Bachelor of Science in Computer Science and Engineering

Fingertip Detection and Finger Identification for Real-time Hand Gesture Recognition using Kinect

By
Ahmed Al Marouf
and
Shaumic Shondipon
Systems and Software Lab(SSL)

Supervised by
Md. Kamrul Hasan, PhD
Assistant Professor, Department of CSE
Systems and Software Lab(SSL)

Department of Computer Science and Engineering (CSE)
Islamic University of Technology (IUT)
November, 2014

Declaration of Authenticity

This is to certify that the work presented in this thesis is the outcome of the analysis and investigation carried out by the candidate under the supervision of Dr. Md. Kamrul Hasan in the Department of Computer Science and Engineering (CSE), IUT, Gazipur. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references has been given.

Signature of the Students:

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Ahmed Al Marouf	Shaumic Shondipon
Student ID: 104441	Student ID: 104415
Academic year: 2013-2014	Academic year: 2013-2014
Date:	Date:

Signature of the Supervisor:

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Md. Kamrul Hasan, PhD
Assistant Professor
Department of Computer Science and Engineering
System and Software Lab(SSL)
Islamic University of Technology
Date:

Table of Contents

List of Figures

List of Tables

Acknowledgements

All praise is for Allah (sbw). Without the grace and blessings of the Almighty we wouldn’t be where we are. We are grateful for the strength, patience and skill He has given us, without which it would have been difficult to complete this thesis successfully.

We would like to thank our supervisor Dr. Md. Kamrul Hasan, Assistant Professor, Department of CSE, IUT for guiding us through the research, and inspiring us.

We would also like to thank Mr. Hasan Mahmud, Assistant Professor, Department of CSE, IUT for helping us in tight spots, giving us valuable suggestions and advices.

There are several individuals we would like to thank. Md. Mozaharul Mottalib, Abdullah Al Tariq were very helpful throughout the research work. Their support and contributions were essential to successful completion of this thesis.

Finally, we would like to thank IUT for providing an excellent environment for research.

Abstract

Fingertip Detection and finger identification is one of the main challenges of gesture spotting and recognition. The topic is mostly related to the Human computer Interaction(HCI) area. We have proposed a system for detection of fingertip and identification of finger by several steps like pre-processing, processing, hand segmentation, palm point identification, fingertip detection, finger identification. Pre-processing includes the thresholding ,RGB to Gray-scale conversion and color & depth calibration. Determining the minimum depth value from the Kinect camera, determining the segmentation threshold, cropping the region of interest and edge detection are the steps of processing. Depth and color segmentation and calibration is done for hand segmentation. For fingertip detection we have merged two existing idea from state of the art in a manner that it minimizes the limitations of both approach. Ellipse fitting technique is applied for palm point determination and point-to-point scanning is done for fingertip locating. For finger identification, we have proposed a new model, 4Y - model based on iterative Hill climbing Algorithm for finger identification. The main contribution in this area would be the new approaches of fingertip detection and finger identification.

Keywords
Human Computer Interaction (HCI), Fingertip Detection, Finger Identification, Palm point Determination, 4Y Model, Hill Climbing Algorithm.

Chapter 1 Introduction

1.1 Introduction:

Human gesture recognition is an interesting area of Human Computer Interaction(HCI). There are many challenging problems of gesture recognition like hand detection, specifically fingertip detection and finger identification. In Human Computer Interaction (HCI), gesture based interface gives a new direction towards the creation of a natural and user friendly environment. Recently in HCI, the detection of finger and finger types has received growing attention in applications like sign language, vision based finger guessing games and in applications related to real-time systems and virtual reality and recognizing pointing gestures in the context of human–robot interaction. [1]

Real time hand trajectory recognition is another important issue in this area. Many researchers have proposed different approaches for static images or template based approach to minimize the limitations of fingertip detection, but the static images itself is identified as a limitations as the human poses are gestures are the continuous. So real time continuous approaches should be made for fingertip detection. We have introduced a new approach in real time hand gesture recognition.

Several depth sensor cameras are available nowadays those takes the image and gives its coordinate values as well as depth value. Depth value is the distance value of each point of the image from the camera. This depth and coordinate information could be used for fingertip detection and identification.

The approaches to solve the problem of detecting fingertip are equation based where some constant or threshold value comes up for the solution or the determination of palm point is not very interesting to solve the problem. This type of approaches have their own limitations and solving step by step is very sophisticated work. Mixture models could be solution to this type of problems. The better sides of different approaches could be sum up to minimize the limitations in both side. We have also proposed an idea for fingertip detection which will incorporate two different approaches and minimize their limitations as well.

Several stages could be proposed to solve the detecting and identification problem like pre-processing, processing, hand segmentation, palm point identification, fingertip detection, finger identification and use this fingertip values for further applications. Pre-processing includes the thresholding ,RGB to Gray-scale conversion and color & depth calibration. Determining the minimum depth value from the Kinect camera, determining the segmentation threshold, cropping the region of interest and edge detection are the steps of processing. Depth and color segmentation and calibration is done for hand segmentation. For fingertip detection we have merged two existing idea from state of the art in a manner that it minimizes the limitations of both approach. Ellipse fitting technique is applied for palm point determination and point-to-point scanning is done for fingertip locating. For finger identification, we have proposed a new model, 4Y - model based on iterative Hill climbing Algorithm for finger identification. The main contribution in this area would be the new approaches of fingertip detection and finger identification.

Our main contribution is on fingertip detection and finger identification. The ellipse fitting technique introduced in paper [2], is being followed to get the palm point. For fingertip detection, the point-to-point scanning is done in order to find the edge points and these points are being processed. And , for finger identification we have proposed our own algorithm which is named as 4Y-Model based on iterative Hill climbing algorithm.

1.2 Motivation of Work:

Human computer Interaction(HCI) is a vast area to work in and the diversity of work depends on the human behavior and usability. The user experience is also an important issue to be considered by the designers. Fingertip detection and finger identification by Kinect is very interesting topic where there is many opportunity to work.

Literature review or the state of the art also helps us to get motivated. Existing applications of fingertip detection are very interesting but not without limitations. So we have decided to minimize the limitations and work in this area. The obtained information about fingertip and fingers could be used in different applications.

Different applications are present where the fingertips and identification of fingers is necessary. So getting more accurate fingertips could lead the applications to be more user friendly and interactive. Working on this area actually makes us more creative, practical and generous. So contributing for this area was the great choice for us.

1.3 Problem Statement:

"To detect the fingertips and identify the fingers solving the previous limitations and use the obtained information's in the future applications."

The quoted statement is our problem statement and the description of the statement is very straight forward.

Fingertip detection and identification of finger is our main purpose of work. The positive description of these two terms could be given as.

Human hand generally consists of five fingers. Each of the fingers have three joints except the thumb. Each of these fingers have a fingertip that we use for many day to day work as well as technological issue like touching the screen of cell phone, fingerprint identification etc. The fingertips are also a key issue for gesture recognition as the gestures are made by the trajectory of the fingertip. Many researchers have introduced different approaches for the determination of five fingertips, but none of them are limitation free. So fingertip detection is one of the main problems to be solved.

Five different fingers of human hand generally could be named as thumb, index, middle, ring and little fingers. These fingers could be used for different gesture recognition techniques and the identification of each finger is one is very important issue. For digital instruments, virtual games or virtual applications the identification of different fingers are important. So finding the names of the fingers are good problem to be solved.

Chapter 2
Related Works

2.1 State-of-the-art:

The state of the art defines the related work in this area. Gesture spotting and recognition is a huge area to contribute and hand gestures are considered as the most popular area of researchers.

The state-of-the-art for our work could be divided into two basic parts: fingertip detection and finger identification.

For fingertip detection we have studied a lot of related papers, journals, conference papers and master thesis papers also. Some selective papers are given below:

- Xin Zhang et al. [2] A dual mode(side and frontal) switching algorithm to identify the hand pose in writing mode and ellipse fitting technique for palm point detection. A physical model for fingertip detection is also given. They have given DSB-MM(Depth Skin Background Mixture Model) hand segmentation algorithm; new approach of Fingertip Detection and MQDF(Modified Quadratic Discriminate Function) for classification.

- M.K. Bhuyan et al. [3] proposed a hand calibration algorithm consists Extraction of Geometric features and Hand modeling. Five circles intersecting with the points in segmented image are considered as fingertip points. They have proposed hand segmentation using Skin

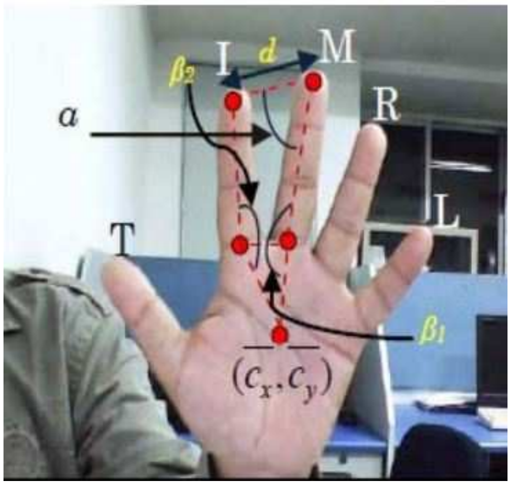
Segmentation, Detection of Centroid, Normalization Constant, Determination of Orientation and Hand and Forearm Segmentation. They have proposed a hand calibration algorithm consists Extraction of Geometric features and Hand modeling.

- Meenakshi et al. [5] proposed a hand gesture recognition system based on shape parameters. For different scenario of situation the system changes its interest and the shape of the parameters also. For different gestures unique code in generated which helps to classify the gesture easily.
- Zhou Ren et al. [6] [7] proposed a system of robust part based hand gesture recognition using different camera. The gesture is recognized based on finger earth mover's distance with a commodity depth camera [6]. But within two years the recognition system he has proposed for the depth Kinect camera.
- Duan Duan Yang et al. [8] an effective robust fingertip detection method for finger writing character recognition system on plane surfaces. Template matching, axis-boundary intersection, conic fitting, contour tracking is done. To detect the fingertip, position is identified from hand contour and then circle feature matching is done.
- Sanjay Iyer et al. [9] detection of fingers with a depth based hand detector in static frames. True positive, false positive approach is taken in case of template matching and area percentage. Skin model is used for segmentation.
- Jagdish et al. [10] new way of tracking fingertips and determining the center of palm. This paper is helpful for palm point identification policy. By applying the distance transform on the inverted binary images of hand. The paper introduced color difference for different hand.
- For edge detection two important papers are followed by us [11][12]. Image segmentation is done based on edge detection using boundary code[11]. But it is for the 2D cameras. For 3D camera such as Kinect A. Lejeune et al. [12] a new jump edge detection method implemented on the basis of canny approximation of edge detection.

- For ellipse fitting algorithm, Walter Gander et al. [13] least-square fitting of circles and ellipses. Andrew Fitzgibbon et al. [14] a direct approach of non-linear least square fitting of ellipse.

For finger identification there is a limited resource because it is an application of fingertip detection and this is not well established by the algorithms. The following papers represent the basic two different algorithms to identify the fingers:

- M.K. Bhuyan et al. [3] proposed a method to identify the fingers by drawing straight lines from the fingertips and connecting them to the centroid. The tip to tip distance value is considered as the main factor to identify the fingers. All the tip to tip distances are calculated by the Euclidean distance equation. A line is drawn from the fingertip to the palm area and the palm area point is connected by a line to centroid. The angle created between is measured and the compared to the defining value, if match found then the finger is said to be identified.

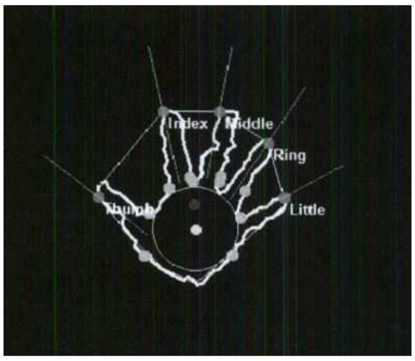


[Fig 1: Finger identification by angle.]

- Yi Li et al. [4] proposed a Three Point Alignment algorithm for fingertip detection and given a finger identification algorithm.

Finger Identification Algorithm:

1. The first and easiest step is to identify the thumb and the index finger, since the distance between them is the largest among all neighboring fingers.
2. The little finger is identified as the farthest finger away from the thumb, meanwhile the middle finger is identified as the closest one to the index finger.
3. The remaining one is the ring finger.



[Fig 2: Finger identification by biometric features]

Other Related Work:

- Finger Extraction from Scene with Grayscale Morphology and BLOB Analysis is proposed by D. D. Nguyen.[15]
- X. Yin et al. [16] a model for finger identification and hand posture recognition for human robot interaction.
- S. K. Kang et al. [17] proposed a Color Based Hand and Finger Detection Technology for User Interaction.
- Different types of camera, input image frames (static and template based), different methods of segmentation and fingertip detection work are proposed.

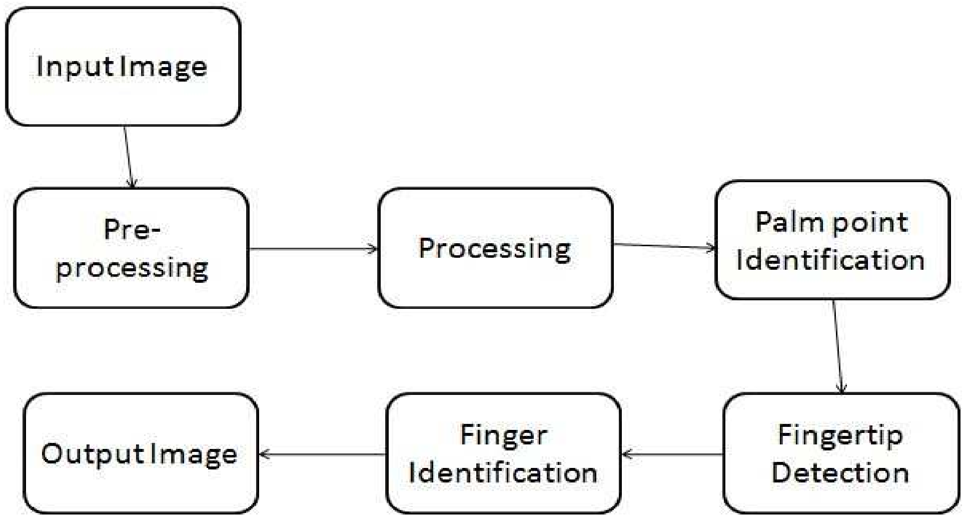
2.2 Literature Limitations:

- In some of the approaches, defining an angle for the finger detection doesn’t gives us the exact fingertip point. Also it goes for only one finger detection or two, not for five fingers.
- Intersection points may vary with the movement. No idea is given about how to track fingertip in real time movement.
- The distance between the tip to tip is considered and the area point is not defined properly. What are the reasons behind this alter specification?
- Template matching is done in some of the approaches, which is very much inefficient for the real time behavior.
- Many research paper does not contain good evaluation process . Only the idea specification is given.

Chapter 3
Proposed System

3.1 System Overview:

We are proposing a system to detect the precise fingertip and identify the fingers name. The work flow of the system could be defined more easily by a flow diagram:



[Fig 3: Work flow of Proposed system.]

The input images are coming from the dataset of Kincet sensor and then they are pre-processed and processed. Hand segmentation is included in the process step. The segmented image is used for fingertip detection and the detected fingertips & the palm point is used for finger identification. These things together could be used for any gesture recognition system. The gesture based applications could use the idea of fingertip detection and identification of fingers.

3.2 Proposed Methodologies:

The input images are taken from the Kinect sensor. The RGB image as well as the depth image is taken as input. The RGB image represents the pixel to pixel points index value and the depth image represents the depth value for 3each points.

Pre-processing includes the image thresholding ,RGB to Gray-scale conversion and color & depth calibration. As Kinect does not calibrate the color and depth image. The color-depth synchronization problem is introduced here. So we propose to calibrate these two image to get the exact depth values for the pre-processing. Boundary extraction could also be done to get the edge points of the binary image.

Determining the minimum depth value from the Kinect camera, determining the segmentation threshold, cropping the region of interest and edge detection are the steps of processing. Depth and color segmentation and calibration is done for hand segmentation.

For hand segmentation Depth and Color Segmentation Model could be applied. Depth and color combination model would be better because only the color or only the depth values are not enough for the segmentation. Several possible problems could be generated because of different segmentation policy used.

For Palm Point Identification:

The center of the palm of the hand is declared as the palm point. The palm point could be determined by various ways. The area of the palm could be estimated and then the center of mass could be found out. That could be considered as the palm point. Again, from all the edge points the center could be determined. The edge points could be given some weight values and then from the weights the center of mass could be generated and considered as the palm point.

But these approaches differ from person to person and situation to situation. So we are going to follow the approach proposed by the Xin Zhang et al. [2]. He proposed that ellipse fitting algorithm could be used to find the palm point. If we can find the palm points, from them we can implement any ellipse fitting algorithm to get the palm point.

For Fingertip Detection:

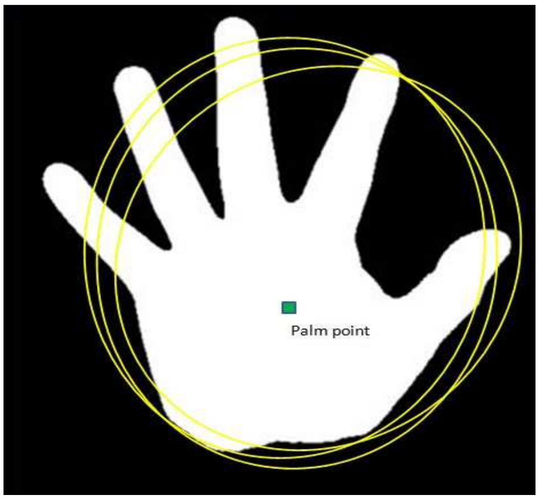
The ellipse fitting technique will draw ellipse considering the base point, right side point and left side point. The ellipse fitting algorithm will return the center of the palm as in palm point. The fingertip points could be identified from graph based solution or implementing basic logic. If the edge points are found from the segmented image, the points will have the five fingertip points and other points also. So extracting the five interest points will work for us.

The fingertip points will be the most far points from the palm points. So extracting the five distant points will give us the five fingertips.

For Finger Identification:

For finger identification we have proposed our own algorithm which is named as 4Y- Model based on simple Hill climbing algorithm. The input for the algorithm would be the five fingertips. The valley points of the hand orientation could be found from the Hill climbing algorithm as the minimum distance points. The distance between the fingertips and the valley points could be calculated and the Euclidean distance could be used for the finger identification.

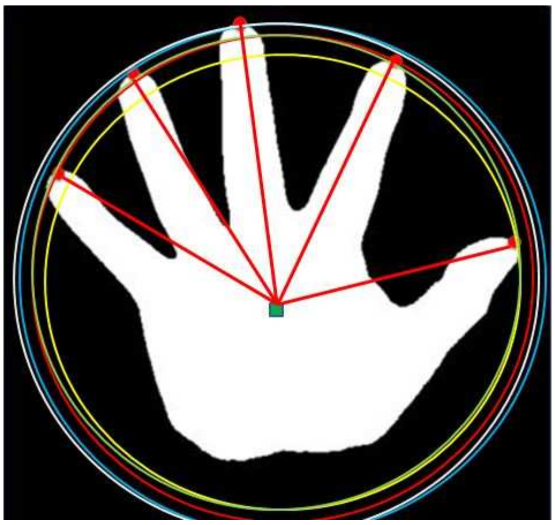
3.2.1 Palm Point Determination:



[Fig 4: Ellipse fitting technique for palm determination. This figure is not experimental result. This represents the way of our proposed algorithm will work]

Ellipse fitting algorithm could be established. The Ohad Gal’s fit_ellipse algorithms could be followed to locate the palm point.

3.2.2 Fingertip Detection:

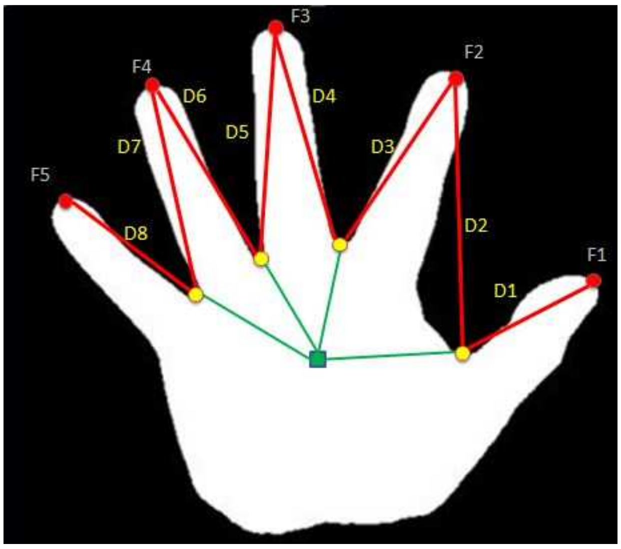


[Fig 5: Fingertip detection using point-to-point scan algorithm. This figure is not experimental result. This represents the way of our proposed algorithm will work]

By point-to-point scan algorithm the first intersection point is found and the circle is drawn taking the palm point as center. Then the radius is decreased by a decrement function and the four other circles are drawn at the fingertips points. The intersections are considered as the fingertips.

3.2.3 Finger Identification:

We are proposing a 4Y-model based on iterative Hill climbing algorithm for finger identification. The algorithms is described below:



[Fig 6: 4Y Model for finger identification. This figure is not experimental result. This represents the way of our proposed algorithm will work.]

Let Po(Xp,Yp) be the palm point. Consider there are four joint or valley points $V_1(X_{v1},Y_{v1})$, $V_2(X_{v2},Y_{v2})$, $V_3(X_{v3},Y_{v3})$, $V_4(X_{v4},Y_{v4})$, respectively and five fingertips' are $F_1(X_{f1},Y_{f1})$, $F_2(X_{f2},Y_{f2})$, $F_3(X_{f3},Y_{f3})$, $F_4(X_{f4},Y_{f4})$, $F_5(X_{f5},Y_{f5})$, respectively. Consider the distances between the fingertips and valley points D1, D2, D3,D4,D5,D6,D7 and D8 , respectively. The distance is calculated by the Euclidean Distance Equation,

Distance = squareroot((x2-x1)^2 + (y2-y1)^2)

Input: Five fingertip points coordinate $F_1(X_{f1},Y_{f1})$, $F_2(X_{f2},Y_{f2})$, $F_3(X_{f3},Y_{f3})$, $F_4(X_{f4},Y_{f4})$,$F_5(X_{f5},Y_{f5})$, respectively and the palm point $Po(X_p, Y_p)$.

Algorithm Steps:

1. Show the five fingers of the hand in front of the camera for better calibration. The hand should be perpendicular to the camera and fully stretched for this algorithm.
2. Start iterative Hill-Climbing Algorithm from the $F_1(X_{f1},Y_{f1})$ contouring up to the last fingertip $F_5(X_{f5},Y_{f5})$ and trace the valley points coordinate $V_1(X_{v1},Y_{v1})$, $V_2(X_{v2},Y_{v2})$, $V_3(X_{v3},Y_{v3})$, $V_4(X_{v4},Y_{v4})$, respectively.

3. Find out all the eight distances between the fingertips and valley points using the Euclidean Distance Equation.
4. As for F1 and F5 there will be only one distance value, starting the identification from these points.
5. If (D1+D2) is greater than (D7+D8), then F1 is in the THUMB finger and F2 is in the INDEX finger. Again, F4 is in the RING finger and F5 is in the LITTLE finger.
6. There is only one finger left which is MIDDLE finger.

Output: Output of this algorithms is to show the name of the finger with the fingertips in the associated image.

This is the proposed algorithm for finger identification. We have implemented the algorithm in our experiment part and the result analysis is also done for the algorithm.

Chapter 4
Experiment and Result Analysis

4.1 Experiment:

The experiment is done as the implementation of the proposed algorithms of fingertip detection and finger identification. Pre-processing, processing, palm point identification, fingertip detection and finger identification are the steps to be done in the experiment in order to validate the proposed algorithms.

The dataset of the NTU-Microsoft-Kinect-HandGesture dataset, which contains the color image and the corresponding depth map obtained from the Kinect sensor. It contains 10 different gestures taken from 10 subjects(persons), and for each gesture, there are 10 variations. Thus, there are 10*10*10 = 1000 cases. The reference for this dataset goes to ACM Multimedia papers [6][18] published in 2011.

As the dataset has 10 different gestures, but our algorithm is only validate for the stretched hand. So only 100 cases could be used for testing this experiment.

The input images are two types: RGB image and depth image. The RGB image represents the RGB pixel values and the depth image represents the depth values. The experiment starts with loading the input images.

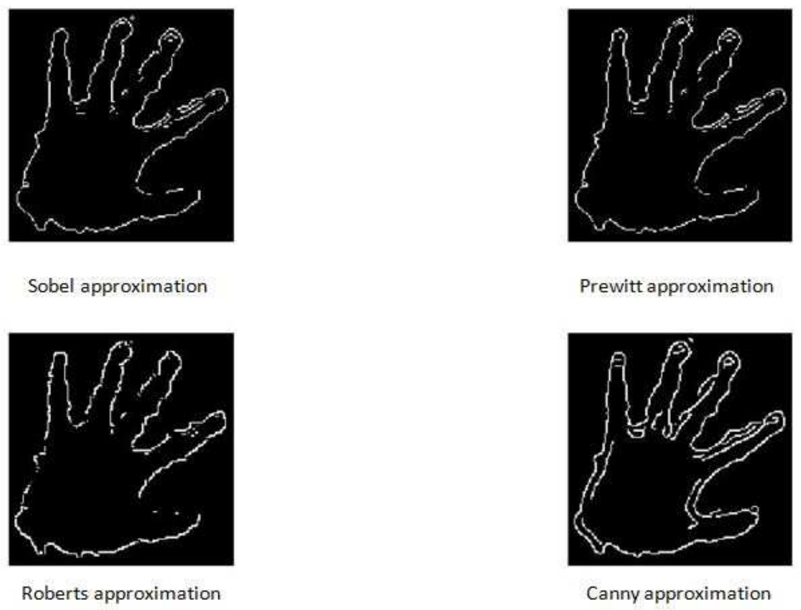
4.1.1 Pre-processing:

The RGB image is converted to gray-scale image and depth image had been loaded to a variable. Then we had to calibrate the color and depth image as Microsoft Kinect does not do calibration for the image and it introduces the color-depth synchronization problem. The calibration is nothing a random process which works as trial and error solving mechanism.

4.1.2 <u>Processing:</u>

The calibrated image is then processed. By processed we mean finding the minimum depth value from the image; determining the segmentation threshold which defines the region of interest from the image; cropping the interest part into another image. So the cropped image is the considering image after this level. The edge detection is done on the cropped image.

There are different edge detection algorithms in the state of the art. Sobel, Canny, Prewitt and Roberts are the main approximation of edge. We have tested all four types of edge detection and found that “Canny’s Approximation” gives the best result.



[Fig 7: Four different edge detection applied to the image.]

4.1.3 <u>Palm Point Identification:</u>

The cropped image is converted in binary image and the dilation operation is done on the binary image. The constant used for the dilation is 10 and the structural element used is disk. After dilation the image only gives the palm region and the indices of the palm region are determined. Then the ellipse fitting algorithm is being used. Ohad Gal's ellipse fitting algorithm is directly used to get the parameters for the image. The input for the algorithm are the boundary points and the output is a structure containing sub axis radius of x and y axis; orientation in radius of the ellipse; center coordinates of the non-tilt ellipse; center coordinates of the tilted ellipse; size of the long and short axis and status of detection of an ellipse. The center coordinates of tilted ellipse is the center of palm which is palm points.

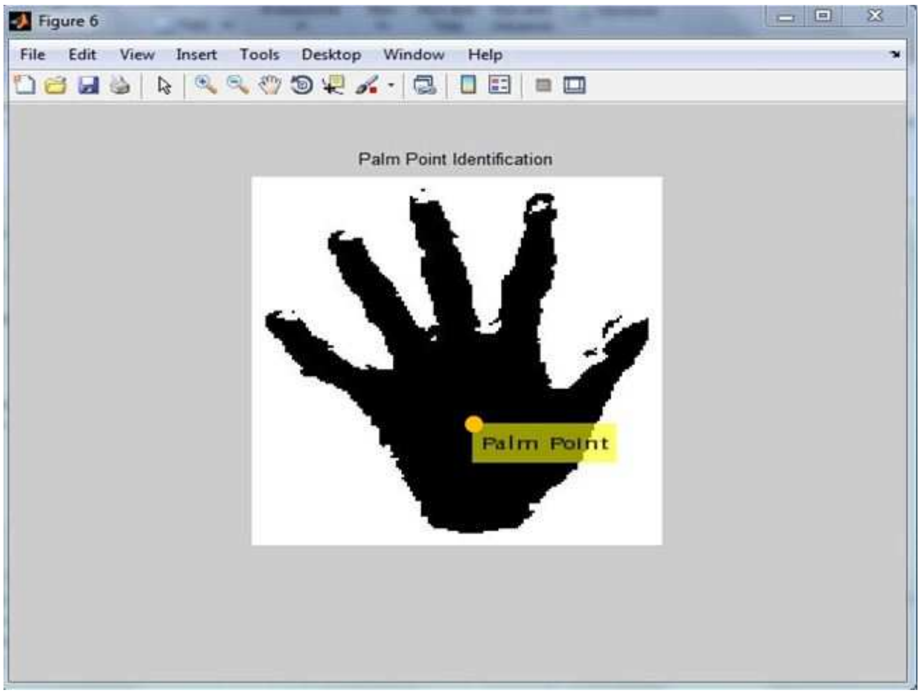
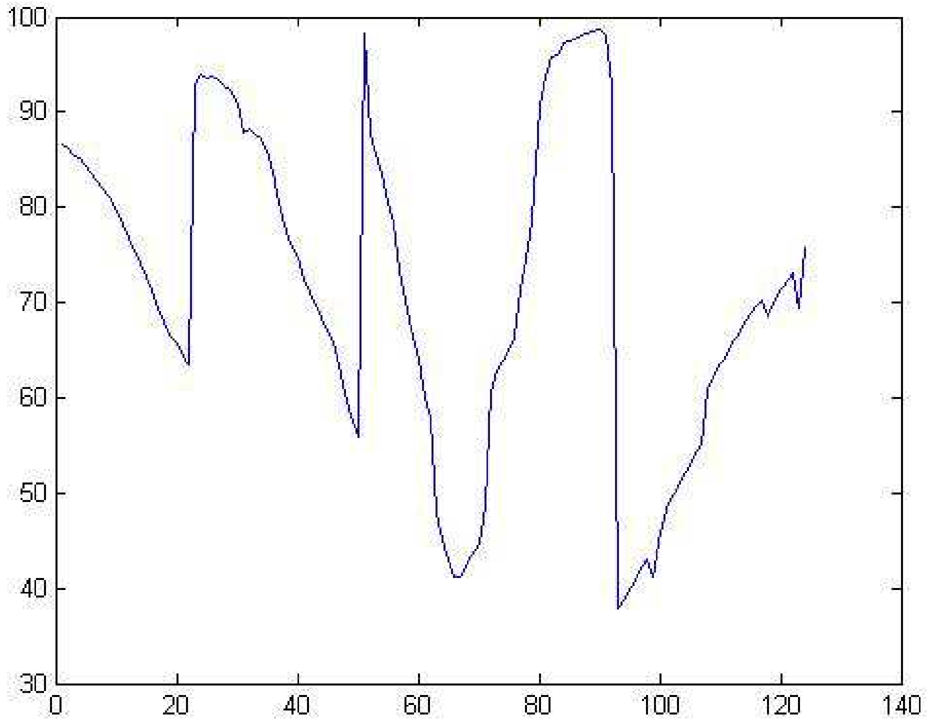


Figure 6
File Edit View Insert Tools Desktop Window Help
Palm Point Identification
Palm Point

[Fig 8: Palm point Identification by Ohad Gal's Algorithm.]

4.1.4 **Fingertip Detection:**

Canny approximation is used to get the edge points of the whole image. The points are get processed to get only one y axis value for each x axis points. All the distance between the points and the palm point are calculated. The most close four are the valley points and the farthest five points are the fingertip points. To determine the farthest and nearest points the iterative Hill climbing algorithm is implemented. The five fingertip points are extracted from the other points.



[Fig 9: The Euclidean distance between the palm point and the edge points.]

The peak points in the graph shows the five fingertips as they are the farthest points from the palm point. Again, the four valleys in the graph indicates the four valley points for the 4Y model.

4.1.5 <u>Finger Identification:</u>

The 4Y model which has been proposed is implemented for identifying the fingers. The algorithm is followed step by step and the input was given accordingly. The output results the five fingertips along with the name of the fingers(Thumb, Index, Middle, Ring and Little).

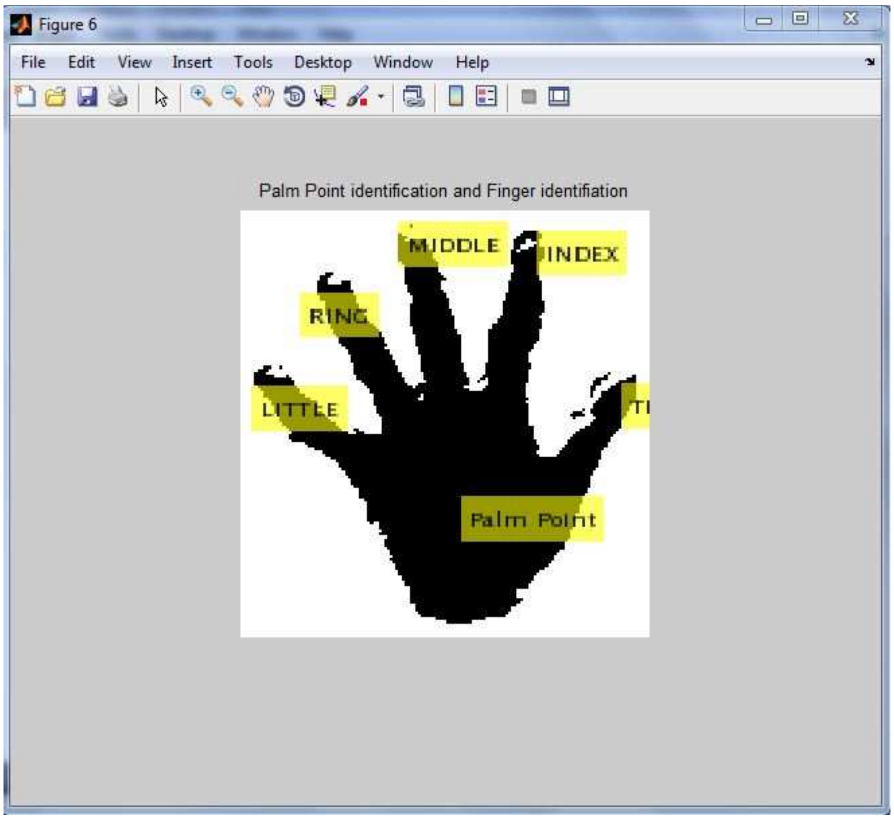
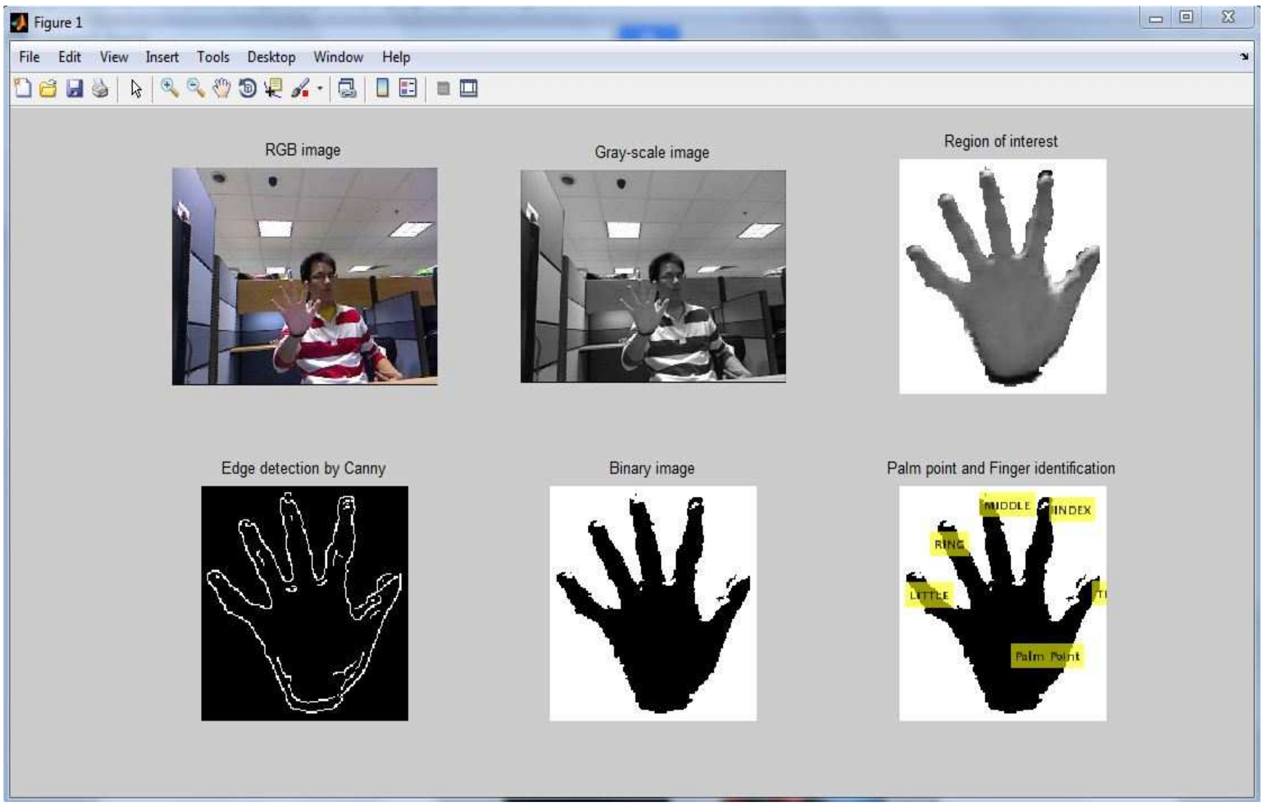


Figure 6
File Edit View Insert Tools Desktop Window Help
Palm Point identification and Finger identifiation
MIDDLE
INDEX
RING
LITTLE
TI
Palm Point

[Fig 10: Five Fingers are identified in the image.]

The whole process of fingertip detection and finger identification could be show in a single figure of Matlab output.



[Fig 11: Implementation of the Proposed 4Y model.]

4.2 <u>Result Analysis:</u>

For result analysis, we have used Matlab 13a and figured out the output image. The output of the experiment were gray-scale image, cropped image which is the region of interest, edged image, binary image and the binary image indicating the fingers name along with the palm point.

The dataset of the NTU-Microsoft-Kinect-HandGesture dataset, which contains the color image and the corresponding depth map obtained from the Kinect sensor. It contains 10 different gestures taken from 10 subjects(persons), and for each gesture, there are 10 variations. Thus, there are 10*10*10 = 1000 cases. The reference for this dataset goes to ACM Multimedia papers [6][18] published in 2011.

As the dataset has 10 different gestures, but our algorithm is only validate for the stretched hand. So only 100 cases could be used for testing this experiment.

We have tested the 4Y Model for finger identification using 100 test images and depth data. The contingency matrix or the confusion matrix [20] gives the actual reading of data that are used to determine the Specificity and Sensitivity of the proposed algorithm.

	Predicted Class					
Actual Class	Finger Names	THUMB	INDEX	MIDDLE	RING	LITTLE
	THUMB	90	7			3
	INDEX	6	92	2		
	MIDDLE		1	96	3	
	RING			3	88	9
	LITTLE	4			6	90

[Table 1: Confusion Matrix.]

Fingers	THUMB	INDEX	MIDDLE	RING	LITTLE
Accuracy(%)	90%	92%	96%	88%	90%
Error(%)	10%	8%	4%	12%	10%

[Table 2: Accuracy and error rate of the findings.]

Sensitivity and Specificity of The Proposed Algorithm:

Sensitivity and specificity are the statistical measures to measure the performance of any binary classification test [19]. As in our algorithm classification is done, so sensitivity and specificity could be a good measurement to judge the algorithm effectively.

Sensitivity measures the proportion of actual positives which are correctly identified as such [19]. The equation to measure this quantity is :

$$\text{sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}$$

Specificity measures the proportion of negatives which are correctly identified as such [19]. The equation to measure this quantity is :

$$\text{specificity} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}}$$

True Positive(TP), True Negative(TN), False Positive(FP), False Negative(FN) are determined from the confusion matrix. These four quantities could be used to measure other five major terms to justify the proposed algorithm.

Precision or Positive predictive value(PPV) is the positivity measurement from the positive findings of the data. The equation for PPV is :

$$\mathrm{PPV} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false positives}} = \frac{\text{number of true positives}}{\text{number of positive calls}}$$

Negative predictive value(NPV) is the negativity measurement from the negative findings of the data. The equation for NPV is :

$$\mathrm{NPV} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false negatives}} = \frac{\text{number of true negatives}}{\text{number of negative calls}}$$

Fall-out or False positive rate(FPR) is the measurement of positivity which are rejected from the positive findings of the data. The equation for FPR is :

$$\frac{FP}{FP + TN}$$

False Discovery Rate(FDR) is the measurement of false alert which is found from one minus the positive predictive value of the findings. The equation for FDR is :

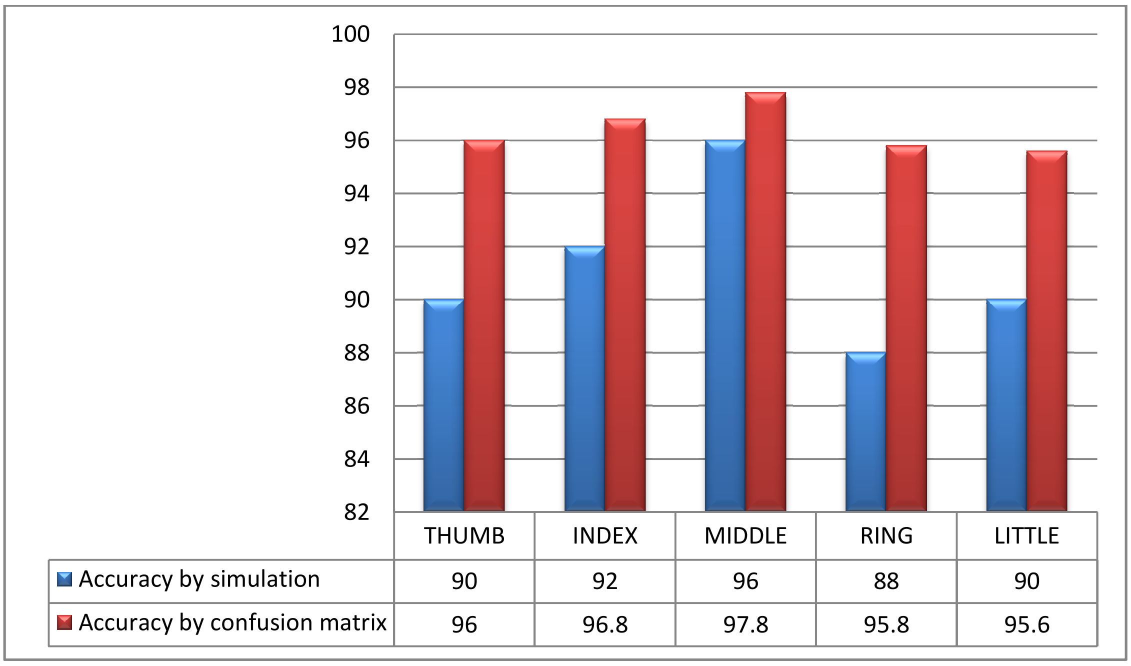
$$\mathrm{FDR} = 1 - \mathrm{PPV} = \frac{\text{number of false positives}}{\text{number of true positives} + \text{number of false positives}} = \frac{\text{number of false positives}}{\text{number of positive calls}}$$

Accuracy from the confusion matrix could be measured by the division of trueness and total positivity and negativity of the findings. The equation for Accuracy is :

$$ACC = (TP + TN)/(P + N)$$

	THUMB	INDEX	MIDDLE	RING	LITTLE
True Positive(TP)	90	92	96	88	90
True Negative(TN)	390	392	393	391	388
False Positive(FP)	10	8	7	9	12
False Negative(FN)	10	8	4	12	10
Sensitivity	0.90	0.92	0.96	0.88	0.90
Specificity(SPC)	0.9750	0.9800	0.9825	0.9775	0.9700
Precision(PPV)	0.900	0.920	0.932	0.907	0.882
Negative Predictive Value(NPV)	0.9750	0.9800	0.9899	0.9710	0.9750
Fall-out/False Positive Rate(FPR)	0.0250	0.0200	0.0175	0.0225	0.0300
False Discovery Rate(FDR)	0.100	0.080	0.068	0.093	0.118
Accuracy	0.960	0.968	0.978	0.958	0.956

[Table 3: Sensitivity, Specificity, Negative Predictive Value, Fall-out or False Positive Rate, False Discovery Rate and Accuracy measurement from the confusion matrix.]



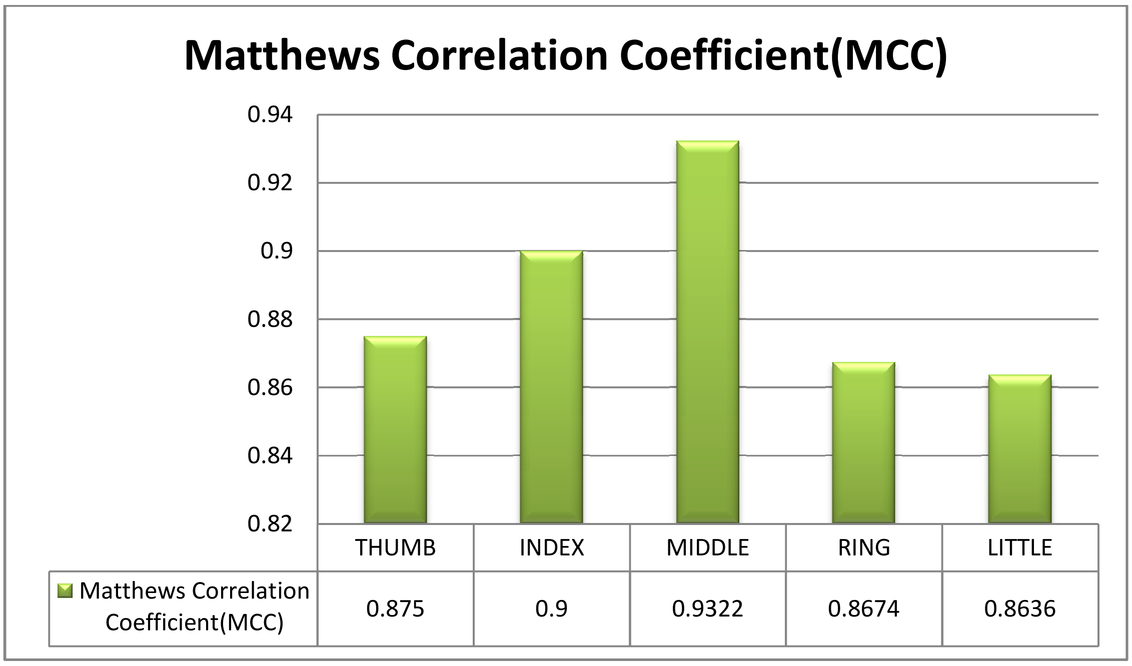
	THUMB	INDEX	MIDDLE	RING	LITTLE
🟦 Accuracy by simulation	90	92	96	88	90
🟥 Accuracy by confusion matrix	96	96.8	97.8	95.8	95.6

[Fig 12: Accuracy by simulation versus accuracy by confusion matrix.]

Matthews Correlation Coefficient(MCC)

The Matthews correlation coefficient is used in machine learning as a measure of the quality of classifications. It takes into account true and false positives and negatives and is generally regarded as a balanced measure which can be used even if the classes are of very different sizes. The MCC is in essence a correlation coefficient between the observed and predicted binary classifications; it returns a value between −1 and +1. [21]

$$\mathrm{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

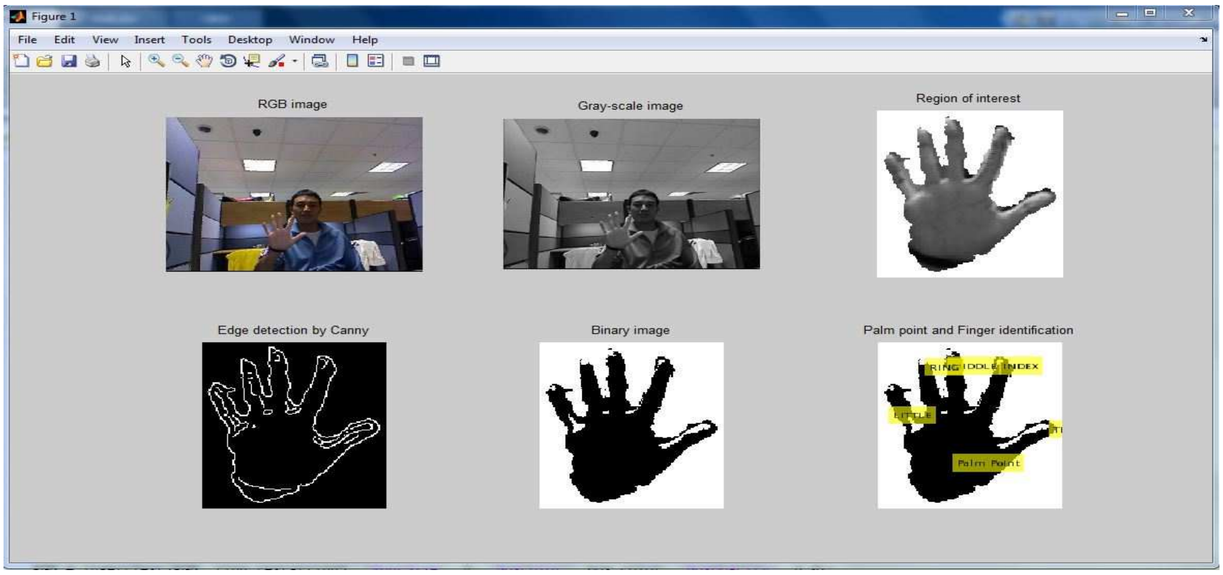


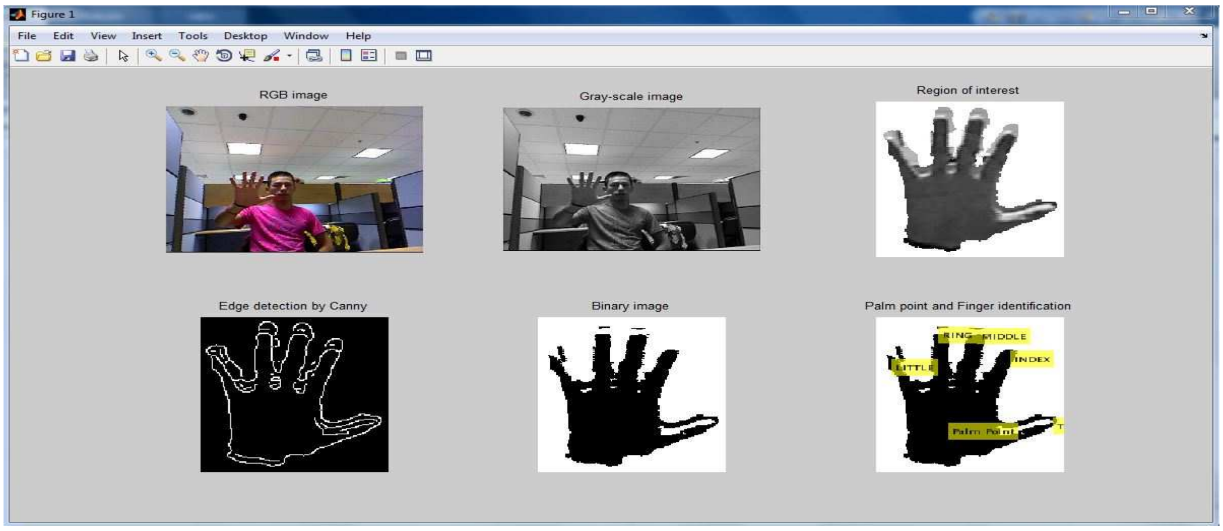
[Fig 13: Matthews Correlation Coefficient for five fingers.]

Serial No	Actual Point	Resulting Output Point	Accuracy(%)	Percentage of Error(%)
1	(76,100)	(79,102)	97.23%	2.77%
2	(55,100)	(50,100)	98.333%	1.667%
3	(42,45)	(45,43)	98.864%	1.136%
4	(48,102)	(48,100)	98.666%	1.334%
5	(56,98)	(56,98)	100%	0%
6	(65,96)	(64,97)	99.135%	0.865%
7	(78,105)	(75,105)	98.336%	1.664%
8	(98,92)	(97, 100)	96.446%	3.554%

[Table 4: The accuracy and percentage of error in determination of palm point along with the actual points.]

Some of the example output:





[Fig 14: Final output of the system. The examples are for different images.]

Chapter 5

Conclusion and Future Prospects

Advantages:

For different Human Computer Interaction(HCI) applications the fingertips location and the identity of the fingers could be used. Sign language, Calculator, Gesture recognition and spotting, Trajectory analysis, Real time Character recognition using hand gesture these are applications where the output ingredients could be used.

Problems We Faced:

1. There are lack of resource for finger identification. again, there is a huge resource for fingertip detection, which confuses us to choose the best one among the algorithms .
2. Lack of dataset is another problem we have faced. Only 100 images as a dataset is not enough for image processing.
3. Error: As the hand in the image from the dataset are not perpendicular to the camera, the data is also biased and the output accuracy lacks behind.

Future Prospects

- Considering huge dataset to get the best out of it.
- Filtering algorithm to get more accurate image and noise reduction.
- Try to implement the jump edge algorithm to find the best edged image.
- Considering different algorithms to get the fingertips.
- Try to implement 4Y model for different 3D camera also.
- Implement any application such as character recognition, sign language recognition etc. in order to use the fingertips detected and fingers identified.

References:

[1] V. I. Pavlovic, R. Sharma and T. S. Huang, “Visual Interpretation of Hand Gestures for Human-Computer Interaction,” IEEE Trans.Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 677–695, 1997.

[2] Xin Zhang, Zhichao Ye, Lianwen Jin, Ziyong Feng and Shaojie Xu (South China University of Technology); “A new writing experience: Finger writing in the air using a Kinect sensor.” Published by IEEE Computer Society Journal of Multimedia at work (October-December 2013).

[3] M.K. Bhuyan, Debanga Raj Neog and Mithun Kumar Kar (Dept of EEE, IIT Guwahati India); “Fingertip detection for hand pose recognition”. Published in International Journal of Computer Science and Engineering(IJCSE) in Vol 4 Num 3 (March 2012).

[4] Yi Li; “Hand gesture recognition using Kinect” Dept. of Computer Engineering and Computer Science(CECS), University of Louisville, Kentucky, USA. (May 2012).

[5] Meenakshi Panwar, Centre for Development of Advanced Computing Noida, Uttar Pradesh, India. “Hand gesture recognition based on shape parameters”. In Proceedings of IEEE International Conference on Image Information Processing(ICIIP 2011), Waknaghat, India, November 2011.

[6] Zhou Ren, Junsong Yuan, Zhengyou Zhang. “Robust Hand Gesture Recognition Based on Finger Earth Mover’s Distance with a Commodity Depth Camera” MM’11, November 28–December 1, 2011, Scottsdale, Arizona, USA.
ACM 2011 id:978-1-4503-0616-4/11/11.

[7] Zhou Ren, Junsong Yuan, Jingjing Meng, Zhengyou Zhang. “Robust Part-Based Hand Gesture Recognition Using Kinect”. Sensor Published on IEEE Transactions on Multimedia, Vol. 15, No. 5, August 2013.

[8] DUAN-DUAN YANG, LIAN-WEN JIN, JUN-XUN YIN, LI-XIN ZHEN, JIAN-CHENG HUANG. “AN EFFECTIVE ROBUST FINGERTIP DETECTION METHOD FOR FINGER WRITING CHARACTER RECOGNITION SYSTEM” Fourth International Conference on Machine Learning and Cybernetics,Guangzhou, 18-21 August 2005.

[9] SANJAY VASUDEVA IYER, Master of Science thesis. “DETECTION OF FINGERS WITH A DEPTH BASED HAND-DETECTOR IN STATIC FRAMES”. THE UNIVERSITY OF TEXAS AT ARLINGTON August 2013.

[10] Jagdish L. Raheja, Ankit Chaudhary, Kunal Singal. “Tracking of Fingertips and Centres of Palm using KINECT” Published on 2011 Third International Conference on Computational Intelligence, Modelling & Simulation.

[11] Takumi Uemura, Gou Koutaki and Keiichi Uchimura. “Image Segmentation based on edge detection using boundary code.” Published on International Journal of Innovative Computing, Information and Control. Vol: 7, No: 10, October,2011.

[12] A. Lejeune, S. Piérard, M. Van Droogenbroeck, J. Verly. Master of Science Thesis from INTELSIG Laboratory, “A new jump edge detection method for 3D cameras”. Montefiore Institute, University of Liège, Belgium. December 2011.

[13] ” Walter Gander, Gene H. Golub, Rolf Strebel. “Least-Squares Fitting of Circles and Ellipses (Institut fur Wissenschaftliches Rechnen Eidgen ossische Technische Hochschule CH-8092 Zurich, Switzerland). Computer Science Department, Stanford University, Stanford, California 94305.

[14] Andrew Fitzgibbon, Maurizio Pilu, and Robert B. Fisher. “Direct Least Square Fitting of Ellipses”. Published on PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 21, NO. 5, MAY 1999.

[15] D. D. Nguyen, T. C. Pham and J. WookJeon, “Finger Extraction from Scene with Grayscale Morphology and BLOB Analysis,” Proc. IEEE Int’l Conf. Robotics and Biomimetics, pp. 324–329, 2008.

[16] X. Yin and M. Xie, “Finger identification and hand posture recognition for human robotinteraction, ”Image and Vision Computing, vol. 25, pp. 1291-1300, 2007.

[17] J. Lin, Y. Wu and T. S. Huang, “Capturing Human Hand Motion in Image Sequences,”Proc. IEEE Int’l Workshop on Motion and Video Computing, IEEE Computer Society, pp. 1–6, 2002.

[18] Zhou Ren, Jingjing Meng, Junsong Yuan, and Zhengyou Zhang."Robust Hand Gesture Recognition with Kinect Sensor", Proc. of ACM Multimedia 2011, (Demo paper)

[19] http://en.wikipedia.org/wiki/Sensitivity_and_specificity

[20] http://en.wikipedia.org/wiki/Confusion_matrix

[21] http://en.wikipedia.org/wiki/Matthews_correlation_coefficient

Appendix

Appendix A

Ellipse Fitting

The fit_ellipse is a matlab function introduced by Ohad Gal in March 2003 and again revised in October 2003. The appendix shows the procedure followed in the algorithm.

fit_ellipse - finds the best fit to an ellipse for the given set of points.

Format: ellipse_t = fit_ellipse(x,y,axis_handle)

Input: x,y - a set of points in 2 column vectors. AT LEAST 5 points are needed !

axis_handle - optional. a handle to an axis, at which the estimated se will be drawn along with its axes

Output: ellipse_t - structure that defines the best fit to an ellipse

a	- sub axis (radius) of the X axis of the non-tilt ellipse
b	- sub axis (radius) of the Y axis of the non-tilt ellipse
phi	- orientation in radians of the ellipse (tilt)
X0	- center at the X axis of the non-tilt ellipse
Y0	- center at the Y axis of the non-tilt ellipse
X0_in	- center at the X axis of the tilted ellipse
Y0_in	- center at the Y axis of the tilted ellipse
long_axis	- size of the long axis of the ellipse
short_axis	- size of the short axis of the ellipse

status - status of detection of an ellipse

Note: if an ellipse was not detected (but a parabola or hyperbola), then an empty structure is returned

Ellipse Fit using Least Squares criterion

We will try to fit the best ellipse to the given measurements. the mathematical

representation of use will be the CONIC Equation of the Ellipse which is:

Ellipse = a*x^2 + b*x*y + c*y^2 + d*x + e*y + f = 0

The fit-estimation method of use is the Least Squares method (without any weights).The estimator is extracted from the following equations:

g(x,y;A) := a*x^2 + b*x*y + c*y^2 + d*x + e*y = f

where:

A - is the vector of parameters to be estimated (a,b,c,d,e)

x,y - is a single measurement

We will define the cost function to be:

Cost(A) := (g_c(x_c,y_c;A)-f_c)'*(g_c(x_c,y_c;A)-f_c)

= (X*A+f_c)'*(X*A+f_c)

= A'*X'*X*A + 2*f_c'*X*A + N*f^2

where:

g_c(x_c,y_c;A) - vector function of ALL the measurements each element of g_c() is g(x,y;A)

X - a matrix of the form: [x_c.^2, x_c.*y_c, y_c.^2, x_c, y_c]

f_c - is actually defined as ones(length(f),1)*f

Derivation of the Cost function with respect to the vector of parameters "A" yields:

A'*X'*X = -f_c'*X = -f*ones(1,length(f_c))*X = -f*sum(X)

Which yields the estimator:

A_least_squares = -f*sum(X)/(X'*X) ->(normalize by -f) = sum(X)/(X'*X)

(We will normalize the variables by (-f) since "f" is unknown and can be accounted for later on). NOW, all that is left to do is to extract the parameters from the Conic Equation. We will deal the vector A into the variables: (A,B,C,D,E) and assume F = -1;

Recall the conic representation of an ellipse:

A*x^2 + B*x*y + C*y^2 + D*x + E*y + F = 0

We will check if the ellipse has a tilt (=orientation). The orientation is present if the coefficient of the term "x*y" is not zero. If so, we first need to remove the tilt of the ellipse. If the parameter "B" is not equal to zero, then we have an orientation (tilt) to the ellipse. We will remove the tilt of the ellipse so as to remain with a conic representation of an ellipse without a tilt, for which the math is more simple: Non tilt conic rep.: A`*x^2 + C`*y^2 + D`*x + E`*y + F` = 0. We will remove the orientation using the following substitution: Replace x with cx+sy and y with -sx+cy such that the conic representation is:

A(cx+sy)^2 + B(cx+sy)(-sx+cy) + C(-sx+cy)^2 + D(cx+sy) + E(-sx+cy) + F = 0

where: c = cos(phi) , s = sin(phi)

x^2(A*c^2 - Bcs + Cs^2) + xy(2A*cs +(c^2-s^2)B -2Ccs) + ...

y^2(As^2 + Bcs + Cc^2) + x(Dc-Es) + y(Ds+Ec) + F = 0

The orientation is easily found by the condition of (B_new=0) which results in:

2A*cs +(c^2-s^2)B -2Ccs = 0 ==> phi = 1/2 * atan(b/(c-a))

Now the constants c=cos(phi) and s=sin(phi) can be found, and from them

all the other constants A`,C`,D`,E` can be found.

A` = A*c^2 - B*c*s + C*s^2 D` = D*c-E*s

B` = 2*A*c*s +(c^2-s^2)*B -2*C*c*s = 0      E` = D*s+E*c

C` = A*s^2 + B*c*s + C*c^2

Next, we want the representation of the non-tilted ellipse to be as:

Ellipse = ((X-X0)/a)^2 + ((Y-Y0)/b)^2 = 1

where: (X0,Y0) is the center of the ellipse

a,b   are the ellipse "radiuses" (or sub-axis). Using a square completion method we will define:

F`` = -F` + (D`^2)/(4*A`) + (E`^2)/(4*C`)

Such that: a`*(X-X0)^2 = A`(X^2 + X*D`/A` + (D`/(2*A`))^2)

c`*(Y-Y0)^2 = C`(Y^2 + Y*E`/C` + (E`/(2*C`))^2)

which yields the transformations:

X0 = -D`/(2*A`)

Y0 = -E`/(2*C`)

a = sqrt(abs(F``/A`))

b = sqrt(abs(F``/C`))

And finally we can define the remaining parameters:

long_axis = 2 * max(a,b)

short_axis = 2 * min(a,b)

Orientation = phi

Appendix B

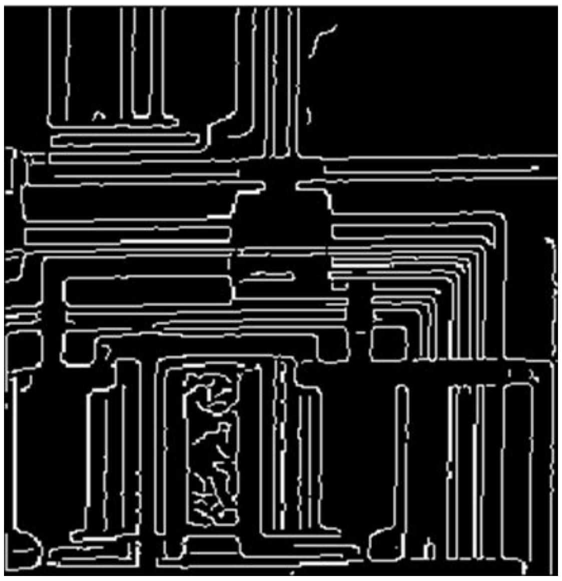
Edge Detection By Canny Approximation

BW = edge(I,'canny') specifies the Canny method.

BW = edge(I,'canny',thresh) specifies sensitivity thresholds for the Canny method. thresh is a two-element vector in which the first element is the low threshold, and the second element is the high threshold. If you specify a scalar for thresh, this scalar value is used for the high threshold and 0.4*thresh is used for the low threshold. If you do not specify thresh, or if thresh is empty ([]), edge chooses low and high values automatically. The value for thresh is relative to the highest value of the gradient magnitude of the image.

BW = edge(I,'canny',thresh,sigma) specifies the Canny method, using sigma as the standard deviation of the Gaussian filter. The default sigma is sqrt(2); the size of the filter is chosen automatically, based on sigma.

[BW,thresh] = edge(I,'canny',...) returns the threshold values as a two-element vector.



[Fig: Canny Edge Detection example.]

The Canny method applies two thresholds to the gradient: a high threshold for low edge sensitivity and a low threshold for high edge sensitivity. edge starts with the low sensitivity result and then grows it to include connected edge pixels from the high sensitivity result. This helps fill in gaps in the detected edges.

In all cases, the default threshold is chosen heuristically in a way that depends on the input data. The best way to vary the threshold is to run edge once, capturing the calculated threshold as the second output argument. Then, starting from the value calculated by edge, adjust the threshold higher (fewer edge pixels) or lower (more edge pixels).

The function edge changed in Version 7.2 (R2011a). Previous versions of the Image Processing Toolbox™ used a different algorithm for computing the Canny method. If you need the same results produced by the previous implementation, use the following syntax:

BW = edge(I,'canny_old',...)

References:

Canny, John, "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence,Vol. PAMI-8, No. 6, 1986, pp. 679-698.

Appendix C

Hill Climbing Algorithm

In computer science, hill climbing is a mathematical optimization technique which belongs to the family of local search. It is an iterative algorithm that starts with an arbitrary solution to a problem, then attempts to find a better solution by incrementally changing a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found.

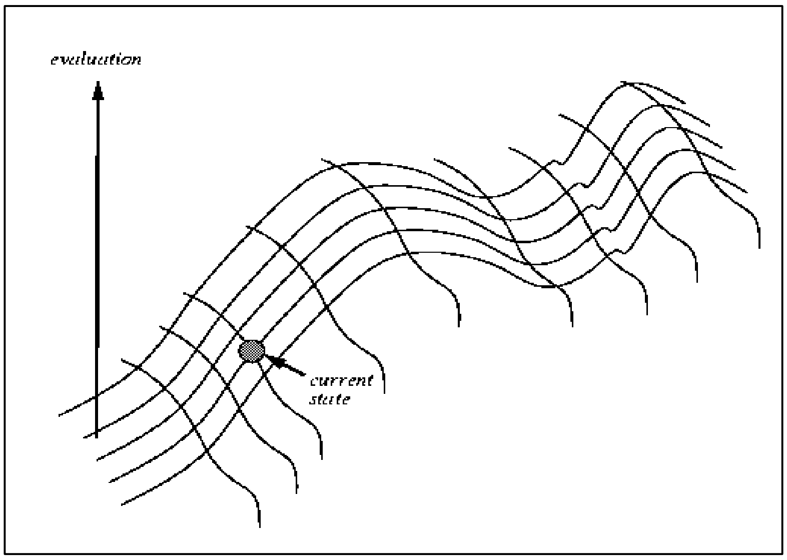
For example, hill climbing can be applied to the travelling salesman problem. It is easy to find an initial solution that visits all the cities but will be very poor compared to the optimal solution. The algorithm starts with such a solution and makes small improvements to it, such as switching the order in which two cities are visited. Eventually, a much shorter route is likely to be obtained.

Pseudo code of Algorithm:

```
Discrete Space Hill Climbing Algorithm
   currentNode = startNode;
   loop do
      L = NEIGHBORS(currentNode);
      nextEval = -INF;
      nextNode = NULL;
      for all x in L
         if (EVAL(x) > nextEval)
              nextNode = x;
              nextEval = EVAL(x);
      if nextEval <= EVAL(currentNode)
         //Return current node since no better neighbors exist
         return currentNode;
      currentNode = nextNode;
```

Problems with Hill climbing:

The main problem with hill climbing (which is also sometimes called ***gradient descent***) is that we are not guaranteed to find the best solution. In fact, we are not offered any guarantees about the solution. It could be abysmally bad. Imagine that you are in the Alps and you want to find the highest peak. You



parachute into the Alps but when you land you are in thick fog. Undeterred you start moving upwards. Eventually, you come to a point where you cannot move upwards anymore. You reason that you have found the highest peak. Of course, that is not necessarily the case. There might be many other higher peaks around you but you cannot see them because of the fog.
And this is exactly the same problem with hill climbing. When we reach a position where there are no better neighbors, it is not a guarantee that we have found the best solution. In fact it could be one of the worst solutions. In terms of the Alps analogy we might be standing on a small hillock instead of at the top of the mountain range. Therefore, the solution that hill climbing returns may not be that good.

We can represent the problem we have just described on this diagram.

You can see that we will eventually reach a state that has no better neighbors but there are better solutions elsewhere in the search space.
The problem we have just described is called a ***local maxima***.
One solution to this problem is to conduct a ***multi-start hill-climb***. This consists of a series of hill climbing searches, each one starting at a random point in the search space.
The best result is saved from all the searches and is returned as the best solution.
But, unless we look at all possible solutions we are still not guaranteed to find to global optimum.

Another type of problem we may find with hill climbing searches is finding a ***plateau.*** If a search hits this type of problem the usual solution is to conduct a random walk until you find an area that starts to give better quality solutions (or you may find that no neighbors are ever of better quality - in which case we can assume we are on the top of table top mountain!)

References

Russell, S., Norvig, P. 1995. *Artificial Intelligence A Modern Approach*. Prentice-Hall