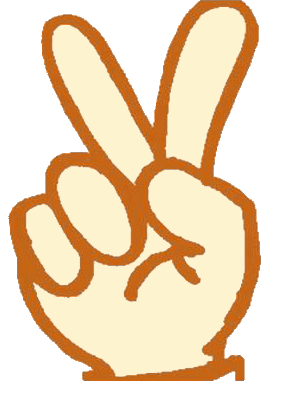


**An-Najah National University**

**Faculty of Engineering**

**Computer Engineering Department**

|  |  |
| --- | --- |
| **Graduation Project Report** | **Hand Gestures Recognition Applications “Virtual Mouse” “Virtual Piano” “Integration with Interactive Game”** |

****

**Supervised by:  
Dr. Luai Malhis.**

**Prepared by:  
Suad Seirafy/ Fatima Zubaidi**

**2011/2012**

**الإهداء**

**الحمد لله كما ينبغي لجلال وجهه وعظيم سلطانه ... الحمد لله حمداً كثيراً مباركاً ملء السموات و ملء الأرض و ملء ما شئت يا ربنا من شيء بعد. و الصلاة والسلام على عبده الذي أسري به إلى المسجد الأقصى, وعرج به إلى السموات العلى..**

**وإلى فلسطين الأبية نقدم ثمرة جهدنا المتواضع, و إلى جامعتنا حماها الله التي وضعنا اسمها " النجاح" نصب أعيننا, إلى من رعانا و مهد لنا سبل النجاح أساتذتنا المخلصين حفظهم الله و رعاهم.. و نخص بالذكر الدكتور لؤي ملحيس والدكتور سامر العرندي وكما نشكر أساتذتنا في قسم هندسة الحاسوب الذين دائما ما حرصوا على ان نصل الى هذه المرحله من العلم والمعرفة , نشكرهم جميعا على جهودهم .**

**وإلى الوالدين العزيزين نبض القلب وشفاء الروح....**

**الى كل هؤلاء نهدي مشروعنا.. اللهم ما كان من توفيق ونجاح فمن عندك.. و ما كان من تقصير و أخطاء فمن أنفسنا.. نسأل الله التوفيق و الإخلاص في العمل..**

# 

# Acknowledgements

We wish to express our deep Gratitude to Allah (SWT), who owns the first and last thank, and our parents in the second place as they are the ones who brought us to this world, raised us up so we can now work on our studies.

We are indebted to our Project supervisor Dr. Luai Malhis and Dr.Samer Al-Arandi (An Najah National University) for his critical reading of an earlier version of this paper and for making many constructive comments and suggestions that have helped us shape it into its present form.

A continuous thank for friends and professors who were giving us the promotion and the courage every time we felt down.

We thank them all for their contribution in a way or another, May Allah blesses them all.

**Abstract:**

This project presents an approach to develop a real-time hand gesture recognition based in “Vision Based” that uses only a webcam and Computer

Vision technology, such as image processing that can recognize several gestures for use in computer interface interaction.

The applications of real time hand gesture recognition in the real world are numerous, due to the fact that it can be used almost anywhere where we interact with computers. An important application of this project is to simulate the mouse as a visual inputting device with all of its tasks such as left click, right click, double-click, dragging and dropping, and scrolling.

Other applications that are applied in our project using the hand gesture recognition are playing a virtual piano using specific gestures for each piano note and playing interactive games by allowing two players to interact with the game using gestures instead of using a controller.

**1.1 TABLE OF CONTENTS:**

**1 TABLE OF CONTENTS AND FIGURES...............................................................1**

**1.1 Table of Contents ………………………………………………………….1**

**1.2 Table of Figures…………………………………………………………….2**

**1.3 Tables ………………………………………………………………………2**

**2 INTRODUCTION...................................................................................................... 3**

**2.1 Project Overview......................................................................................... 3**

**2.2 Project Applications.................................................................................... 4**

**2.3 Report Overview…….................................................................................. 5**

**3 SYSTEM DESCRIPTIONS ………...........................................................................6**

**3.1 System Environment………………………..……………………….……..6**

**3.2 Software Information……………………….……………………….……..6**

**3.3 System Constraints ….…………..………………………………….……...7**

**3.4 System Flow Chart .…………..………………………………….…….......7**

**4 DETECTION............................................................................................................... 8**

**4.1 Image acquisition………………………..………..….….………….…..…..8**

**4.2 Image processing and hand detection ..……….…..….………….………….8**

**4.3 Image Processing Life Cycle……………………………………………………..12**

**4.4 Detection Problems and Solutions ..………………..….…..…………….…..….15**

**5 RECOGNITION .........................................................................................................16**

**5.1 Recognition Notes………………………………………………………….16**

**5.2 Hand Gesture Recognition ………………………………………………..17**

**5.3 Recognition Problems and Solutions……………………………………...19**

**6 Event Generation and Applications ……..................................................................21**

**6.1 FIRST APPLICATION: Virtual Mouse.....................................................21**

**6.2 SECOND APPLICATION: Virtual Piano.................................................24**

**6.3 THIRD APPLICATION: Integrating the project with existing game….28**

**7 CONCLUSIONS......................................................................................................... 32**

**7.1 Project Goals ................................................................................................32**

**7.2 Further Work …….......................................................................................32**

**8 REFERENCES .......................................................................................................... 33**

**9 APPENDIX.................................................................................................................. 34**

**9.1 Recognition regions and values .................................................................. 34**

**9.1.1 Regions ........................................................................................... 34**

**9.1.2 Sequences and Thresholds ........................................................... 34**

**9.2 Allowable Gestures Deviations ................................................................... 35**

**1.2 Table of Figures**

Figure 1: System Environment…………………………………………..7  
Figure 2: System Flow Chart…………………………………………....15  
Figure 3: Image processing steps………………………………………..20

Figure 4: Convex Hull and Convexity Defects for hand.……… …………..22

Figure 5: hand gestures and their related mouse actions………………..25

Figure 6: Snapshot for the Piano………………………………………...27

Figure 7: Hand Gestures and their related Piano notes………………….28

Figure 8: Snapshot for the stick fighters 2 player game……………...….29

Figure 9: Hand Gestures and their related game even………………......29

Figure 10: Deviation allowed in gesture 0…………………………........35

Figure 11: Deviation allowed in gesture 1………………………………35

Figure 12: Deviation allowed in gesture 2……………………………....36

Figure 13: Deviation allowed in gesture 3……………………………....36

**1.3 Tables:**

Table 1: hand gestures and their related mouse actions……………..….21

Table 2: Left and right hand gestures and their related note…..…..……24

Table 3: Hand gestures and their related game actions…..…………..…28

Table 4: Sequences corresponding rows and threshold values used id

Recognition……………………………………………………..34

**2. Introduction:**

Human computer interaction (HCI) is a growing field in which computer scientists study novel ways in which humans can interact with computers naturally and intuitively. One of the most widely researched topics in this field is hand gesture recognition, where hand movements are used to control computers.

The key problem in gesture interaction is how to make hand gestures understood by computers. The approaches present can be mainly divided into “Data-Glove based” and “Vision Based” approaches. The Data-Glove based methods use sensor devices for digitizing hand and finger motions into multi-parametric data. The extra sensors make it easy to collect hand configuration and movement. However, the devices are quite expensive and bring much cumbersome experience to the users. In contrast, the Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices. These systems tend to complement biological vision by describing artificial vision systems that are implemented mostly in software. This approach is the cheapest, and the most lightweight. Moreover, such systems must be optimized to meet the requirements, including accuracy and robustness.

**2.1 PROJECT OVERVIEW:**

In our project, we implemented software using the second approach “Computer Vision” that handles the 2-D real- time video from a webcam and analyzed it frame by frame to recognize the hand gesture at each frame. We used the image processing techniques to detect hand poses from the image by passing the image through many filters and finally apply our own calculations on the binary image to detect the gesture.

The software must satisfy several conditions, including real time function. This is needed for full interactivity and intuitiveness of the interface. This is measured through fps (Frames per Second) which essentially provides information about the refresh rate of the application. If the refresh rate is long, then there is a delay between the actual event and the recognition. If gestures are performed in rapid succession, the event may not be recognized at all. This is why real rime function is crucial.

Another required condition is flexibility, and how well it integrates with new applications as well as existing applications. In order to be a candidate for practical applications, the software must be able to accommodate external programs easily. This is for the benefit of the application developer and the user. Finally, the software must be reasonably accurate enough to be put to practical use.

To achieve all of these requirements, we took a huge number of training sets of hand gestures for different people and in different environments and apply several conditions to them in order to detect the correct gesture. Besides we have to choose a suitable programming language to deal with the real time tracking for the hand, and after a long search about this topic we decided to implement our software image processing part with the help of Intel’s Open Source Computer Vision (OpenCV) Library in C/C++ environment using Microsoft Visual Studio 2010 due to its advantages in real-time image processing.

During the developing period of our project, we faced several problems related with the accuracy, environment, light and the tracking speed. We faced these problems as much as possible – we will talk in details about them later- until we reach to a point that our project is accurate enough to be put to practical use.

**2.2 PROJECT APPLICATIONS**

The applications that we applied our hand gestures recognition in them are:

**1- Virtual Mouse:**

By applying vision technology and controlling the mouse by natural hand gestures, we simulated the mouse system. This mouse system can control all mouse tasks, such as clicking (right and left), double clicking, dragging and dropping and scrolling. We employ several image processing techniques and we developed our own calculations to implement this.  
In this application, we needed to track one hand only. Besides we got used of the windows API to implement the mouse events.

**2- Virtual Piano:**

In this application, we needed to keep track of both hands to get the correct combination of gestures to relate them with their specific Piano notes.  
We used the Wrapper Library for Windows MIDI API in order to get the Piano sounds.

**3- Our Gesture recognition Project can also be easily integrated into already existing interactive applications:**

As an example of this, we integrated it with online interactive 2 player game so that two players can compete in the game each with his hand.

**2.3 Report Overview:**

* Description of system environment, software used in our project and system constraints will be covered in chapter 3.
* The design of Detection process in which image acquired and converted during Image processing lifecycle until the wanted image needed for recognition reached will be covered in chapter 4.
* Using the result of Detection process in order to determine the gesture which called Recognition will be covered in chapter 5.
* The events that will be generated according to gestures will be explained for each project separately in chapter 6.
* The Conclusion will be displayed in chapter 7.
* The References that helped us will be mentioned in chapter 8.
* Values used in recognition and the deviation allowable for each gesture will be explained in chapter 9.

**3 SYSTEM DESCRIPTIONS:**

**3.1 System Environment**

****

The system will use a single, color camera mounted perpendicularly above a dark background surface next to the computer (see Figure 1).

The output of the camera will be displayed on the monitor. The user will not be required to wear a colored wrist band and will interact with the system by gesturing in the view of the camera. Shape and position information about the hand will be gathered using detection of skin.

Figure 1: System Environment.

**3.2 Software Information**

**Microsoft Visual Studio C/C++ environment.**

**OpenCV:**

In our project, to process video frames, we used OpenCV 2.0 library. OpenCV means Intel® Open Source Computer Vision Library. It is a collection of C functions and few C++ classes that implement some popular algorithms of Image Processing and Computer Vision. OpenCV is cross-platform middle-to-high level API that consists of a few hundreds (>300) C functions. It does not rely on external numerical libraries, though it can make use of some of them at runtime.

**Wrapper Library for Windows MIDI API:**

In order to produce the piano notes for our second application, we used Wrapper Library for MIDI API.

**3.3 System Constraints:**

1. The picture of the hand must be taken against a dark background.

2. The camera should be perpendicular to the hand.

3. The hand must be placed upright (inclination angle is 0) or tilted at maximum 20 degrees to left or 20 degrees to right.

4. The hand must remain on the same height, on which initial dimensions of hand were taken.

5. The program recognizes a limited number of gestures and the actions performed depending on the way the gesture occurred.

**3.4 System Flow Chart:**

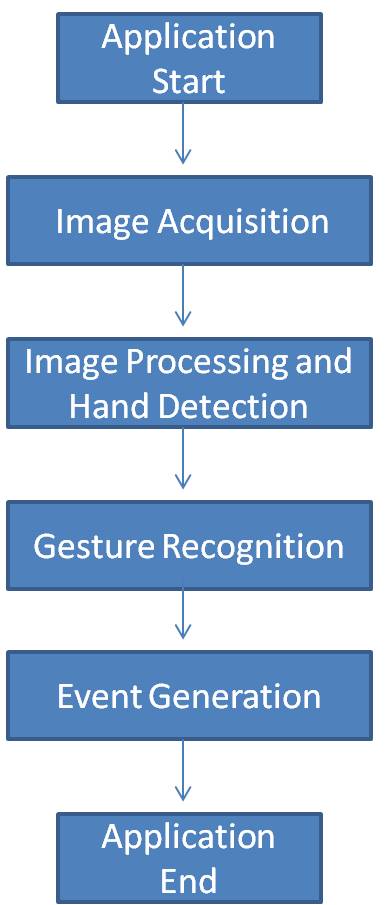
****

Figure 2: System Flow Chart.

**4. DETECTION:**

**4.1 Image acquisition**

Read a video stream from the camera then continuously get one frame to be processed.

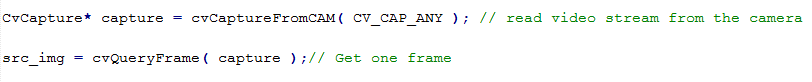


Figure: Get a video stream and take one frame to be processed.

**4.2 Image processing and hand detection**

Step 1: The image is converted into gray scale and smoothed using a Gaussian

Kernel.

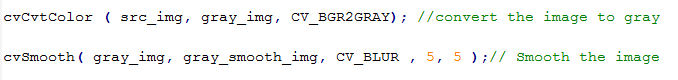


Figure: Gray Scale.

Step 2: Convert the gray scale image into a binary image. Set a threshold so that the

Pixels that are above certain intensity are set to white and those below are set to black.



Figure: Binary Image with specific threshold.

Step 3: Find contours, then remove noise and smooth the edges to smooth big contours and melt numerous small contours.



Figure: Draw the Contour.

Step 4: The largest contour is selected as a target, approximate it by a polygon, bound this new contour by an initial rectangle.

(In the case of two hands the largest 2 contour are selected, approximated and bounded by two initial rectangles)

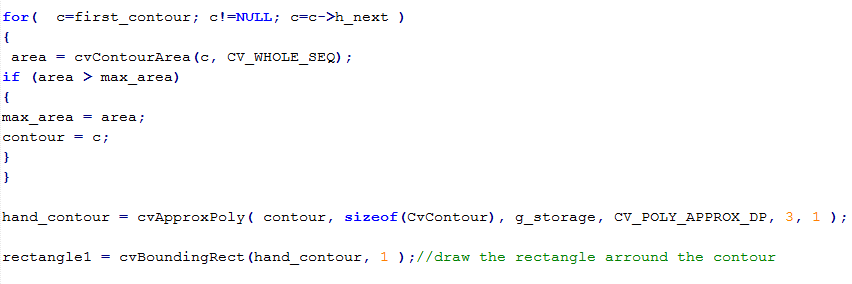


Figure: Selecting the largest Contour and approximate it to polygon.

Step 5: If the user is new, he will be asked to put his hand closed in order to take hand’s dimensions required (closed hand width will considered the original width), else continue to step 7 (in the case of two hands, both hands must be put closed and separated in order to take the width for both hands)

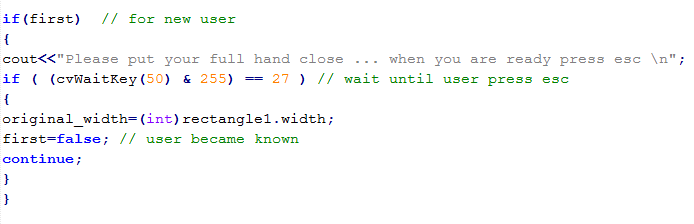


Figure: Taking the new user hand dimensions.

Step 6: User now is known return to step 1.

Step 7 : Take the hand until the wrist by taking hand length equal to 1.7 of original hand width , bound this new contour by a second rectangle and set this rectangle ROI (region of interest), (in the case of two hands, hand length depends on the width of the hand so lengths of two hands may differ)

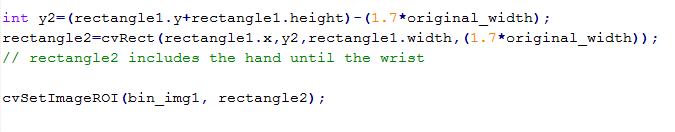


Figure: Determining the wrist.

Step 8: Normalization, Do dual resizing

* Resize image to rectangle2 size



Figure: First Normalization.

* Second resizing depends on whether thumb is hidden or apparent in image (If rectangle2 width larger than original width this mean thumb is apparent, else thumb is hidden).

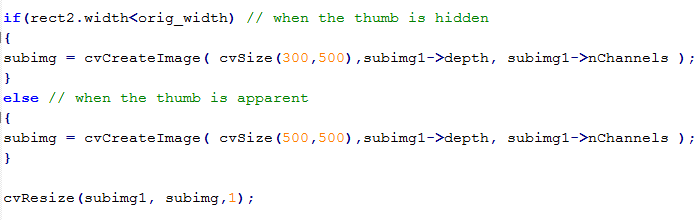
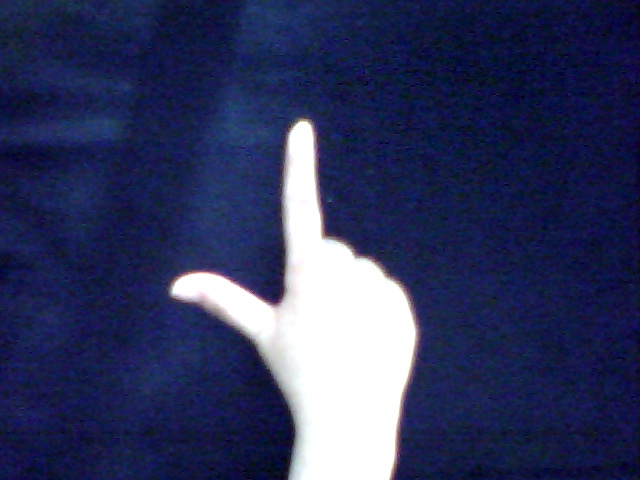


Figure: Second Normalization.

Now image is processed and hands are detected, ready to be recognized.

* 1. **Image processing lifecycle**

****

****

(2) Gesture done by the user

Image size 640\*480

1. Closed hand to get original width

Image size 640\*480

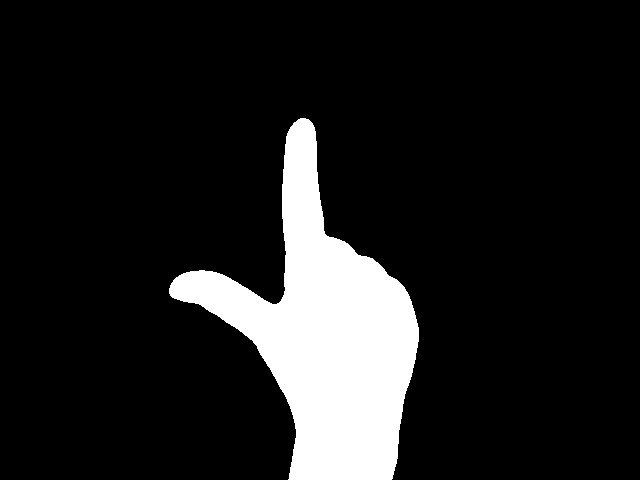
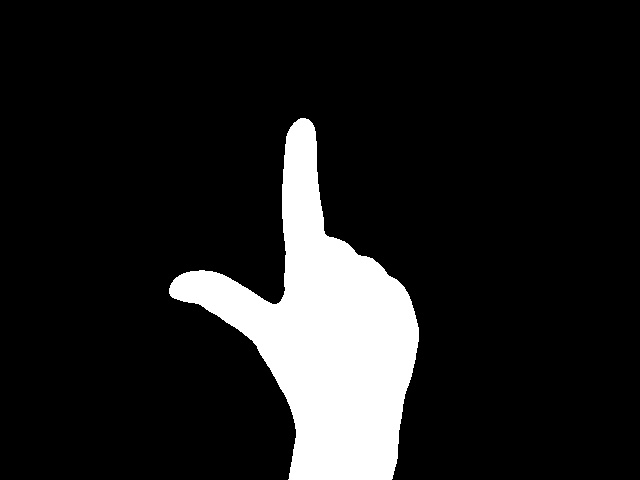
** **

(4) Smoothed Image

Image size 640\*480

(3) Gray Image

Image size 640\*480

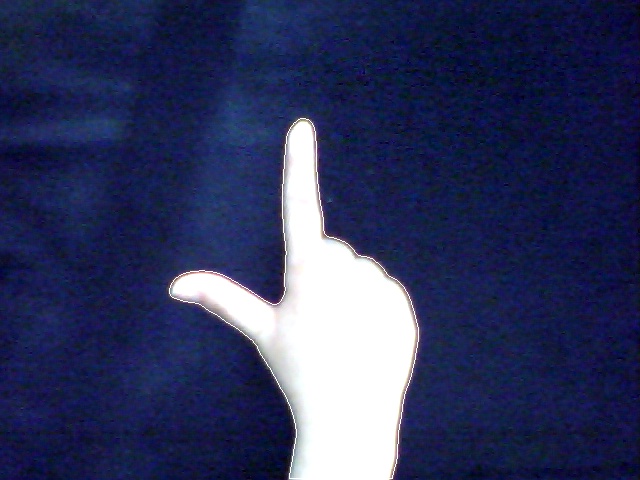
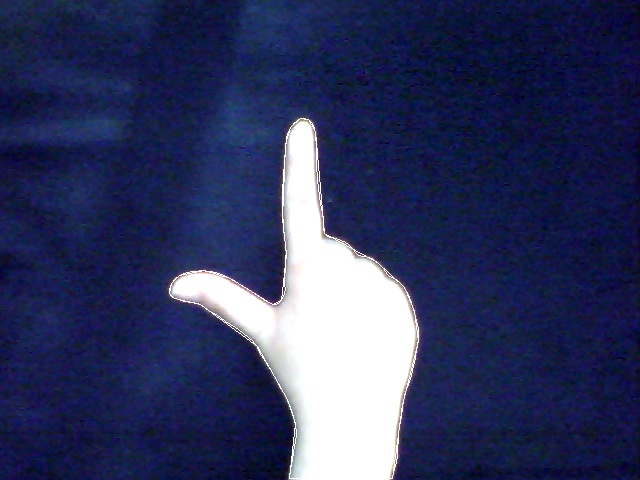
** **

(6) Cloned (binary image)

Image size 640\*480

(5) After applying threshold

Image size 640\*480

** **

(8) Approximate the contour to polygon

Image size 640\*480

(7) Drawing contour around hand

Image size 640\*480

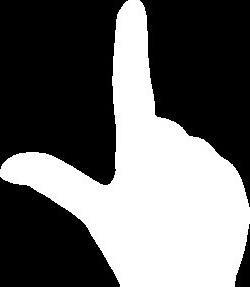
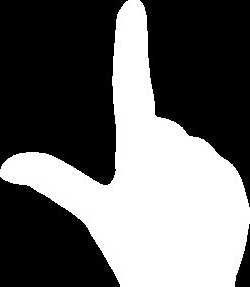
** **

(10) hand length= 1.7 of original width

Image size 640\*480

(9) Draw rectangle around contour

Image size 640\*480

** **

(12) Normalization step1

Image size equal rectangle 2 size

(11) Set hand as ROI

Image size 640\*480

** **

(14) Combining each 25 pixel together

(we don’t save this binary result)

(13) Normalization step2

Image size 500\*500

Figure 3: Image processing steps.

* 1. **Problems that we faced in this stage:**

Problem 1: The appropriate threshold value to extract hand pixels from the total image, this threshold depends on skin color, as we know humans differ in skin colors.

Solution: We took a huge training set and tuned threshold to a value approximately correctly applied to all skin colors.

Problem 2: Previously we extracted the hand just depending on a threshold and processed the result but this was a real problem that took a lot of our time in recognition since not just the hand appeared but sometimes forearm may be appeared in image also.

Solution: we solved this big problem by taking just the hand until the wrist and ignore the rest part, this was not achieved easily we took a huge training set and applied some calculations until we reached a semi-fact helps us in our project, this fact says that if we take hand length equal 1.7 times of closed hand width, we will take the needed region in recognition.

**5. Recognition:**

**5.1 Recognition Notes:**

* In this part we take the deviation of the hand into consideration.
* Gestures that are difficult and user can’t do with hand only are ignored and don’t checked.
* In the case of two hands we take each hand and recognize it separately.

We take specific regions into consideration:

(Region: how many white pixels in this region)

1. Upper/lower regions.
2. Upper/lower diagonals.
3. Left/right regions.
4. region1, region2 and region3 to detect pinkie finger.

We take specific rows into consideration:

(Sequence: how many changes from black to white in a specific row)

1. Sequence1 to detect how many fingers are open from (ring, middle, index) fingers and RowX (how many white pixel in this row) to ensure the result of sequence1.
2. Sequence11 and sequence12 to detect if all fingers are open or not in the case the hand is upright.
3. Sequence21 and sequence22 to detect if all fingers are open or not in the case the hand is tilted to the left.
4. Sequence31 and sequence32 to detect if all fingers are open or not in the case the hand is tilted to the right.

**5.2 Gesture Recognition:**

Step 1: loop the image starting from top left and combine each 25 pixel intensities into one value (black (0) or white (1))(in the case of left hand we will start from bottom right in order to recognize it in the same way as right hand), if value is white make several comparisons:

1. If pixel is below diagonal increase lower diagonal counter, else increase upper diagonal counter.
2. If pixel is in the upper region increase upper region counter, else increase lower region counter.
3. If pixel is in region1 increase region1 counter.
4. If pixel is in region1, region2 or region3 increase the corresponding counter.
5. If pixel is in rowX increase rowX counter.
6. If pixel is in rowX and the previous pixel is black increase sequence1.
7. If pixel in sequence11 corresponding row and the previous pixel is black increase sequence11.
8. If pixel in sequence12 corresponding row and the previous pixel is black increase sequence11.
9. If pixel in sequence21 corresponding row and the previous pixel is black increase sequence11.
10. If pixel in sequence22 corresponding row and the previous pixel is black increase sequence11.
11. If pixel in sequence31 corresponding row and the previous pixel is black increase sequence11.
12. If pixel in sequence11 corresponding row and the previous pixel is black increase sequence32.

Step2: we decided previously if thumb is hidden or apparent depending on this decision we can eliminate our choices for gesture

* If thumb is hidden gesture possibilities :
* Gesture 0 : when the following conditions achieved :

1. Sequence1 equal 1; just one finger from ring, middle and index fingers is open.
2. rowX within specific thresholds (larger than threshold1 and lower than threshold2); to ensure condition 1.
3. Region1 lower than threshold3; to ensure that pinkie is closed.

* Gesture 1 : when the following condition achieved:

1. Sequence1 equal 2; two fingers from ring, middle and index fingers a open.
2. rowX within a specific thresholds(larger than threshold4 and lower than threshold5); to ensure condition 1.
3. Region2 below threshold6; to ensure that pinkie is closed.

* If thumb is apparent gesture possibilities :
* Gesture 2: when following conditions achieved :

1. RowX specific thresholds (larger than threshold7 and lower than threshold8); to ensure that just one finger from ring, middle and index fingers is open.
2. Region1 lower than threshold9; to ensure that pinkie is closed.
3. Upper region larger than lower region by 1.8 times.
4. Upper diagonal region larger than lower diagonal region by 1.5 times.

* Gesture 3: when one of the following conditions achieved :

1. First Condition:

* Sequence11 equal 3 or 4; to ensure that ring, middle and index fingers are open.
* Sequence12 equal 1; to ensure that thumb is open.
* Region2 larger than threshold10; to ensure that pinkie is open.

1. Second Condition:
   * + - Sequence21 equal 3 or 4; to ensure that ring, middle and index fingers are open.
       - Sequence22 equal 1; to ensure that thumb is open.
       - Region1 larger than threshold11; to ensure that pinkie is open.
2. Third Condition:

- Sequence31 equal 3 or 4; to ensure that ring, middle and

Index fingers are open.

* Sequence32 equal 1; to ensure that thumb is open.
* Region3 larger than threshold12; to ensure that pinkie is open.

Now gesture is determined.

**5.3 Problems that we faced in this stage:**

Problem1: We tried using image subtraction; this way was inefficient and slow since we needed to compare each processed image to a set of images stored in database to determine the correct gesture, also we cannot reach a static thresholds for the result of image subtraction.

Problem 2: We tried to count number of fingers in image using ConvexHull and ConvexityDefects, but the result was not accurate since false points appeared and we cannot eliminate them completely.

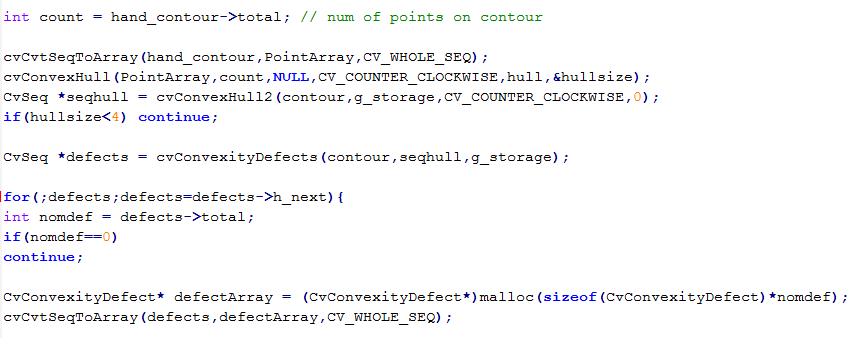


Figure: Convex Hull and Convexity Defects.

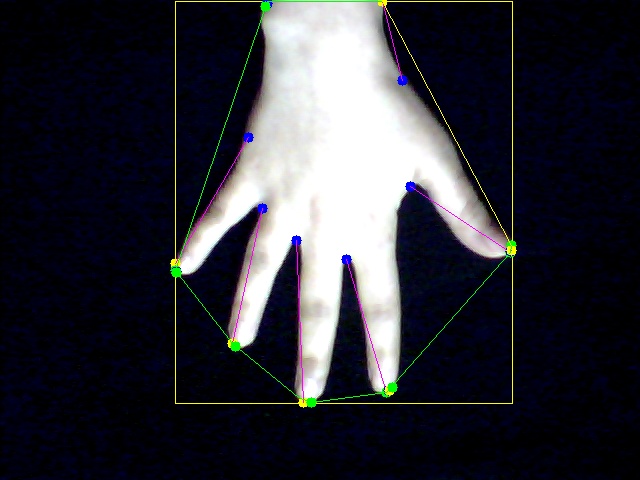


Figure 4 : Convex Hull and Convexity Defects of hand.

Solution: We solved both problems by implementing our own way of recognition that takes special parts in binary image and applies some calculations to determine the gesture without needing any reference text or images stored previously to compare our results to, this point makes our recognition way efficient, fast and differs from other applications.

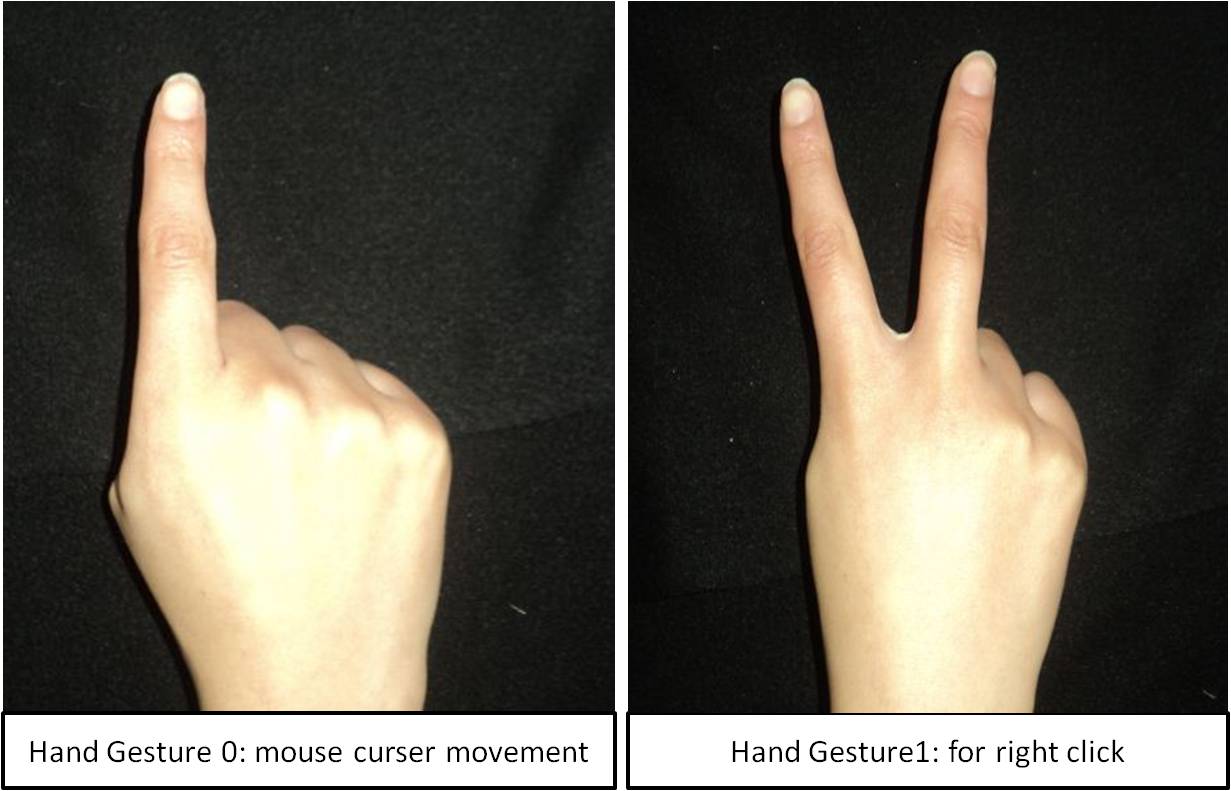
**6. Event Generation**

**6.1 Virtual Mouse Application:**

Depending on the gesture the cursor may be moved or not and the appropriate event will be generated.

Table 1: hand gestures and their related mouse actions.

|  |  |
| --- | --- |
| Hand Gesture | mouse action |
| 0 | Move Mouse |
| 1 | Right click |
| 2 | Single click |
| 3 | Hand  (scrolling up/down, show/hide desktop) |



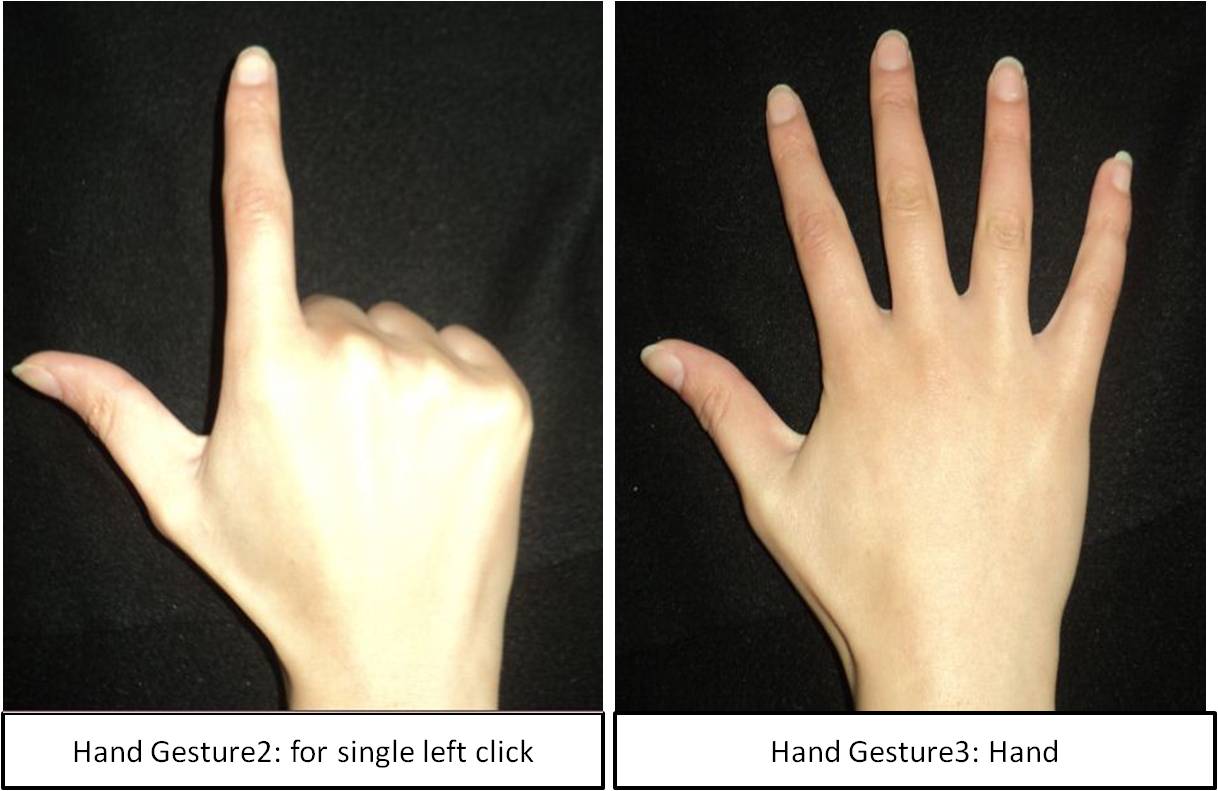


Figure 5: hand gestures and their related mouse actions.

* Case “mouse Move”:
* Find the fingertip position and scale it to screen position, also get the current cursor position.
* Check if mouse left key is pressed, release it.
* Check if the previous gesture is single click set flag indicate single->mouse gestures sequence occurred , this flag is needed to perform double click later.
* Check if single click->mouse->single click sequence was occurred then double click event will be generated now on the current cursor position.
* Check if previous gesture was mouse then mouse cursor will be moved now, mouse will be moved from current position to fingertip scaled position.
* Update previous gesture to mouse.
* Case “right click”:
* Get current cursor position
* press right mouse key down
* Wait for very short period.
* Release right mouse key.
* Update previous gesture to right click.
* Case “single click”:
* Find the fingertip position and scale it to screen position, also get the current cursor position.
* If left mouse key is not pressed single click->mouse sequence is checked
* if it is true single click->mouse->single click sequence flag is set; in order to be checked later in mouse gesture to perform double click or not.
* Else left mouse key will be pressed.
* Else perform drag and drop event by moving mouse cursor from current position to fingertip scaled position.
* Update previous gesture to single click.
* Case “full hand’’ (scrolling up/down, show/hide desktop):
* If user move his hand to right or left show/hide desktop event will be generated.
* If user moves his hand up/down scrolling cursor event will be generated.
* Update previous gesture to full hand.
* Problems that we faced in this stage:

Problem: deciding the sequence for double click, there was a conflict between it and the drag and drop action, since both needed single click gesture, in this stage if the user makes single click, drag and drop action starts.

Solution: We solved this problem by starting the drag/drop action after 3 single click gestures, and double click sequence as follow mouse->single click->mouse->single click->mouse then double click event is generated.

**6.2 Virtual Piano application**

User puts his both hands separated and closed over dark background below a camera then it can run our application and plays our own virtual piano which includes do, re, mi, fa, sol, la, si, do, do#, re#, fa#, sol# and la# notes.

* Depending on the gesture the corresponding note will be generated and remains until another correct gesture occurred, by this assumption we can control the length of note that allow user to produce notes in an interesting way such that resulting music will be as real piano .
* Notes will be generated as follow :

Table 2: Left and right hand gestures and their related note.

|  |  |  |  |
| --- | --- | --- | --- |
| Left Hand Gesture | Right Hand Gesture | Note Number | Note Name |
| Gesture 3 | Gesture 3 | 0 | Do |
| Gesture 3 | Gesture 0 | 1 | Re |
| Gesture 3 | Gesture 2 | 2 | Mi |
| Gesture 3 | Gesture 1 | 3 | Fa |
| Gesture 0 | Gesture 3 | 4 | Sol |
| Gesture 0 | Gesture 0 | 5 | La |
| Gesture 0 | Gesture 2 | 6 | Si |
| Gesture 0 | Gesture 1 | 7 | Do |
| Gesture 2 | Gesture 3 | 8 | Do# |
| Gesture 2 | Gesture 0 | 9 | Re# |
| Gesture 2 | Gesture 2 | 10 | Fa# |
| Gesture 2 | Gesture 1 | 11 | Sol# |
| Gesture 1 | Gesture 3 | 12 | La# |

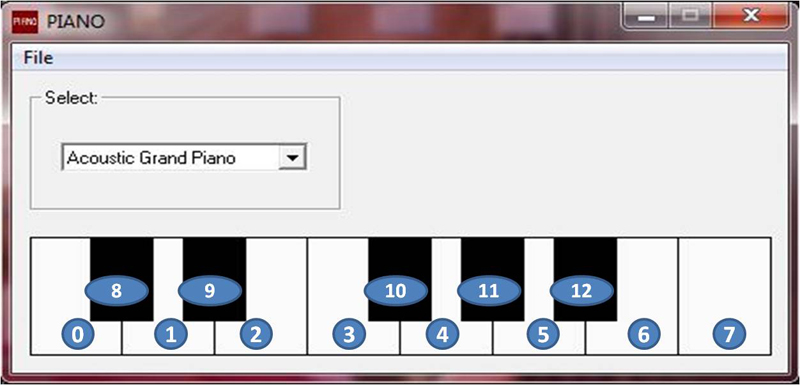
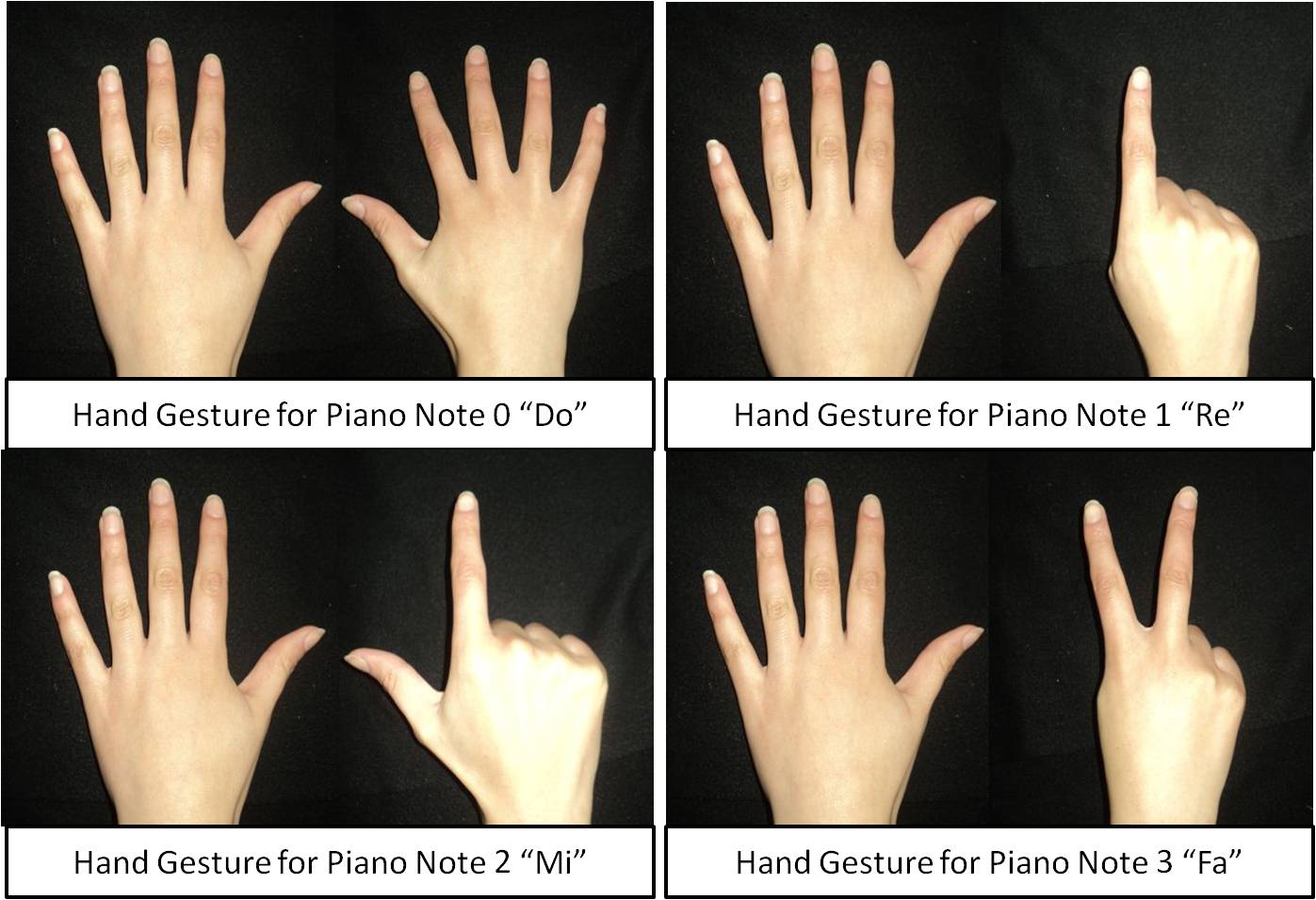
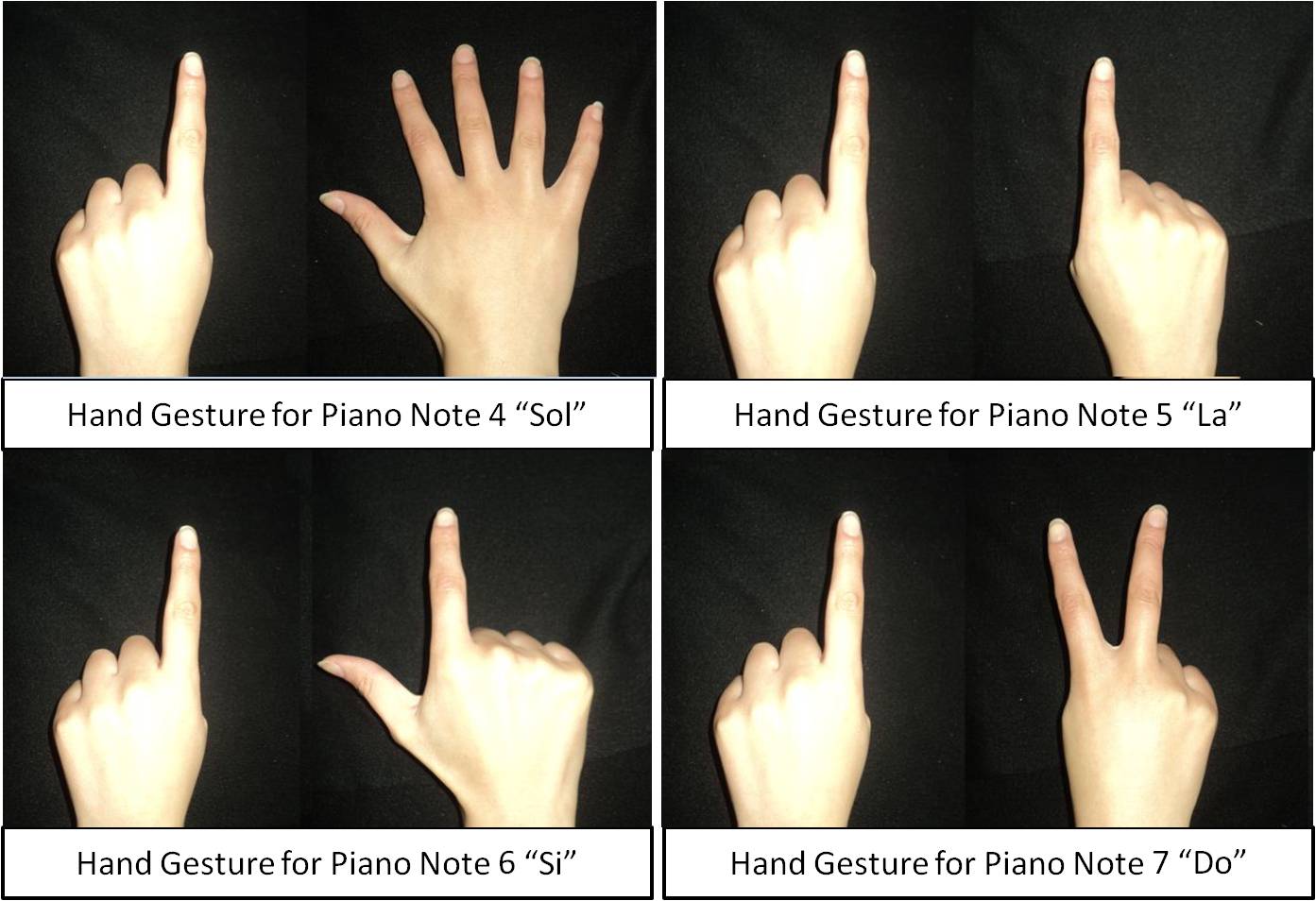


Figure 6: Snapshot for the Piano.

****

****

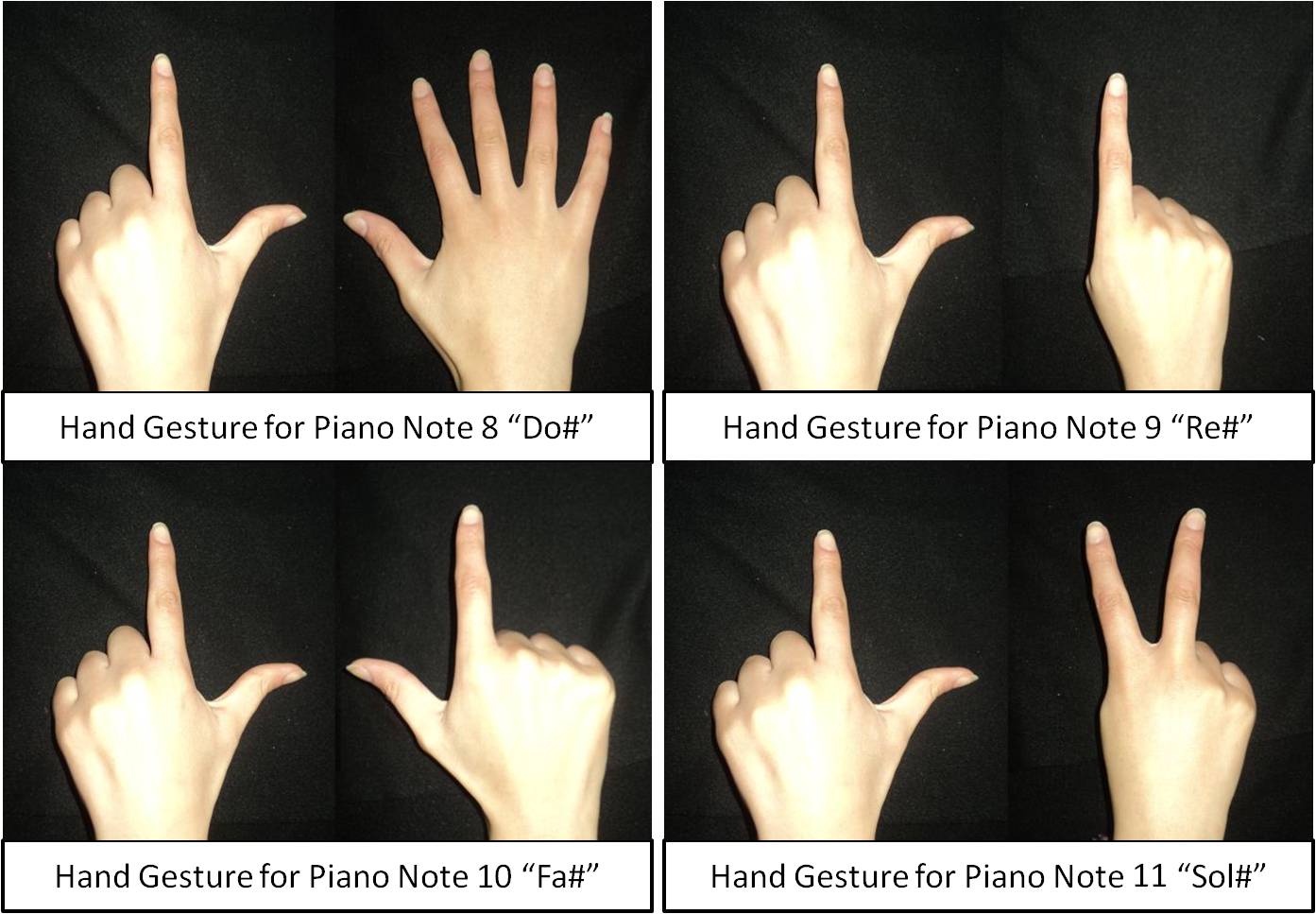
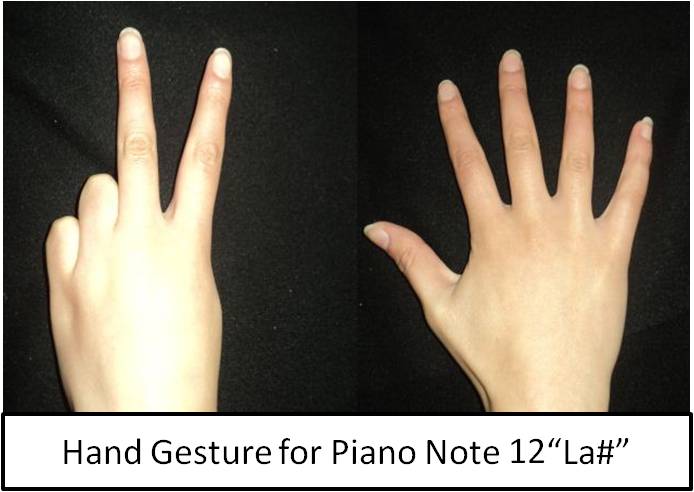
****

Figure 7: Hand Gestures and their related Piano notes.

**6.3**  **2 Players Interactive Game**

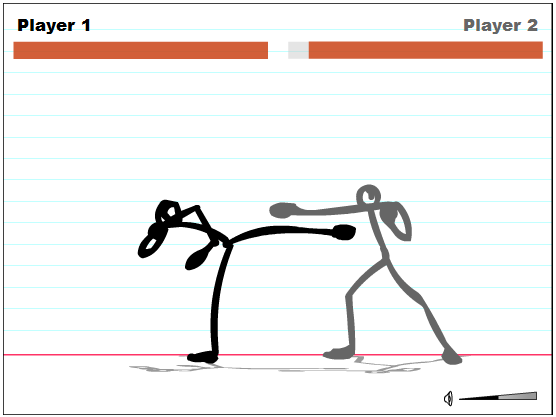


Figure8: Snapshot for the stick fighters 2 player game.

Users put their both right hands separated and closed over dark background below a camera then it can run our application and play the game by using their hands which allow each user to control his player by moving it left or right, fighting by his leg or hand.

Actions generated as follow:

Table 3: Hand gestures and their related game actions.

|  |  |
| --- | --- |
| Hand Gesture | Action |
| 0 | Fighting by hand |
| 1 | Moving right |
| 2 | Moving left |
| 3 | Fighting by leg |

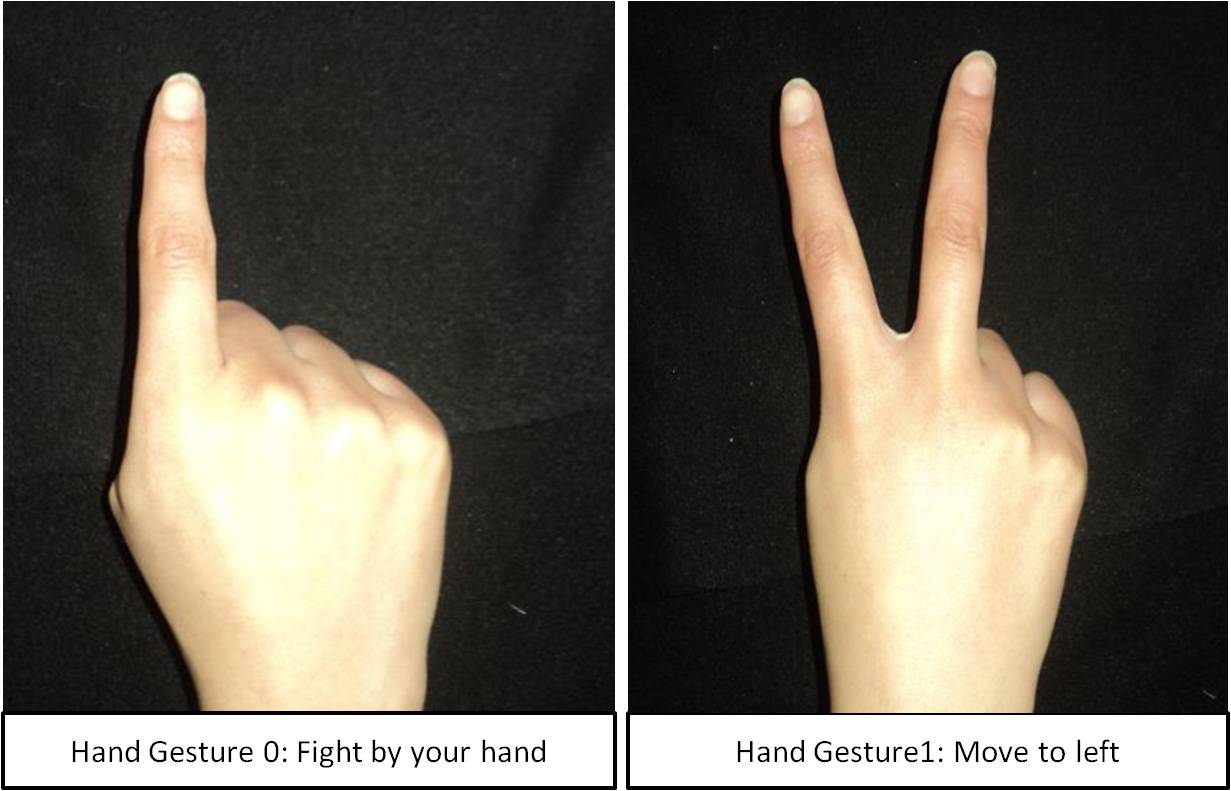
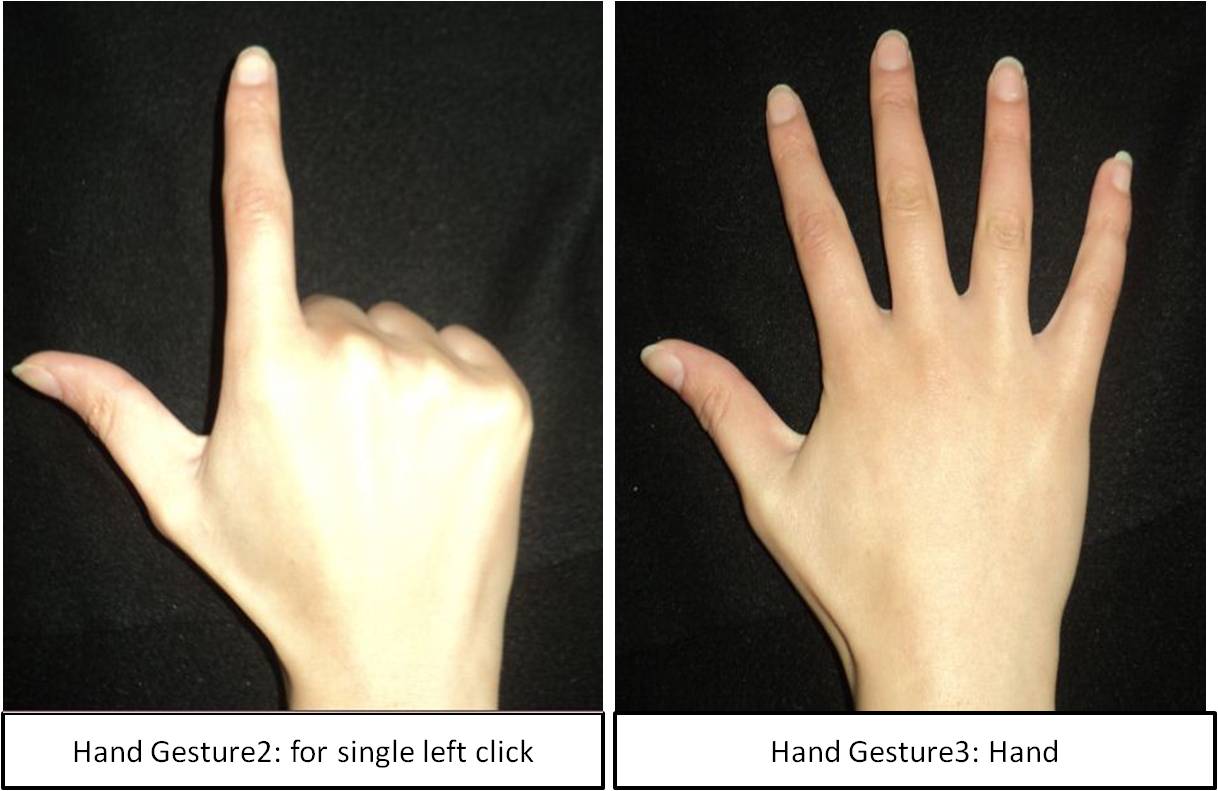


Figure9: Hand Gestures and their related game events .

* Case ‘Fighting by hand’ :
* Palyer 1 :



keyboard key will be pressed



Figure: Keyboard Key ‘/’ Event.

* Player 2 :



keyboard key will be pressed



Figure: Keyboard Key ‘3’ Event.

* Case ‘moving right’ :
* Player 1:



keyboard key will be pressed until the user change the gesture



Figure: Keyboard Key ‘>’ Event.

* Player 2:



keyboard key will be pressed until the user change the gesture



* Case ‘moving left’ :
* Player 1:

keyboard key will be pressed until the user change the gesture





Figure: Keyboard Key ‘<’ Event.

* player 2 :

keyboard key will be pressed until the user change the gesture





Figure: Keyboard Key ‘V’ Event.

* Case ‘Fighting by leg’:
* Player 1:



Keyboard key will be pressed



Figure: Keyboard Key ‘,’ Event.

* Player 2:



Keyboard key will be pressed



Figure: Keyboard Key ‘1’ Event.

**7. CONCLUSION:**

In today’s digitized world, processing speeds have increased dramatically, with computers being advanced to the levels where they can assist humans in complex tasks. Yet, input technologies seem to cause a major bottleneck in performing some of the tasks, under-utilizing the available resources and restricting the expressiveness of application use. Hand Gesture recognition comes to rescue here. Computer Vision methods for hand gesture interfaces must surpass current performance in terms of robustness and speed to achieve interactivity and usability.

**7.1 Project Goals**

The goal of this project was to create a system to recognize a set of gestures oh one hand or both hands at real time, and apply the gestures in order to generate the appropriate events for our applications.

**7.2 Further Work**

Two-handed 3D: It would be possible to detect the gestures by both hands whilst both are in the frame in 3d (using more than one camera). A method would have to be devised to detect a gesture (or range of gestures) that is represented by a partially occluded hand. This method would be considerably harder to implement.  
Because we need to process more than one frame at a time – from more than one camera- to recognize the gestures. We may use these gestures to apply them on the full American sign language.

**8. REFERENCES:**

1. OpenCV library

<http://opencv.willowgarage.com/wiki/>

1. Introduction to programming with openCV

[http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv- intro.html](http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-%20intro.html)

1. OpenCV Tutorial 7

<http://dasl.mem.drexel.edu/~noahKuntz/openCVTut7.html>

1. learning openCV

<http://learningopencv.wordpress.com/>

1. Image processing tutorial-basic concepts

<http://ccdware.com/Files/CCDStack%20Tutorial.pdf>

(6) [Introduction to image processing](http://www.google.ps/url?sa=t&rct=j&q=image%20processing%20&source=web&cd=6&ved=0CF4QFjAF&url=http%3A%2F%2Fwww.spacetelescope.org%2Fstatic%2Fprojects%2Ffits_liberator%2Fimage_processing.pdf&ei=8_XsTq3POozZ8QOaiO2gCQ&usg=AFQjCNFXzgtH1Kl0rZoJ7QARJ0J7ODscfA&cad=rja) <http://www.spacetelescope.org/static/projects/fits_liberator/image_processing.pdf>

# (7) Image Processing and Analysis Reference

# <http://www710.univ-lyon1.fr/~bouakaz/OpenCV-0.9.5/docs/ref/OpenCVRef_ImageProcessing.htm>

(8) C++ MIDI library

<http://www.jdkoftinoff.com/main/Free_Projects/C++_MIDI_Library/>

**9 APPENDIXES**

**9.1** **Recognition regions and values:**

**9.1.1 Regions:**

Upper diag.

Upper

Lower

Left

Right

Lower diag.

Figure: image partitioning in recognition.

**9.1.2 Sequences and Thresholds**

Table 4: Sequences corresponding rows and threshold values used id recognition

|  |  |
| --- | --- |
| **Sequences** | |
| Sequence number | Row number |
| Sequence1 | 75 |
| Sequence11 | 77 |
| Sequence12 | 35 |
| Sequence21 | 82 |
| Sequence22 | 40 |
| Sequence31 | 72 |
| Sequence33 | 43 |
| **Thresholds** | |
| Threshold Number | Threshold Value |
| Threshold1 | 10 |
| Threshold2 | 23 |
| Threshold3 | 10 |
| Threshold4 | 20 |
| Threshold5 | 42 |
| Threshold6 | 3 |
| Threshold7 | 10 |
| Threshold8 | 25 |
| Threshold9 | 10 |
| Threshold10 | 10 |
| Threshold11 | 20 |
| Threshold12 | 10 |

**9.2 Allowable Gestures Deviations:**

* **Gesture 0:**

****  ****

Figure 10: Deviation allowed in gesture 0

* **Gesture 1:**

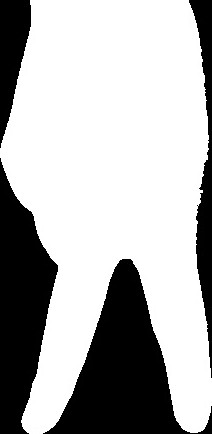
**  **

Figure 11: Deviation allowed in gesture 1

* **Gesture 2:**

**  **

Figure 12: Deviation allowed in gesture 2

* **Gesture 3:**

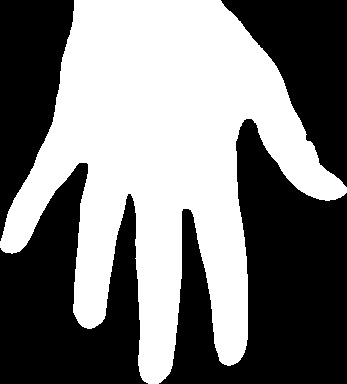
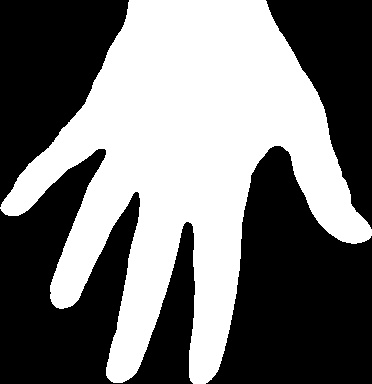
**  **

Figure 13: Deviation allowed in gesture 3