedings of Firth International Conference on Pervasive Technologies Related to Assistive Environments, Greece, July 2008

C.W. Ng, S. Ranganath, Real-time gesture recognition system and application. Image Vis. Comput. **20**(13–14), 993–1007 (2002)

C. Hu, Q. Yu, Y. Li, S. Ma, in *Extraction of Parametric Human Model for Posture Recognition Using Genetic Algorithm*. Proceedings of Fourth IEEE International Conference on Automatic Face and Gesture Recognition, Grenoble, France, pp. 518–523, 28–30 Mar 2000

S. Malassiotis, M.G. Strintzis, Real-time hand posture recognition using range data. Image Vis. Comput. **26**(7), 1027–1037, 2 July 2008

F. Tsalakanidou, F. Forster, S. Malassiotis, M.G. Strintzis, in *Real-Time Acquisition of Depth and Color Images Using Structured Light and Its Application to 3D Face Recognition*. Real-Time Imaging, Special Issue on Multi-Dimensional Image Processing, vol. 11, Issue 5–6, Oct–Dec 2005

S.N. Sivanandam, S.N. Deepa, *Principles of Soft Computing* (Wiley India Edition, New Delhi, 2007)

J.L. Raheja, R. Shyam, U. Kumar, P.B. Prasad, in *Real-Time Robotic Hand Control Using Hand Gesture*. Proceedings of the Second International Conference on Machine Learning and Computing, Bangalore, India, pp. 12–16, 9–11 Feb 2010

Y. Wang, G. Mori, in *Max-Margin Hidden Conditional Random Fields for Human Action Recognition*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Miami, Florida, USA, pp. 872–879, 20–25 June 2009

D. Huang, W. Hu, S. Chang, Gabor filter-based hand-pose angle estimation for hand gesture recognition under varying illumination. Expert Syst. Appl. **38**(5), 6031–6042 (2010)

E. Stergiopoulou, N. Papamarkos, Hand gesture recognition using a neural network shape fitting technique. Eng. Appl. Artif. Intell. **22**(8), 1141–1158 (2009)

L.A. Zadeh, Fuzzy sets. Inf. Control **8**, 338–353 (1965)

T. Schlomer, B. Poppinga, N. Henze, S. Boll, in *Gesture Recognition with a Wii Controller*. Proceedings of the Second International Conference and Embedded Interaction, Bonn, Germany, pp. 11–14, 18–20 Feb 2008

G. Trivino, G. Bailador, in *Linguistic Description of Human Body Posture Using Fuzzy Logic and Several Levels of Abstraction*. Proceedings of the IEEE Conference on Computational Intelligence for Measurement Systems and Applications, Ostuni, Italy, pp. 105–109, 27–29 June 2007

H.-K. Lee, J.H. Kim, HMM based threshold model approach for gesture recognition. IEEE Trans. Pattern Anal. Mach. Intell. **21**(10), 961–973 (1999)

M.M. Zaki, S.I. Shaheen, Sign language recognition using a combination of new vision based features. Patt. Recog. Lett. **32**(4), 572–577 (2011)

J.H. Shin, J.S. Lee, S.-K. Kil, D.F. Shen, J.G. Ryu, E.H. Lee, H.K. Min, S.H. Hong, Hand region extraction and gesture recognition using entropy analysis. Int. J. Comput. Sci. Netw. Secur. **6** (2A) (2006)

J. Alon, V. Athitsos, Q. Yuan, S. Sclaroff, Simultaneous localization and recognition of dynamic hand gestures. IEEE Workshop Mot. Video Comput. **2**, 254–260 (2005)

S. Zou, H. Xiao, H. Wan, X. Zhou, in *Vision-Based Hand Interaction and Its Application in Pervasive Games*. Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry, Yokohama, Japan, pp. 157–162 (2009)

C. Chang, C. Liu, W. Tai, Feature alignment approach for hand posture recognition based on curvature scale space. Neurocomput. Neurocomput. Vis. Res. Adv. Blind Sig. Process. **71**(10–12), 1947–1953 (2008)

H. Suk, B. Sin, S. Lee, Hand gesture recognition based on dynamic Bayesian network framework. Pattern Recogn. **43**(9), 3059–3072 (2010)

T.S. Huang, V.I. Pavlovic, in *Hand Gesture Modeling, Analysis and Synthesis*. Proceedings of International Workshop on Automatic Face and Gesture Recognition, Zurich, pp. 73–79 (1995)

F.K.H. Quek, *in Toward a Vision-Based Hand Gesture Interface*. Proceedings of the Virtual Reality System Technology Conference, pp. 17–29 (1994)

C. Wang, D.J. Cannon, in *A Virtual End-Effector Pointing System in Point-And-Direct Robotics for Inspection of Surface Flaws Using a Neural Network-Based Skeleton Transform*. Proceedings of IEEE International Conference on Robotics and Automation, vol. 3, pp. 784–789, 2–6 May (1993)

A. Just, S. Marcel, A comparative study of two state-of-the-art sequence processing techniques for hand gesture recognition. Comput. Vis. Image Underst. **113**(4), 532–543 (2009)

Wikipedia.org (Online). Available: http://en.wikipedia.org/wiki/Cluster_analysis#Fuzzy_c-means_clustering ([WIKIa])

Chapter 4
Hand Image Segmentation

It has been discussed in Chap. 3 that a number of upcoming as well as existing applications are based on hand gesture recognition (HGR) techniques involving a bare hand. Such techniques allow a natural communication with machines. HGR-based systems face many problems in skin segmentation due to luminance and intensity in images which is a major cause of noise in the preprocessed results. The HGR-based systems mostly make assumptions about the hand direction. This causes restrictions in the natural expression of humans. Many systems are based on the assumption that the users have to show the hand in straight position, with the fingers pointing upward. Processing time is another key factor that needs to be considered when designing image-based processing algorithms.

In this chapter, the focus is on direction invariant segmentation of a natural hand from captured image frame with real-time performance. It is considered that there is no restriction on the gesture direction. Moreover, in the context of natural computing, there is no requirement for gloves, sensors, or color strips to segment the hand. The only assumption is that the user presents the hand to the system such that the palm faces the system while the direction of the hand is not restricted. The user is free to move the hand in any direction naturally as the hands move. This chapter presents a novel image cropping preprocessing algorithm, which fastens the gesture recognition process. This is done by performing cropping and hence reducing the number of processing pixels. A detailed discussion on hand geometry parameters (HGP), i.e., fingertips and COP detections is presented in Chap. 6. Figure 4.1 presents a block diagram of the entire process.

A novel method of hand segmentation with the help of MS KINECT is also discussed in this chapter. KINECT facilitates by providing the depth information of foreground objects. The gesture parts are segmented using the depth vector given by KINECT depth sensor.

A. Chaudhary, *Robust Hand Gesture Recognition for Robotic Hand Control*,
DOI 10.1007/978-981-10-4798-5_4

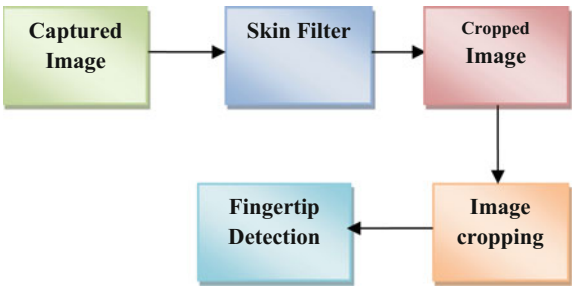

Fig. 4.1 Algorithm flow for the preprocessing method

4.1 Related Approaches

A lot of work has been done in this area of dynamic hand gesture recognition using different techniques. In Chap. 3, it was discussed that there are several issues in the existing approaches that are yet to be solved. Garg (Garg et al. 2009) used 3D images in their method to recognize the hand gesture. However, this process is complex and also not time-efficient. The processing time is one of the very critical factors in real-time applications (Ozer et al. 2005). Aznaveh (Aznaveh et al. 2008) presented a RGB vector-based method for skin detection in images.

Researchers (Brown and Thomas 2000; Crowley et al. 1995; Quek et al. 1995; Tomita and Ishii 1994) have assumed that the hand is always pointing upward to get a precise localization. Few other studies are dependent on specialized instruments and setup like the use of infrared camera (Oka et al. 2002a), stereo camera (Ying et al. 2008), and a fixed background (Crowley et al. 1995; Quek and Mysliwiec et al. 1995) or use of markers on hand for segmentation. Researchers (Oka et al. 2002a, b; Sato et al. 2000) have used infrared cameras to get a reliable segmentation. Generally, image-based models operate pixel by pixel and perform hand segmentation working only on skin pixels, which is the region of interest. However, most hand segmentation methods cannot perform accurate hand segmentation under some conditions, for instance, fast hand motion, cluttered background, and poor light condition (Hardenberg and Berard 2001). If the hand segmentation is not accurate, then the detection of fingertips could become questionable.

Few researchers (Crowley et al. 1995; Hardenberg and Berard 2001; Keaton 2002; Quek et al. 1995; Tomita and Ishii 1994; Wu et al. 2000) in their work limit the degree of the background clutter, finger motion speed or light conditions to get a reliable segmentation. Few others also use 3D mapping using specialized devices like MS KINECT for hand segmentation. This chapter describes a novel method of dynamic hand segmentation without using any kind of sensor or marker.

4.2 Hand Segmentation

A video is a sequence of image frames taken at a fixed rate. In this experimental setup, all images are captured continuously with a simple web camera in 2D and are processed one by one as shown in Fig. 4.2. The process of hand segmentation is discussed in three steps. First, an HSV color space based skin filter is applied on the images for hand segmentation. Second, an intensity-based histogram is generated for the hand direction detection. Third, the image is cropped so that the resultant image is composed of only the gesture pixels.

4.2.1 Skin Filter

The skin filter used on the input image can be based on HSV or YC_bC_r color space to reduce the light effect to some extent. In the HSV color space, the skin is filtered using the chromaticity (hue and saturation) values while in the YC_bC_r color space, the C_b, C_r values are used for skin filtering. A HSV color space based skin filter is applied to the current image frame for hand segmentation. This color space separates three components: hue (H), saturation (S), and brightness (I, V or L).

Essentially HSV color spaces are deformations of the RGB color cube and they can be mapped from the RGB space via a nonlinear transformation. The reason behind the selection of this color space in skin detection is that it allows users to intuitively specify the boundary of the skin color class in terms of the hue and saturation. As I, V, or L give the brightness information, they are often dropped to reduce illumination dependency of skin color. The result of applying skin filter is shown in Fig. 4.3.

The skin filters are used to create a binary image with the background in black color and the hand region in white. Binary images are bi-level images where each pixel is stored as a single bit (0 or 1). Smoothening of the resultant image is needed, as the output image may contain some jagged edges. This binary image is smoothened using the averaging filter.

Fig. 4.2 System prototype

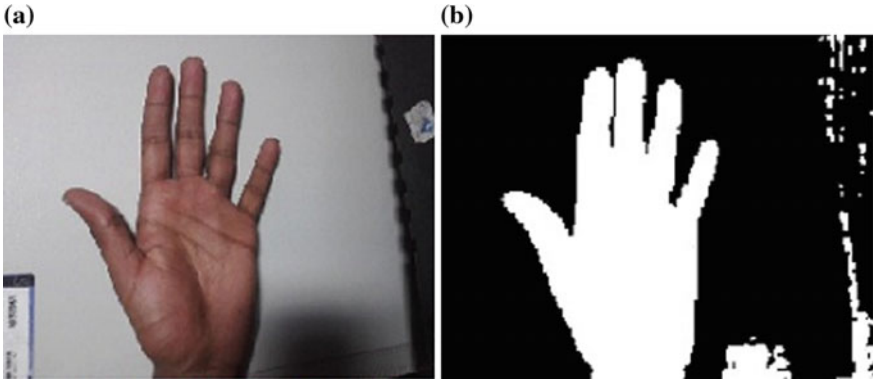

Fig. 4.3 Skin-filtering results. **a** Initial hand image. **b** Binary silhouette

There can be false positives in the results due to falsely detected skin pixels or some skin color objects (like wood) in the background. This can generate few unwanted spots in the output image as shown in Fig. 4.3b. Hence to remove these errors, the biggest BLOB (Binary Linked Object) is considered as the hand mask and rest are considered in the background as shown in Fig. 4.4a. The biggest BLOB represents hand coordinates as '1' and '0' for the background. The filtered hand image obtained after removing all errors is shown in Fig. 4.4b. The only limitation of this filter is that the BLOB for the hand should be the biggest one. In this masking, the background is eliminated. Therefore, falsely detected skin pixels do not exist in the background.

4.2.2 Hand Direction Detection

In this system, the user can give direction free input by presenting a hand gesture to the camera. For obtaining better and unambiguous results, it is necessary to find out the direction of the hand. For doing so, a 4-way scan of the preprocessed image is performed as shown in Fig. 4.5 and histograms are generated based on the skin color intensity in each direction. In all four scans, the maximum value of skin pixels is selected from the histograms. It is oblivious that the maximum value of skin pixels in the image represents the wrist end and the opposite end of the scan represents the finger end. The functions employed for generating the intensity histograms are presented in (4.1) and (4.2).

$$H_x = \sum_{y=1}^{n} \mathrm{imb}(x, y) \tag{4.1}$$

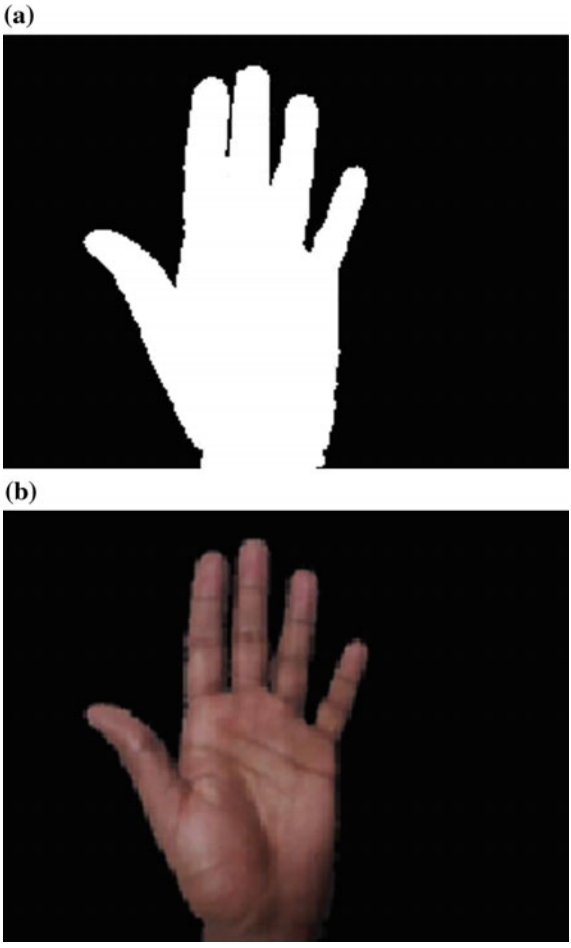

Fig. 4.4 BLOB results. **a** Biggest BLOB. **b** Hand after filtration

$$H_y = \sum_{x=1}^{m} \text{imb}(x, y), \tag{4.2}$$

where imb represents the binary silhouette and m and n represent the rows and columns, respectively, of the matrix imb.

The yellow bar shown in Fig. 4.5 corresponds to the first skin pixel in the binary silhouette scanned from the left direction. The green bar corresponds to the scan performed from the right direction, the red bar corresponds to the scan performed from the downward direction, and pink bar corresponds to the scan performed from the upward direction.

It is clear that the red bar has a higher magnitude compared to the other bars for this particular gesture image. Hence, it can be inferred that the wrist end is in the downward direction of the frame and consequently the direction of fingers is in the upward direction. Hence, the direction from the wrist to the fingers is identified.

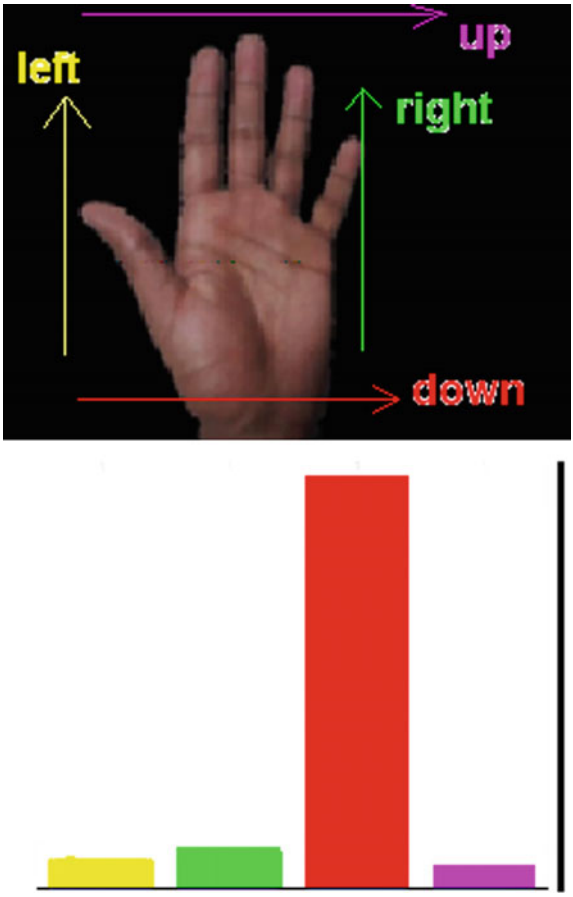

Fig. 4.5 Image scanning and corresponding bars

4.2.3 Hand Cropping

In Sect. 4.2.1, the region of interest has been detected from the image frame, while the remaining part of the image is rendered useless. The part of the image shown in Fig. 4.4b that is black (which is the undesired part of the image) forms a significant portion of the complete image. Hence, if it is possible to reduce or crop the image such that only the ROI is present, a significant amount of computational time can be reduced. In the histograms which were generated in Sect. 4.2.2, it was observed that at the point where wrist ends, a steep inclination of the magnitude of skin pixels in the histogram starts, the slope m of which can be obtained from

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{4.3}$$

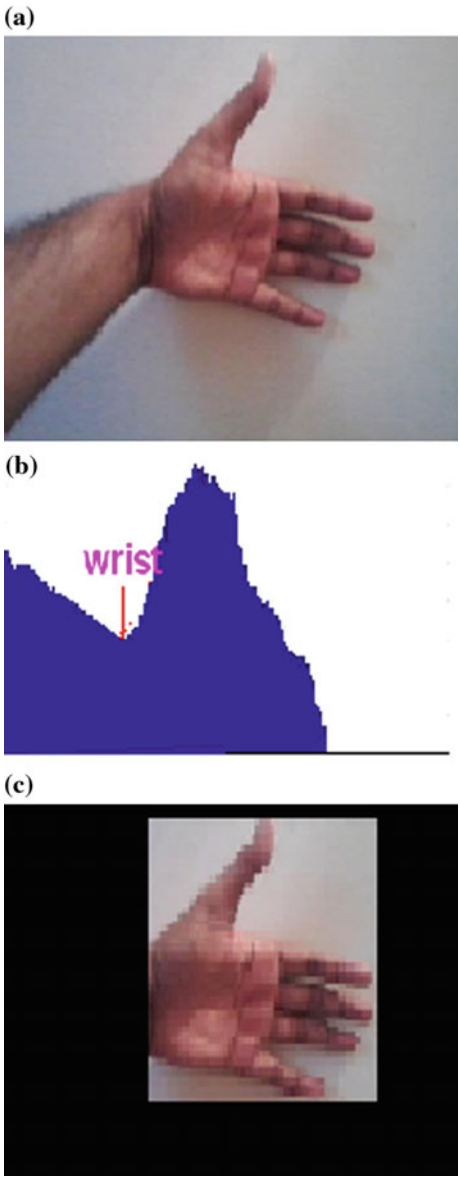

Fig. 4.6 Hand cropping process: images shown are **a** initial image, **b** histogram of binary silhouette where the wrist end is clearly detected, **c** cropped hand image

Figure 4.6 presents the intensity histogram for skin color, where a sharp inclination is clearly visible. Now, consider the last pixel of the skin color in each direction and consequently obtain the tangent on that point. The image is cut at these tangents in all four directions. As the hand direction is already known from Sect. 4.2.2, start a scan from the wrist end of the resultant image of Sect. 4.2.1. The inclination point in the binary image is obtained and the image is cut at this point. In

other three directions, start a scan in the image and stop when the first skin color pixel in each direction is obtained.

The points corresponding to the first skin pixel known from performing scanning from all sides are obtained. Then, crop the image at these points by drawing virtual tangents at these points. The expressions, based on which the image cropping is performed, are given by:

$$\text{imcrop} = \left\{ \begin{array}{ll} \text{origin}_{\text{image}} & ; X_{\min} < X < X_{\max} \\ & ; Y_{\min} < Y < Y_{\max} \\ 0, & ; \text{elsewhere} \end{array} \right\}, \tag{4.4}$$

where imcrop represents the cropped image and $X_{\min}$, $Y_{\min}$, $X_{\max}$, and $Y_{\max}$ represent the boundary of the hand in the image.

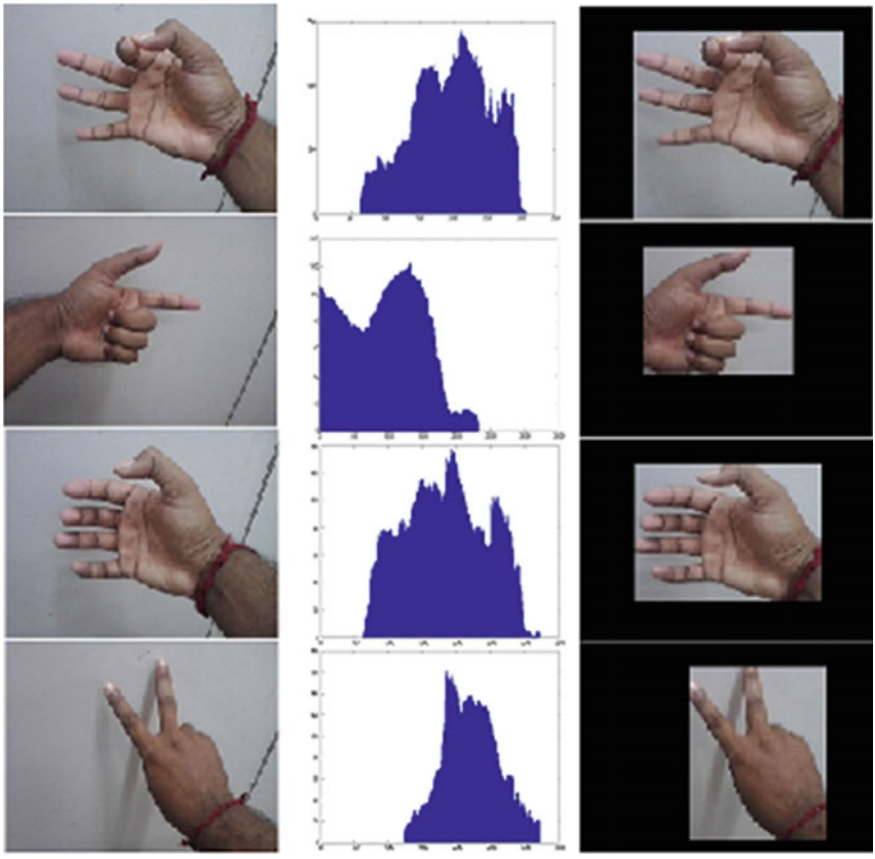

Fig. 4.7 Results of the hand cropping process obtained from live images

Some results with processing steps for hand cropping are shown in Fig. 4.7. In all of the histograms presented in Fig. 4.7, it is clear that at the wrist point a steep inclination begins in the scanning direction.

4.3 Hand Segmentation Using KINECT

Many researchers have proposed different methods for dynamic hand gesture recognition using various sensors. This has led to an era of research that is focussed on the existing problem in the area of HGR of obtaining the depth information of the object. Microsoft KINECT is an example. KINECT is able to detect individual finger motion in three dimensions. A model of KINECT is shown in Fig. 4.8.

4.3.1 Microsoft KINECT Architecture

KINECT provides a RGB view and an infrared view of the object as well as the distance information of the object. The internal architecture of MS KINECT is shown in Fig. 4.9. It consists of an infrared camera and a PrimeSense sensor to compute the depth of the object while the RGB camera is used to capture the images. As Frati (Frati and Prattichizzo 2011) stated "It has a webcam–like structure and allows users to control and interact with a virtual world through a natural user interface, using gestures, spoken commands or presented objects and images."

It is clear that KINECT is a robust device and can be used in various complex applications. The depth images and RGB images of the object can be obtained at the same time. It has a 3D scanner system called light coding which employs a variant of image-based 3D reconstruction. The depth output of KINECT is 11 bit with 2048 levels of sensitivity [WIKIb].

Fig. 4.8 MS KINECT

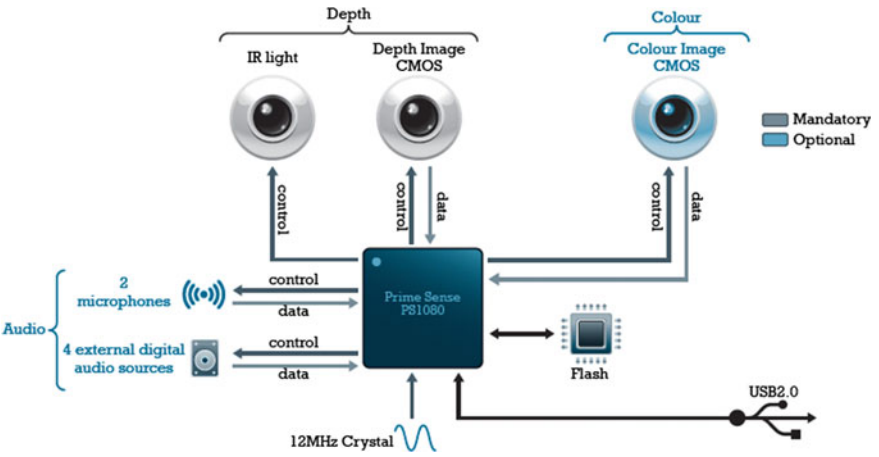

Fig. 4.9 MS KINECT architecture (TET 2011)

4.3.2 Related Approaches

Garg (Garg et al. 2009), Nolker (Nolker and Ritter 2002) and a few researchers have used 3D models in their techniques for recognizing the hand gesture. However, this process is complicated and inefficient. It is essential that due importance is given to accuracy as well as efficiency because processing time is a very critical factor in real-time applications. Few other methods based on specialized instruments for better hand segmentation (Oka et al. 2002a; Ying et al. 2008; Crowley et al. 1995; Quek et al. 1995) or use of markers on hand have been proposed. These methods are more robust and reliable as they are based on the depth information provided by the KINECT, while for a simple webcam these are based on segmentation methods.

4.3.3 Hand Segmentation in 3D

Now we know that the infrared sensor provides the depth information of objects that are in its view. The different depth values are represented by different colors on the screen. The image with its depth information is presented in Fig. 4.10.

The depth value d_{raw} of a point in 3D can be defined through calibration procedure (Frati and Prattichizzo 2011), namely

$$d = K \tan(Hd_{\text{raw}} + L) - O, \tag{4.5}$$

where d is the depth of that point in cm, H equals 3.5×10^{-4} rad, K equals 12.36 cm, L equals 1.18 rad, and O equals 3.7 cm. This tangential approximation has a sum of squared difference of 0.33 cm^2 for the calibration data. After getting

Fig. 4.10 Depth image acquired using KINECT

the depth information, the 3D coordinates of any object can be computed. Let (i, j) be the coordinates of the perspective projection of any point onto the KINECT camera frame that forms a 3D system for this point represented by (x, y, z). Then, the 3D vectors can be obtained from (Fisher 2010).

$$x = (i - c_x) f_x d \tag{4.6}$$

$$y = (j - c_y) f_y d \tag{4.7}$$

$$z = d, \tag{4.8}$$

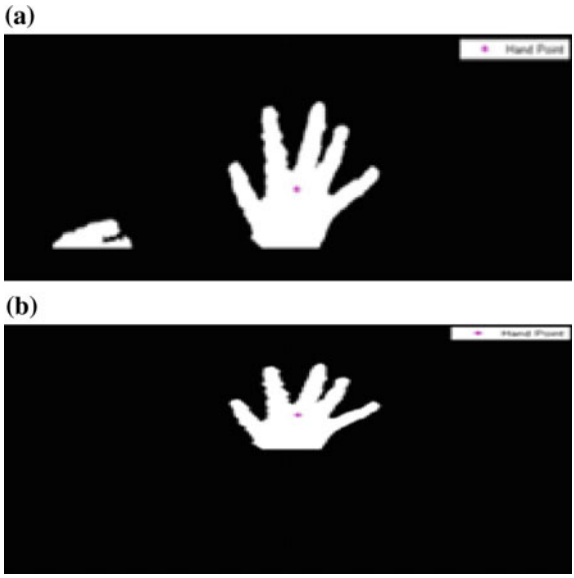

Fig. 4.11 Segmentation using KINECT results **a** threshold image, **b** image of one hand

where f_x equals 0.5942143, f_y equals 0.5910405, c_x equals 339.3078, and c_y equals 242.739.

After obtaining the 3D points, hand tracking and point detection are performed using NITE modules, which use Bayesian object localization for hand detection. OpenNI modules, which provide C++ based APIs, is used for hand detection and tracking.

The depth image is segmented after applying a calculated threshold on the depth of hand points and the hands are detected by choosing the blob which contains the hand point. The result obtained when this process is implemented is presented in Fig. 4.11.

4.4 Conclusion

In this chapter, techniques are presented for hand segmentation implemented on a live stream of images. The hand images from the live video are segmented using a simple webcam as well as the MS KINECT. Encouraging results have been obtained. The presented method is able to segment image irrespective of the hand direction. This is a stepping stone toward the target, i.e., hand gesture recognition.

Different users of the system would have different skin colors and the skin color also would be varied in changed light intensity. For the robust hand gesture recognition, a light intensity invariant method is needed. This light conditions invariant HGR technique is discussed in Chap. 5.

References

M.M. Aznaveh, H. Mirzaei, E. Roshan, M. Saraee, A new color based method for skin detection using RGB vector space, in *Proceedings of the Conference on Human System Interactions*, pp. 932–935, 25–27 May 2008

T. Brown, R.C. Thomas, Finger tracking for the digital desk, in *Proceedings of First Australasian User Interface Conference*, Canberra, Australia (2000), pp. 11–16

J.L. Crowley, F. Berard, J. Coutaz, Finger tacking as an input device for augmented reality, in *Proceedings of International Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland (1995), pp. 195–200

M. Fisher, Kinect study, 2010. [Online]. Available: http://graphics.stanford.edu/~mdfisher/Kinect.html

V. Frati, D. Prattichizzo, Using kinect for hand tracking and rendering in wearable haptics, in *Proceedings of the IEEE World Haptics Conference*, pp. 317–321, 21–24 June 2011

P. Garg, N. Aggarwal, S. Sofat, Vision-based hand gesture recognition. World Acad. Sci. Eng. Technol. **49**, 972–977 (2009)

C.V. Hardenberg, F. Berard, Bare hand human computer interaction, in *Proceedings of the ACM workshop on Perceptive user interfaces*, Orlando, Florida, USA (2001), pp. 1–8

T. Keaton, S.M. Dominguez, A.H. Sayed, SNAP & TELL: a multimodal wearable computer interface for browsing the environment, in *Proceedings of Sixth International Symposium on Wearable Computers*, Seattle, WA, USA (2002), pp. 75–82

C. Nolker, H. Ritter, Visual recognition of continuous hand postures. IEEE Trans. Neural Networks **13**(4), 983–994 (2002)

K. Oka, Y. Sato, H. Koike, Real time tracking of multiple fingertips and gesture recognition for augmented desk interface systems, in *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, Washington, D.C., USA (2002a), pp. 411–416

K. Oka, Y. Sato, H. Koike, Real time fingertip tracking and gesture recognition. IEEE Comput. Graphics Appl. **22**(6), 64–71 (2002b)

I.B. Ozer, T. Lu, W. Wolf, Design of a real time gesture recognition system: high performance through algorithms and software. IEEE Signal Process. Mag. **22**, 57–64 (2005)

F.K.H. Quek, T. Mysliwiec, M. Zhao, Finger mouse: a free hand pointing computer interface, in *Proceedings of International Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland (1995), pp. 372–377

Y. Sato, Y. Kobayashi, H. Koike, Fast tracking of hands and fingertips in infrared images for augmented desk interface, in *Proceedings of Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, Grenoble, France (2000), pp. 462–467

TET, The Teardown, Engineering & Technology, **6**(3), 94–95 (2011)

A. Tomita, R.J. Ishii, Hand shape extraction from a sequence of digitized gray-scale images, in *Proceedings of Twentieth International Conference on Industrial Electronics, Control and Instrumentation*, Bologna, Italy, vol. 3, pp. 1925–1930, 5–9 September 1994

Y. Wu, Y. Shan, Z. Zhangy, S. Shafer, VISUAL PANEL: from an ordinary paper to a wireless and mobile input device, Technical Report, MSRTR 2000, Microsoft Research Corporation (2000)

H. Ying, J. Song, X. Renand, W. Wang, Fingertip detection and tracking using 2D and 3D information, in *Proceedings of the Seventh World Congress on Intelligent Control and Automation*, Chongqing, China (2008), pp. 1149–1152

Chapter 5
Light Invariant Hand Gesture Recognition

Recently, there has been a growing interest in the field of light invariant recognition. For advanced applications, a system can be set up in the laboratory with ideal conditions. However, in practical scenarios the hand gesture recognition systems may have applications in various environments. In image processing applications, the light intensity plays an important role because it significantly affects the segmentation of the ROI from the original image frame. The light intensity may not be same everywhere.

Different light sources and their intensities would show the same object with different luminance. If the light intensity changes, then the threshold for skin filter also has to be changed. Hence a robust system that operates in all types of light conditions is required. This motivates the development of techniques that are applicable to different light intensities. As this book is devoted to the natural interaction with machines using hand gestures, in this chapter the analysis of the effect of light intensity on barehand gesture recognition is performed. The light intensity at different times of the day, specifically, every 2 hours, is considered.

The hand gestures are interpreted by the system during the hand gesture recognition process, whether it is a predefined valid gesture or not. If the gesture is included in the list of the gestures, then the system responds to the corresponding action as predefined in the system. The light intensity in predefined gesture and light intensity in the current system can be different, but the system is supposed to recognize it as the same gesture.

5.1 Related Approaches

There are many previous studies that have extracted certain features of the hand for gesture recognition in order to make them robust. Keskin (Keskin et al. 2005) have developed an automatic tutor for sign language. Some common features extracted include hand silhouettes (Shimada 1998; Messery 1998; Sonka et al. 1999),

A. Chaudhary, *Robust Hand Gesture Recognition for Robotic Hand Control*,
DOI 10.1007/978-981-10-4798-5_5

contours (Starner and Pentland 1995), key points distributed along the hand, i.e., fingertips and joints. Yoon (Yoon et al. 2001) have also proposed a recognition scheme using combined features of location, angle, and velocity. Locken (Locken and Fitzgibbon 2002) proposed a real-time gesture recognition system, which can recognize 46 MAL, letter spelling alphabet and digits. There is, however, still a requirement for a system for efficient and robust technique finger detection. Identification of a hand gesture can be performed in many ways, which are selected on the basis of the problem to be solved, as is discussed in Chap. 3.

5.2 Pattern Recognition

The goal of pattern recognition is to classify the objects of interest into one of a number of categories or classes (Therrien 1989). The objects of interest are generally called patterns. They can be printed letters or characters, biological cells, electronics waveforms or signals, states of a system or any other items that one may desire to classify. Any pattern recognition system consists of two components, namely feature transformation and classifier (Therrien 1989) (see Fig. 5.1).

The observation vector X is first transformed into another vector Y whose components are called features. These are intended to be fewer in numbers than the observations collected but must collectively represent most of the information needed for the classification of patterns (Therrien 1989). By reducing the observations to a smaller number of features, one may design a decision rule which is more reliable. There are several transformations that can be applied in order to obtain better features. However, the feature extraction is mostly problem domain dependent.

The feature vector Y is passed to the classifier, which makes decisions about the pattern. The classifier essentially partitions the feature vector space into a number of disjoint regions as shown in Fig. 5.2. If the feature vector corresponding to the pattern falls into region R_i, the pattern is assigned to class ω_i.

For a pattern recognition problem, feature extraction is a major step. The goal of feature extraction is to find a transformation from an n-dimensional observation

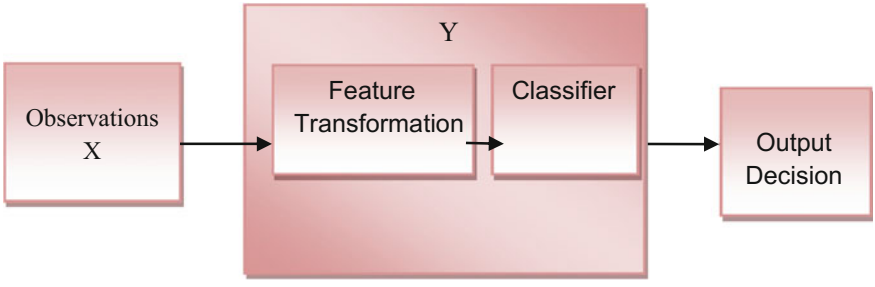

Fig. 5.1 Approach to feature extraction

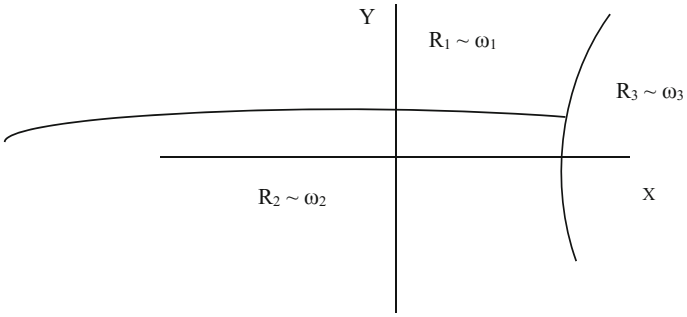

Fig. 5.2 Partitioning of feature space

space X to a smaller m-dimensional feature space Y that retains most of the information needed for pattern classification. There are two main reasons for performing feature extraction. First, the computational complexity for pattern classification is reduced by dealing with the data in a lower dimensional space. Second, for a given number of training samples, one can generally obtain more accurate estimates of the class conditional density functions and thus formulate a more reliable decision rule (Therrien 1989). In the past several methods have been used for feature extraction. Generally, the features are suggested by the nature of the problem.

5.3 Orientation Histogram

Orientation Histogram (OH) technique for feature extraction was developed by McConnell (McConnell 1986). The major advantage of this technique is that it is simple and robust to lighting changes (Freeman and Roth 1994). If the pixel intensities approach is taken, certain problems arise due to varying illumination (Messery 1998). If pixel by pixel difference for the same gesture is taken from two different images, while the illumination conditions are different, the distance between them would be large. In such scenarios, the picture itself acts as a feature vector.

The main motivation for using the orientation histogram is the requirement for lightening and position invariance. Another important aspect of gesture recognition is that irrespective of the orientation of the hand in different images, for the same gesture system must produce the same output. This can be done by forming a local histogram for local orientations (Liang and Ouhyoung 1998). Hence, this approach must be robust for illumination changes and it must also offer translational invariance.

Another important requirement is that it should match with the same gestures regardless of where they occur in the image. The pixel levels of the hand would

vary considerably with respect to light, on the other hand, the orientation values remain fairly constant. The local orientation can be calculated from the direction of the image gradient. Then, the local orientation angle θ will be a function of position x and y, and the image intensities $I(x, y)$. The angle θ is defined as:

$$\theta(x, y) = \arctan[I(x, y) - I(x - 1, y), I(x, y) - I(x, y - 1)] \tag{5.1}$$

Now form a vector Φ of N elements, with the ith element showing the number of orientation elements $\theta(x, y)$ between the angles $\frac{360^\circ}{N}\left[i - \frac{1}{2}\right]$ and $\frac{360^\circ}{N}\left[i + \frac{1}{2}\right]$. Where Φ is defined as:

$$\Phi(i) = \sum_{x,y} \begin{cases} 1 & \text{if} \left|\theta(x, y) - \frac{360^\circ}{N} i\right| < \frac{360^\circ}{N} \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

5.4 Light Invariant System

The recognition system works on the principle of the computer vision in 2D space. The basic methodology has been shown in Fig. 5.3. The system has an interface with a small camera which captures users' gestures. The input to the system is image frame of moving hand in front of a camera captured as a live video. The preprocessing of image frame was done as discussed in Chap. 4. The resulting image would be ROI, i.e., only gesture image. Once the ROI is available, next step is to find out feature vectors from the input image to recognize it with the help of ANN.

As this system was for proof of concept, only six different gestures are taken in the data set as many researchers were also have tested their methods with six gestures in the past. These six different gestures used in this research are shown in Fig. 5.4. The images of each gesture, used for ANN training, were different in the skin color and light intensity.

Once the gesture would get recognized the corresponding action takes place which could be associated with it. In the developed system, the audio description of the matched gesture was attached as the corresponding action. On recognition of the gesture, the audio file corresponding to the recognized gesture would be played.

The implementation of system is discussed in different steps.

5.4.1 Data Collection for Training Purpose

The training images for the ANN were collected from different sources including online search and manually collection. This was to ensure the robustness of the method as images from different sources would contain different skin color,

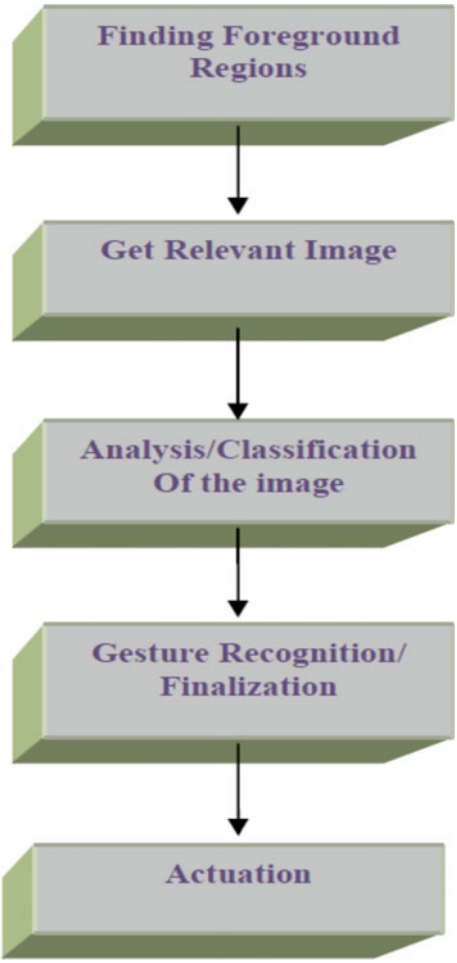

Fig. 5.3 Gesture recognition methodology

different light intensity, and different hand shape. Skin color has the property that it looks different in the different light intensities. Fourteen different images are used for each gesture to train ANN.

5.4.2 Preprocessing of Images

The ROI is needed from the images with the random background for training purpose and for recognition also. If the images have only ROI, then the training of the ANN would be better. All images, used for training, were converted into same resolution as the webcam was capturing the video. The preprocessing of the images is described in Chap. 4.

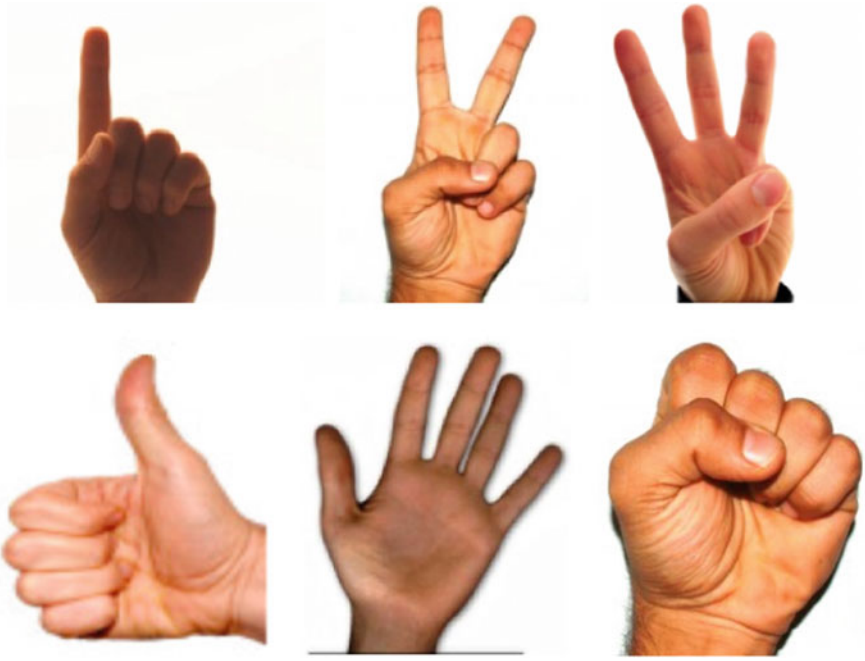

Fig. 5.4 Hand gestures to be used in the system

5.4.3 *Feature Extraction*

To train the ANN and for gesture recognition, the features need to be extracted from the preprocessed images. The algorithm used for feature extraction results in an orientation histogram for a given gesture. The same algorithm was applied for all the gestures present in the database in order to generate a training pattern. These training patterns were stored and applied to the neural network to train it. For gesture recognition purpose the same algorithm was applied. The algorithm is described in the steps below:

1. This algorithm is based on orientations of edges, so we have to find the edges in preprocessed images. Two simple filter vectors were used to for this, for X direction $X = [0\ -1\ 1]$ and for Y direction $Y = [0\ -1\ 1]^T$.
2. They were applied to each pixel in the ROI image to calculate image gradients dx and dy. The local orientation was calculated using image gradients and then arctan transformation was applied as shown in (5.1). This would result in gradient orientation vector.
3. The image blocks were rearranged into columns using MATLAB®. This was converted from radian values to degrees. So, a scan could be done on the orientation element values ranging from 0° to 190°. As mentioned in the (5.2), vector Φ of N elements where $N = 19$ is obtained.

4. The column matrix was used in order to plot orientation histogram. Using these plots the closeness of gesture recognition could be identified.

Now average the adjacent histogram bins in order to reduce the noise and allow interactions. For this research $N = 19$ was used in step 3 of the algorithm and this is purely an empirical estimate after observing the results with various N values like 20, 25, 15, etc. Euclidean distance can also be used to measure the difference between the histograms of two gestures. This is given in (5.3).

$$\sum_i (\Phi_1(i) - \Phi_2(i))^2 \tag{5.3}$$

5.4.4 Light Invariant Gesture Recognition

Gestures should be recognized same regardless of where they occur within the camera's field of view. This translation invariance can be achieved by the drastic step of ignoring position altogether, simply tabulating a histogram of how often each orientation direction occurred in the image. Clearly, this throws out some information, and few distinct images will be confused by their orientation histograms. For the purpose of feature extraction illustration, take one example of forefinger gesture raised and their corresponding OH as shown in Figs. 5.5 and 5.6.

Here it can be seen that the similarity between these orientation histograms even the skin color was very much different. This skin color difference could be because of different people or the same person in different light intensity. These similarities would be more clearly observed if we plot the OH for another gesture. It is a general assumption that positions of fingertips in the human hands are relative to the palm and it is almost always sufficient to differentiate a finite number of different gesture (Ahmad and Tresp 1993; Davis and Shah 1994; Kuno et al. 1994). Let us consider the gesture as all five fingers open and it is OH as shown in Fig. 5.7. From this, it is clear that OH plotted for two different gestures would be very much different while for same gesture it would show same OHs, only amplitude of vector can vary w.r.t. to light intensity.

As six predefined gestures are taken to test developed method, these samples would be used for training. The gesture and its corresponding OH are shown in Fig. 5.8 for all six samples.

5.5 Neural Networks Implementation

Neural networks are efficient for classification and have been used and lead to very satisfying results. The main difficulty lies in the training and all the preprocessing it requires ready. Neural networks can be used to solve some of the problems in vision

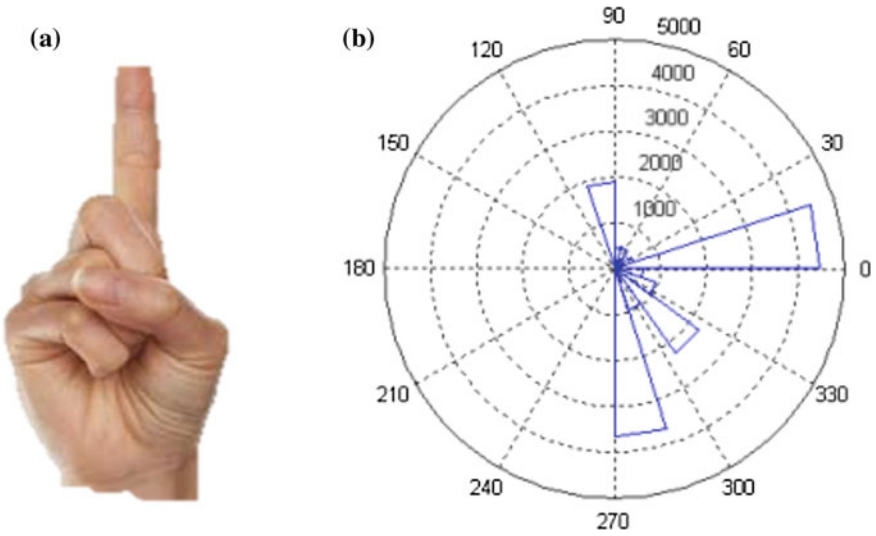

Fig. 5.5 **a** Gesture I and **b** OH of gesture I

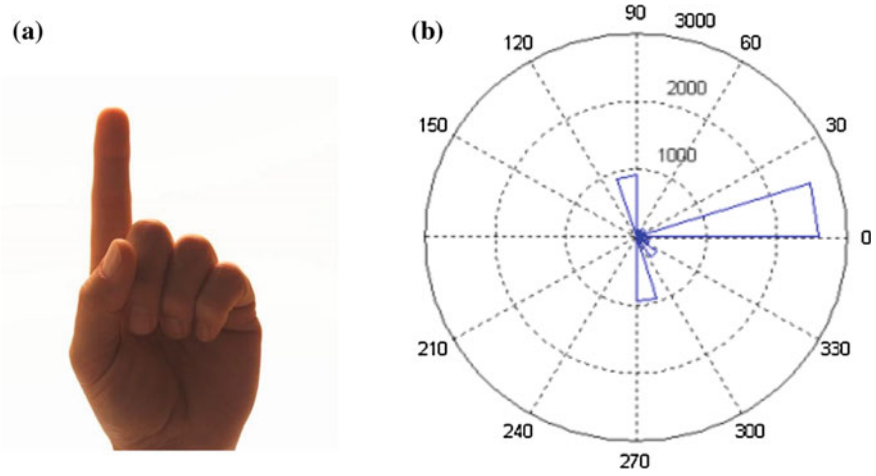

Fig. 5.6 **a** Gesture II and **b** OH of gesture II

for which procedural algorithms are difficult to generate and optical computing can be considered as a means to provide the greater computing power required for real-time and more general-purpose vision systems (Maung 2009). There are a variety of benefits that an analyst realizes from using neural networks in their work (Maung 2009).

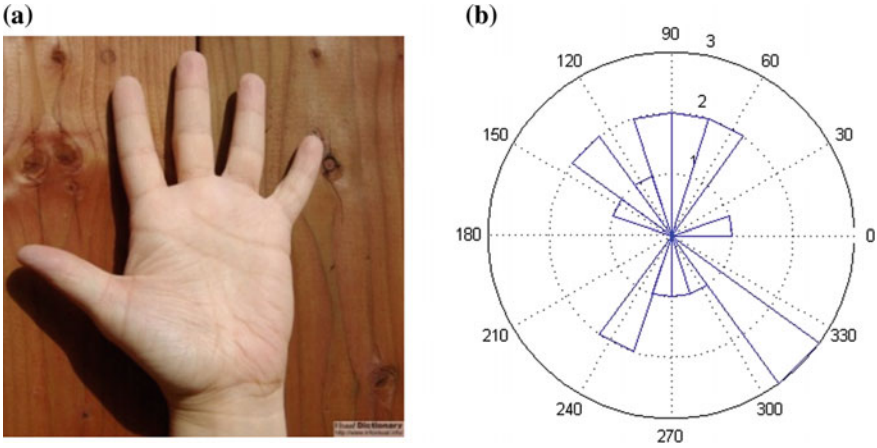

Fig. 5.7 **a** Gesture III and **b** OH of gesture III

(a) Pattern recognition is a powerful technique for harnessing the information in the data and generalizing about it. Neural nets learn to recognize the patterns which exist in the data set.
(b) The system is developed through learning rather than programming. Programming is much more time-consuming for the analyst and requires the analyst to specify the exact behavior of the model. Neural nets teach themselves the patterns in the data freeing the analyst for more interesting work.
(c) Neural networks are flexible in a changing environment. Rule-based systems or programmed systems are limited to the situation for which they were designed when conditions change, they are no longer valid. Although neural networks may take some time to learn a sudden drastic change, they are excellent at adapting to constantly changing information.
(d) Neural networks can build informative models where more conventional approaches fail. Because neural networks can handle very complex interactions they can easily model data which is too difficult to model with traditional approaches such as inferential statistics or programming logic.
(e) The performance of neural networks is at least as good as classical statistical modeling, and better on most problems. The neural networks build models that are more reflective of the structure of the data in significantly less time.

The neural network designed consists of 18 neurons at the input layer, 9 hidden neurons, and 6 neurons at the output. The architecture of the neural network is shown in Fig. 5.9.

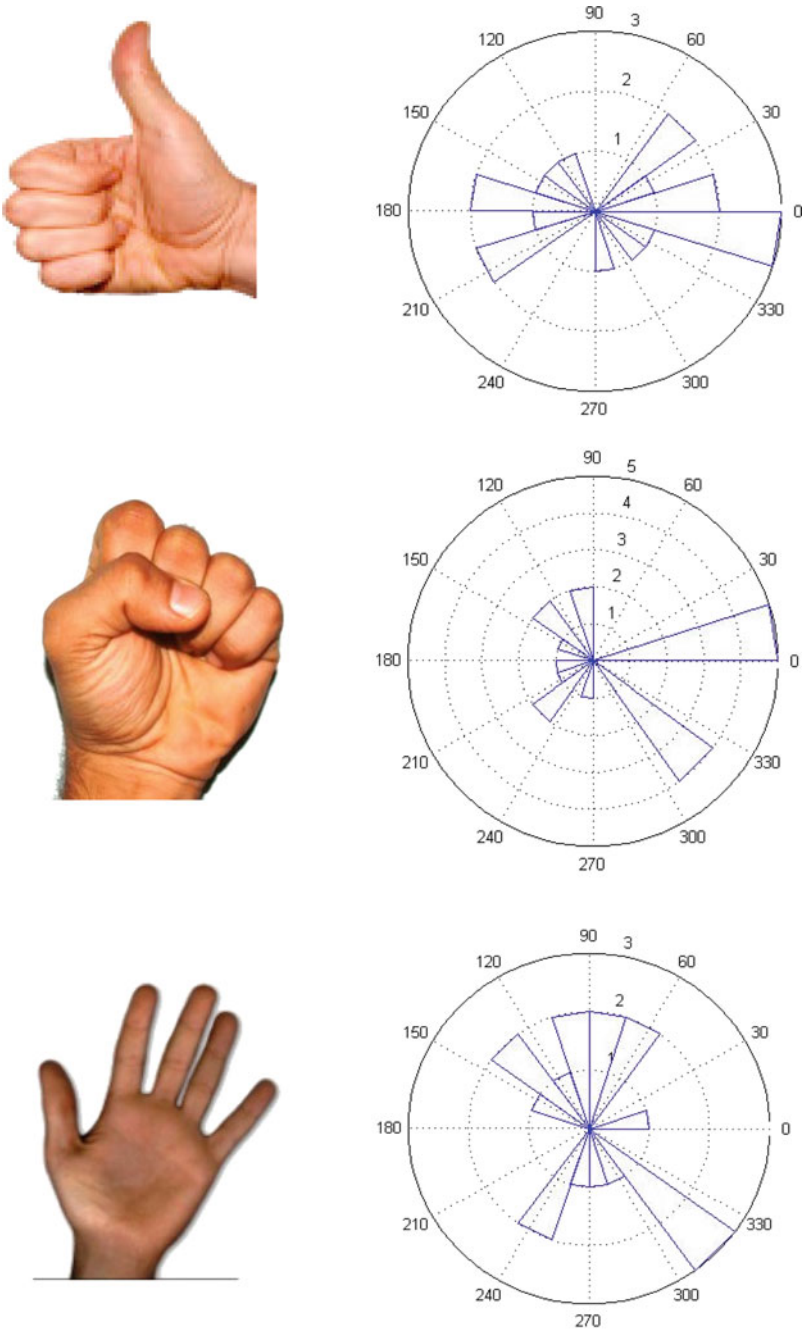

Fig. 5.8 Gestures used in the system and their OHs

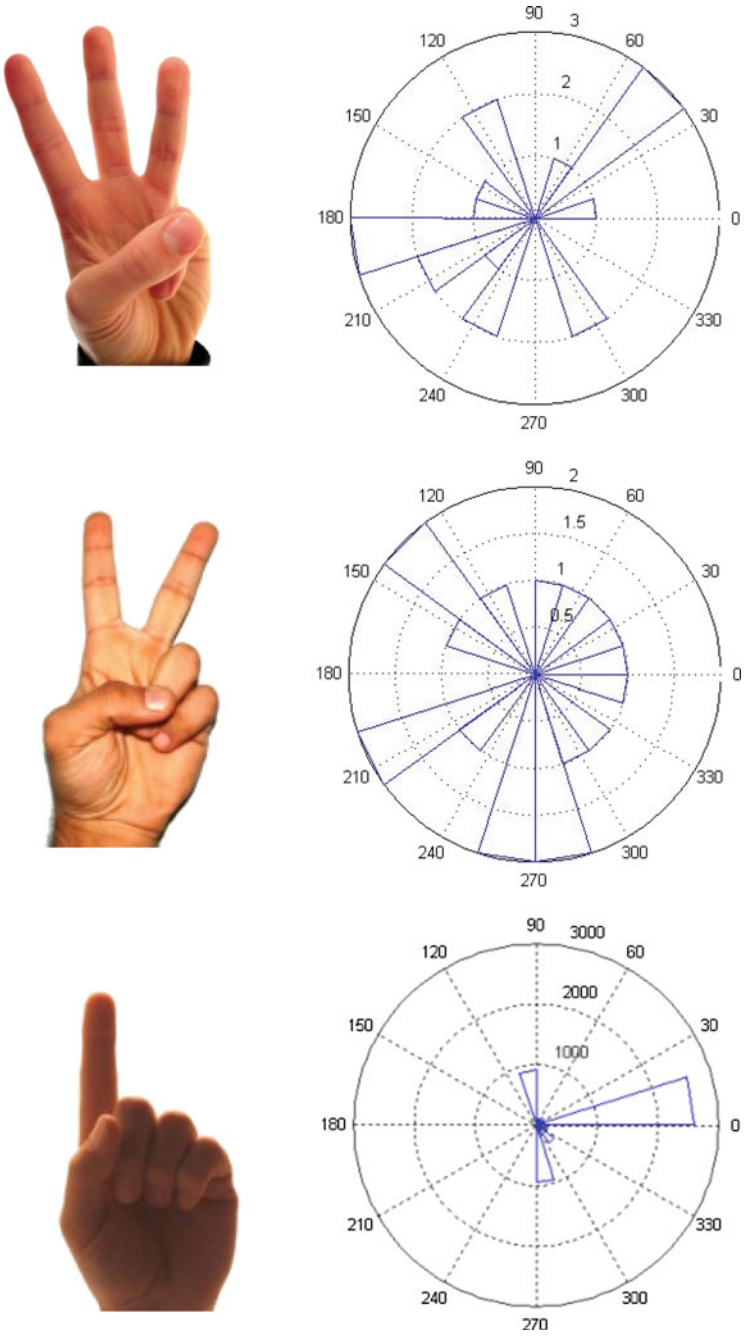

Fig. 5.8 (continued)

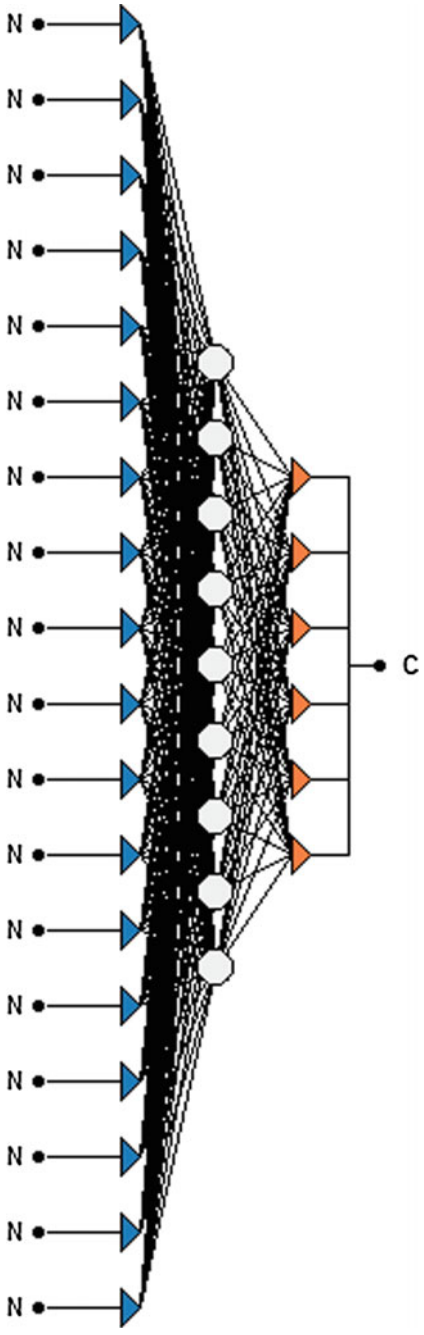

Fig. 5.9 Neural network architecture

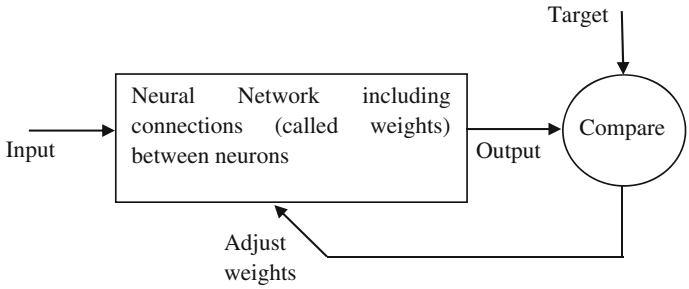

Fig. 5.10 Neural network block diagram (Maung 2009)

5.5.1 ANN Training

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. A neural network can be trained to perform a particular function by adjusting the values of the connections (weights) between elements.

Commonly neural networks are adjusted, or trained so that a particular input leads to a specific target output. Such a situation is shown in Fig. 5.10. Here the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically, many such input/target pairs are used in this supervised learning to train a network.

Neural networks have been trained to perform complex functions in various fields of applications including pattern recognition, identification, classification, speech, and vision and control systems. Today neural networks can be trained to solve problems that are difficult for conventional computers or human beings. The supervised training methods are commonly used but other networks can be obtained from unsupervised training techniques or from direct design methods. Unsupervised networks can be used, for instance, to identify groups of data. Certain kinds of linear networks and Hopfield networks are designed directly. In summary, there are a variety of kinds of design and learning techniques that enrich the choices that a user can make. For this system supervised learning has been used while back-propagation algorithm has been used for training. The process of training the network is simply a matter of altering the connection weights systematically to encode the desired input–output relationship.

5.5.2 Backpropagation Algorithm

Backpropagation algorithm has been selected since it is simple to implement and also it is a standard method and generally works well. It requires a teacher that

knows or can calculate, the desired output for any input in the training set. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). Backpropagation requires that the activation function used by the artificial neurons (or "nodes") be differentiable. The back-propagation learning algorithm can be divided into two phases: propagation and weight update.

Phase 1: Propagation

Each propagation in the ANN involves the following steps:

1. Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
2. Backward propagation of the propagation's output activations through the neural network using the training pattern's target in order to generate the deltas of all output and hidden neurons.

Phase 2: Weight update

For each weight synapse:

1. Multiply its output delta and input activation to get the gradient of the weight.
2. Bring the weight in the opposite direction of the gradient by subtracting a ratio of it from the weight.

This ratio influences the speed and quality of learning; it is called the learning rate. The sign of the gradient of a weight indicates where the error is increasing; this is why the weight must be updated in the opposite direction. Repeat the phase 1 and 2 until the performance of the network is good enough.

There are two modes of learning to choose from: One is online learning and the other is batch learning. In online learning, each propagation would be followed immediately by a weight update. In batch learning, much propagation occurs before weight updating occurs. Batch learning requires more memory capacity, but online learning requires more updates. As the algorithm's name implies, the errors propagate backward from the output nodes to the inner nodes. Backpropagation calculates the gradient of the error of the network regarding the network's modifiable weights (Werbos 1994). This gradient is almost always used in a simple stochastic gradient descent algorithm to find weights that minimize the error. Backpropagation usually allows quick convergence on satisfactory local minima for error in the kind of networks to which it is suited.

Backpropagation networks are necessarily multilayer perceptrons (usually with one input, one hidden, and one output layer). In order for the hidden layer to serve any useful function, multilayer networks must have nonlinear activation functions for the multiple layers: a multilayer network using only linear activation functions is equivalent to some single layer, linear network. Nonlinear activation functions that are commonly used include the logistic function, the softmax function, and the

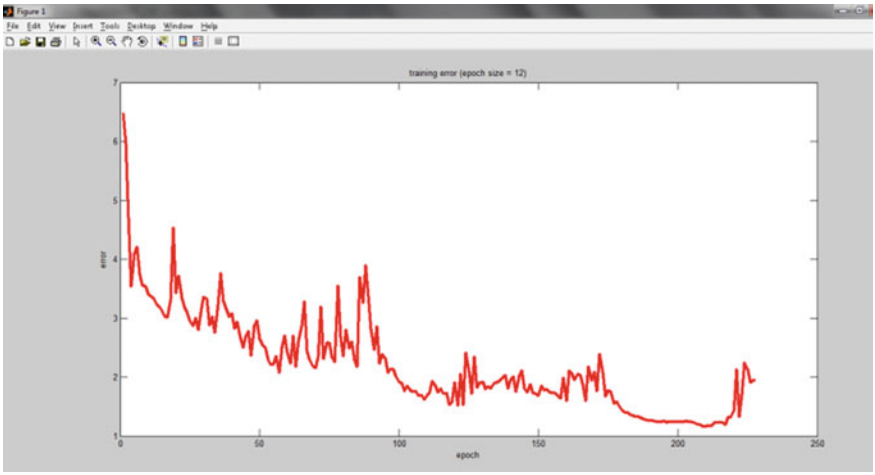

Fig. 5.11 Training error for epochs 220

gaussian function. The backpropagation algorithm is applied for training gestures. This system is using a transformation that converts an image into a feature vector which was used to train the neural network. Features were extracted from all sample images and were feed to ANN. After training, the feature vectors of a test image would be sent to this ANN for classification.

The feature vectors are scaled in the range of −5 and +5. These are applied to the neural network. The error is calculated by subtracting the output value from the target value. Then the sum-squared error was calculated. The error graphs of the network training for different values are shown in Figs. 5.11, 5.12 and 5.13.

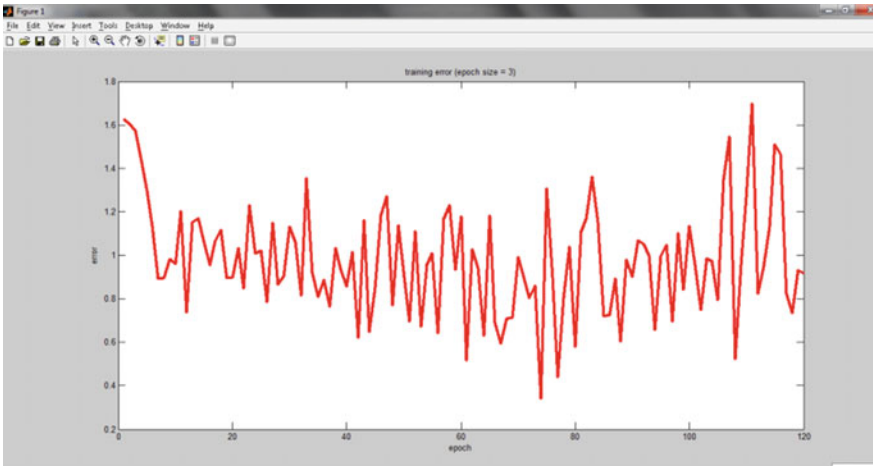

Fig. 5.12 Training error for epochs 120

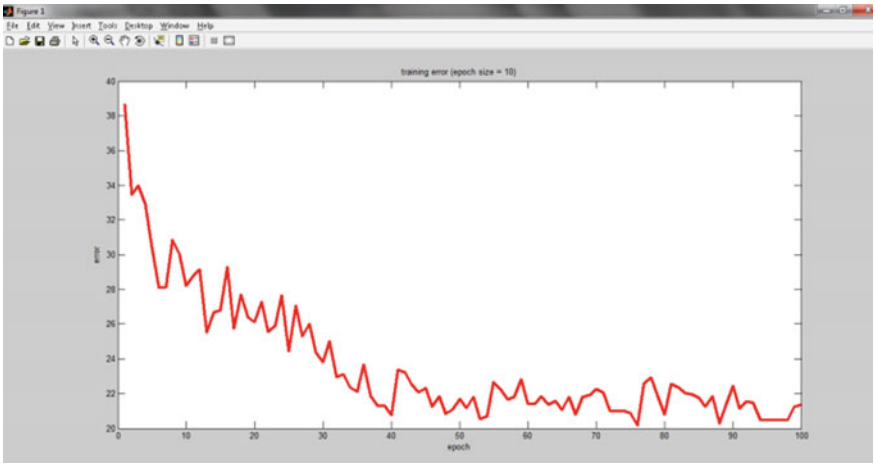

Fig. 5.13 Training error for epochs 100

5.6 Experimental Results

The target vectors for six sample gestures are defined in Table 5.1. The gesture recognition of test image was performed with matching the images stored in the database, first with Euclidean distance method and then using a neural network to improve results. The Euclidean method was giving satisfactory results but the false detection rates for few gestures were high.

Figure 5.14 shows the preprocessing results from live image frames, which was done as the procedure is discussed in Chap. 4. The Euclidean distance method was applied to a varied set of images several times and the outputs obtained are tabulated in Table 5.2. It shows the results of one test image with all the six type of gestures stored in the database. The minimum difference between two feature vectors would be considered a match. The matched gesture is shown in bold font in Table 5.2.

Where pattern 1–6 represents two fingers gesture, five fingers gesture, thumbs up gesture, single finger gesture, three fingers gesture, and closed fingers gesture respectively.

After this method, the supervised artificial neural network is used for gesture classification. The outputs obtained for different test gestures shown by the user to system webcam, are recognized by the developed system are shown in Figs. 5.15, 5.16 and 5.17.

From the above output, it can infer that though the lighting conditions were different in both the images, test image was correctly mapped to the required gesture as shown in Fig. 5.15. Let us consider another gesture of closed fingers and its recognition as shown in Fig. 5.16.

Similarly result for two open fingers gesture recognition is shown in Fig. 5.17.

Table 5.1 Gestures and their target vectors

Gesture	Target vectors
	000001
	000010
	000100
	001000
	010000
	100000

Fig. 5.14 Test image captured at real time and output after skin filtering

Table 5.2 Euclidean distance

S. No	Test image	Single finger	Two fingers	Five fingers	Closed fingers	Thumb up	Three fingers
1	Pattern 1	0.625	**0.435**	1.457	0.667	1.297	0.734
2	Pattern 2	3.366	3.084	**1.661**	1.974	2.597	0.346
3	Pattern 3	1.874	2.0997	2.368	2.070	**1.637**	2.254
4	Pattern 4	**0.4122**	0.593	1.810	1.644	0.834	0.700
5	Pattern 5	0.599	0.350	1.992	1.897	1.897	**0.227**
6	Pattern 6	0.3723	0.3783	0.2654	**0.23407**	0.394	0.204
7	Pattern 1	0.563	**0.367**	1.524	0.733	1.379	0.648
8	Pattern 2	3.431	3.257	**1.749**	2.697	1.943	3.628
9	Pattern 3	1.874	2.099	2.368	2.070	**0.637**	2.254
10	Pattern 4	**0.313**	0.496	2.180	1.650	2.647	3.890
11	Pattern 6	0.3527	0.3266	0.1962	**0.1850**	0.2606	0.363
12	Pattern 5	0.3174	0.3240	0.2625	0.2735	0.2450	**0.150**
13	Pattern 1	0.6094	**0.3677**	1.5240	0.7340	1.3791	0.609
14	Pattern 2	0.241	0.2596	**0.1549**	0.19562	0.17498	0.269
15	Pattern 3	0.1818	0.1504	0.1662	0.1214	**0.106**	0.204
16	Pattern 4	**0.4123**	0.4777	1.8512	1.0046	1.7786	0.599
17	Pattern 6	0.3266	0.3381	0.1962	**0.1857**	0.2606	0.350
18	Pattern 5	0.4167	0.7008	2.0689	1.2571	1.9901	**0.227**
19	Pattern 1	0.9133	**0.5649**	1.2087	0.75132	1.2002	0.858
20	Pattern 2	0.3257	0.3429	**0.185**	0.2785	1.934	0.355
21	Pattern 3	1.818	1.5046	1.662	1.0635	**0.2914**	2.045
22	Pattern 4	**0.4776**	1.233	1.8512	1.7786	0.5999	1.004
23	Pattern 5	0.5125	0.5013	2.0689	1.2571	1.891	**0.127**
24	Pattern 6	2.4372	3.2424	3.2793	**0.27191**	2.3341	3.554

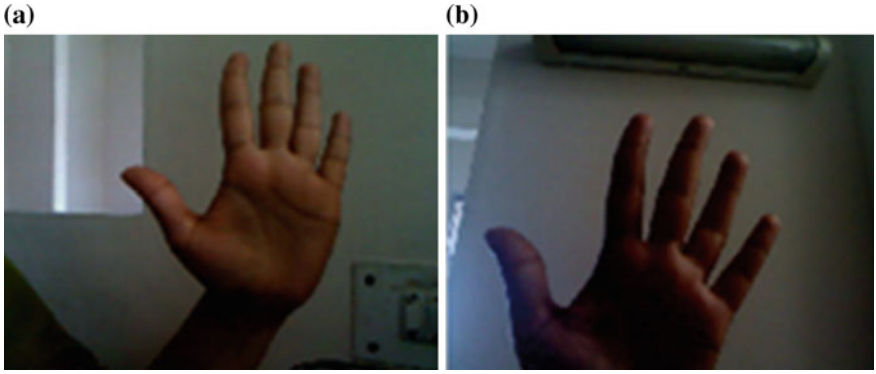

Fig. 5.15 Test I: output after applying recognition algorithm: **a** test image and **b** image in the database

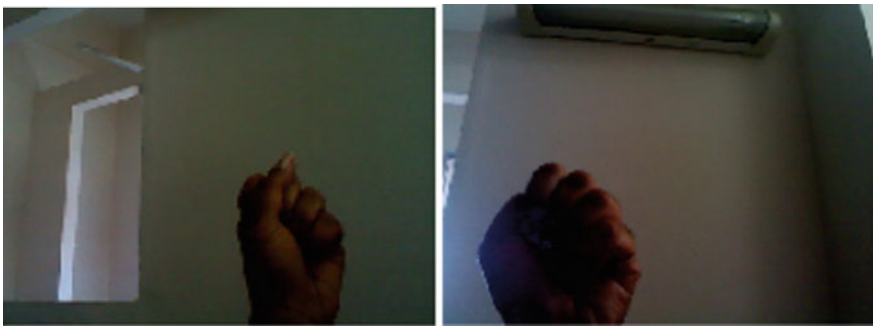

Fig. 5.16 Test II: output after applying recognition algorithm

Fig. 5.17 Test III: output after applying recognition algorithm

For comparison purpose, 14 different images for each gesture is applied to both methods and prepared their confusion matrix. Table 5.3 shows the confusion matrix for supervised neural network-based gesture recognition.

Each column in the confusion matrix corresponds to the number of images mapped to a particular gesture while each row contains the total number of images applied for testing. For example, the test pattern 6 which is three fingers gesture, out of 14 testing images, 13 gestures were mapped correctly and the rest 1 gesture has been mapped wrongly as the five fingers gesture. Similarly, for the single finger gesture, 13 were mapped correctly and the rest 1 was mapped wrongly to two fingers gesture. Overall with this method out of 84 testing images, 78 images were mapped correctly. The accuracy of recognition is obtained to be 92.86%.

Confusion matrix obtained for Euclidean distance method is shown in Table 5.4. The classification was done based on the Euclidean distance between the template and the testing image. Only 62 testing images were mapped correctly out of 84. The efficiency of recognition with this method came as 73.8%.

So, the efficiency was improved 19.06% with a neural network. The main advantage of using neural networks is that conclusions can be drawn from the

Table 5.3 Confusion matrix with neural network

Target / Test	Two Fingers	Single Finger	Five Fingers	Thumb Finger	Closed Fingers	Three Fingers
1	14	0	0	0	0	0
2	1	13	0	0	0	0
3	1	0	13	0	0	0
4	0	0	1	13	0	0
5	0	0	0	2	12	0
6	0	0	1	0	0	13

Table 5.4 Confusion matrix with Euclidean distance

Target / Test	Two Fingers	Single Finger	Five Fingers	Thumb Finger	Closed Fingers	Three Fingers
1	10	2	0	0	1	1
2	1	9	0	2	2	0
3	1	0	12	0	1	0
4	1	2	1	8	0	2
5	0	0	0	1	12	1
6	0	1	1	1	0	11

network output. If a vector is not classified correctly, its output can be checked and work out a solution. Figure 5.18 shows the comparison between the successes with neural network and with Euclidean distance method for each gesture using a bar graph. In the graph, blue color represents the number of successes with the neural network and the red color represents the number of successes with Euclidean distance. Here gesture 1–6 represents two fingers, single finger, five fingers, thumb, closed fingers, and three finger gestures respectively. This has been plotted after testing each method with 14 test images for each gesture.

Random collected test data is applied a number of times on these two methods and collected results to get a better view of performance. Table 5.5 summarizes the collected results for each gesture.

Figure 5.19 represents Table 5.5 in a line graph to show a visual comparison of accuracies. The x-axis numbers are gesture sequence same as shown in Table 5.5.

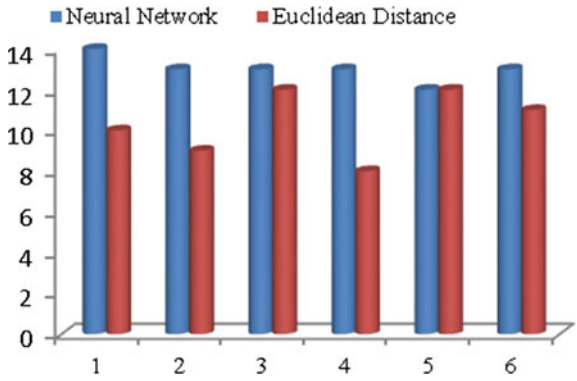

Fig. 5.18 Comparison graph

Table 5.5 Gesturewise accuracy comparison

Gesture	Accuracy (with Euclidean distance) (%)	Accuracy (with ANN) (%)
Two fingers	71	99
Single finger	64	92
Five fingers	85	95
Thumb fingers	57	92
Closed fingers	85	85
Three fingers	81	90

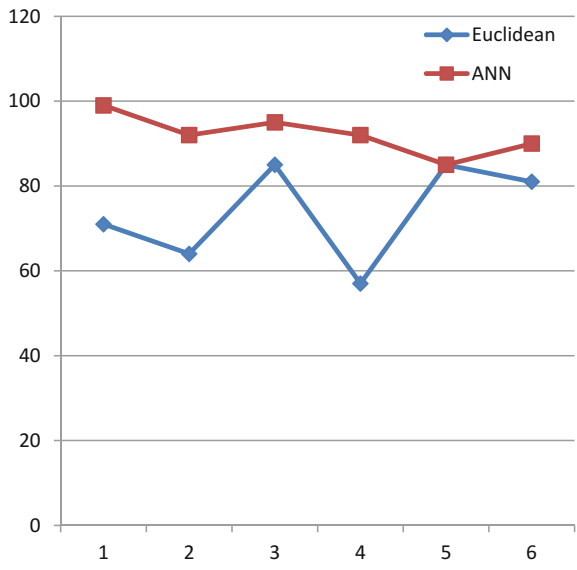

Fig. 5.19 Accuracy comparison

5.7 Conclusion

This chapter presents a light invariant gesture recognition system which would be a great impact on the current state of the art. The system is tested with Euclidean distance method and also implemented ANN with backpropagation algorithm. The system can be made more robust by improving the training data set. The main advantage of using neural networks is that conclusions can be drawn from the network output. The presented method used a database of images to match with test gesture, but it is not possible to store all movements of the hand. So the hand geometry needs to be identified to detect correct hand posture in the image frame. Next chapter discusses the hand geometry parameters detection from the preprocessed images which will help in HGR.

References

C. Keskin, O. Aran, L. Akarun, Real time gestural interface for generic applications, in *Proceedings of European Signal Processing Conference*, Antalya (2005)

N. Shimada et al., Hand gesture estimation and model refinement using monocular camera—ambiguity limitation by inequality constraints, in *Proceedings of IEEE Third Conference on Face and Gesture Recognition*, pp. 268–273, 14–16 Apr 1998

T. Messery, Static hand gesture recognition report

M. Sonka, V. Hlavac, R. Boyle, *Image Processing, Analysis, and Machine Vision* (Brooks/Cole Publishing Company, 1999)

T. Starner, A. Pentland, Real-time American sign language recognition from video using hidden Markov models, in *Proceedings of International Symposium on Computer Vision*, pp. 265–270, 21–23 Nov 1995

H.-S. Yoon, J. Soh, Y.J. Bae, H.S. Yang, Hand Gesture recognition using combined features of location, angle and velocity. Pattern Recogn. **34**(37), 1491–1501 (2001)

R. Locken, A.W. Fitzgibbon, Real gesture recognition using deterministic boosting, in *Proceedings of the British Machine Vision Conference* (2002)

C.W. Therrien, *Decision Estimation and Classification: An Introduction to Pattern Recognition and Related Topics* (Wiley, New York, 1989)

R.K. McConnell, Method of and apparatus for pattern recognition, U.S. Patent No. 4567610, Jan 1986

W.T. Freeman, M. Roth, Orientation histograms for hand gesture recognition, TR-94-03a, Dec 1994

R. Liang, M. Ouhyoung, A real-time gesture recognition system for sign language, in *Proceedings of Third International Conference on Automatic Face Gesture Recognition*, pp. 558–567, 14–16 Apr 1998

S. Ahmad, V. Tresp, Classification with missing and uncertain inputs, in *Proceedings of International Conference on Neural Networks*, Amsterdam, vol. 3 (1993), pp. 1949–1954

J. Davis, M. Shah, Recognizing hand gestures, in *Proceedings of European Conference on Computer Vision*, Stockholm, pp. 331–340, 2–6 May 1994

Y. Kuno, M. Sakamoto, K. Sakata, Y. Shirai, Vision-based human computer interface with user centered frame, in *Proceedings of Intelligent Robots and Systems*, pp. 2023–2029, 12–16 Sept 1994

T.H.H. Maung, Real time hand tracking and gesture recognition system using neural networks. World Acad. Sci. Eng. Technol. **50**, 466–470 (2009)

P.J. Werbos, *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting* (Wiley, New York, 1994)

Chapter 6
Fingertips Detection

Fingertip detection forms an important component of HGR when image-based models are employed for constructing or detecting hand positions. In this chapter, the focus is on direction invariant fingertip and center of palm detection of natural hand with real-time performance. The HGP is detected in 2D using a simple webcam and in 3D using KINECT. Low-level image processing methods are used to detect HGP in 2D, while KINECT facilitates by providing the depth information of foreground objects. The gesture parts are segmented using the depth vector and the centers of palms are detected using distance transformation on an inverse image.

6.1 Related Approaches

A number of studies have been conducted in the area of dynamic hand gesture recognition using fingertip detection. Fingertip detection should be near to the real time when a video is processed. Yang (Yang et al. 2005) analyses the hand contour to select fingertip candidates, then finds the peaks in their spatial distribution and checks the local variance to locate fingertips. This method is not invariant to the orientation of the hand. There are other methods which use directionally variant templates to detect fingertips (Kim and Lee 2008; Sanghi et al. 2008). Poor hand segmentation performance usually invalidates fingertip detection methods. Moreover, some of the fingertip detection methods cannot localize accurately multidirectional fingertips.

6.2 HGP Detections

HGP includes fingertips and the center of the palm. In Chap. 4 the hand is segmented as a part of preprocessing. The automatic center of palm (COP) detection in a real-time input system is a challenging task, but it opens a new set of applications

A. Chaudhary, *Robust Hand Gesture Recognition for Robotic Hand Control*,
DOI 10.1007/978-981-10-4798-5_6

where hand gesture recognition can be used. This section continues with the results of segmentation from Chap. 4 to detect HGP.

6.2.1 Fingertips Detection

At this point from the segmented results, one smaller image is available which contains mostly skin pixels (hand gesture shape). Since the hand direction is known, the direction of fingertips is also known. To determine all fingertips in the cropped hand image, a scan of the cropped binary image from wrist to finger ends is initiated. Consequently, the numbers of pixels are calculated for each row or column based on the hand direction, whether it is in the up-down or left-right position. Then, the intensity of each pixel is assigned values between 1 and 255 in increasing order from wrist to finger end in equal distribution. The process is presented in Fig. 6.1.

Hence, each skin pixel at the edges of the fingers is assigned a high-intensity value of 255 as it would be the last skin pixel. Therefore, all the fingertips contain a pixel value 255 and have the brightest intensity. The fingertip detection process can be represented mathematically as

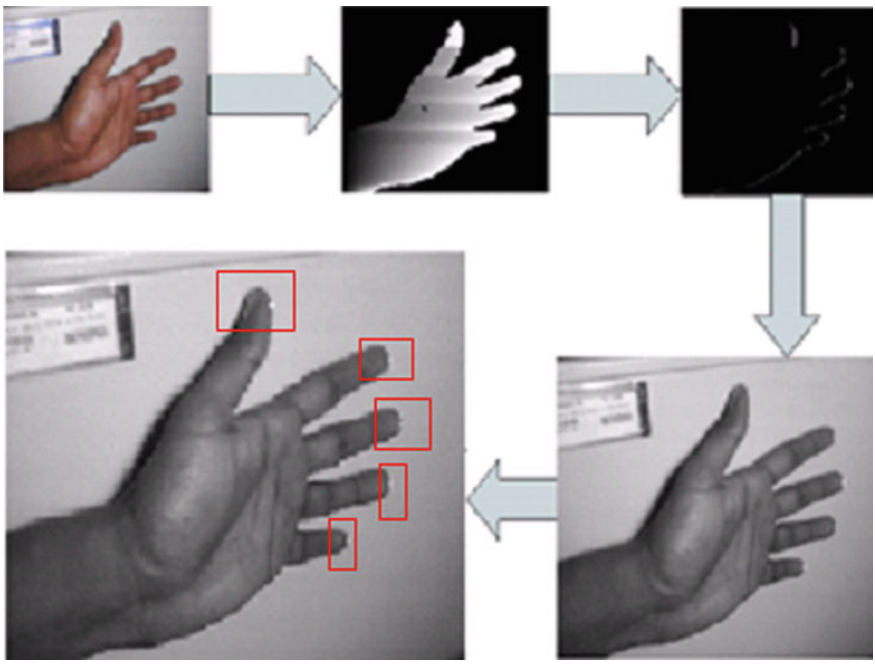

Fig. 6.1 Fingertip detection process

$$\text{pixel}_{\text{count}}(y) = \sum_{X=X\min}^{X\max} imb(x, y) \tag{6.1}$$

$$\text{modified}_{\text{image}}(x, y) = \text{round}(x \times 255/\text{pixel}_{\text{count}}(y)) \tag{6.2}$$

$$\text{Finger}_{\text{edge}}(x, y) = \left\{ \begin{matrix} 1 & \text{if modified}_{\text{image}}(x, y) = 255 \\ 0 & \text{otherwise} \end{matrix} \right\} \tag{6.3}$$

where $\text{Finger}_{\text{edge}}$ gives the boundary of the finger.

The line that has the highest intensity pixel is first indexed and it is checked whether the differentiated value lies inside an experimentally set threshold. If it does, then it represents a fingertip. The threshold value changes with respect to the direction of the hand. The threshold can be set after the detection of the hand direction, which is already known. For a frame resolution of 240 × 230 where hand direction is left to right, the set threshold is found to be 7. The enhanced results of fingertip detection are presented in Fig. 6.2 where the detected pixels are marked in different colors. The fingertips have been detected in the captured frame for further operations carried out in this book.

6.2.2 COPs Detection

In order to determine the hand geometry with a greater precision, there is a need to detect the center of palm in the same image. The exact location of the COP in the hand image can be identified by applying a mask of dimension 30 × 30 to the cropped image and counting the number of skin pixels lying within the mask. This process is made faster by employing the summed area table of the cropped binary image for calculating the masked values (Crow 1984). In the summed area table, the value at any point (x, y) is the sum of all the pixels above and to the left of (x, y), inclusive. As shown in (6.4).

$$\text{sum}(x, y) = \sum_{\substack{x' \le x \\ y' \le y}} i\left(x', y'\right) \tag{6.4}$$

The summed area table can be computed efficiently in a single pass over the image using (6.5)

$$\text{sum}(x, y) = i(x, y) + \text{sum}(x - 1, y) + \text{sum}(x, y - 1) - \text{sum}(x - 1, y - 1) \tag{6.5}$$

Once the summed area table is computed, the task of evaluating any rectangle can be accomplished in constant time with just four array references (see Fig. 6.3) using (6.6).

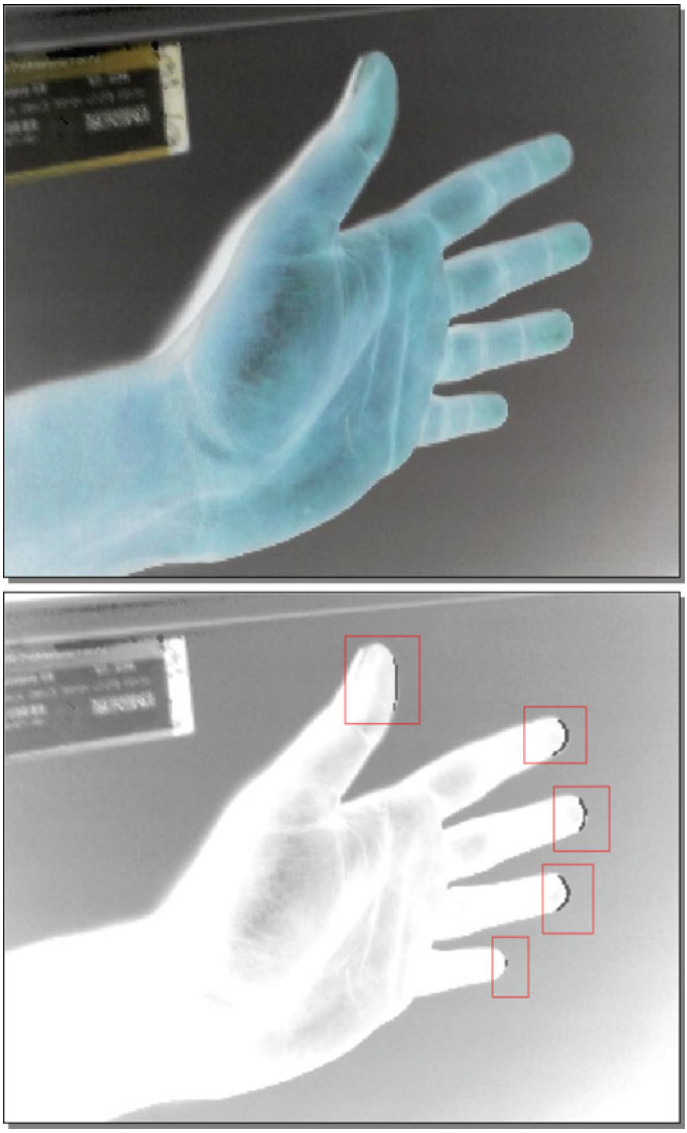

Fig. 6.2 Results of fingertip detection in the original image frame

$$\sum_{\substack{A(x)<x'\le C(x)\\ A(y)<y'\le C(y)}} i(x',y') = \mathrm{sum}(A) + \mathrm{sum}(C) - \mathrm{sum}(B) - \mathrm{sum}(D) \tag{6.6}$$

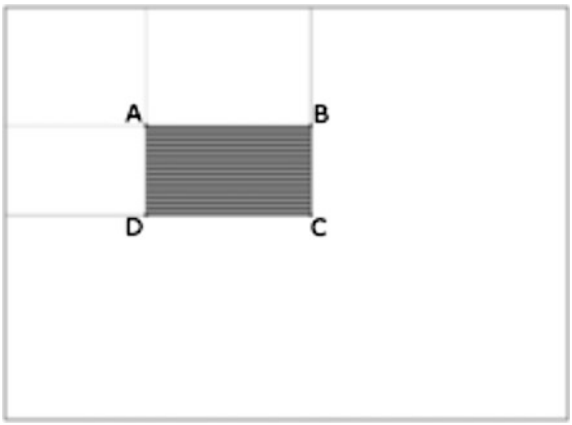

Fig. 6.3 Finding the sum of a rectangular area [WIKIc]

Fig. 6.4 Fingertips and center of palm detected in a real-time system

The value of the rectangular mask over a region can be calculated by simply four lookups. This improves the speed of computation by a factor of 250. The COP is calculated as the mean of the centers of all the regions that have a sum of more than a calculated threshold, as shown in Fig. 6.3.

From experiments, this threshold was obtained as 832. Results of fingertips and COP detection are presented in Fig. 6.4. Enhanced results are also presented in Fig. 6.5.

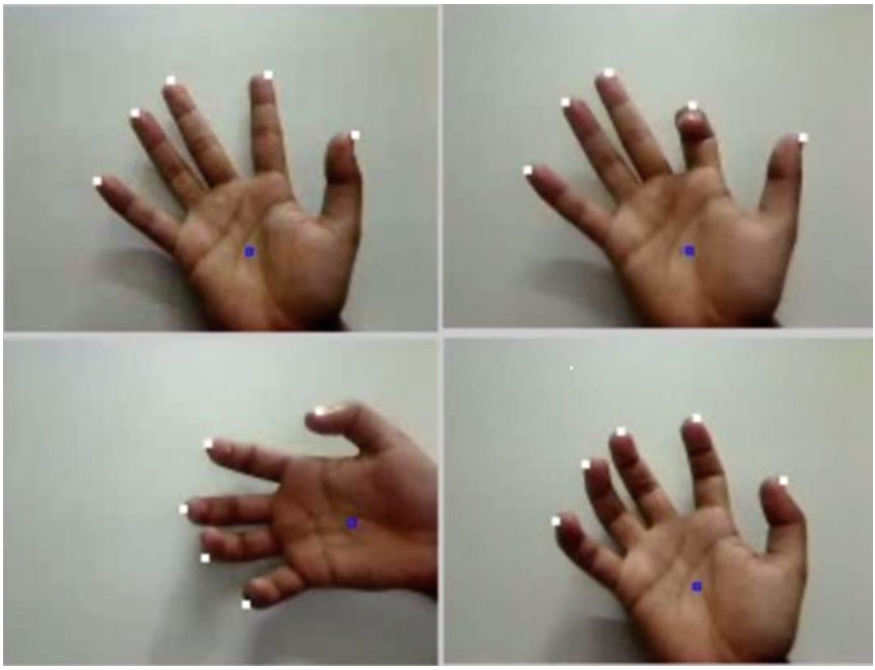

Fig. 6.5 Enhanced results of fingertips and center of palm detection

6.3 HGP Detection Using KINECT

In Chap. 4 hand segmentation was performed using KINECT. This section describes a novel method for fingertips and center of palm detection in dynamic hand gestures generated by either one or both hands with KINECT.

6.3.1 Fingertip Detection in 3D

The approach of fingertip detection using a special device KINECT uses segmented results presented in Sect. 4.3.3. To detect the fingers in the ROI, first the palm of the hand is detected, which is done by applying a big circular filter to the image. Thereby all the fingers in the images are removed. Now, the palm of the hand is subtracted from the original hand image to obtain the segmented finger masks as shown in Fig. 6.6. Figure 6.7 presents the result of applying the mask to the depth image collected from KINECT sensors. The resultant image consists of only fingers and then the fingertips are detected.

After examining the depth map, it can be easily determined whether the value of the fingertip depth is the minimum. A minimum depth implies that the fingertips are

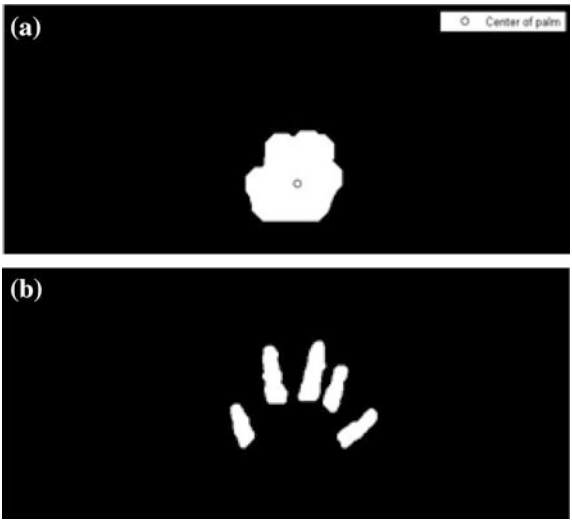

Fig. 6.6 Results of palm subtraction **a** palm in one hand image, **b** fingers mask for one hand

Fig. 6.7 Segmented fingers in the depth image

the closest to the camera compared to the remaining objects. Thereby the fingertips are detected by determining the minimum depth in each finger. The results are presented in Fig. 6.8.

6.3.2 COP Detection Using KINECT

The centers of the palms are detected by applying the distance transform on the inverted binary images of the hand. The results are shown in Fig. 6.9. It is noted

Fig. 6.8 Result of fingertip detection in real time

that the maximum of the distance transform gives the center of the palm on one segment.

6.3.3 Results

From the lab setup, fingertips and the center of palm are identifiable very accurately even when the fingers were bent at a large angle. This is because KINECT has nothing to do with the spatial information and this method is based on the depth information provided by the device. The accuracy for fingertips detection when all fingers are open is approximately 100%. When the fingers are fully bent, sometimes confusion is created because in this position the finger joints are closer to the device. When the center of the palm is detected results are approximately 90%

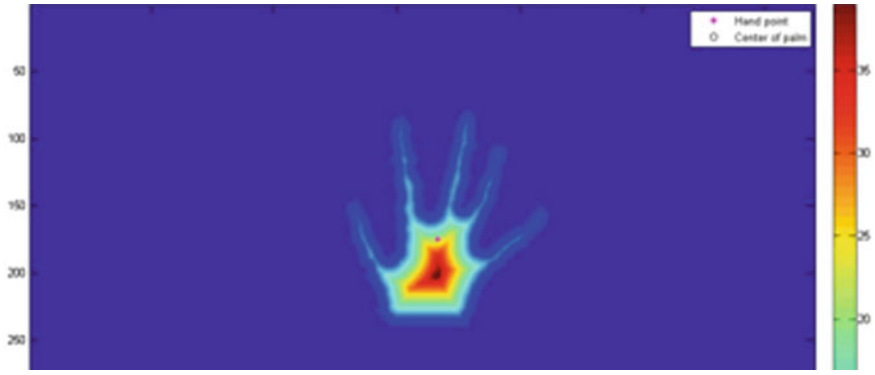

Fig. 6.9 Distance transform of the hand

Fig. 6.10 Final result showing hand point, the center of palm, and fingertips

accurate. The whole system is implemented based on real-time requirements and the results were very encouraging (see Fig. 6.10).

6.4 HGP Detection for Both Hands

The fingertips and COP for one hand have been successfully detected using KINECT. The hand can be right or left, depending on whether the user is right-handed or left-handed. The same process for HGP detections works for both hands and can be applied simultaneously to both hands. The results obtained are very encouraging. The results for real-time fingertips and COP detections for both hands are presented in Fig. 6.11. The orientation of the hands is not a constraint in this approach, as the 3D sensor is able to detect hands in any direction.

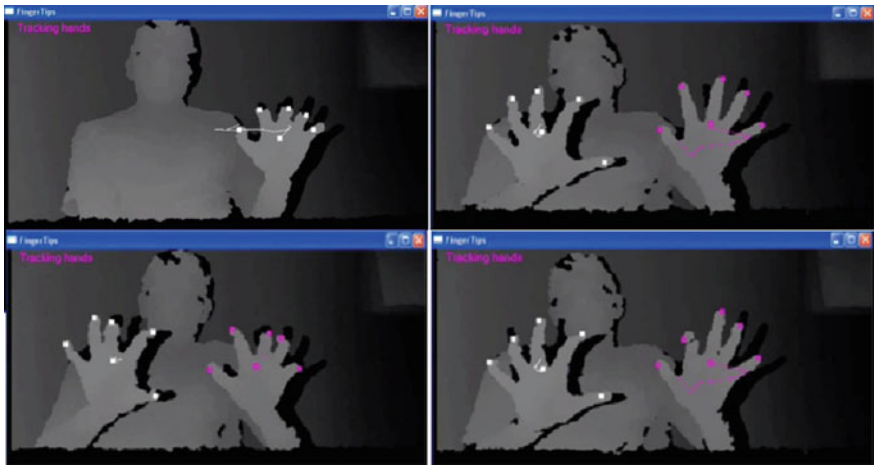

Fig. 6.11 Results of fingertip detection for both hands

As this experiment can be used for both hands, we have developed techniques for clearly distinguish between the hands. As shown in Fig. 6.11, for a single hand the fingertips and the center of palm are detected in white color, while if both hands are detected in the image, the right hand is detected as white and the left hand is detected as pink. This color difference ensures that the center of the palm of one hand does not match the other hand details.

6.5 Conclusion

This chapter is a milestone leading toward the ultimate goal of natural communication with the machine. This chapter presents an algorithm to detect the hand geometry parameters with bare hands. The results are encouraging, specifically, 6 frames per second are processed in the MATLAB® setup, which is a significantly fast real-time performance. The fingertips and the center of palm are detected clearly with the known hand direction. Moreover, this pre-processing method is direction invariant. The user can depict the gesture in any direction and consequently the parameters are detected in that particular direction.

The detection of fingertips and the center of palms are also done using a special device, i.e., KINECT, which satisfies all the requirements of the system considered. Using this device single or both hands can be segmented irrespective of the background and light intensity. Moreover, the results of the direction invariants are also available. KINECT is a sophisticated device that provides a number of features and it is hence significantly more expensive than a simple device such as a webcam.

HGP detection enables to get hand posture estimation which can be used in many applications. This book also includes controlling of a robotic hand which would mimic the hand gesture. To control a robotic hand, the fingers' bending angles calculation is needed which is discussed in the next chapter.

References

D. Yang, L.W. Jin, J. Yin et al., An effective robust fingertip detection method for finger writing character recognition system, in *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, Guangzhou, China (2005), pp. 4191–4196

J.M. Kim, W.K. Lee, Hand shape recognition using fingertips, in *Proceedings of Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, Jinan, Shandong, China (2008), pp. 44–48

A. Sanghi, H. Arora, K. Gupta, V.B. Vats, A fingertip detection and tracking system as a virtual mouse, a signature input device and an application selector, in *Proceedings of the Seventh International Caribbean Conference on Devices, Circuits and Systems*, Cancum, Mexico, pp. 1–4, 28–30 April 2008

F. Crow, Summed-area tables for texture mapping, in *Proceedings of Eleventh Annual Conference on Computer Graphics and Interactive Techniques*, New York, USA, (1984), pp. 207–212

Chapter 7
Bent Fingers' Angles Calculation

Hands play an important role in the execution of many important tasks in the daily lives of humans as well for a number of other special purposes. The shape of the human hand is such that it is able to easily perform a number of otherwise tedious tasks. It can bend its fingers to different angles to pick up or to hold objects and to apply force via fingers or the palm area. In a number of scenarios, a human hand can perform the tasks much more efficiently than a machine shaft. This is due to the ability of a human hand to operate over a number of degrees of freedom and its ability to bend fingers at different angles.

However, in some scenarios, it may not be suitable to use a human hand, while the use of a machine may be preferable. For instance, in situations like bomb detection and diffusion, execution of suspected ammunitions and landmine removal, if humans are led into the field, then casualties may occur. Hence, there is a need for a robotic hand which can perform the same operations as a human hand. The robotic hand should be able to bend fingers like a human does and it should be easily controllable. The robotic hand should have joints in the fingers, which it can bend like a human in interaction mode. Hence, in general, the robotic hand should be able to perform all the operations of a human hand in real time.

The methods presented in this chapter can be used to control a remotely located robotic hand which is able to perform the same operations as a human hand. The user shows his natural hand (without wearing any mechanical-electronic equipment) to the camera and the palm should face the camera. The behavior of human hand is detected by the camera and the robotic arm is made to operate accordingly.

This chapter describes a novel method to determine the bending angles of all fingers in the hand. The hand geometry parameters have already been determined in previous chapters. This information could be used to control an electro-mechanical robotic hand by gestures generated by the human hand. The user can show any hand to the system (right of left). There is no restriction on the direction of the hand. If the palm faces the camera, the hand can be in any direction to control the electro-mechanical hand. This vision-based system detects fingertips in the real time from user input and passes the information of the fingers' bending angles to the

A. Chaudhary, *Robust Hand Gesture Recognition for Robotic Hand Control*,
DOI 10.1007/978-981-10-4798-5_7

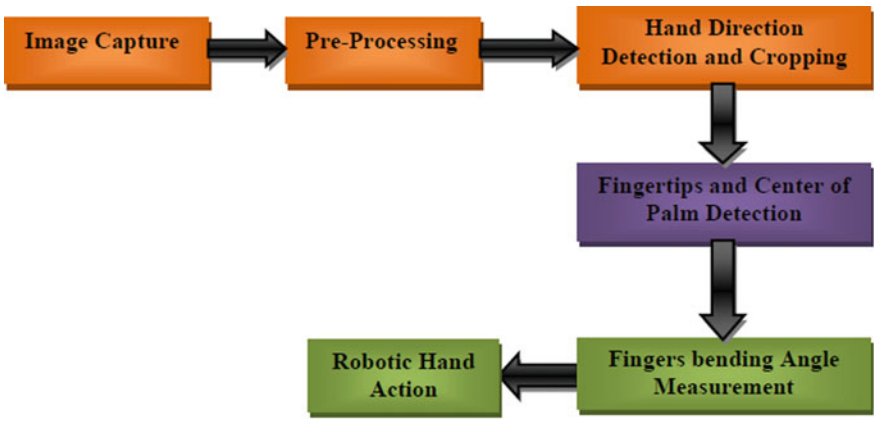

Fig. 7.1 Block diagram flow of the system

robotic hand. When the user bends his fingers to hold an object (virtual object), and the robotic hand performs the same operation i.e. holds the object. The movement of the user's hand changes the movement of the robotic hand in real time.

As already discussed in this book, this work comes under natural computing and requires no sensors, color or paper clips to detect gestures in the image. In the past researchers have done a significant amount of work in this area, but they used a wired glove in which sensors are planted or have used colors on fingers to recognize the gesture to control robots.

The process involved in the implementation of a human gesture on a robotic hand action is described in Fig. 7.1. The video captured from a webcam in 2D is preprocessed, as explained in Chap. 4 and fingertips and center of palm are detected.

7.1 Related Approaches

Many applications can be found in literature of real-time robotic control in context of human-computer interaction, computer games control (Freeman 1999), human-robot interaction (Triesch and Malsburg 1997) and sign language recognition (Starner and Pentland 1995; Rashid et al. 2009; Alon et al. 2009). Bhuyan (Bhuyan et al. 2005) developed a gesture recognition system using edge detection and hand tracking and FSM, TGR classification techniques for developing a platform for communication with robots. Dastur (Dastur and Khawaja 2010) controlled a robotic arm by gestures recognition using the HAAR classifier. Hardenberg (Hardenberg and Berard 2001), Hoff (Hoff and Lisle 2003), Li (Li et al. 2007), Man (Man et al. 2005) and Mohammad (Mohammad et al. 2009) have also used gesture recognition to control robots/electro-mechanical gadgets in their applications.

Raheja (Raheja et al. 2010) controls the robotic hand using human hand movement where he uses a PCA based pattern matching.

Many researchers (Gastaldi et al.2005; Kim and Lee 2008; Lee and Chun 2009; Lee and Park 2009; Nguyen et al. 2009; Nolker and Ritter 2002; El-Sawah et al. 2007; Shin et al. 2004; Verma and Dev 2009; Zhou and Ruan 2006) have used fingertip detection in their research to obtain the information about the human hand according to their applications.

Very little work has been done in the area of bent fingers' angle calculation. Nolker (Nolker and Ritter 2002) presents GREFIT where she focuses on a large number of 3D hand postures. She uses fingertips in hands as a natural determinant of hand posture to reconstruct the image. It takes grayscale images of 192 × 144 resolutions to process. Nolker uses ANN based layer approach to detect fingertips. After obtaining fingertip vectors, it is transformed into finger joint angles to an articulated hand model.

7.2 Angle Calculation

Hand Gesture based applications are mostly based on hand movements or open finger counting method. A number of research studies are currently focused on determining the number of fingers that are bent and the angles that they are bent at. Generally, methods consider a partially bent finger as open and a fully bent finger as closed. However, in natural gesture positions, it is not possible that the hand will remain straight and all open fingers would point upwards.

The information about the angle at which each finger is bent is can be very useful in specialized applications like remote cruise control, robotic hand control and even in unmanned military weapons. As the hand shape parameters (fingertips and center of palm) are known, they could be used to determine the angles for the captured gesture in real time. In this section, firstly, the geometrical angle calculation method is discussed. Then, the ANN-based method is discussed.

7.2.1 Distance Measurement Between COP and Fingertips

The distance between each fingertip and COP can be calculated by subtracting their coordinates as shown in Fig. 7.2. Here, the origin of the coordinate system is the midpoint of the wrist line. As this experiment is performed in the spatial domain, all values are the differences of pixel values. Hence, the difference is the number of pixels between the COP and a particular fingertip.

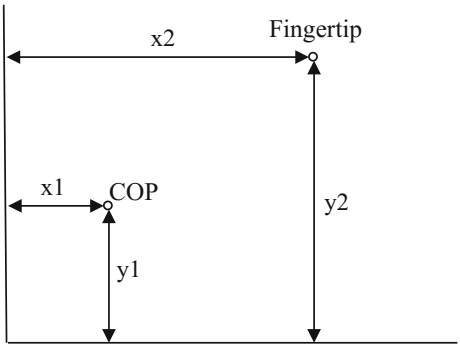

Fig. 7.2 Distance calculation between COP and fingertips

7.2.2 Fingers' Bending Angles Calculation

The process of fingers' angle calculation is very smooth. Initially, the user has to show all fingers' open gesture to the system, as shown in Fig. 7.3. This is recorded as the reference frame for this session. From the reference frame, the initial angles are stored as 180° of all fingers.

The distance between any fingertip and the COP is the maximum in this position. As the user starts bending the fingers in either direction (forward or backward), distances among fingertips and COP start decreasing. The distance values calculated from the reference frame is stored for each finger. If the user changes the position of any of his fingers, the distance between COP and fingertips is compared with the reference distances as shown in Fig. 7.4.

The angles for bent fingers are calculated by performing a comparison of these distances. In presented method, the angles can have values ranging between 90° and 180° as after bending more than 90° the base of the fingers in the hand is detected as a fingertip. Through experiments, the distance between the COP and the fingertip is

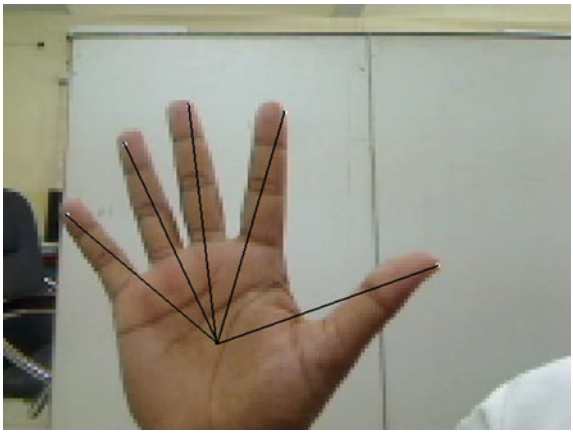

Fig. 7.3 The reference frame for angle calculation

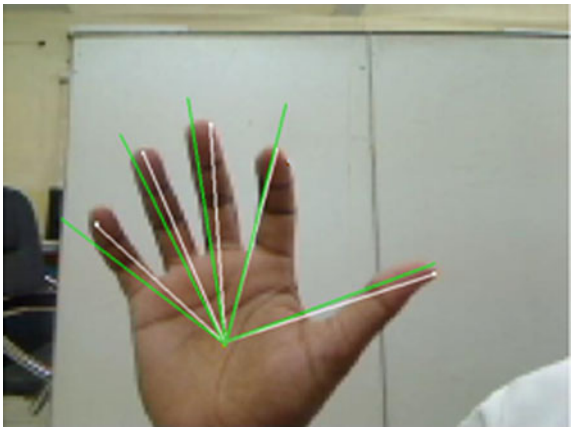

Fig. 7.4 Comparisons with reference frame: *green lines* show the reference distances and *white lines* show the current distances

assumed to be 1/3rd of the reference distance on 90° and when the angle is 180°, the distance between the COP and the fingertip becomes equal to the reference distance, as shown in Fig. 7.5.

From the Fig. 7.5 it is clear that when $d = d_{\text{ref}}$, angle $a1 = 0°$ and when $d = d_{\text{ref}}/3$, angle $a1 = 90°$. Hence, the angle $a1$ can be expressed as

$$a1 = 90° - \frac{d - d_{\text{ref}}/3}{2d_{\text{ref}}/3} \times 90° \tag{7.1}$$

The angle of finger bending i.e. $a2$, which is shown in Fig. 7.5, can be calculated by simple trigonometry as shown in (7.2), (7.3) and (7.4).

$$a2 = 180° - a1 \tag{7.2}$$

$$a2 = 180° - 90° + \frac{d - d_{\text{ref}}/3}{2d_{\text{ref}}/3} \times 90° \tag{7.3}$$

$$a2 = 90° + \frac{d - d_{\text{ref}}/3}{2d_{\text{ref}}/3} \times 90° \tag{7.4}$$

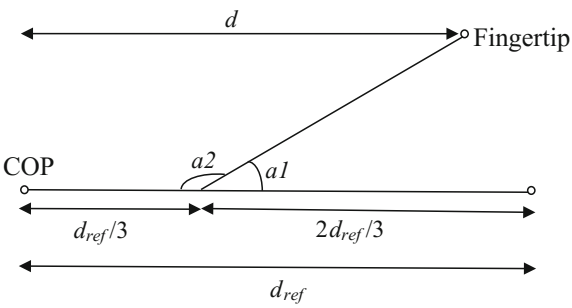

Fig. 7.5 Angle approximation method

In the same way, bending angles for all five fingers can be calculated simultaneously. Figure 7.6 presents the results of angle detection.

Table 7.1 presents results obtained from calculating the distances among fingertips and COP and their corresponding angles for the respective user inputs. This method works on any hand gesture input shown in any direction. Even if the user moves the hand position, this information can also be passed to a remote robotic hand and the robotic hand would also move in the same direction. Different hand gestures and detection of hand geometry parameters are shown in Fig. 7.7. It is clearly visible that hand gestures can be in any direction and the detection of parameters is robust, which is clearly visible in the image.

7.2.3 *Performance*

As a real-time system is needed to calculate the bent fingers' angle, the time taken by different components need to be measured in this system. Table 7.2 shows the comparative analysis of the time taken by the system in different steps. The simulation of the system is performed in MATLAB® running on Windows XP® and Pentium®4, 2.80 GHz processor. The image capturing frequency is configurable, currently, it takes 6 images which give a feel of real time video input and calculate the fingers' position angles in different gestures. There can be varying lighting conditions, in which the user provides input to the system. The maximum time is consumed during the preprocessing stage, which is followed by image cropping and

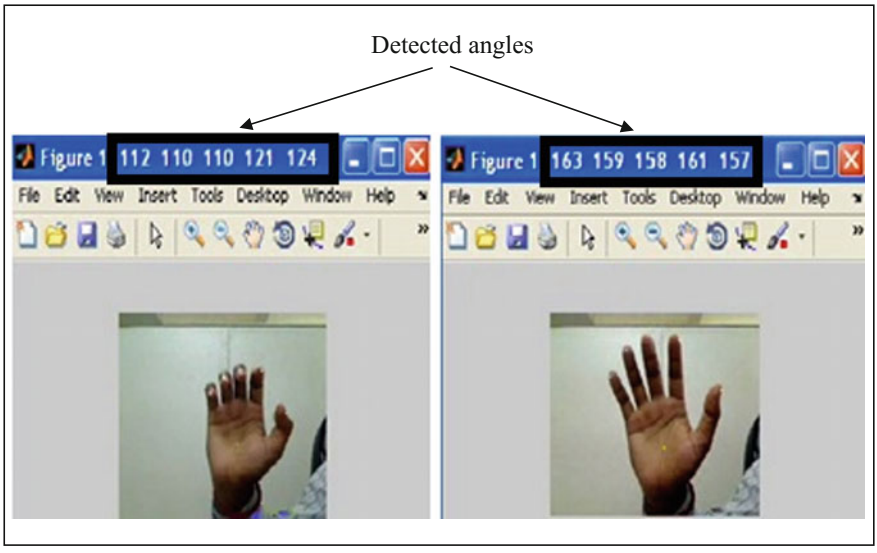

Fig. 7.6 Angle detection in one hand

Table 7.1 Distances (number of pixels) between COP and fingertips and corresponding angles (in degrees)

S. No.	Image	Finger 1		Finger 2		Finger 3		Finger 4		Finger 5	
		Distance	Angle	Distance	Angle	Distance	Angle	Distance	Angle	Distance	Angle
1.		207.06	180	229.506	180	254.283	180	255.765	180	246.14	180
2.		176.78	160.3	211.32	169.3	237.103	170.9	201.479	151.3	236.24	174.6
3.		188.138	167.7	214.308	171.1	235.936	170.3	243.298	173.4	237.191	175.1
4.		125.032	126.5	192.276	158.1	199.063	150.7	206.461	154	142.302	123
5.		144.9	139.5	149.933	133.2	146.512	122.8	106.075	101	138.679	121.1

Fig. 7.7 Fingertips and COP detections in several hand postures

hand direction detection stage. It is clear that for the robustness of the system, there is no condition on the user. The user gives an input with his hand (one hand at a time, either right or left) in random directions.

The system is tested in different conditions for a long period of time to check its sustainability in the commercial deployment and it performed excellently with different users. It is independent of the user's hand geometry and works the same for anyone. In the past Bhuyan (Bhuyan et al. 2005) performed experiments on dynamic gestures and obtained an accuracy of 80–95%. Li and Greenspan (2011) showed 88–96% accuracy for dynamic gesture recognition. Raheja (Raheja et al. 2010) obtained an accuracy of 90% in his robotic hand control work. In fact, the work presented in this chapter is the first attempt towards the angle calculation in

Table 7.2 Tabulation of computational time

S. No.	Action	Time taken (in ms)
1.	BS formation	45 (30.6%)
2.	Hand direction and cropping	39 (26.5%)
3.	COP detection	22 (15%)
4.	Fingertips detection	25 (17%)
5.	Angle of bending	16 (10.9%)
6.	Total	147 (100%)

2D under natural computing, hence there is no previous study to compare this work with. Currently, the system is working with an accuracy of 90–95% and we are trying to improve the system to make it more robust.

7.3 ANN Based Angle Calculation

This section discusses a novel method for the calculation of the positions of robotic fingers' angle using supervised Artificial Neural Network. The user has to show a gesture to the system with a bare hand as in Sect. 7.2.

7.3.1 System Description

The input to this section includes the same parameters i.e. the fingertips locations and center of the palm, which have previously been calculated in Chap. 6. The distance between the COP and fingertips has also been measured in Sect. 7.2.1. Here these distances are classified with supervised ANN. The block diagram of the proposed system is presented in Fig. 7.8. The ANN has five inputs for fingers' bending angle distances and five outputs for the angle for each finger.

7.3.2 Neural Network Architecture

The differences between COP and detected fingertips are obtained from each processed frame, which are the distances in pixels unit. They can be used for training the neural network. A supervised ANN is implemented using the

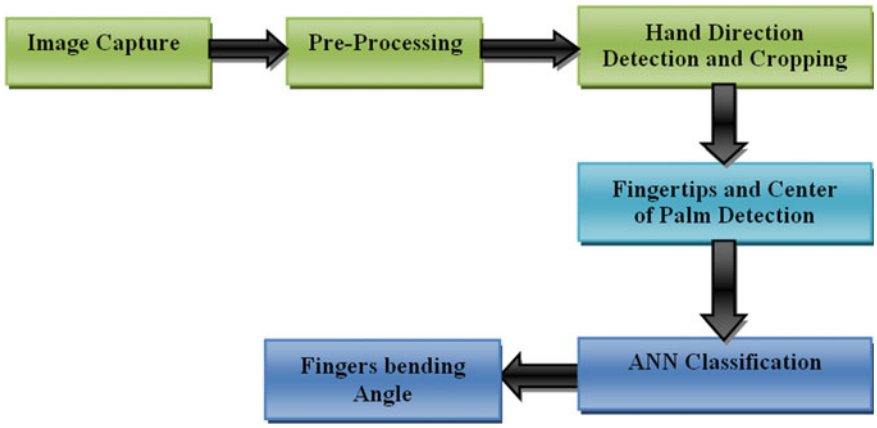

Fig. 7.8 Block diagram of the angle calculation system

Levenberg-Marquardt algorithm (Wilamowski and Chen 1999) with 8000 sample data for all fingers in different positions. This data is collected after storing the distances for different gestures depicted by different users and ANN is trained for 1000 iterations on this data. The architecture of ANN is shown in Fig. 7.9 with 5 input layers for five finger positions and 5 output layers for the angle of each finger.

The system was implemented for a number of different ANN architectures and performed a comparative study. The details of different architectures considered are shown in Table 7.3. It was determined that for the 6th combination presented in the table the test error is the minimum, while the fitness values are the maximum. Hence, this is the best fit for this application. The ANN design includes 19 neurons for processing.

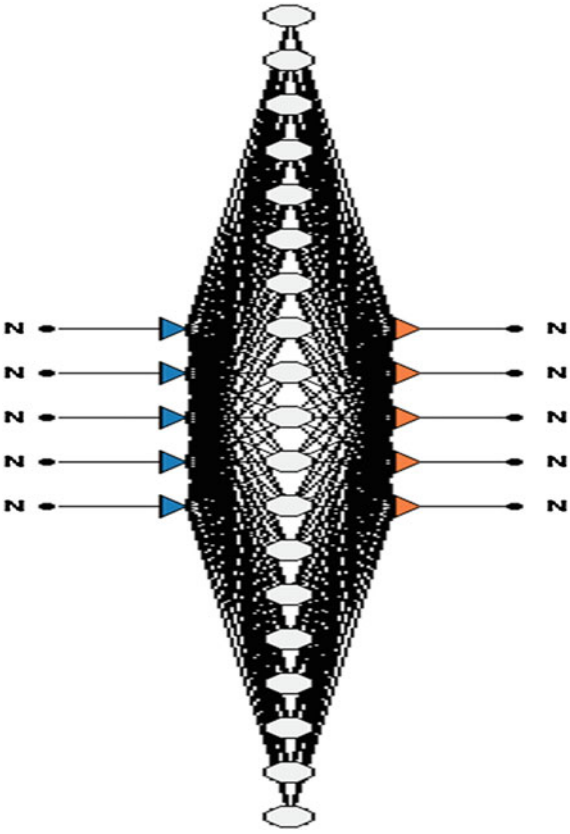

Fig. 7.9 Architecture of ANN

Table 7.3 Architecture comparison for ANN

ID	Architecture	Fitness	Train error	Validation error	Test error
1	[5-1-5]	0.007762	111.6609	123.3729	128.8286
2	[5-9-5]	0.029136	21.07543	25.35231	34.32153
3	[5-14-5]	0.030816	17.03908	26.58807	32.45031
4	[5-17-5]	0.028309	22.56162	26.80747	35.32473
5	[5-18-5]	0.031086	20.17483	25.85577	32.1686
6	**[5-19-5]**	**0.037425**	**12.60105**	**22.93995**	**26.71978**
7	[5-20-5]	0.034877	12.1308	25.62003	28.67238
8	[5-21-5]	0.034308	12.13067	24.02896	29.14752
9	[5-23-5]	0.03166	14.48353	22.33495	31.5859

7.3.3 Neural Network Training

A number of algorithms are available for the ANN training. The Levenberg-Marquardt algorithm is designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feed-forward networks) then the Hessian matrix can be approximated as

$$\mathbf{H} = \mathbf{J}^{\mathbf{T}}\mathbf{J} \tag{7.5}$$

And the gradient can be computed as

$$\mathbf{g} = \mathbf{J}^{\mathbf{T}}\boldsymbol{e} \tag{7.6}$$

where **J** is the Jacobian matrix that contains the first derivatives of the network errors with respect to the weights and biases and **e** is a vector of network errors. The Jacobian matrix can be computed through a standard back propagation technique which is much less complex than computing the Hessian matrix. The Levenberg-Marquardt algorithm uses this approximation (Wilamowski and Chen 1999) to the Hessian matrix using the following (7.7).

$$\mathbf{x}_{\boldsymbol{k}+1} = \mathrm{x}_k - \left[\mathbf{J}^{\mathbf{T}}\mathbf{J} + \mu\mathbf{I}\right]^{-1}\mathbf{J}^{\mathbf{T}} \tag{7.7}$$

When the scalar μ is zero, then (7.7) represents simply a quasi-Newton's method using the approximate Hessian matrix. When μ is large (7.7) represents a gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, hence the aim is to shift toward the Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step increases the performance function. Hence, the performance function always reduces in each iteration of the algorithm.

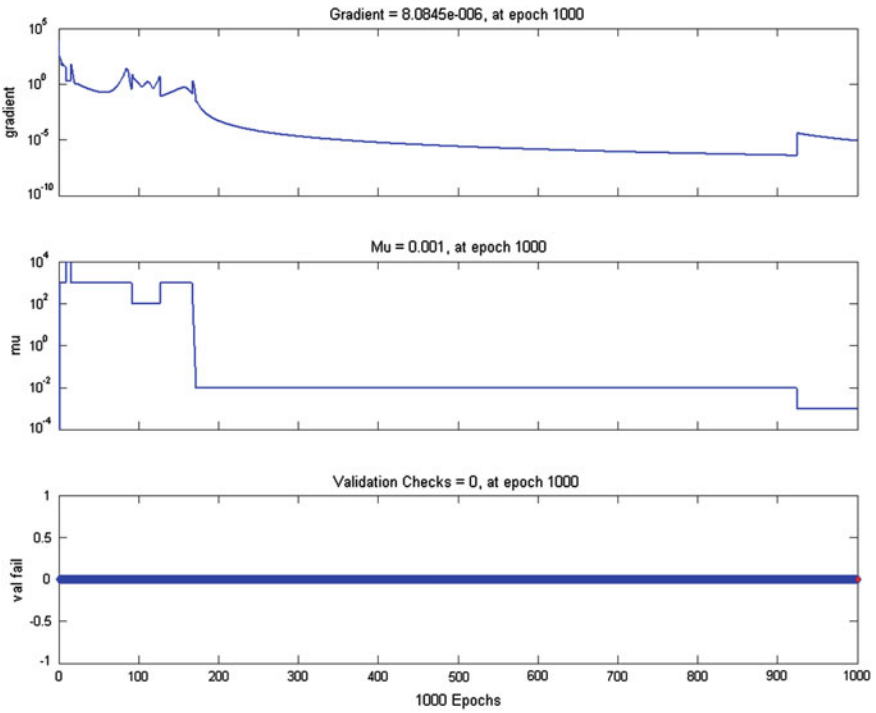

Fig. 7.10 Training state using 1000 iterations

This algorithm appears to be the fastest method for training moderate-sized feed forward neural networks (up to several hundred weights). The training state during iterations and data validation for this system is shown in Figs. 7.10 and 7.11

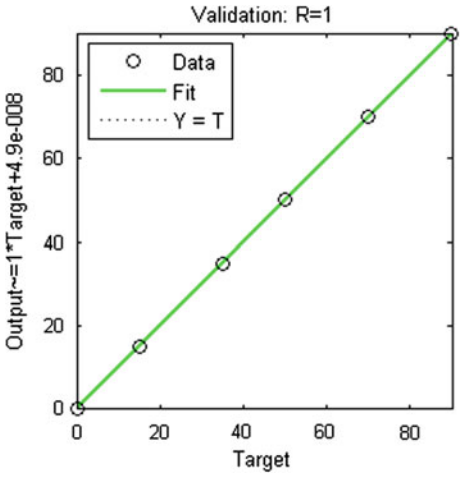

Fig. 7.11 Data validation state graph

Table 7.4 Distances from the center of palm to each fingertip in pixels

Angles	Index	Middle	Ring	Little	Thumb
0	81	87	82	75	57
15	79	84	79	70	53
30	77	80	76	67	50
45	73	72	68	64	45
60	60	63	61	60	40

respectively. The input data includes the distances (in terms of pixels) from COP to fingertips for all fingers at different angles. Table 7.4 presents the values for one random test for all fingers. The Mean squared error in ANN is of the order of 10^{-12} as shown in Fig. 7.12.

7.3.4 Experimental Results

The ratio of the new distance in the image frame to reference distance is sent to ANN to classify it and ANN provides the corresponding angles for all the five fingers. The response time of the system to display angles for one gesture is

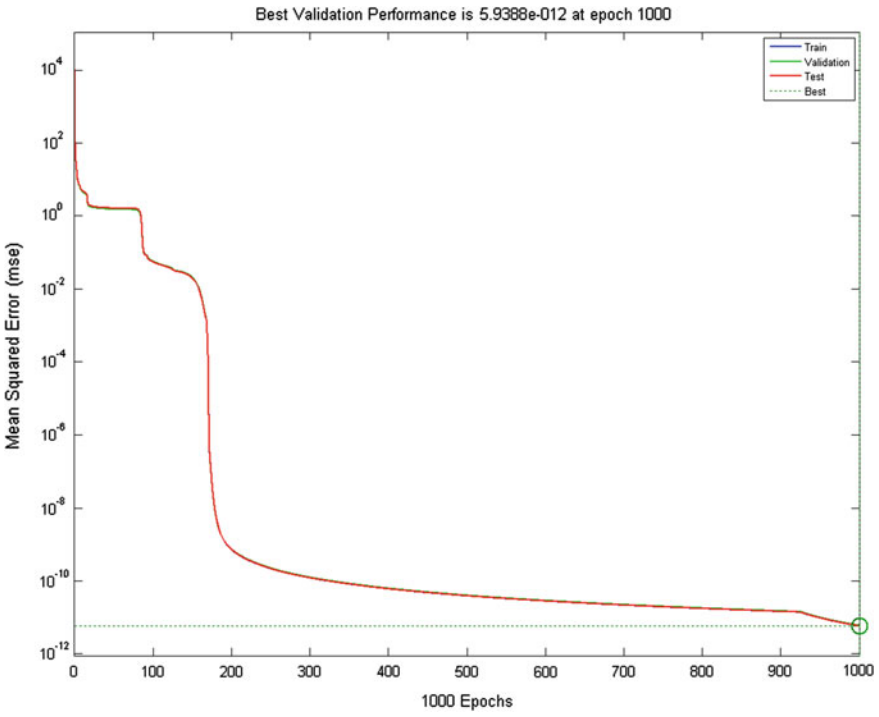

Fig. 7.12 Mean squared error in the ANN

0.001 s, which is very satisfactory and is close to real time. The system captures frames of 160 × 120 resolutions in RGB format, which is processed according to the technique previously described in Chap. 4. The reference distance is taken from the first frame of the hand and this reference distance stays the same throughout the session of the system. If the hand disappears from the camera view, the system will remove the reference distance and will take the new reference distance in the same session. Hence, different users can operate in the same session. Figure 7.13 shows few snapshots of the angle calculation system.

The five values shown give the angles of fingers in sequence as they appear in the input image. The system takes live input from the user and calculates angles in real time. Hence, these values can be used to feed to electro-mechanical robotic hand control software so that it can do the same operation there. The simulation of the robotic hand has been done in the blender® software, where the simulated hand operates on the basis of the values provided by the developed system.

Fig. 7.13 Results of fingers' bending angle computation

7.4 Conclusion

This work presents a new research direction in gesture recognition techniques. Previously very little work has been done on closed/bent fingers detection. This chapter discusses the angle calculation process for one hand from the hand gesture shown to the system. This is done for both the right and left hand. The bending angles of fingers are calculated using a time efficient geometric modeling method and the same are obtained using ANN. The user can control the robotic hand using his gesture without wearing any gloves or markers. The results are satisfactory and this technology can be used in a number of real life applications where it is preferable to employ a robotic hand than a human hand.

As, in the real scenario, the user uses both hands together to show gesture. It is necessary to find both hands' gesture to fulfill natural computing requirement. Also, many applications need both hands to work together, so angle calculations for both hands would be required. A parallel algorithm for both hands angle calculation is discussed in detail in next chapter.

References

J. Alon, V. Athitsos, Q. Yuan, S. Sclaroff, A unified framework for gesture recognition and spatio-temporal gesture segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **31**(9), 1685–1699 (2009)

M.K. Bhuyan, D. Ghosh, P.K. Bora, Designing of human computer interactive platform for robotic applications, in *TENCON IEEE Region,* vol. 10, pp. 1–5, 21–24 Nov 2005

J. Dastur, A. Khawaja, Robotic arm actuation with 7 DOF using Haar classifier gesture recognition, in *Proceedings of the Second International Conference on Computer Engineering and Applications*, vol. 1, pp. 27–29, March 2010

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, in *Proceedings of the Fourth Canadian Conference on Computer and Robot Vision*, pp. 495–502, Montreal, Canada, 28–30 May 2007

W.T. Freeman, Computer vision for television and games, in *Proceedings of International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pp. 118 (1999)

G. Gastaldi, A. Pareschi, S.P. Sabatini, F. Solari, G.M. Bisio, A man-machine communication system based on the visual analysis of dynamic gestures, in *Proceedings of IEEE International Conference on Image Processing*, Genoa, Italy, vol. 3, pp. III-397–400, 11–14 Sept 2005

C.V. Hardenberg, F. Berard, Bare hand human computer interaction, in *Proceedings of the ACM workshop on Perceptive user interfaces*, Orlando, Florida, USA, (2001), pp. 1–8

W.A. Hoff, J.C. Lisle, Mobile robot control using a small display, in *Proceedings of International Conference on Intelligent Robots and Systems*, Golden, CO, USA, vol. 4, pp. 3473–3478, 27–31 Oct 2003

J.M. Kim, W.K. Lee, Hand shape recognition using fingertips, in *Proceedings of Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, Jinan, Shandong, China (2008), pp. 44–48

B. Lee, J. Chun, Manipulation of virtual objects in marker-less AR system by fingertip tracking and hand gesture recognition, in *Proceedings of the Second International Conference on*

Interaction Science: Information Technology, Culture and Human, Seoul, Korea, pp. 1110–1115 (2009a)

D. Lee, Y. Park, Vision-based remote control system by motion detection and open finger counting. IEEE Trans. Consum. Electron. **55**(4), 2308–2313 (2009b)

H. Li, M. Greenspan, Model-based segmentation and recognition of dynamic gestures in continuous video streams, in *Pattern Recognition*, August 2011

Z. Li, S. Wachsmuth, J. Fritsch, G. Sagerer, View-adaptive manipulative action recognition for robot companions, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1028–1033, 29 Oct 2007

W.T. Man, S.M. Qiu, W. K. Hong, ThumbStick: a novel virtual hand gesture interface, *IEEE International Workshop on Robot and Human Interactive Communication*, pp. 300–305, August 2005

Y. Mohammad, T. Nishida, S. Okada, Unsupervised simultaneous learning of gestures, actions and their associations for human-robot interaction, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2537–2544, 10–15 Oct 2009

D.D. Nguyen, T.C. Pham, J.W. Jeon, Fingertip detection with morphology and geometric calculation, in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, USA, pp. 1460–1465, 11–15 October 2009

C. Nolker, H. Ritter, Visual recognition of continuous hand postures. IEEE Trans. Neural Netw. **13** (4), 983–994 (2002)

J.L. Raheja, R. Shyam, U. Kumar, P.B. Prasad, Real-time robotic hand control using hand gesture, in *Proceedings of the Second International Conference on Machine Learning and Computing*, Bangalore, India, pp. 12–16, 9–11 Feb 2010

O. Rashid, A. Al-Hamadi, B. Michaelis, A framework for the integration of gesture and posture recognition using HMM and SVM, in *IEEE International Conference on Intelligent Computing and Intelligent Systems*, Shanghai, pp. 572–577, 20–22 Nov 2009

M.C. Shin, L.V. Tsap, D.B. Goldgof, Gesture recognition using Bezier curves for visualization navigation from registered 3-D data. Pattern Recogn. **37**(5), 1011–1024 (2004)

T. Starner, A. Pentland, Real-time American sign language recognition from video using hidden Markov models, in *Proceedings of International Symposium on Computer Vision*, pp. 265–270, 21–23 Nov 1995

J. Triesch, C.V.D. Malsburg, Mechanical gesture recognition, in *Gesture Workshop*, pp. 233–244 (1997)

R. Verma, A. Dev, Vision-based hand gesture recognition using finite state machines and fuzzy logic, in *Proceedings of the International Conference on Ultra-Modern Telecommunications and Workshops*, pp. 1–6, 12–14 Oct 2009

B. Wilamowski, Y. Chen, Efficient algorithm for training neural networks with one hidden layer, in *Proceedings of International Joint Conference on Neural Network*, vol. 3, pp. 1725–1728 (1999)

H. Zhou, Q. Ruan, A real-time gesture recognition algorithm on video surveillance, in *Proceedings of Eighth International Conference on Signal Processing*, Beijing, China, pp. 1754–1757, November 2006

Chapter 8
Both Hands' Angles Calculation

In natural communication, people use both hands to express themselves while talking. The bent fingers' angle for one hand can be detected using the method discussed in the previous Chapter. There is a need to detect the angles for both hands to make the gesture-based system more robust. In this chapter, a novel method for angle approximation of both hands' bent fingers is presented and its application to a robotic hand control is discussed. The system uses a simple camera and a PC.

8.1 Issues

In the case of both hands, it is obvious that the computational time required is greater than that for a single hand. The approach described in Chap. 7 can also be used for this purpose with some modifications in the algorithm. However, the process is time-consuming, specifically, almost twice the amount of time (for a single hand) is required when both hands are considered. It is not always true that the previous approach will consume twice the amount of time if the directions of both the hands are the same. The time consumption will be the same as that of the single hand, but in the real sense, the directions of both the hands need not always be the same.

8.2 Both Hands' Angle Calculation

For calculating the angle for both hands, the algorithm (presented in Chap. 7 for a single hand) can be applied twice to the image frame. This leads to additional time consumption, which is not desirable in real-time applications. It is essential that the computational time should be less for real-time performance. Hence, a new approach

A. Chaudhary, *Robust Hand Gesture Recognition for Robotic Hand Control*,
DOI 10.1007/978-981-10-4798-5_8

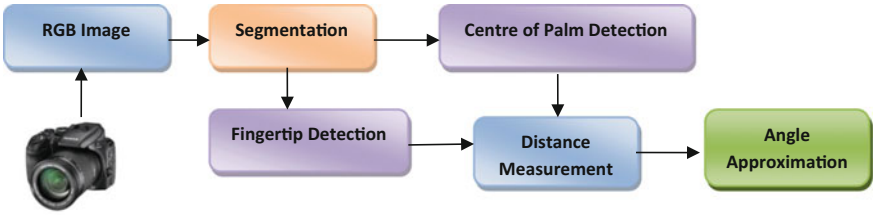

Fig. 8.1 Algorithmic flow for angle calculation for both hands

is developed for calculating the angle of the fingers of both hands. This approach is presented by the block diagram shown in Fig. 8.1. The COP and the fingertips are detected from the segmented image and both hands fingers' angle were calculated in parallel. As for both hands, the detection process is carried out in parallel. Hence, the time consumed is smaller than that consumed when the algorithm presented in Chap. 7 is implemented twice, separately for both hands. A detailed discussion on parallel image processing is presented in (Bräunl et al. 2001).

8.2.1 Pre-Processing

As there are two hands in the captured gesture image and it is a need now to detect both of them. For this, a change in segmentation technique is needed. The HSV color space-based skin filter is used to form the binary silhouette of the input image. The hands are segmented using the same method as discussed in Sect. 4.2.1, where the ROI is determined. After the formation of the binary image, two binary linked objects (BLOBs) are collected and the BLOB analysis based on 8 connectivity criteria is performed. As a result of that two hands would be distinguished from each other. It makes sure that while calculating the distance between COP and fingertips, the system does not make any mistake by considering the fingertip of one hand and COP of another hand. The main purpose of the BLOB analysis is to extract the two biggest BLOBs to eliminate the fault detection of skin and the skin color found in the background and to distinguish the two BLOBs from each other. Figure 8.2 presents the results for hand segmentation. The brighter BLOB corresponds to the right hand of the main frame and another BLOB corresponds to the left hand.

8.2.2 Fingertip Detection

The fingertip detection discussed in Chap. 7 is suitable for the simultaneously simultaneous processing of both the hands if their directions are not same. To process both hands simultaneously, a new approach is used that is based on the circular separability filter (CSF) and concentric circular filter (CCF) (Raheja et al. 2010).

Fig. 8.2 Result of both hands' segmentation

The CSF has the shape of a square with a concentric circle inside as shown in Fig. 8.3. After performing a number of experiments it was decided to consider the radius of the circle to be equal to 5 pixels and the bounding square to have a size of 20 pixel length.

After applying the filter, the filter response is computed for all the points of the region of interest in the binary image. The filter responses for the fingertip regions are found to be distinctively different from that of other regions. The candidate fingertip locations are determined by using an appropriate threshold condition.

The exact fingertip location is calculated in two steps. First, an approximate location for the fingertip is determined and then the exact location is calculated by further calculation using the orientation of the finger. The 8-connected points that satisfy the threshold condition for the filter response of the CSF are grouped together. The groups that have a number of pixels greater than a set threshold are selected and the centroids of the groups are taken to approximate the fingertip position.

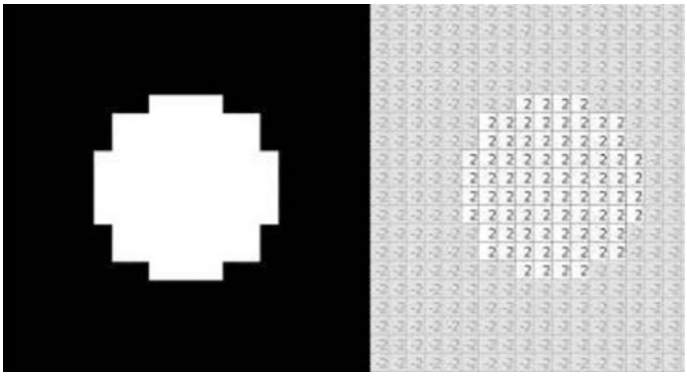

Fig. 8.3 Circular separability filter

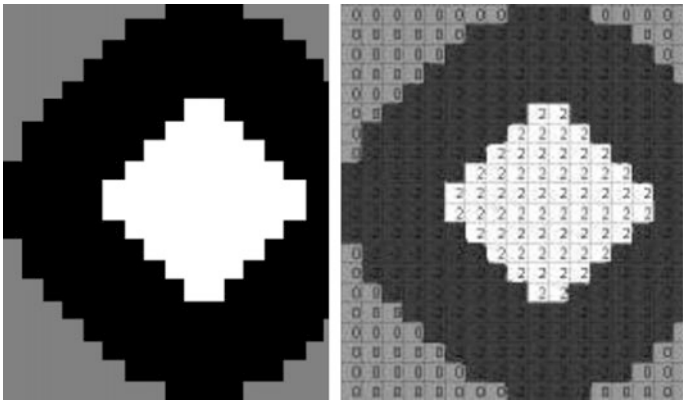

Fig. 8.4 Concentric circular filter and assigned element values

For the exact fingertips location, the orientation of each finger is determined using a filter with 2-concentric circular regions. The CCF is shown in Fig. 8.4. The diameters of the inner and outer circular regions of the filter are 10 and 20 pixels, respectively. The points inside the inner circle are assigned a value of +2, the points outside the inner circle but inside the outer circle are assigned the value −2 and the points outside the outer circle but inside the bounding square are assigned the value 0.

The filter is then applied to the binary silhouette of the hand image at the previously detected approximate fingertip locations. The pixels that lie in the −2 region are grouped under the 8-connectivity criteria. Then, the largest group is selected and the centroid of the group is calculated as previously discussed. The orientation of the finger is calculated as the angle (say θ) made by the line joining the centroid of the largest group and the previously calculated approximate fingertip location with the horizontal axis. This is graphically shown in Fig. 8.5.

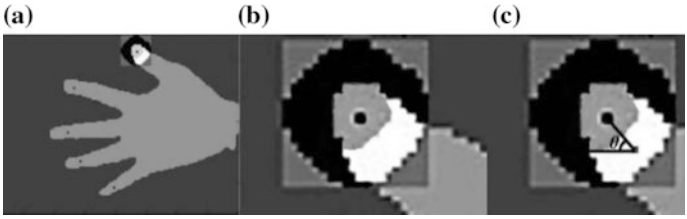

Fig. 8.5 Fingertip detection **a** CCF applied on the approximate thumb tip location **b** zoomed view of the thumb tip region **c** position of the centroid of the largest 8-connected group (the region in *white*) and the angle (θ) with respect to the horizontal

Fig. 8.6 Result of both hand COP and fingertip detection

After moving several steps in this direction with an incremental distance (say r) using (8.1) till the edge of the finger was reached.

$$R_{\text{new}} = R_{\text{old}} + r\cos(-\theta) \tag{8.1}$$

$$C_{\text{new}} = C_{\text{old}} + r\sin(-\theta) \tag{8.2}$$

where R_{old} and C_{old} are the 2D coordinates of the previous trace point while R_{new} and C_{new} are the 2D coordinates of the current tracepoint. The values of R_{new} and C_{new} obtained after performing a few iterations provide the exact coordinates of the fingertips. The results of fingertip detection are presented in Fig. 8.6 by white points.

8.2.3 Center of Palm Detection

Center of Palm (COP) detection is performed based on the same approach as discussed previously in Chap. 6. The exact locations of the COP in the hands are

identified by applying a mask of dimension 30 × 30 to the binary silhouette of the image and counting the number of pixels lying within the mask. If the numbers of pixels are within a set threshold then the center of the mask is considered as the nominee of COP and finally the mean of all the nominees found in a BLOB is considered as the COP of the hand represented by that BLOB. Since there are two BLOBs, results contain two COPs identifying the COP of both the hands. Figure 8.6 presents the result of COPs detection as yellow dots.

8.3 Angle Calculation

Here is the extension of the approach described in Sect. 7.2.2 to determine the fingers' angles in both hands. This geometrical method does not require any training or data to calculate the angles for both hands. The distance between each fingertip and the COP can be calculated by subtracting their coordinates as discussed in Sect. 7.2.1. Initially, the user has to show a reference frame to the system in which all fingers are open and the bending angle of all fingers are 180°. The user can move his hands in front of the camera as it is not necessary to have his hand and/or arm static. The method would also calculate the angles in such a scenario.

The method given in Sect. 7.2.2 is based on trigonometry and performs a comparison between a reference distance and the new distance. For both hands, measure the distances between the COP and the fingertips and then using the analysis from Chap. 7, the bending angle (say $a2$) can be calculated as described is (8.3). For more information about the trigonometric concepts see Fig. 6.5.

$$a2 = 90^\circ + \frac{d - \frac{d_{\text{ref}}}{3}}{\frac{2d_{\text{ref}}}{3}} \times 90 \tag{8.3}$$

where the angle $a2$ approximates the value of finger bending angle. Figure 8.7 shows the result of fingers' angle detection for both hands. The angles are shown at the top of the window according to the finger shown sequence from the first to the last one as they appear in the captured image frame.

8.4 Experimental Results

The method uses no training data and both the hands can be oriented in any direction. The usage of the system is very similar to that discussed in Chap. 7. The experimental environment is based on Intel® i5 processor and 4 GB RAM desktop computer. The discussed method is implemented in MATLAB® on Windows® XP. The live video is captured using Logitech® HD webcam with image resolution 240 × 230. If single hand finger's angle calculation method is applied to detect both hands fingers' angle, the computational time is 294 ms. On the other hand, the

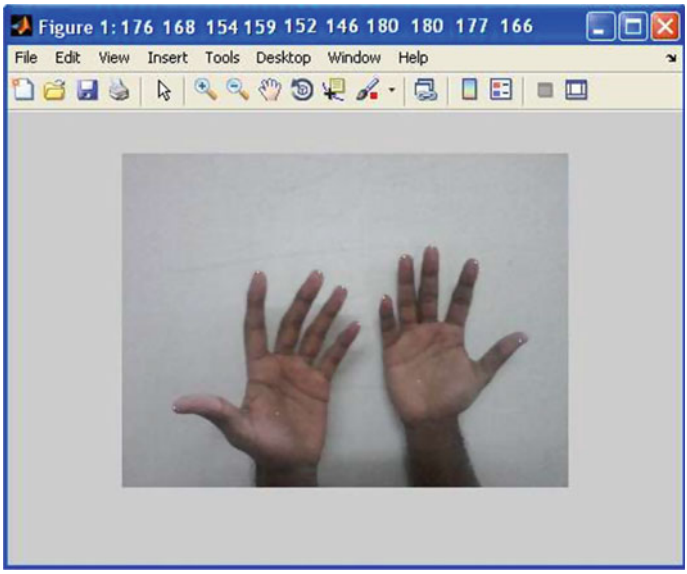

Fig. 8.7 Finger bending angle calculation of double hand

method proposed in this paper takes only 198 ms to perform the same computations. Both the hands are recognized distinctly, as the system remembers both hands' parameter separately. If there is only one hand shown, the system will work perform a single hand gesture analysis. This method provides an accuracy of around 90–92% on live input in different light conditions.

8.5 Conclusion

This chapter presents a novel technique for the detection of the angles defining bent fingers in human hands. The technique presented in this paper carries tremendous significance since it can be adopted for identifying human gestures, which are often depicted during communication using both hands. The system considered uses no training data and the hands can be used in any direction. This approach minimizes the processing time of the last presented algorithm for determining the angles of a single hand. The processing time is reduced by 96 ms, which corresponds to a reduction of approximately 33%.

This book has presented issues from the literature and discussed the light invariant technique for gesture recognition. Also for the controlling of the robotic hand, angles from the human hands are also calculated.

References

T. Bräunl, S. Feyrer, W. Rapfand, M. Reinhardt, Parallel Image Processing, Springer Verlag, Germany (2001)

J.L. Raheja, R. Jain, P. Mohapatra, Visual approach to dynamic hand gesture recognition for human computer interface, Int. J. Recent Trends Eng. Technol. **3**(3), 190–194