DHITI
APOGEE 2015
FINAL REPORT

COLLEGE NAME: Birla Institute of Technology and Science

TEAM LEADER: MANIT GUPTA

TEAM MEMBERS:

CONTACT: manitgupta95@gmail.com
+91-9649861186

AREA OF THE PROJECT: Open

TITLE OF PROJECT: Virtual Braille

FINAL REPORT

TITLE OF PROJECT: **Virtual Braille**

II. PROBLEM STATEMENT:

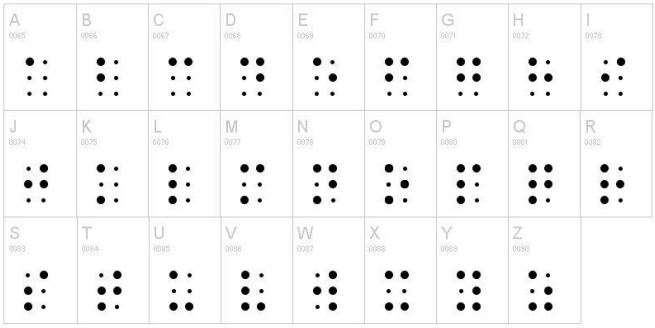
The purpose of our project is to make computers more accessible to the visually challenged section of the society. For the visually impaired, day-to-day interaction with the modern day computers become a challenge. It takes an extraordinary amount of skill and effort for a visually impaired person to become proficient at using a computer. The project aims to improve the communication system between a Visually Challenged Person and a computer.

III. BACKGROUND

Human – Computer Interaction in our project uses nothing but bare hands as Input Media. No sensor, marker or colour is used on hands to segment skin in the images. The Human Gesture Recognition consists of three parts:

1. Hand detection : Module I
2. Finger identification : Module II
3. Gesture recognition. : Module III

The so called "Braille Keyboard" basically consists of 26 hand gestures which accurately map to the Braille characters used in the Braille script as shown:



The first three fingers in each hand i.e. **the Index Finger, the Middle Finger and the Ring Finger** on each hand **represent the 6 dots** used in the Braille Script. Showing appropriate combination of these fingers to Kinect results in gesture recognition and subsequent call to a 'Key Press Event' which types Text onto the Screen.

III. ABSTRACT

1. OBJECTIVE:

A **gesture-based Braille Keyboard** will be developed which will greatly enhance the experience of performing daily tasks on the computer. This would allow them to become more independent, adapt and use the latest technologies in more fulfilling ways. Most importantly, it will also serve as a basis for development of new input systems rather than the traditional Mouse and Keyboard. The above will be achieved using a Kinect sensor for XBOX, and a real-time Human Gesture Recognition (HGR) system which will be implemented using Microsoft Visual Studio 2013. The programming language mainly used will be C#.

2. BENEFICIARIES:

Till now, majority of the aids development for visually impaired people are mere extensions or simple modifications of the existing equipment/technology. In either case, they have to learn how to interact and use the system in contrast to a visually unimpaired person to which interaction mechanisms with a computer come naturally. The project aims to bridge this gap and provide a novel method for interacting with the computer. Since our project uses **Braille Script** for gestures, using it now will not be a difficult task for a visually impaired person since it will come naturally to him.

3. VALUE OF RESULTS:

A lot of active research is going on in the field of Gesture Recognition. Several algorithms have already been developed for Hand Detection using Image Processing Devices and more work is still being done. As far as the cost is concerned, the project does not involve any recurring equipment to be purchased. **We have started active development of the project and have already finished off the first two modules of the project (as stated in description).** The project is progressing strictly as per the conceived timeline.

IV. RESEARCH:

Fingertip and Hand recognition using Kinect without the aid of color segmentation markers is a classic paradigm which has been researched upon by variety for Scholars. Even some research papers have been published:

[1] Hand gesture recognition using Kinect, Yi Li; Comput. Eng. & Comput. Sci., Univ. of Louisville, Louisville, KY, USA

[2] Tracking of Fingertips and Centers of Palm Using KINECT, Raheja, J.L. ; Digital Syst. Group, Machine Vision Lab., Pilani, India ; Chaudhary, A. ; Singal, K.

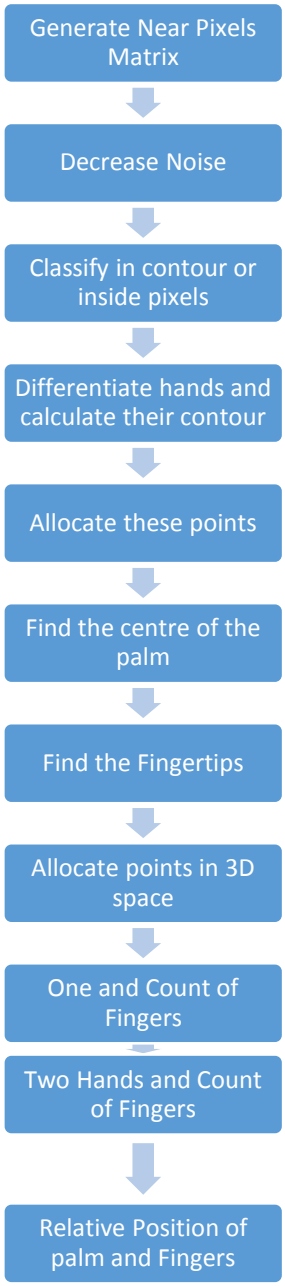
[3] Robust Part-Based Hand Gesture Recognition Using Kinect Sensor, Zhou Ren ; Nanyang Technol. Univ., Singapore, Singapore ; Junsong Yuan ; Jingjing Meng ; Zhengyou Zhang

However, the algorithm which has been used in this application modifies the the existing research to obtain more accurate results for the described problem. Moreover, the idea/objective behind this project is entirely novel and no existing literature uses it for such an application.

V. PROPOSED METHOD (TECHNICAL REPORT):

As soon as the application is launched, the Kinect sensor gets activated and starts detecting the user input by capturing the depth image. At this point, the user should place both his hands in front of the Kinect sensor to start Gesture-Recognition.

For hand-based gesture recognition, the program has to walk through all these steps.



1. Generate near pixels matrix

The first step we need to accomplish in order to start to work with the depth data, is to decide which pixels we are going to take into account, to carry on the tracking. The Kinect can catch the distance of the points which are visible to the camera, between the values minDist and maxDist (Figure 1).

We are going to base our choice using the proximity to the Kinect. We could choose if a pixel is near, by one of this method:

Absolute depth: A pixel is near if its depth is lesser than a constant value that means that the pixel is between minDist and a predefined value which is greater than minDist and lesser than maxDist.

Relative depth: First of all, we calculate the minimum depth, and in base of that depth we select a maximum depth (for example by adding a constant value to the minimum depth). So, if the depth is between these two values, the pixel is near. This method allow us to have greater mobility, because it does not force us to stay in the same position the whole time. In the Figure 1 the minimum depth is d and we add a constant s to it, so every point between minDist and d+s will be accepted as near. After selecting a method, we must select the resolution ofthe depth image. It is possible to select it from three different options 80x60, 320x240 or 640x480. The best option is obviously the 640x480 resolution, but in order to improve the efficiency of the code, we can choose the 320x240 resolution, because it gives us enough definition to distinguish the contour of the hands. This choice greatly reduce the number of operations, and consequently, it will improve the efficiency of the code. Once we have the depth data, we should generate a matrix of the corresponding size, which contains if each pixel is close enough or not, following the criterion selected at the beginning. This matrix will let us to access the data in an easier way, due to many of the algorithms need to know the adjacent pixels of a given one. One final improvement is to set to false the values of the borders of the matrix. This way, we have not to check each time if a matrix access is out of bounds. That will increase the efficiency in the contour algorithm, and we do not lose almost information.

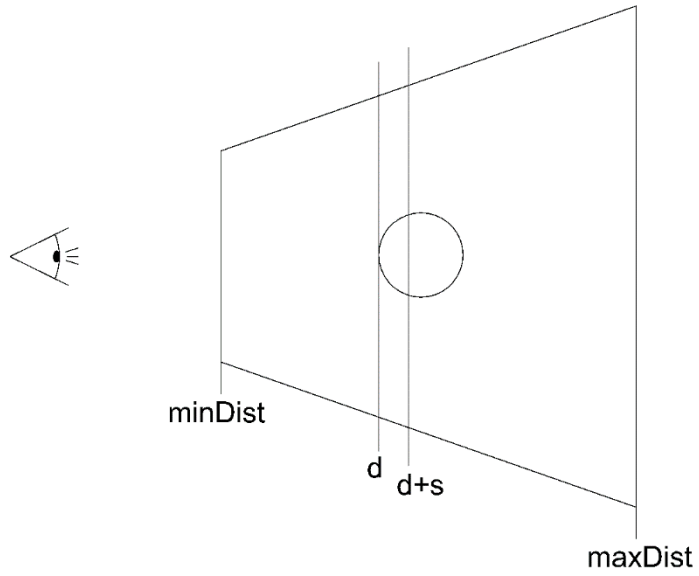


Figure 1: Relative depth method representation

2. Decrease noise

This step is optional, but sometimes could help if we are too far from the Kinect or there too much noise.

The dilation and erosion are used for expanding and contracting the shapes contained in a binary image respectively.

In this case, the hands are these shapes. Both methods apply a mask B over a matrix A. Applying dilation using a circle as the mask will round the contour of the hand, and will decrease the noise. But, if the dilation is too strong we can lose the shape of some fingers, so after the dilation we could apply erosion with the same mask. Dilation is not the opposite operation of erosion, so the final image will contain a rounder shape which preserves the same size of the first one.

These methods are not cost effective, they are only needed in case that the algorithm cannot find the fingertips.

3. Classify in contour or inside pixels

Finally we have a binary matrix which contains the shape of the hands. Each pixel in this matrix points out if the pixel is part of the hand (is valid) or not. The

valid pixels of this matrix must be divided in contour or interior pixels. A pixel is interior if every adjacent pixel (up, down, left and right pixels) are also valid, and contour if one of its adjacent pixels are not valid.

This information should be stored in two different lists, otherwise we have to go over the whole matrix several times, in order to calculate the sorted contour and the palm centre.

4. Differentiate hands and calculate their contour

It seems that the contour is already calculated, but we need a sorted contour for calculating the fingertips and to improve the efficiency of the calculation of the centre of the palm. A sorted contour means that the contour must be a list of points, in which, the previous and next point of each point in the list are adjacent in the contour of the hand, and the first and last point are also adjacent. The contour is calculated applying the "turtle algorithm". Each time a point is added to the contour of one hand, that point is marked as visited. So, after finishing the contour of one hand, the application looks for other contour point to calculate the sorted contour of another hand. If all the contour points has been visited, there are no more hands to analyse. Must take into account that could be some noise that create some small shapes. If the contour is so small compared to the total number of contour points, we must discard it.

5. Allocate interior points

At this point, we have already calculated the number of hands we have, and their contour. The next step is to allocate each interior point into the current hand. One heuristic could be to calculate the container box. The container box of a figure is delimited by two points. These two points delimit the smallest box which contains every pixel of the figure. We can calculate the container box using the contour points, and with the container box we can classify very quickly if a interior point is inside the container box and therefore in the figure. There are some problems when the container boxes overlap, and the same interior point is inside more than one box. In this case, the pixel is wrongly classified.

These kind of errors does not affect the final result, due to the objective of classify the interior points into the correct hand is to find the centre of the palm, and the misclassified pixels can never be the centre of the palm.

6. Find the center of the palm

The center of the palm is one of the points that gives us a lot of information. This point indicates where the hand is, and together with the fingertips, could be used for calculating other relevant information which, for example, could help us to identify gestures. The center of the hand is, usually, the center of the biggest circle that fits inside the contour of the hand. In order to find this point, we calculate the minimum distances from each interior point to every contour point, and from these distances we select the maximum. The point that corresponds to that distance is the center of the hand. This algorithm consume a lot of time, so we must improve its efficiency. We have done it in two ways:

1. While calculating the minimum distance of an interior point to every contour point, if the current minimum is lesser than the current maximum, we can assure that this point is not the center, so there is no need of calculating the distance to the rest of contour points for this interior point.
2. Due to in the list of the interior and contour points, consecutive points are near in the hand, we can only do the calculations for 1 in N consecutive points. Using a small value of N greater than 1, the error is negligible and the efficiency is improve by 1=N2. Actually, with values around 8 the results are quite acceptable.

In the 2obtained from [Ste] we can observe what are we looking for when we use the algorithm. The circle in the figure is the biggest one that the hand shape can contains, so the center of that circle should be the center, and indeed it is the center. Sometimes we could have problems if the arm is showed, and the algorithm allocate the center in the arm.

7. Find the fingertips

The method used to find the fingertips is the k-curvature algorithm. The main idea is for each point P(i) of the contour, get the P(i-k) and P(i+k) points, and using these points generate two vectors and calculate the minimum angle they form. The vector are formed by P(i-k) - P(i) and P(i+k) - P(i). If the angle is lesser than a value, it is a fingertip. The more robust values found were k=22 and a = 40. One of the problems of this algorithm is that it could produce some false positive in the valley between two fingers, because they have similar properties to the fingers. To avoid this false positive, we calculate the distance between the center of the hand an two points, P(i) and the middle point of P(i-k) and P(i+k). If the distance is greater in the first case, it is fingertip, in other case is valley. Moreover to improve the efficiency we introduce a heuristic. Each time we deduce that P(i) is a fingertip, the nextpoint we should deal with is not P(i+1), it should be P(i+m), because fingertips are not near to each other. The value of m depends on the size of the contour of the hand, around a 10% of the number of points of the contour seems to be reasonable. We have two cases, the first one try to calculate the angle a that is formed by the lines P(i+k) - P(i)

and P(i-k) - P(i). In this case the angle is around 30º, so the point P(i) it could be a fingertip.

But, in the second case we calculate the angle b using he lines P(x+k) - P(x) and P(x-k) - P(x), and the resulting angle is equal to 180º more or less, so the point P(x) cannot a fingertip. The value of k in this sample is too small, and we can find more points than we should, so the idea is to find the value k that fits better with the right side of the figure.

8. Allocate points in a 3D space

Everything explained above is used for hand and finger recognition in 2D space. But how can we extend these 2D points into 3D? The solution is really simple using Kinect, due to we have the depth of each 2D point, we can normalize the values and add the depth data to each point. This way we have the fingertips and the center of the palm located in a 3D space. To normalize the points we have decided to use the same normalization that Kinect uses for skeleton tracking. The data we obtain from each point is a X and Y value contained in the box whose left-upper corner is (0, 0), and the right-down corner is (width, height). The Z value is between 400 and 8000, the vision space the Kinect can catch measured in centimeters, although Kinect is not too accurate for points further than 4 meters.

9. One Hand and Count of Fingers

Fig 1: The Backspace Gesture

In case only hand is detected, irrespective of the number of fingers shown, the application prompts the user to “Show Both Hands” using audio output. Once both hands are placed in front of Kinect, the user may use the appropriate symbols to type and edit (Fig. 1).

10. Two Hands and Count of Fingers

Firstly, the hands detected are assigned to two variables of a List (data type), which hold the coordinates of the centre of the palm. In case, the Y-coordinate of Hands[0] is greater than Hands[1], they are swapped. This ensures that Hands[0] always holds the coordinates of the Left-hand and Hands[1] hold the coordinates of Right-Hand.

Next, the number of fingers on each hand is detected. Based upon the number of Fingers, the control of the program accesses different portions of the code, for

eg, when the user shows 1 finger on each hand, only the part of code which stores the gestures for C, E or I is used. This ensures fast gesture recognition.

11. Relative Position of Palms and Fingers

After the numbers of fingers have been detected, the program moves onto compare the relative position of each finger. This is done by calculating the pixel distance between successive fingertips recognized. When these values fall into the thresholded data stored inside the application for each gesture, the Keys BOT is called. This *initiates* a key press event which writes text on the screen. Finally, the alphabet corresponding to the detected gesture is passed to the TTS engine.

12. Experiments with the Prototype/Workability

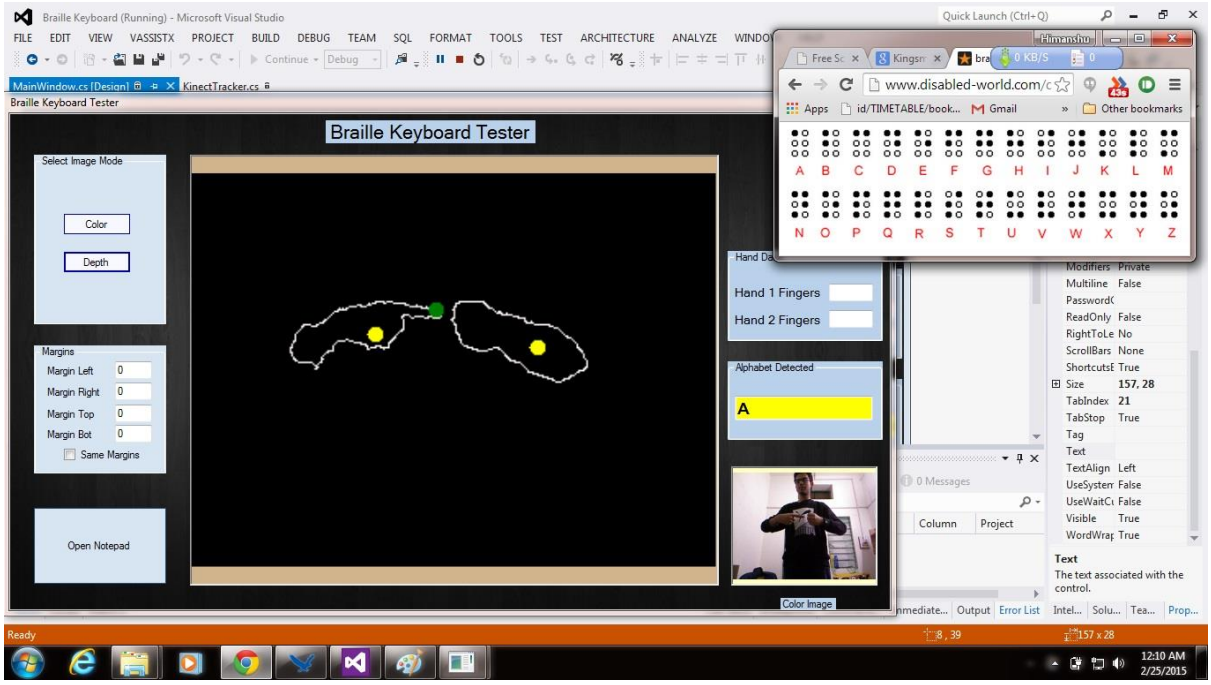


Fig 2: The Alphabet ‘A’

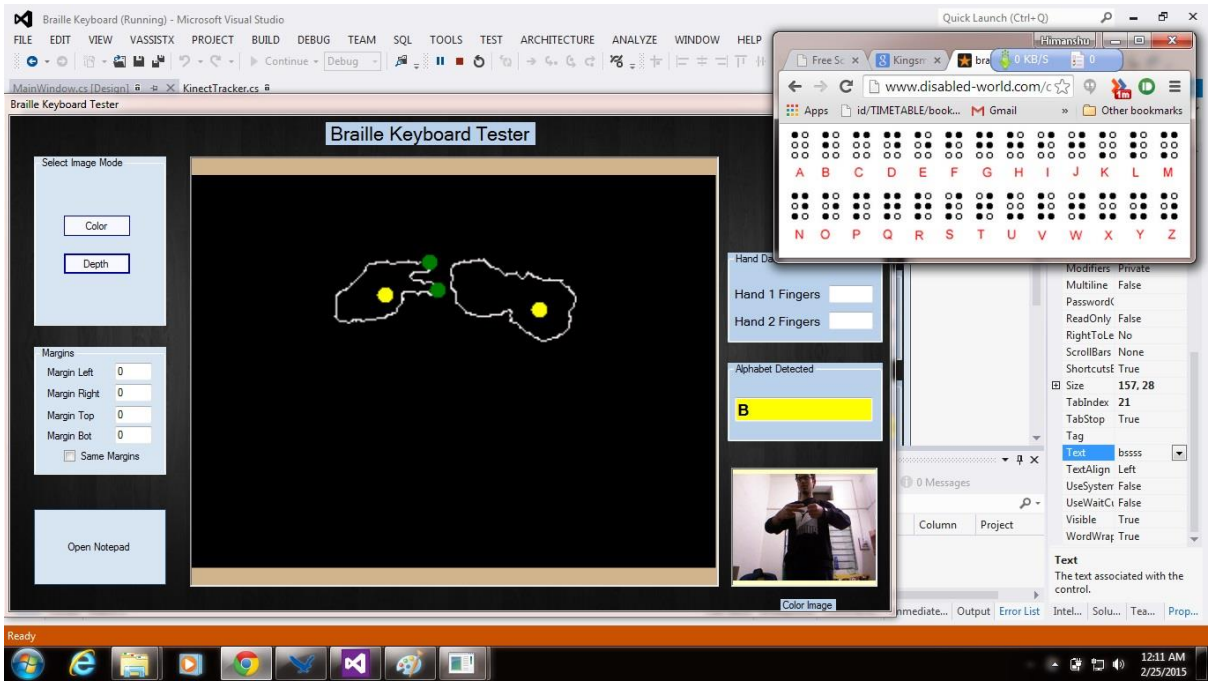


Fig 3: The Alphabet 'B'

Video LINK of the Project DEMO:

http://youtu.be/5zy1_OcHMzI

----------END OF REPORT----------