الجامعـــة الإسلاميـــة للتكنولوجيا
UNIVERSITE ISLAMIQUE DE TECHNOLOGIE
ISLAMIC UNIVERSITY OF TECHNOLOGY
DHAKA, BANGLADESH
ORGANISATION OF ISLAMIC COOPERATION

HAND GESTURE RECOGNITION USING DEPTH INFORMATION AND DTW

By

MOUNTAPMBEME ABOUBAKAR
Student Id: 104445

Supervised By

Mr. HASAN MAHMUD
Assistant Professor
CSE DEPARTMENT, IUT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
October, 2014

BOARD BAZAR, GAZIPUR-1704, BANGLADESH
Tel: 9291250, 9291252 PABX: 9291254-59 Fax: 880-2-9291260 e-mail: vc@iut-dhaka.edu website : www.iutoic-dhaka.edu

DECLARATION

This work, **Hand Gesture Recognition using Depth Information and DTW,** is the output of the research carried out by Mountapmbeme Aboubakar (Student Id. 104445) under the supervision of Mr. Hasan Mahmud (Assistant Professor, CSE Department, IUT). Completed and submitted to the Department of Computer Science and Engineering (CSE), in Partial Fulfilment of the Requirements leading to the award of the degree of Bachelor of Science in Computer Science and Engineering (B.Sc. Engg. CSE).

Mountapmbeme Aboubakar

Student Id: 104445

Mr. Hasan Mahmud

Assistant Professor, CSE, IUT

Prof. Dr. M. A. Mottalib

Head, Department of CSE, IUT

DEDICATION

To my elder brother, ASSANEGOU SOULE, who endlessly advises and supports me to become successful not only in this world, but also in the Hereafter.

ACKNOWLEDGEMENT

All the praises and thanks be to Allah, the Lord of the Alamin (mankind, jinn and all that exists). Qur'an (V. 1:2).

First and foremost we thank the Almighty Allah (SWT) for blessing us with this beautiful thesis idea and for giving us the knowledge, strength, determination, spirit of team work and for accompanying us throughout the successful completion of this thesis.

Our special thanks goes to Mr Hasan Mahmud, our thesis supervisor, who directed and supported us throughout the successful completion of this thesis.

We express our sincere gratitude and thanks to Prof. Dr. M. A. Mottalib, head of the Computer Science and Engineering Department of the Islamic University of Technology and the entire staff of the department for the work and efforts they are doing to teach us the knowledge of computer science and engineering.

Abstract

Human computer interaction has introduced a new paradigm for the construction of computer interfaces. This requires the construction of easy to use, natural and intuitive computer interfaces. Various methodologies have been considered to build such interfaces. Gestures and especially hand gesture represent the most easy to use, natural, intuitive and memorable way to communicate with computers. However, gesture recognition systems are difficult to design. These systems are often used in complex environments with cluttered background and different lighting conditions. These factors act as major constraints in the design of gesture recognition systems. However, the introduction of depth sensing devices such as the Microsoft Kinect makes it easy to capture hand gestures in complex backgrounds. The accuracy and efficiency of gesture recognition systems are often affected by the choice of features to be used to represent the hand shape and the algorithm to be used for classifying the hand gestures. In this thesis, we try to implement a hand gesture recognition system that makes use of the time-series representation of the contour points of the hand shape and uses Dynamic Time Warping (DTW) for classification of hand gestures. DTW is well known for its accuracy and effectiveness in matching time-series representations. Our proposed system makes use of the depth information of the scene provided by the Microsoft Kinect. This allows our system to be used in challenging backgrounds. We evaluate our system and compare it to some contemporary systems. The results of evaluation shows that our hand gesture recognition system is accurate and efficient with a mean accuracy of 94.6% and mean running time of 0.5179s. Our system is also invariant to scale, rotation and translation and runs effectively in complex background settings. We also survey some of the methods and algorithms used for recognising hand gestures. Finally we explore some of the application areas of gesture base interfaces. Interaction using gestures has proven to be useful in many domains such as in medicine, in classroom for teaching and presentations, in sign language recognition, interactive emotion recognition, simulations and in the gaming industry.

Keywords

Human Computer Interaction, HCI, Gesture Recognition, Dynamic Time Warping, DTW, Microsoft Kinect.

Contents

List of Tables

List of Figures

CHAPTER 1

INTRODUCTION

1.1. THE CONTEXT

Recently, a lot of efforts and work is been made to reduce the gap between humans and computers. This involves bringing the interaction between humans and computers to the human level where humans can communicate with computers in natural and intuitive ways. This issue among others is addressed by Human-Computer Interaction (HCI). Among the various interaction techniques used to achieve this goal, interaction using gestures appears to be the most efficient in terms of naturalness and intuitiveness. However, modelling gesture based interactive systems is not an easy task and presents a lot of challenges. Many devices and algorithms have been developed to recognize and classify human gestures for communication with computers [1]. Among the human gestures used for interacting with computers, hand gestures represent the simplest and most easy to use. Hand gestures are expressive motion of the hand, including the fingers in order to communicate. Humans frequently gesture with their hand while they communicate. Thus interaction between humans and computers using hand gestures provide the most effective way to model naturalness. However, effective and efficient hand gesture recognition systems are yet to be developed.

The Microsoft Kinect [2] [3] is a motion sensing input device developed by Microsoft originally for use with the Xbox gaming console. However, after introducing this device, its potentials for used with computers uncovered and as a result Microsoft released the Microsoft Kinect for Windows alongside the Microsoft Kinect for Windows SDK to help developers build computer applications by exploiting the motion sensing and audio input capabilities of this device. The Microsoft Kinect has been used recently for capturing hand gestures for used in gesture recognition systems [2] [4] [5].

1.2. THE PROBLEM

As mentioned above, designing efficient hand gesture recognition systems still poses a lot of challenges and such systems are yet to be developed. Despite these challenges a lot of efforts have been made in the past years. Generically, a hand gesture recognition system is composed of 3 steps: image acquisition and hand segmentation, feature extraction & representation, and gesture recognition. These steps individually pose a lot of challenges. Different researches [6] [7] have been conducted to improve the efficiency of the individual steps and of the system as a whole. However, it is difficult to find a system that is efficient in all the three steps combined. Therefore designing a systems that is efficient as a whole requires considering efficiency at the modular level as well as how to integrate these modules to increase the overall efficiency of the system. This is of primary concern.

1.3. THE SOLUTION

In this thesis, we design, implement and evaluate a hand gesture recognition system that uses techniques that have been proven efficient at the individual levels and which combine efficiently to produce a robust hand gesture recognition system. Our system uses the Microsoft Kinect for image acquisition and segmentation, a feature representation based on the global features of the hand and Dynamic Time Warping (DTW) for the gesture recognition module. This thesis also surveys the different type of methods and techniques used to recognise hand gestures as well as describes some of the application areas of gesture based interactions.

1.4. INNOVATIVE ASPECTS

We have demonstrated the efficiency of DTW in classifying hand gestures based on contour information represented as time-series. Our hand gesture recognition systems overcomes some of the shortcomings of the existing systems. This systems operates well under cluttered background i.e. no constraint is place on the environment of the gesturer. The systems is also translation, rotation and scale invariant. This system can be adapted to recognise any set of predefined gestures.

1.5. STRUCTURE OF THE THESIS

As mentioned above, this thesis designs a new gesture recognition system as well as surveys the different methods and techniques used so far for recognising gestures. Chapter 2 gives a brief overview of the different types of human gestures used for interacting with computers. Chapter 2 also discuss in a concise way the various devices, methods and techniques used for recognising these gestures. We conclude this chapter with a brief overview of some of the application areas of gesture recognition systems. Chapter 3 describes the problem associated with some of the gesture recognition systems introduced in chapter 2. Chapter 4 describes our proposed system in detail. In chapter 5 we present our experimental results. Chapter 6 cover some of the related work as well as the comparison of our systems with these systems. Finally, chapter 7 concludes our system as well as presents some future directions of our work.

CHAPTER 2

BACKGROUND

2.1. GESTURES

Designing gesture based interaction systems naturally begins with understanding what gestures are. This requires us to understand the different types of gestures expressed by humans, their mechanism of operation, how gestures convey information and their context of use. As stated in [1], gestures are expressive, meaningful body motions involving physical movements of the fingers, hands, arms, head, face, or body with the intent of conveying meaningful information or interacting with the environment. From this definition, we observe that we have facial gestures, that is those gestures involving the face which are mostly emotional in nature, gestures involving the hands, arms and fingers which are the most common and widely used in communication especially in sign language and finally we have gestures involving the motion of the whole body. Naturally, we have two main categories of gestures and this affects how gestures are recognised [1]. We have static and dynamic gestures. Static gestures are simply poses, where a person assumes a certain pose for a fixe time period such a making a sign with the hand to indicate 'stop'. Dynamic gestures are those which change in time and space where the gesture takes a particular path or trajectory to convey a meaningful information such a 'waving' to say good bye. Intuitively, dynamic gesture are more difficult to recognise than static gestures. In [1], you will find detail description of the different types of human gestures as well as some of the techniques and methods used to recognise each type of these gestures.

2.2. DEVICES USED FOR CAPTURING GESTURES

In order to recognise gestures, you need to track and capture the gesture. Many devices have been develop for tracking or capturing hand gestures.

Until recently, gestures, especially hand gestures have been tracked or captured accurately by using optical and electromechanical sensors [1] [8]. Electromechanical sensing devices capture hand gestures accurately and efficiently but they do not provide the naturalness and easy to use paradigm stated by HCI. These devices need to be attached to or worn by the user in order to recognised gesture. As we can see, they are cumbersome, add weight to the user and even hinder the free movement of the user thus eliminating the naturalness sought. Some of these devices include data gloves, tracking suites etc. Electromechanical devices are very efficient in capturing hand gestures in the sense that they directly track the motion of the hand and convey the signal accurately to the computer. A small wearable three-axial inertial sensor is used in [8] to track the hand motion and subsequently send signal to the computer.

Due to the limitations of electromechanical motion sensing devices, optical or vision based sensors where introduced in gesture recognition [1]. This include simple cameras. The camera is used to track the hand motion in video frames and these frames are then used for recognising

gestures. These sensors provide an improvement in the naturalness over the electromechanical sensors but in exchange, the accuracy and efficiency are greatly reduced due to the limitations of optical sensors. The optical sensors often require special lighting conditions to accurately track a gesture, they are sensitive to cluttered background and the obtained hand pose is highly distorted in most of the cases [1].

To overcome the limitations of electromechanical sensors and vision based sensors, depth based sensors were recently introduced for capturing hand gestures [2] [6] [9]. Depth based sensors such as the Microsoft Kinect introduced in chapter 1 are increasingly being used for capturing hand gestures due to their efficiency and accuracy. These sensors use the depth information of a video frame to correctly segment the hand region from the image, therefore they rely on the 3D data of the frame. As a result, their contactless nature as opposed to electromechanical sensors and their ability to rely on depth information rather than the 2D colour information used by vision based sensors makes them the most effective and robust device currently in used for capturing hand gestures. Depth based gesture recognition systems have proven to be resistant to cluttered background and lighting conditions as described in [6] [9].

The Microsoft Kinect is increasingly being used in gesture recognition systems for image acquisition and segmentation [2] [4] [5] [9]. Briefly, the Microsoft Kinect (also called Kinect) is a physical device that contains a camera, a microphone array, and an accelerometer as well as a software pipeline that processes colour, depth, and skeleton data.

Figure 1: Microsoft Kinect

The Microsoft Kinect has the following components and specifications

- An RGB camera that stores three channel data in a 1280x960 resolution
- An infrared (IR) emitter and an IR depth sensor
- A multi-array microphone, which contains four microphones for capturing sound

- A 3-axis accelerometer configured for a 2G range, where G is the acceleration due to gravity.

Other than the Microsoft Kinect, other 3D sensing devices are common nowadays. Examples include the Wavi Xtion by ASUS. The detail description of the hardware components and specifications of the Microsoft Kinect can be found in [3] or in the Microsoft Website itself.

2.3. ALGORITHMS COMMONLY USED IN GESTURE RECOGNITION

Many algorithms have been used so far for recognising gestures [1]. Theoretically, recognising gestures involves the application of principles and concepts from diverse domains. This includes image processing concepts, statistical models, pattern recognition concepts, computer vision techniques and many others. Tools for recognising gestures include among others Hidden Markov's Models (HMM), Principal Component Analysis, Artificial Neural Networks (ANN), Dynamic Time Warping (DTW), connectionist approach, template matching and many others. In [1] and [7] a detail description of each of these algorithms and approaches have been given and a description of how each of these methods are used in the recognition of the different types of gestures stated above.

The type of gesture determines the type of algorithm to be used. Static gestures are usually recognised using template matching. Template matching usually involves measuring the similarity or dissimilarity between feature representations of hand shapes. The Finger-Earth Movers Distance (FEMD) has been used as a dissimilarity measure in [9].

Hidden Markov's Model (HMM)

HMMs [1] [10] are useful tools for recognising dynamic gestures. As stated in [1], an HMM is a double stochastic process governed by: 1) an underlying Markov chain with a finite number of states and 2) a set of random functions, each associated with one state. A HMMs models spatio-temporal information such as gestures efficiently. HMM is mostly used in the recognition part of the gesture recognition process. An HMM is made of states, which generate observation symbols according to state transition probabilities and output probabilities as described in [1] and [10]. In [11], they have used a method where an adaptive mean shift algorithm is applied to the depth histogram of the gesturing hand in order to track the trajectory of the gesturing hand and subsequently passes this information to a HMM in order to recognise the dynamic hand gesture. The disadvantage of using HMMs in recognising gestures is that HMMs usually require the definition of the observation sequence as well as transition probabilities. Also training the HMM for the set of gestures to be recognised is not always easy.

Dynamic Time Warping (DTW)

DTW is another popular method that has been used in gesture recognition systems [6].

The DTW algorithm is briefly described in [6] as follows:

Let $M = (M_1, \ldots, M_m)$ be a model sequence in which each M_i is a feature vector and let $Q = (Q_1, \ldots, Q_n)$ be a query sequence in which each Q_j is another feature vector.

A warping path W defines an alignment between M and Q. Formally, $W = w_1, \ldots, w_T$, where $\max(m,n) \leq T \leq m + n - 1$. Each $w_t = (i,j)$ specifies that feature vector M_i of the model is matched with feature vector Q_j . The warping path is typically subject to several constraints:

- **Boundary conditions:** $w_1 = (1,1)$ and $w_T = (m,n)$. This requires the warping path to start by matching the first frame of the model with the first frame of the query, and end by matching the last frame of the model with the last frame of the query.
- **Temporal continuity:** Given $w_t = (a,b)$ then $w_{t-1} = (a',b')$, where $a - a' \leq 1$ and $b - b' \leq 1$. This restricts the allowable steps in the warping path to adjacent cells along the two temporal dimensions.
- **Temporal monotonicity:** Given $w_t = (a,b)$ then $w_{t-1} = (a',b')$ where $a - a' \geq 0$ and $b - b' \geq 0$. This forces the warping path sequence to increase monotonically in the two temporal dimensions.

There are exponentially many warping paths that satisfy the above conditions. However we are only interested in the path that minimizes the warping cost:

$$DTW(Q,C) = min\left\{\sqrt[2]{\sum_{t=1}^{T} w_k}\right\}$$

The min is calculated for all $w_1, \ldots, w_T$.

This path can be found using dynamic programming to evaluate the following recurrence, which defines the cumulative distance $\gamma(i,j)$ as the distance $d(i,j)$ found in the current cell and the minimum of the cumulative distances of the adjacent elements:

$$\gamma(i,j) = d(q_i, c_j) + min\{\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)\}$$

The Euclidean distance between two sequences can be seen as a special case of DTW where the kth element of W is constrained such that $w_k = (i,j)_k$, $i = j = k$. Note that it is only defined in the special case where the two sequences have the same length. The time and space complexity of DTW is $O(nm)$.

CHAPTER 3

STATE-OF-THE-ART

As mentioned earlier, robust hand gesture recognition is still a problem especially when the fingers are involved. In [12], a method has been introduced for tracking the finger tips and centre of palms using Microsoft Kinect. The method makes use of the depth sensing capability of the Kinect device. It assumes that the finger tips are the closest parts of the hand facing the Kinect. It uses the distance transform algorithm to detect the centre of the palm. This method is developed as part of a research work to build gesture controlled robots. The disadvantage with this method is that it tracks only fingertips and centre of palms. It does not recognise the number of fingers and the trajectory of the gesturing hand.

Another method involving finger tracking and hand gesture recognition has been proposed by [4]. This method uses the Kinect sensor to detect the gesturing hand. It then uses Median filtering technique to remove noise from the detected hand during the segmentation phase. After obtaining the hand segment, the contour of the segmented hand is calculated. Equidistant point are taken on this contour. From this contour point, a centroid distance function is calculated. A Fourier descriptor is calculated from this function. Finally a feature vector is obtained based on this Fourier descriptor. The system is trained by computing the feature vector of each gesture in the set of gestures to be recognised. Gesture recognition is done through template matching. This technique successfully differentiated the number of fingers in a hand in a gestured hand. However, it requires the fingers to be widely opened and straight, it requires the gesturing hand to be parallel with the camera plane and finally it only models static gestures.

Yang *et.al.* [11] have proposed a hand tracking algorithm using depth image by calculating hand weighted probabilities. The method for detecting the hand gesture uses an adaptive mean shift algorithm. The trajectory of the hand is tracked using a 3D feature vector and the gesture is recognised by a HMM. This method was used to implement a gesture controlled media player. However, this method only tracks the hand trajectory and does not take into consideration the configuration of the fingers.

In [6], a method for recognising dynamic hand gestures has been developed. However, this does not take into account the configuration of the fingers. In [6], DTW is used to recognise the dynamic gestures.

Ren *et.al.* [9] [13] [14] have proposed a part-base hand gesture recognition system using the Microsoft Kinect. They have employed a technique called Finger Earth Mover's Distance (FEMD) to measure the dissimilarity between two hand shapes. This method is part based, meaning using fingers (global features) to differentiate the hand shapes instead of local features such as contours used by classic shape recognition methods. Our hand gesture recognition systems is based on some of the work done by Ren et al. [9]. For better understanding, the FEMD framework proposed by Ren et al. is briefly described below.

Our system is related to this framework at the level of image acquisition, segmentation and feature extraction. Our framework defers from the FEMD framework at the level of feature representation and gesture recognition. We were motivated by the performance of FEMD and DTW.

The FEMD framework is shown below

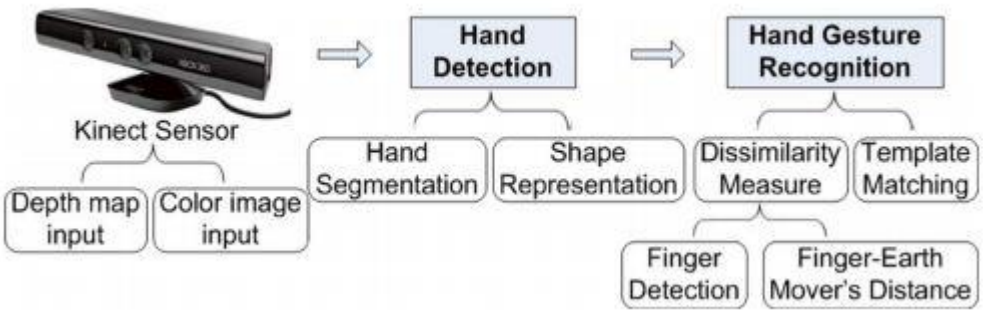

Figure 2: FEMD framework, Ren.et.al [9]

Image acquisition and segmentation in the FEMD framework is the same as described for our systems. This framework also extracts the contour points and represents them as a time-series curve. The difference comes in the representation of this time-series information as a feature vector or signature for the gesture recognition step.

This framework represents the input hand shape by global features. That is each finger in the hand shape represents a cluster in the signature. Ren *et.al.* [9] defines a hand signature R as follows

$$R = \{(r_1, w_{r1}), \dots, (r_m, w_{rm})\}$$

Where r_i is a cluster (finger) representative and w_{ri} is the weight of the cluster.

r_i is defined as the angle interval between the end points of each finger segment *i.e.*

$$r_i = \left[r_{ia,} r_{ib}\right]$$

The weight of a cluster, w_{ri} is the normalized area within the finger segment.

This is shown below.

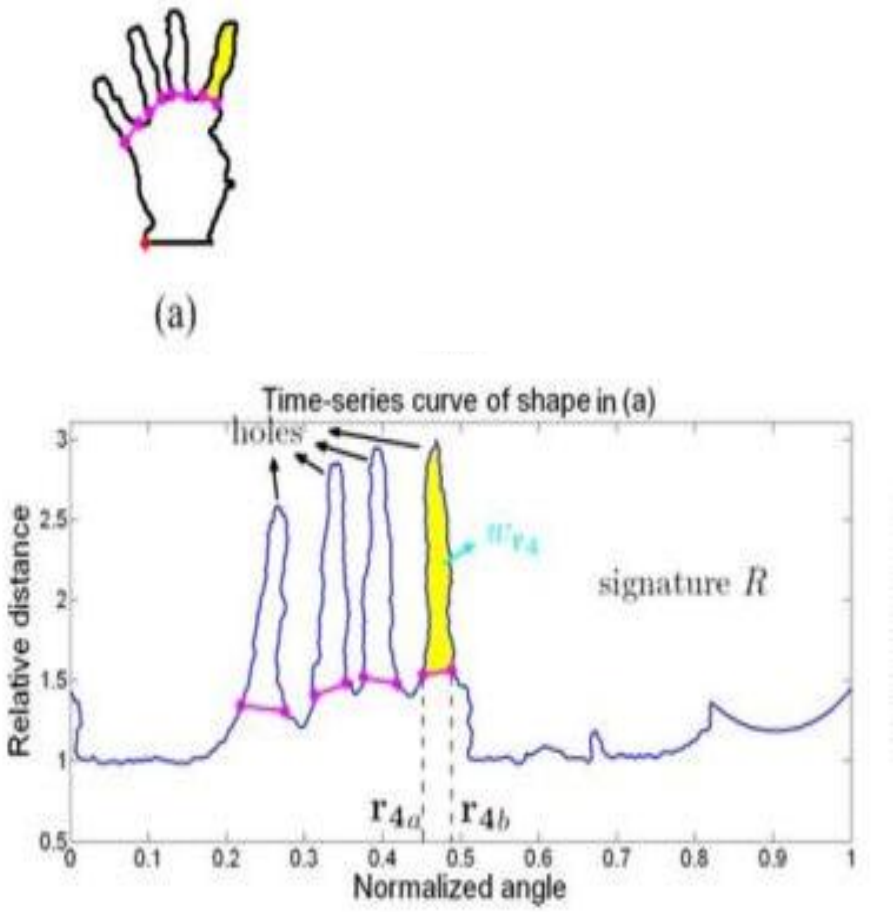

Figure 3: signature definition in FEMD, (source, [9])

This framework uses two methods to identify the finger segments from the time series representation. The first method is Thresholding decomposition where the height information in the time-series curve is used to segment the fingers. The second method is the Near-Convex decomposition. More about these methods can be found in [9] [13] [14].

After representing two hand shapes as signatures as described above, their FEMD distance can be then calculated. FEMD stems from the Earth Mover's Distance (EMD) defined by Rubner *et al* [15].

For two signature *R* and *T*, their FEMD distance is given by

$$\text{FEMD}(R,T) = \beta E_{move} + (1 - \beta)E_{empty}$$

$$= \frac{\beta \sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}f_{ij} + (1-\beta)\left|\sum_{i=1}^{m} w_{ri} - \sum_{j=1}^{n} w_{tj}\right|}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}}$$

Where $D = [d_{ij}]$ is the ground distance matrix for *R* and *T* such that d_{ij} is the ground distance from r_i to t_j.

d_{ij} is defined as the minimum moving distance for interval $[r_{ia}, r_{ib}]$ to totally overlap $[t_{ja}, t_{jb}]$, i.e.:

$$d_{ij} = \begin{cases} 0, & r_i \; totally \; overlap \; with \; t_j \\ \min\left(|r_{ia} - t_{ja}|, |r_{ib} - t_{jb}|\right), & otherwise. \end{cases}$$

f_{ij} is the flow from r_i to t_j and constitutes the flow matrix *F*.

Consult [9] for the full definition of the flow matrix, *F*.

Finally, gesture recognition in FEMD is also achieved through template matching. However, as mention earlier, FEMD is a dissimilarity measure. That is the input hand is recognized as the class with which it has the minimum dissimilarity distance

$$c = \arg\min \text{FEMD}\,(H, T_c),$$

Where *H* is the input hand signature and *T* is a template of class c.

Some of the disadvantages of the FEMD framework include:

1. Setting the finger threshold: it is difficult to set a threshold that would correctly identify the finger clusters in all the cases.
2. It has a high confusion rate for some gestures as will be seen in a later chapter

CHAPTER 4

THE PROBLEM

Our main objective is to design and evaluate an efficient and accurate hand gesture recognition system. In order to understand the problem we are trying to solve, it is important to categories it according to the individual steps involved in hand gesture recognition. At this point, it is worth noting that our designed systems deals only with static hand gestures.

First and foremost is the problem of acquiring an image and segmenting the hand shape from the image of the gesturer. Vision based techniques are suitable for image acquisition of the hand gesture [1] [6]. However, segmenting the region of interest, i.e. the hand region from the 2D image is usually inefficient and results in a distorted hand shape. Skin detection techniques [7], based on skin colour have been used to segment the hand from the rest of the image. However, this method is inefficient especially when the hand overlaps with the face in which case there will be ambiguity between the colour of the hand and face. Many hand detection approaches using RGB cameras have been proposed in the literature [1] [6]. Despite these efforts, proper segmentation of the hand region is still a problem. Using vision based approaches, the segmented hand region is usually sensitive to cluttered background, lighting conditions and the resolution of the camera. The contour of the segmented hand is usually distorted in most of the cases.

The next problem we tried to tackle in our systems is the feature extraction and representation from the segmented hand shape. Ideally, a hand gesture recognition system or any other classification system in general should be invariant to certain parameters or changes. Particularly, a hand gesture recognition system should be rotation, translation and scale invariant. To maintain these invariabilities requires careful selection of features to represent the hand shape. Approaches such as skeletal representation used in skeleton matching methods [f] usually are sensitive to contour distortions. Histograms have also been used for representing the hand shape [f]. Correspondence-based approaches [f] and shaped context approaches also suffer from some of these constraints.

After extracting and representing the hand shape comes the problem of recognising the hand gestures. Many algorithms have been used for recognising hand gestures [1] [7]. Each algorithm having its advantages and disadvantages. The algorithm to choose also depends on the feature representation of your hand shape. As stated in the previous chapter, recognising static gestures usually involves a similarity or dissimilarity measure through template matching. Choosing the correct algorithm to use usually involve a lot of trade-offs. The usually affect the efficiency of the system.

Finally, the problem of integrating the different modules to form a complete hand gesture recognition system. The modules may be efficient separately but when combined together yield a system with low efficiency than predicted. For example, a good feature representation may not be suitable as input to an efficient algorithm.

In the next chapter, we describe how we tackled some of these problems in our designed system.

CHAPTER 5

THE PROPOSED APPROACH

In this chapter, we discuss our proposed system in great detail relative to the problem presented in chapter 3. Our approach of description is based on the 3 generic steps involved in hand gesture recognition. That is, image acquisition & hand segmentation, feature extraction & representation, and gesture recognition.

A graphical representation of our system given below.

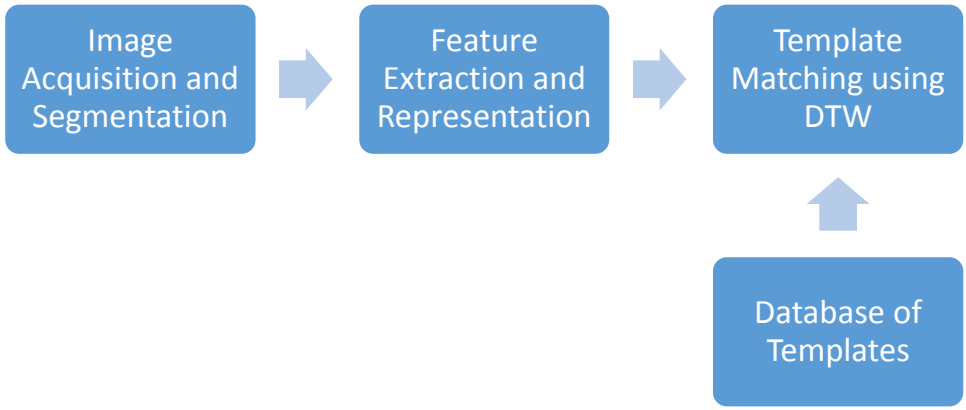

Figure 4: Proposed Gesture Recognition System

5.1. IMAGE ACQUISITION AND SEGMENTATION OF HAND SHAPE

We use the Kinect sensor as the input device to our system. Using the Kinect sensor, we capture the RGB image and the depth image of the gesturer. In particular, we used the Microsoft Kinect for Windows SDK version 1.8 during the design of the system. The RGB and depth images are all 640 x 480 pixels. The depth values are stored in millimetres. The Kinect sensor has proven to be robust in capturing images for used in hand gesturing recognition [2] [5] [9]. Depending on the version of the Microsoft Kinect used, some pre-processing is required in order to calibrate the RGB and Depth Images. After calibration, the input is ready for hand segmentation.

Segmentation of hand shape involves isolating the region of interest from the input image. That is the gestured hand shape. Other background information are not required. In most systems such as in [9], hand segmentation is usually represented under the feature extraction & representation module. However, how the hand is segmented depends on the device or technique used to acquire the image. This justifies why we join them as one module in our system.

Before segmenting the hand shape, some pre-processing is performed on the RGB image. This involves converting the RGB image into grey-scale. We then used the depth information in the depth image to segment the hand shape from the RGB image. First, we locate the smallest depth value from the depth image. This corresponds to the closest point of the hand to the camera plane. We call this value minimum-distance. Next, a threshold value is added to the minimum-distance to give the segmentation threshold. This segmentation threshold is then used to segment the hand region from the rest of the image.

It is important to note that the hand should be the closest object to the camera for correct segmentation of the hand region.

Ren *et.al.* [9] have used a similar technique to segment the hand shape. However, for proper segmentation of the region of interest, they require the gesturer to wear a black belt on the wrist of the gesturing arm.

To allow flexibility in gesturing, we follow the approach of wearing a black belt on the wrist of the gesturing hand as described by Ren *et.al*. The figure below shows the steps through image acquisition and segmentation of hand shape.

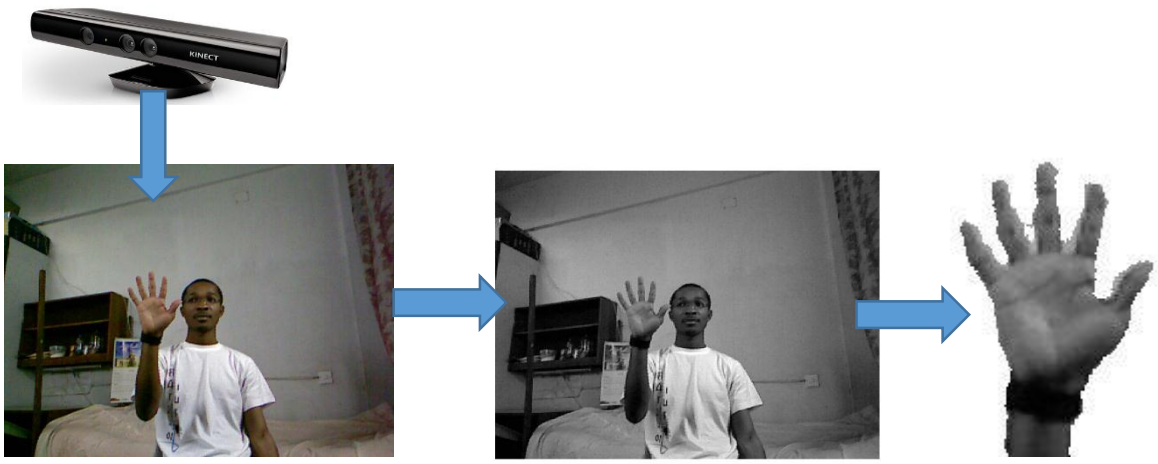

Figure 5:Image Acquisition and Segmentation of Hand Shape

5.2. FEATURE EXTRACTION AND FEATURE REPRESENTATION

After segmenting the region of interest from the input image, we are now ready for feature extraction and representation. As stated earlier, the choice of the features should be such that the system will present a high degree of invariance to scaling, translation and rotation. The representation of the features depends on the algorithm to be used to recognise the gestures. From the illustrated figure of our system above, DTW is used for gesture recognition. As described earlier, DTW is suitable for finding similarities between time-series. We choose the contour of the hand shape as our features and represent these features as a time-series. Contour information has been used successfully by [4] and [9]. In [9], the information is represented as a time-series. Whereas in [4], representation is based on a discrete Fourier transform of the contour information.

The contour points of the hand shape are suitable for use as feature representation because no matter the orientation and location of the hand shape, the contour of the hand remains the same when represented as a time-series.

Before extracting the contour points or vertices, we need to identify the centre point of the hand shape as well as the black belt's pixels.

We applied a distance transform function on the bitwise image of the segmented hand shape in order to obtain the centre point. The index of maximum distance returned by this function is the centre of the hand shape. Distance transform has been used by [9] and [12] this to locate the centre point.

The RANSAC algorithm was applied on the identified black belt pixels to fit a line that will enable use locate our starting point.

In order to obtain the contour points, we first convert the hand shape from grey-scale to binary. To get rid of any distortion that may result from segmentation, we apply a region filling method to the binary image. Now, the image is ready for the extraction of the boundary points. We apply a simple boundary extraction technique which consist of eroding the images with a structuring element and subtracting the output from the original binary image.

That is

$$\beta(A) = A - (A \Theta B)$$

Where A is the input image, $(A \Theta B)$ is the erosion of A by structuring element B and $\beta(A)$ is the extracted boundary of A.

Using the same time-series representation as used by Ren *et.al.* [9], we define the vertical axis of our time-series to be the Euclidean distance between each contour vertex and the centre point of the hand shape. The horizontal axis represents the angle between each contour vertex and the initial point relative to the centre point. The initial point is determine based on the line drawn through the black belt's pixels using the RANSAC algorithm.

The final task in this step involves defining a feature vector for the next step in our framework. The time-series defined above is then converted to a 2D feature vector$f$.

f is defined as follows:

$$f = \{(d_i, a_i), \dots, (d_n, a_n)\}$$

Where d_i and a_i are respectively the Euclidean distance and angle of vertex i and n is the number of vertices in the boundary of the extracted hand shape.

After this step, the feature vector representation f is ready to be passed as input to DTW in the next module.

Below is a sequence of images pictorially describing feature extraction and representation.

Figure 6: Extracted Hand shaped

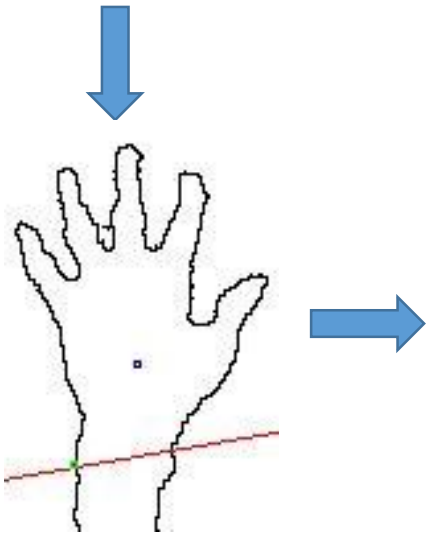

Figure 7: Hand contour, with green point showing the initial point and blue point showing the centre point

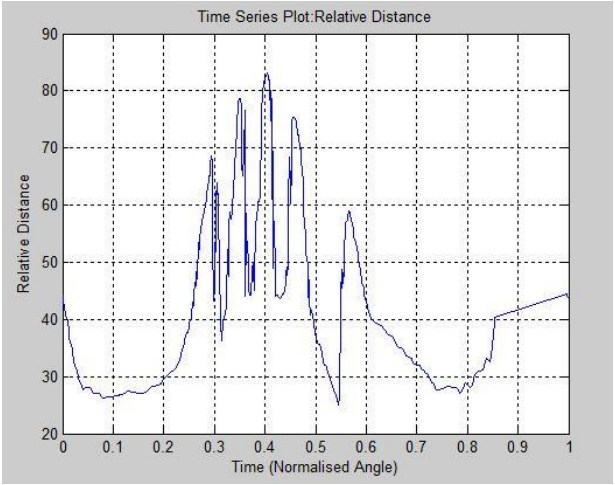

Figure 8: Time-series curve of contour points

5.3. GESTURE RECOGNITION (TEMPLATE MATCHING USING DTW)

Our gesture recognition step involves a similarity measure through template matching using DTW. DTW has been used in [6] for recognising dynamic hand gestures. In this step, DTW is applied between the feature vector representation of the incoming unknown hand shape and a set of predefined templates stored in our database. The unknown hand shape is classified as belonging to the class of the template with which it has the minimum DTW value. That is the unknown gesture is classified under class c, such that

$$c = \arg\min DTW(U, T_c)$$

Where U is an unknown input gesture representation, T is a template from the database of template and c is the class label of T.

5.4. DATASET AND DATABASE OF TEMPLATES

We created a new dataset which contains 200 samples. We had two people perform each gesture type 10 times. We combined our dataset with the dataset provided by Ren *et.al.* [9] which contains 1000 samples. So in total we used 1200 samples to evaluate our system. This samples are all taken in complex scenes with cluttered background. This dataset defines 10 different classes of hand gestures as shown below.

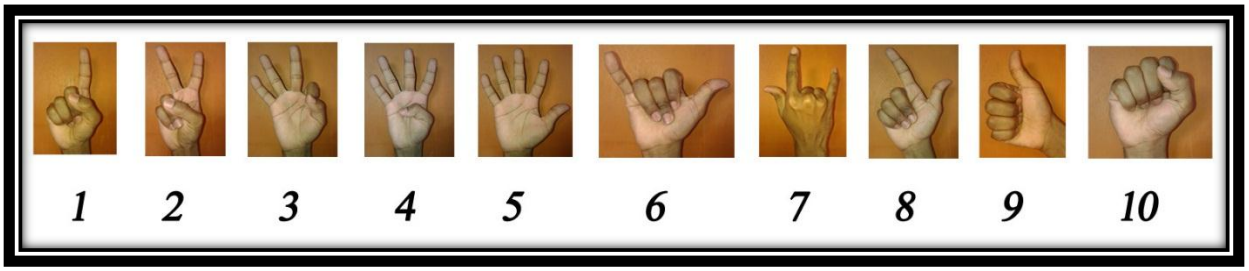

Figure 9: Classes of gestures defined in the dataset

From this dataset, we created a database of templates. That is we computed the feature vector representation of each of these gestures and stored them in the database. This was then used for template matching against the unknown gestures.

The experimental results and analysis of our system are given in the next chapter.

CHAPTER 6

EXPERIMENT AND RESULT ANALYSIS

6.1 EXPERIMENTAL SETUP

All the experiments were conducted on an Intel® Core™ i3-2120 CPU @ 3.30 GHz with 4GB of RAM. This was running a Window 7 32-bit operating system.

Our images were captured using Microsoft Kinect for Windows SDK version 1.8.

Our system was implemented on MATLAB R2013a.

6.2 EXPERIMENTAL RESULTS

In order to determine the accuracy and efficiency of our proposed system, we ran a series of tests on our system. As mentioned in the previous chapter, we generated a new dataset and combined with the dataset provided by Ren et.al. [9]. There are ten class labels representing each gesture. There are 120 images per class.

We analysed our results based on the following criteria.

1. Robustness to cluttered background and lighting sensitivity
2. Invariance to scale, translation and rotation
3. Accuracy and running time

Robustness to cluttered background and lighting sensitivity

Our system proved to be robust to cluttered background and occlusions from the scene. The images from our dataset were taken in challenging scenes. This robustness is justified by the fact that our segmentation is based on the depth information of the scene. So provided the hand is always the closest part of the body to the camera the system will always perform well under any type of background. Also, the Microsoft Kinect requires no special lighting conditions. Therefore the system is also insensitive to lighting condition. Normal bright light is sufficient.

Invariance to scale, translation and rotation

The gestures in a particular class of our dataset were performed by different people in different orientations. This system still proved to be invariant to these factors. This is because our feature representation is based on the contour of the hand shape. The contour information capture the topology of the hand no matter the orientation.

Accuracy and Running time

Finally, we measured the accuracy and running time of our system. A series of test cases were run for each class. Specifically, we tried to classify 100 unknown images from each class of gestures defined above. The classification results per class are shown in the table below, where the 'True' column denotes the number of correctly classified samples and the 'False' column the number of incorrectly classified samples.

Class Label	True	False
1	97	3
2	92	8
3	95	5
4	89	11
5	88	12
6	96	4
7	95	5
8	98	2
9	96	4
10	100	0

Table 1: Results per class label

From the above results, we obtained a mean accuracy of **94.6%** and a mean running time of **0.5179s.** The mean accuracy was obtained by averaging the number of correctly classified images from the table above. The mean running time was also obtained from by averaging the running time of all the test cases.

In order to further compare our results with the FEMD method proposed by Ren *et.al.* [9][13] [14] . We developed the following confusion matrix.

	1	2	3	4	5	6	7	8	9	10
1	97		1						2	
2	5	92		1				2		
3	2		95			2			1	
4		2	4	89			5			
5			4	6	88		2			
6	2					96			2	
7			2			1	95	2		
8	1							98	1	
9						1		1	96	2
10										100

Figure 10: Confusion Matrix of our system (Left)

	1	2	3	4	5	6	7	8	9	10
1	95	1						3	1	
2	3	86	4	2			1	4		
3		2	94	2			2			
4			4	87	6		3			
5				7	89	3	1			
6	1	2				95			2	
7			1			1	96			
8	6	2						92		
9	1					1			98	
10										100

Figure 11: Confusion Matrix of FEMD system [9] (Right)

As we can see from the confusion matrices above, there is a high confusion rate in class 2 of the FEMD system. This class is highly confused with class 1, class 3 and class 8. However, in our system, the confusion rate in class 2 is low and is highly confused only with class 1.

The table below compares our mean accuracy and mean running time to that of FEMD

	Mean Accuracy	Mean Running time
Our System	94.6 %	0.5004s
Thresholding decomposition + FEMD	93.2%	0.5179s

Table 2: Mean Accuracy and Mean Running time of our system and that of FEMD [9]

The above results demonstrate the accuracy and efficiency of our system.

CHAPTER 7

APPLICATIONS OF GESTURE INTERACTIVE SYSTEMS

Gesture interactive systems find applications in a wide variety of fields.

They are mostly used in the gaming industry, in the medical field, in education, and many other field. Below are some of the application of gesture based interactive systems as given in [1]

- Developing aids for the hearing impaired;
- Enabling very young children to interact with computers;
- Designing techniques for forensic identification;
- Recognizing sign language;
- Medically monitoring patients' emotional states or stress Levels;
- Lie detection;
- Navigating and/or manipulating in virtual environments;
- Communicating in video conferencing;
- Distance learning/tele-teaching assistance;
- Monitoring automobile drivers' alertness/drowsiness levels, etc.

CHAPTER 8

CONCLUSION AND FUTURE WORKS

8.1. CONCLUSION

In conclusion, we have designed and developed an accurate and efficient gesture recognition system that performs well in uncontrolled environments. After evaluating our system with a gesture set of 10 classes, we obtain a mean accuracy of **94.6%** and a mean running time of **0.5179s**. The system has proven to be robust to cluttered background as well as insensitive to lighting conditions. The system is also invariant to scaling, translation and rotation. Our contribution is the definition of a 2D feature vector based on the time-series representation of the contour of the hand shape. Our system also demonstrated the power of DTW in matching time-series representation. We also achieved our initial objective which was to find out whether classification accuracy could be improved by using DTW with the time-series representation of the hand shape rather than using the FEMD metric.

8.2. FUTURE WORKS

As our future work, we plan in studying the feasibility of upgrading our system to recognise dynamic hand gestures by using HMM. As stated earlier, HMMs are useful tools for recognising dynamic hand gestures. It is a double stochastic process that models spatio-temporal processes such as gestures efficiently. That is, processes that change with both time and space. Dynamic hand gestures recognition is still lacking behind as compared to static hand gesture recognition. Most dynamic hand gesture recognition systems do not take into consideration the configuration of the fingers, i.e. they only differentiate the path taken by the hand as a whole. For example, waving with five fingers opened and waving with 2 fingers opened will be recognised as the same gesture. However, in order to expand the grammar of gesture based interfaces, there is a need to differentiate between such types of gestures. Thus with efficient systems like ours and the FEMD, we plan on studying how to incorporate dynamism in the gesture recognition system.

Optimising our system for better accuracy and performance is also part of our future work. Finally, we intend to develop some real-life applications to demonstrate the efficiency and accuracy of our system.

References

[1] S. Mitra And T. Ach, "Gesture Recognition: A Survey," *Ieee Transactions On Systems, Man, And Cybernetics,* Vol. 37, No. Part C: Applications And Reviews, Pp. 311-324, 2007.

[2] K. K. Biswas And S. K. Basu , "Gesture Recognition Using Microsoft Kinect®," In *Automation, Robotics And Applications (Icara), 2011 5th International Conference*, Wellington, 2011.

[3] M. Avancini, "Using Kinect To Emulate An Interactive," Facoltà Di Scienze Matematiche, Fisiche E Naturali, Università Degli Studi Di Trento, 2012.

[4] A. Kulshreshth, C. Zorn And J. J. Laviola Jr, "Poster: Real-Time Markerless Kinect Based Finger Tracking And Hand Gesture Recognition For Hci," In *3d User Interfaces (3dui), 2013 Ieee Symposium On* , Orlando, Fl , 2013 .

[5] Y. Li, "Hand Gesture Recognition Using Kinect," Department Of Computer Engineering And Computer Sciences ,University Of Louisville, Louisville, 2012.

[6] P. Doliotis, A. Stefan, C. Mcmurrough, D. Eckhard And V. Athitsos, "Comparing Gesture Recognition Accuracy Using Color And Depth Information," In *Proceedings Of The 4th International Conference On Pervasive Technologies Related To Assistive Environments* , New York, 2011.

[7] A. Chaudhary, J. L. Raheja, K. Das And S. Raheja, "Intelligent Approaches To Interact With Machines Using Hand Gesture Recognition In Natural Way: A Survey," *International Journal Of Computer Science & Engineering Survey (Ijcses),* Vol. 2, Pp. 122-123, 2011.

[8] K. Barczewska And A. Drozd, "Comparison Of Methods For Hand Gesture Recognition Based On Dynamic Time Warping Algorithm," In *Proceedings Of The 2013 Federated Conference On Computer Science And Information Systems*, Krako, 2013.

[9] Z. Ren, J. Yuan, J. Meng And Z. Zhang, "Robust Part-Based Hand Gesture Recognition Using Kinect Sensor," *Ieee Transactions On Multimedia,* Vol. 15, Pp. 1110-1120, 2013.

[10] A. D. Wilson And A. F. Bobick, "Hidden Markov Models For Modeling And Recognizing Gesture Under Variation," In *Hidden Markov Models*, River Edge,, World Scientific Publishing Co., Inc, 2002, Pp. 123 - 160.

[11] C. Yang, Y. Jang, J. Beh, D. Han And H. Ko, "Gesture Recognition Using Depth-Based Hand Tracking For Contactless Controller Application," In *Ieee International Conference On Consumer Electronics (Icce),* 2012.

[12] J. L. Raheja, A. Chaudhary And S. K, "Tracking Of Fingertips And Centre Of Palm Using Kinect," In *In Proceedings Of The 3rd Ieee International Conference On Computational Intelligence, Modelling And Simulation*, Malaysia, 2011.

[13] Z. Ren, J. Meng And J. Yuan, "Depth Camera Based Hand Gesture Recognition And Its Applications In Human-Computer-Interaction," In *Information, Communications And Signal Processing (Icics)* , Singapore, 2011.

[14] Z. Ren, J. Yuan And Z. Zhang, "Robust Hand Gesture Recognition Based On Finger-Earth Mover's Distance With A Commodity Depth Camera," In *Mm '11 Proceedings Of The 19th Acm International Conference On Multimedia* , New York, 2011.

[15] S. Marcel, O. Bernier, J. Viallet And D. Collobert, "Hand Gesture Recognition Using Input–Output Hidden Markov Models," In *4th Ieee International Conference On Automatic Face And Gesture Recognition (Fg 2000)*, Grenoble, 2000.

[16] Y. Rubner, C. Tomasi And L. . J. Guibas, "The Earth Mover's Distance As A Metric For Image Retrieval," *International Journal Of Computer Vision*, Vol. 40, No. 2, Pp. 99 - 121 , 2000 .

Appendix

Source Code

Function: Feature Extractor

```
1. %This function is the feature Extractor
2. %It takes as input a depth image and the correspoinding colour image
3. %The output is a 2D feature vector representing the time-series
4. %inputs
5. %   depthImgPath : path to depth image
6. %   colorImgPath : path to colour image
7. %Output
8. %   featureVector : 2D feature vector
9. function [ featureVector] = featureExtractorDtw( depthImgPath,
   colorImgPath)


10.      delimiter = ',';
11.      filename = depthImgPath; %this is the depth image file in txt

12.      depthimg = importdata(filename, delimiter); %convert depth file
   to image matrix

13.      img = imread(colorImgPath); %this is the corresponding colour
   image


14.      rec = [0 0 590 470];

15.      imggray = rgb2gray(img); %convert colour image to grayscale

16.      %figure,imshow(imggray);

17.      %%below we calibrate the depth and colour images so that pixel
   positions
18.      %%correspond in both images.note Kinect does not calibrate the
   images. it
19.      %%is a random process like trial and error
20.      %if your images are calibrated, you can delete the calibration
   code and
21.      %start from SEGMENTATION AND FEATURE EXTRACTION below


22.      %% CALIBRATION STARTS
23.      depthimg = imcrop(depthimg, rec);
```

```
24.      %depthimg = imresize(depthimg, [480 640]);
25.      %imggray = imcrop(imggray, rec);

26.      [i j] = size(depthimg);
27.      imggray = imresize(imggray, [i j]);

28.      imggray = imresize(imggray, [520 710]);
29.      %figure, imshow(imggray);
30.      imggray = imcrop(imggray, [66 28 i j]);


31.      % CALIBRATION ENDS
32.      %end calibration. now depth and colour pixel now correspond in
   a 1 to 1
33.      %mapping


34.      %% SEGMENTATION AND FEATURE EXTRACTION

35.      %%we locate the minimum depth of the image below
36.      min = 10000; %holds the value of the min depth to the camera
37.      for r = 1:i
38.      for c = 1:j
39.      if ( depthimg(r,c) ~= 0)
40.      if(depthimg(r,c) < min)
41.      min = depthimg(r,c);
42.      end

43.      end

44.      end
45.      end

46.      threshold = min + 70; %segmentation threshold. depth of the
   hand region
47.      %from camera for segmentation


48.      xmin = 100000; %min x coordinate  of hand contour
49.      xmax = -1000;  %maz x coordinate  of hand contour
50.      ymin = 100000;  %min y coordinate  of hand contour
51.      ymax = -1000;  %max y coordinate  of hand contour

52.      %this is to crop the hand region from the whole image


53.      %% we segment the hand region below. all pixels other than the
   hand pixels
54.      %%are set to white

55.      for r = 1:i
56.      for c = 1:j
```

```
57.      if(depthimg(r,c) > threshold || depthimg(r,c) == 0 )

58.      row = r;
59.      col = c;
60.      if(col < 1)
61.      col = c;
62.      end
63.      if( row > i)
64.      row = r;
65.      end
66.      imggray(row,col) = 255;
67.      end


68.      end
69.      end


70.      %figure, imshow(imggray); %imggray now represents our segmented
   hand image




71.      %% cropping the hand region
72.      %%obtaining the min x,y and max x,y vals of the hand region for
   cropping

73.      for y = 1:j
74.      for x = 1:i

75.      if( (depthimg(x,y) < threshold) && ((imggray(x,y) > 0) &&
   imggray(x,y)~= 255) )

76.      if( x < xmin)
77.      xmin = x;
78.      end
79.      if(x > xmax)
80.      xmax = x;
81.      end

82.      if( y < ymin)
83.      ymin = y;
84.      end
85.      if(y > ymax)
86.      ymax = y;
87.      end
88.      end
89.      end
90.      end


91.      %% cropping using imcrop
```

```
92.      height = xmax - xmin;
93.      width = ymax - ymin;

94.      rect = [ymin-5 xmin-5 width+10 height+10];

95.      cropimg = imcrop(imggray, rect);


96.      cropimg2 = cropimg;
97.      %figure, imshow(cropimg2);

98.      %cropimg and cropimg2 now represent our segmented hand region
   which is
99.      %approximately 100x100 pixels

100.     %% Identifying the black belt pixels below

101.     [row col] = size(cropimg);
102.     index = 1;

103.     for i = 1:row
104.     for j = 1:col

105.     if(cropimg(i,j) < 30)
106.     cropimg(i,j) = 255;
107.     pts(1,index) = j;  %pts holds our black belts pixels
108.     pts(2,index) = i;


109.     index = index + 1;
110.     end
111.     end
112.     end

113.     %figure, imshow(cropimg);
114.     % now, pts contains our black belt's pixels locations.


115.     %% use ransac to draw a straight line through our black belt's
   pixels

116.     index = index -1;
117.     st = pts(2,1);
118.     en = pts(2,index);

119.     iterNum = 300;
120.     thDist = 2;
121.     thInlrRatio = .1;
122.     [t,r] = ransac(pts,iterNum,thDist,thInlrRatio);
123.     k1 = -tan(t);
124.     b1 = r/cos(t);
125.     len = 1:col;
```

```
126.    ypts = uint8(k1*len+b1); %% y points generated from ransac

127.    %% below we extract the boundary of our handshape

128.    cropimg2 = ~(im2bw(cropimg2, 0.9)); %convert hand image to
   binary image
129.    dstimg = cropimg2;  %dstimg copy of image to be used for
   distance transform
130.    % to obtain the centre point
131.    cropimg2 = imfill(cropimg2, 'holes');
132.    SE = strel('square',3);

133.    cropimg2 = cropimg2 - (imerode(cropimg2,SE)); %boundary
   extraction



134.    cropimg2 = ~cropimg2; %invert the binary pixels

135.    %now cropimg2 holds only the boundary or contour of the
   handshape


136.    %% perform distance transform on dstimg to find centre point

137.    dstimg = bwdist(~dstimg); %distance transform algorithm

138.    [mpv, indp] = max(dstimg); %mpv = row vector containing max
   distance for each column
139.    %indp =  index of these values

140.    %md = max(mpv);  %md = maximum distance after distance
   transform

141.    maxval = max(dstimg(:));
142.    [md,rowind] = max(mpv); %rowind = row index value of centre
   point

143.    colind = indp(rowind); %colind = column index value of centre
   point

144.    centrept = [rowind colind]; %%this is our centre point

145.    %% below we obtain intersection point of ransac line and hand
   contour

146.    plotimg = cropimg2; %temporary  image to be used as background

147.    plotimg(:,:) = 1; %% white background of same size as our hand
   image to be

148.    % used to draw ransac line for obtaining intersection point
```

```
149.    %figure, imshow(dstimg, []); %display our distance tranformed
   hand shaped


150.    %% we look for intersection point of line and image below
151.    %set(gcf,'visible','off');

152.    figure('visible','off'), imshow(plotimg), hold on;
153.    %set(gcf,'visible','off');
154.    plot(len,k1*len+b1,'r'), hold on; %ransac line plot

155.    f = getframe;              %Capture screen shot
156.    [im,map] = frame2im(f);    %Return associated image data which
   contains our
157.    %ransac line
158.    if isempty(map)            %Truecolor system
159.    rgb = im;
160.    else                       %Indexed system
161.    rgb = ind2rgb(im,map);   %Convert image data
162.    end

163.    im = rgb2gray(im);
164.    im = im2bw(im);
165.    im = imresize(im,[row col]); %im now is an  image of our ransac
   line through
166.    %the black colour pixel.


167.    %figure, imshow(im);

168.    ab = cropimg2+im; % now we obtain the intersection point of the
   handshape
169.    % and the ransac line

170.    %ab = im2bw(ab);  % ab is an image with black pixels
   represented the
171.    %intersection point
172.    set(gcf,'visible','off');
173.    %figure, imshow(ab);

174.    %% now we extract the starting  points from ab

175.    linedir = int16(ypts(col) - ypts(1)); %denotes orientation of
   line so to
176.    %easily locate start point


177.    [row col] = size(ab);

178.    if(linedir > 0 )
179.    fin = 0;
```

```
180.    for i = 1:row
181.    for j = 1:col

182.    if( ab(i,j) == 0)
183.    stpt = [i j];
184.    fin = 1;
185.    break;
186.    end
187.    end
188.    if(fin == 1)
189.    break;
190.    end
191.    end

192.    elseif(linedir <= 0)

193.    fin = 0;
194.    for i = row:-1:1
195.    for j = 1:col

196.    if( ab(i,j) == 0)
197.    stpt = [i j];
198.    fin = 1;
199.    break;
200.    end
201.    end
202.    if(fin == 1)
203.    break;
204.    end
205.    end
206.    end

207.    % stpt now represents our starting point.

208.    %% locate end point for proper segmentation

209.    if(linedir > 0 )
210.    fin = 0;
211.    for i = 1:row
212.    for j = 1:col

213.    if( ab(i,j) == 0)
214.    endpt = [i j];
215.    end
216.    end
217.    end

218.    elseif(linedir <= 0)

219.    fin = 0;
220.    for i = row:-1:1
221.    for j = 1:col
```

```
222.    if( ab(i,j) == 0)
223.    endpt = [i j];

224.    end
225.    end
226.    end
227.    end

228.    %%



229.    %% now time series representation of our handshape
230.    ctpt = [centrept(2) centrept(1)];
231.    [row col] = size(ab);
232.    k = 1;
233.    if(linedir > 0 )

234.    for i = 1:endpt(1)
235.    for j = 1:col

236.    if( cropimg2(i,j) == 0)
237.    cp = [i j];     %% cp implies current position
238.    eudist = sum((ctpt-cp).^2).^0.5;    %eudist = euclidean
   distance between cp and
239.    %initial point, centrept
240.    rdist(k) = eudist; %% rdist is an array that holds the
241.    %the eudist for each contour point

242.    %now we calculate the angle between each contour point
243.    %and the initial pt, stpt relative to centrept

244.    v1 = cp - ctpt;
245.    v2 = stpt - ctpt;
246.    ang = atan2(v1(1)*v2(2)-v2(1)*v1(2),v1(1)*v2(1)+v1(2)*v2(2));
247.    Angle = mod(180/pi * ang, 360);

248.    nangle(k) = Angle/360; %nangle is an array of angles



249.    k = k+1;
250.    end
251.    end

252.    end

253.    elseif(linedir <= 0)

254.    k = 1;
255.    for j = 1:col
256.    for i = 1:stpt(1)

257.    if( cropimg2(i,j) == 0)
```

```
    cp = [i j];     %% cp implies current position
    eudist = sum((ctpt-cp).^2).^0.5;    %eudist = euclidean
  distance between cp and
    %initial point, centrept
    rdist(k) = eudist; %% rdist is an array that holds the
    %the eudist for each contour point


    %now we calculate the angle between each contour point
    %and the initial pt, stpt relative to centrept

    v1 = cp - ctpt;
    v2 = stpt - ctpt;
    ang = atan2(v1(1)*v2(2)-v2(1)*v1(2),v1(1)*v2(1)+v1(2)*v2(2));
    Angle = mod(180/pi * ang, 360);

    nangle(k) = Angle/360; %nangle is an array of angles

    k = k+1;

    end
    end

    end
    end

    %% below we draw the time series curve of the hand shape
    count1=timeseries(rdist,nangle);
    count1.Name = 'Relative Distance';
    count1.TimeInfo.Units = 'Normalised Angle';
    %figure, plot(count1), grid on

    %% smoothing version of the time series curve of the hand shape
    ys = smooth(nangle,rdist,1,'rloess');
    %plot(x,y2,x,ys)
    count1=timeseries(ys,nangle);
    %normangles = count1.time;
    %figure, plot(count1), grid on, hold on;
    featureVector = [count1.time count1.data]; %2D feature Vector
  to be used as
    %input to DTW
```