GESTURE BASED CONTROL

A PROJECT REPORT

Submitted by

ARVIND M

DEEPAK S

DEEPAK S

KARTHIKRAJA S

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

COLLEGE OF ENGINEERING GUINDY

ANNA UNIVERSITY :: CHENNAI 600 025

APRIL 2016

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**GESTURE BASED CONTROL"** is the bonafide work of "**ARVIND M (2012105510), DEEPAK S (2012105515), DEEPAK S (2012105516) and KARTHIKRAJA S (2012105541)"** who carried out the project work under my supervision.

SIGNATURE

Dr. S. MUTTAN,
HEAD OF THE DEPARTMENT,
Department of Electronics and Communication,
College of Engineering Guindy,
Anna University,
Chennai – 600 025.

SIGNATURE

MR. K. PRAVEEN,
ASSISTANT
PROFESSOR (Sr. Gr.)
Department of Electronics and Communication,
College of Engineering Guindy,
Anna University,
Chennai – 600 025.

ABSTRACT

The main objective of this project is to create a virtual projection of the operating system which could be controlled by means of gestures. The gestures one could use for command could be anything ranging from a winking of an eye to the movement of the whole body. In this project, only hand gestures are used for controlling.

By means of a portable projector the screen display is projected on a plain background which provides the visual interaction with human beings and a camera for detecting the gestures and mapping it to the corresponding action.

The finger tip of the hand is detected and it is mapped to the mouse cursor present on the projected screen. A GUI window is created which has the functions for Photo capturing, Photo Viewing, Number Pad and OCR for digitalizing printed text.

Hand detection is done using image processing methods like convex hull point detection, convexity defects detection and histogram back projection. In Photo viewing option, the captured images can be viewed and also can be resized. In the Number Pad option, a virtual keypad is projected on which the cursor can be moved and clicked for getting the value. In OCR option, it captures the text and displays along with playing the first YouTube result for the detected text.

This principle can be extended to a variety of applications wherein the portable gadgets can be activated and operated by means of gestures.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 GENERAL

From the beginning, there have been efforts to close the gap between the real world and the virtual as much as possible. One way is to improve the manner in which people interact with the digital world i.e., invent more user-friendly interfaces. The design trends are such that man-to-device interaction is made as close as possible to mimic man-to-man interaction. The early interfaces such as keypads and mouse have now evolved to touchscreens and voice commands. Gesture-based control is a step taken along this path to bring the virtual world into the real world. Gestures have been part of human communication since early ages. The idea behind this project is to make digital devices understand the human gesture and act accordingly.

1.2 OBJECTIVE

The objective of this project is to create a device whose primary user interface is gestures made with the human hand (the five fingers of a hand). The device is portable. The device allows the user to capture images, view the captured images later in the device gallery, a virtual keypad (number pad) and OCR facility to store the text present in the captured images in digital format. The device sees the gestures through a camera. The feed from the camera is sent by a local processor on the device to the server where it is processed and the gestures are identified. The control signals are then sent back to the processor on the device which in turn carries out the appropriate function. The local processor used in the project is Raspberry Pi.

The captured pictures are stored in the local processor. These pictures can be viewed through a Pico-projector with the help of the gallery application. The Pico-projector being handy and portable allows the user to project the display on any surface anywhere the user goes and in any position the user may be in.

The following chapters explain the methods and algorithms used to implement the above mentioned features.

Chapter 2 explains about the algorithm used in hand detection.

Chapter 3 explains about the algorithm used in identifying the gestures made using the fingers of a single hand and mapping these gestures to the appropriate functions.

Chapter 4 explains about the OCR functionality of the device. This function helps in picking out the text from images captured using the devices camera and using the text to surf the web, for example, to watch a video on YouTube.

The general block diagram of the above mentioned functions is shown as in the Fig. 1.1. The implementation of each operation is explained in detail in upcoming chapters.

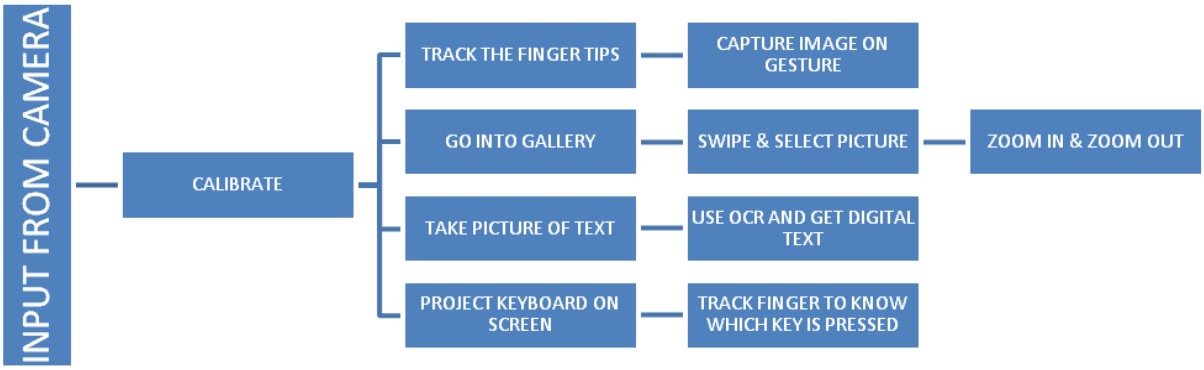

Fig. 1.1 Block Diagram for Different Operations

CHAPTER 2

HAND AND FINGER TIP DETECTION USING IMAGE PROCESSING

2.1 IMAGE PROCESSING

Software:

- OpenCV 3.0
- Python 3.4

2.1.1 General Overview

In general, Image Processing is a technique which is used for analyzing images and videos by storing it in a matrix format. A matrix of an image or a single frame of a video consists of intensity values of the image pixels. The pixels of an image is referred by means of x and y co-ordinates. Thus the matrix can be used for accessing each and every pixel in an image and the necessary transformation is applied on it.

Image Processing is just a technique or a concept which explains about how images and videos can processed. For implementing this there are variety of tools available as open source, among them the most fundamental and dedicated tool for Image processing is OpenCV. OpenCV is an abbreviation for Open Computer Vision, is a tool for Image Processing. The OpenCV tool generally includes library functions of basic image processing techniques like color conversion, image blurring, image transformation by means of thresholding, histogram calculation etc. These functions can be accessed by creating an object for the OpenCV module and using them appropriately.

OpenCV is again just a library file consisting of predefined functions for Image Processing. So to use it, coding languages like C, C++ and scripting

language like Python can be used. Since Python is a scripting language, it is easy to use and write programs for operations, which could be understood with some basic knowledge in programming. Thus in this project OpenCV along Python is used for developing algorithms. The versions of OpenCV used is 3.0 and that of Python is 3.4

2.2 HAND DETECTION

Hardware:

- Camera

2.2.1 Algorithm

The camera module is used to get a live video relay. From that live video a single frame is extracted which is then converted into a gray scale image. Once after that the gray scale image is blurred using Gaussian Blurring function which is available in OpenCV, this is the most efficient blurring function. In this Gaussian Blur function the rows and columns of the matrix are specified as arguments for which a matrix is generated with values that will be used for applying over the image. Then the function makes use of that matrix formed and produces the blurring effect. After this, a thresholding function is applied over the blurred image. The thresholding function then converts the blurred image into a black and white image as shown in the right side of the Fig. 2.1.

Fig. 2.1 Hand Detection using Image Processing

In this method OTSU's thresholding method is used, this algorithm assumes that the image contains two classes of pixels following bi-modal histogram (foreground pixels and background pixels), it then calculates the optimum threshold separating the two classes so that their combined spread (intra-class variance) is minimal, or equivalently (because the sum of pairwise squared distances is constant), so that their inter-class variance is maximal. After this step a binary image (i.e. black and white image) is obtained. For the binary image, the Histogram should be calculated.

The Histogram of an image gives an overall idea about the intensity distribution of the image. By looking into the Histogram of an image, an intuition about contrast, brightness etc. can be obtained. Histogram is nothing but a plot of pixel values in X-axis and its corresponding intensity values in Y-axis as shown in Fig. 2.2. From the Fig. 2.2 it can be seen that Left region of histogram shows the amount of darker pixels in image and right region shows the amount of brighter pixels. From the Histogram, it can be seen that dark region is more than brighter region, and amount of mid tones (pixel values in mid-range, say around 127) are very less.

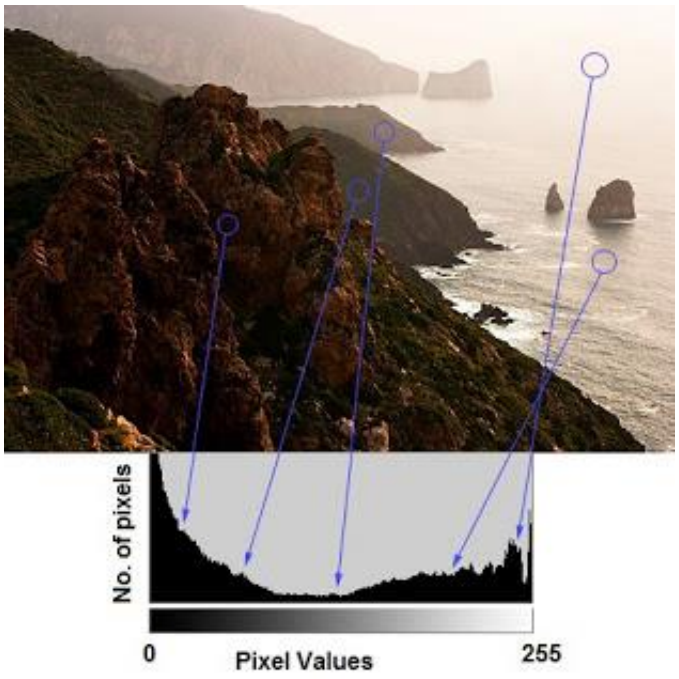

Fig. 2.2 Histogram of a Sample Image

Once after the Histogram is calculated it is normalized. The normalized image's pixel values are used in Histogram Back Projection, which is used for image segmentation or finding objects of interests in the image. In simpler words, the output image will have our object of interest in more white compared to remaining part. Thus the Histogram Back Projected image is taken as the final. After performing all these operations the output will be like the one shown in right side of Fig. 2.1.

2.3 FINGER TIP DETECTION

Software:

- OpenCV 3.0
- Python 3.4

Hardware:

- Camera

2.3.1 Algorithm

The input video feed will be obtained in 680X400 frame size, which is resized to fit to the screen display for fingertip detection purpose. The binary image obtained in the previous stage is now used to find the contours present in it. The contours in the image are obtained by means of OpenCV's contour detection function. Since in an image there may be more than one contour present and in order to extract the hand portion alone in the image, the contour having the largest area is chosen. Once after that the centroid of the largest contour obtained is calculated and stored.

The Convex Hull-points in the image are calculated using the contour obtained in the last step. In Mathematics, the convex hull or convex envelope of a set X of points in the Euclidean plane or Euclidean space is the smallest convex set that contains X. For instance, when X is a bounded subset of the plane, the convex hull may be visualized as the shape enclosed by a rubber band

stretched around X. Formally, the convex hull may be defined as the intersection of all convex sets containing X or as the set of all convex combinations of points in X. In an image sense, the convex hull points are the convex set enclosing the hand region (interested region).

As shown in Fig. 2.3 the red line bounding the hand is convex hull. It is a convex set meaning that if we take two any two points inside the red region and join them to form a line then the line entirely lies inside the set.

Fig. 2.3 Convex Hull Points

Once after the convex hull-points in the image is obtained the convexity defects are calculated. The Convexity Defects identifies those places on the contour that are concave. In the case of the hand, this is the space between fingers. Most importantly, the end points of these convexity defects correspond to fingertip locations. In other words, the convexity defects are the points in the valley like region of the hand as shown in Fig. 2.4. Thus the defect-points in the contour are calculated using OpenCV.

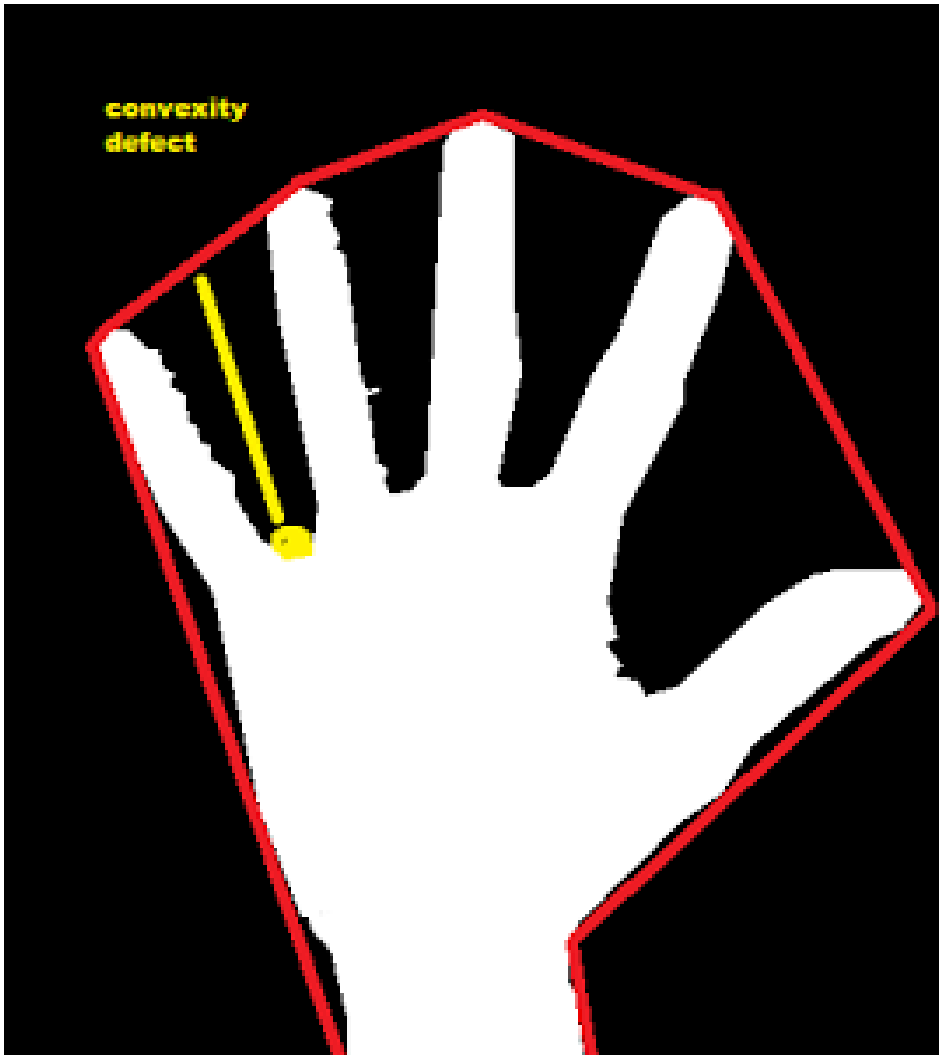

Fig. 2.4 Convexity Defects

The defects points thus calculated are now used in determining the number of fingers shown in the image. From the defect points the Start, End, and Far points are obtained. The Start and End points taken corresponds to a single pair i.e. Start 1 and End 1; Start 2 and End 2 etc., which can be inferred from Fig. 2.5. Thus a single pair of start and end points takes two finger tips. Now three values a, b and c are calculated where 'a' represents the distance between the two finger tips, 'b' represents the length of one finger and 'c' represents the length of another finger. The three values calculated are considered as the three sides of a triangle and the angle between 'b' and 'c' are determined using the following Trigonometric formula,

Angle between two fingers = a*cos((b*b+c*c–a*a)/(2*b*c))*57

Since the angle between any two consecutive fingers will be definitely less than 90 degrees, thus using this condition the number of fingers shown in the image is determined by applying the same formula for each pair of start and end points. This will result in the detection of number of fingers shown in the image.

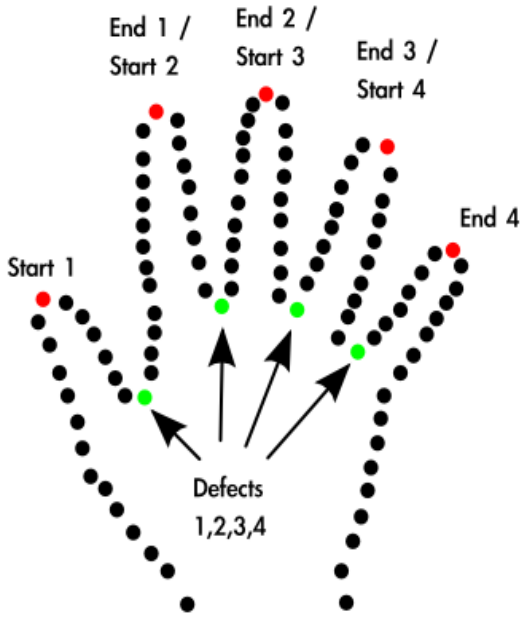

Fig. 2.5 Start and End Points

Each pair of start and end points (either one, since a fingertip is represented by both start point of a pair and end point of next pair) is appended to a temporarily created list. This list now contains the fingertips of the fingers shown at a particular time. Among these finger tips a single fingertip is extracted as follows,

- The distance between the centroid and the first fingertip is calculated.
- The value is stored in a variable.
- The distance between the centroid and the second fingertip is calculated.
- The calculated value is compared with the previous value, if it lesser then it is discarded otherwise the currently calculated value is stored in the variable.
- The above steps are repeated for the remaining finger tips available in the list.

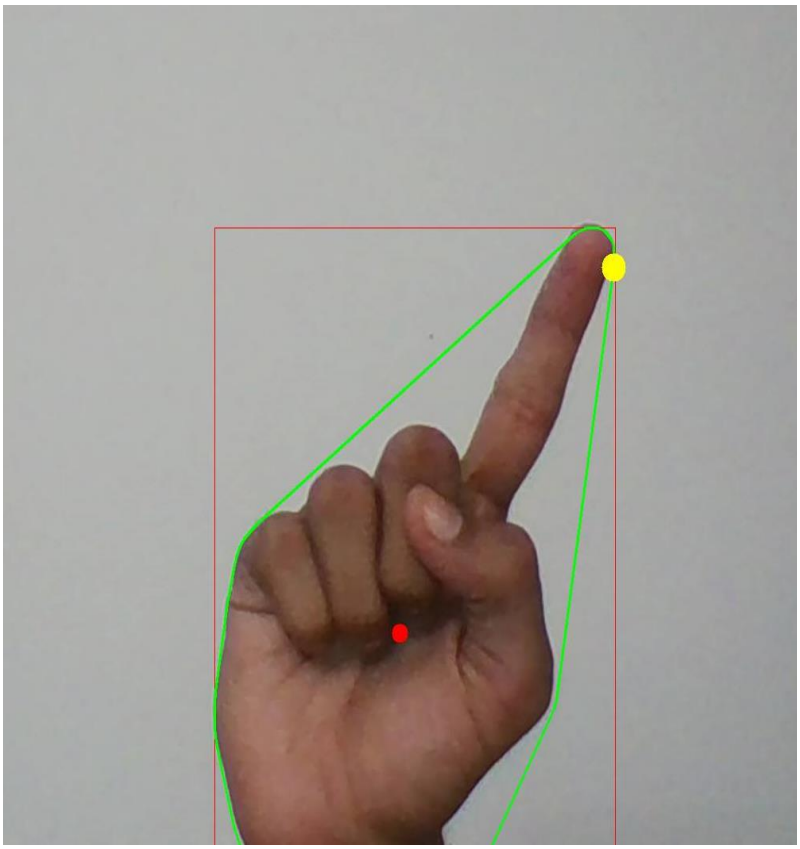

Fig. 2.6 Circular Yellow Dot on Finger Tip

Thus after implementing the above steps the final value held by the variable will be the one that corresponds to the fingertip present at the farthest distance than others. To indicate the position of the fingertip a circular yellow dot is put on screen display as shown in Fig. 2.6.

The position of the fingertip thus detected will be used along with Pyautogui library file for mapping it to the mouse cursor movement action. The implementation of it is explained in next chapter.

CHAPTER 3

GUI INTERFACE AND FINGER TIP MAPPING USING PYAUTOGUI

3.1 PYAUTOGUI

Softwares:

- OpenCV 3.0
- Python 3.4

3.1.1 General Overview

The Pyautogui is a library function available in OpenCV which is used for controlling different operations of the operating system. This library function consists of functions like moving the mouse cursor, making a left click, right click or double click etc. Depending upon the need, the required action is performed. Thus in this project only two functions are used one is for moving the mouse cursor and another one is for performing a click action. This library function can be used by importing and creating an object for it, through which all the functions available under it can be accessed.

3.2 GUI INTERFACE

Softwares:

- OpenCV 3.0
- Python 3.1

3.2.1 Creation of GUI:

A GUI of needed functional operations is designed using OpenCV and Pyautogui modules and libraries. The GUI comprises of a menu with the above mentioned options namely Photo capturing, Photo viewing, Number Pad and

OCR for digitalizing text as shown in Fig. 3.1. Similarly the same library functions are used for creating Number Pad which is used for storing the values entered as shown in Fig. 3.3.

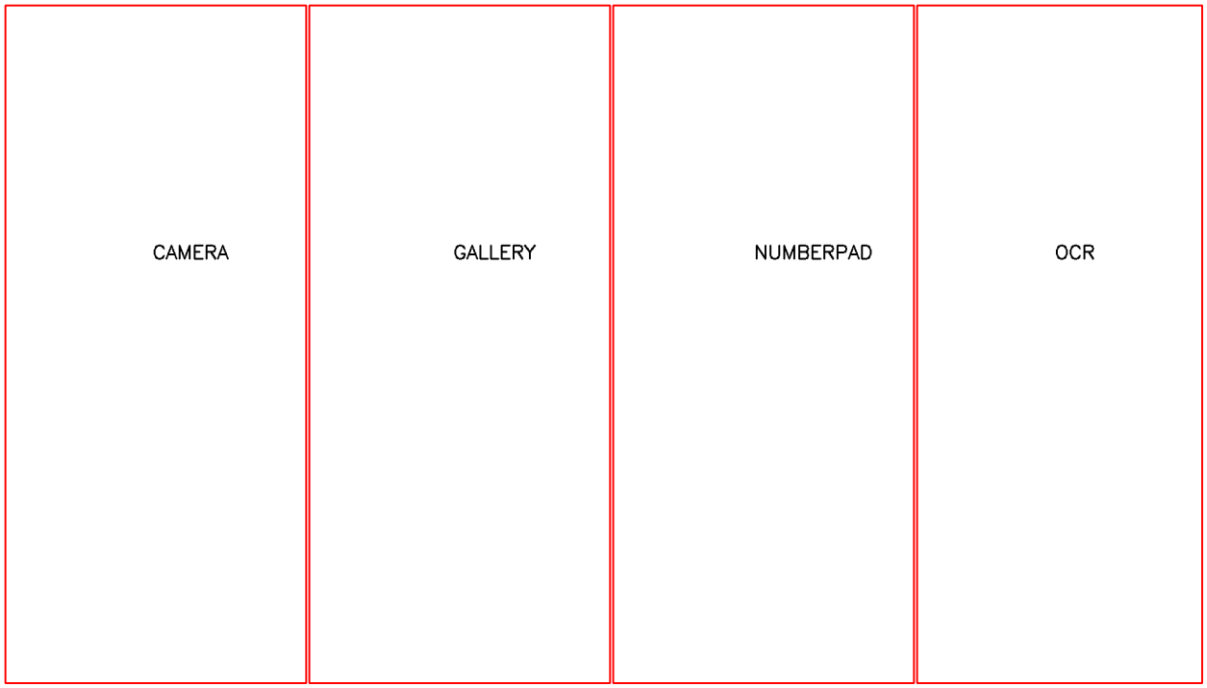

Fig. 3.1 GUI window for different functions

As it can be seen from Fig. 3.2 that the GUI window consists for four different operations, each operation in turn contains functions within it. For accessing a particular operation the mouse cursor is moved over to that and it is clicked. Once after that the functions pertaining to each operations are executed consecutively.

The Camera option is for taking picture, which when clicked runs the program necessary for taking a picture. Once the program finishes, the camera connected to the processor takes a picture and stores it in the Gallery option.

In the Gallery option, the pictures taken can be viewed just be swiping action which will be explained later. In addition to viewing, the image can zoomed in and zoomed out for better viewing.

The Number Pad window will be shown once after clicking on the Number Pad option in the Fig. 3.2. In the Number Pad each and every number will be displayed along with options for back space. The mouse cursor is moved over each Number and clicked on it to get that number. For clicking as said earlier, two fingers are shown.

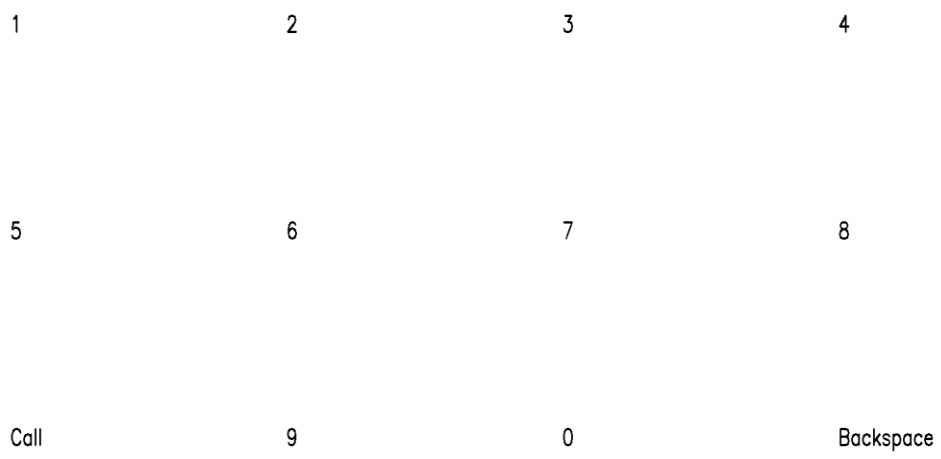

Fig. 3.2 Number Pad window

Finally in the OCR option, similar to camera operation there will be window opened which will be taking pictures of printed text which is then given as input to the Tesseract once after applying the necessary pre-processing techniques to convert the printed text into digitalized format.

3.3 FINGER TIP MAPPING

Softwares:

- OpenCV
- Python 3.4

Hardware:

- Camera

3.3.1 Mapping:

The Pyautogui library function is used for mapping the gesture actions performed. Thus in the previous chapter a single fingertip alone is extracted and its position is stored. This value is now used as an input to the Pyautogui function. The fingertip is used for controlling the mouse cursor movements, thus the position of the fingertip is given as input to the mouse cursor move function, which maps the fingertip movements into mouse cursor movements. As said earlier, instead of moving the mouse cursor every time a circular yellow dot is put on the screen display for indicating the position of the mouse cursor as shown in Fig. 3.1.

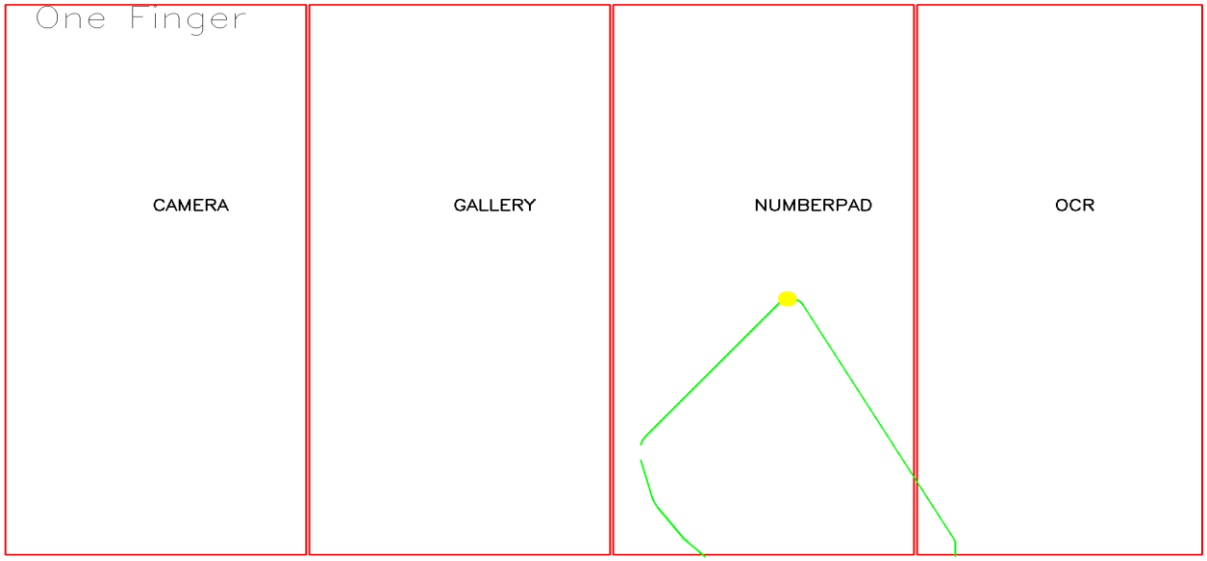

Fig. 3.3 Finger Tip mapping in GUI

In this figure it can be seen that there is a display of how many fingers shown at a particular instant of time.

3.3.2 Controls

There are five different types of controls implemented namely:

- Click action
- Zooming in
- Zooming out
- Swiping
- Closing windows

3.3.3 Gestures

The five different controls mentioned above are implemented by means the gesture action defined for each one.

- Single finger gesture is for Swiping action
- Two finger gesture is for Click action
- Three finger gesture is for Zooming in
- Four finger gesture is for Zooming out
- Five finger gesture is Closing Windows

3.3.4 Mapping of Gestures

Each and every control listed above can be implemented by means of a gesture specified.

In Swipe action, the single fingertip is used, where the mouse cursor is moved in such a manner that it covers larger distance in a short time period. By identifying this movement, it is concluded that a swipe action is performed so that the images stored are scrolled through.

In Click action, the mouse cursor is moved i.e. the circular yellow dot is moved to a desired location and two fingers are shown for performing the click action. Once two fingers are shown, a call to the Click function of the Pyautogui is made which performs the click action.

In Zooming in, the image is zoomed in by showing Three fingers and similarly in Zooming out, the image is zoomed out by showing Four fingers. For closing a currently opened window Five fingers at a time is shown.

In Zooming in and Zooming out the actual size of the image is resized to value greater than the previous value and to value lesser than the previous value respectively. Thus each and every gesture performs its own operation which makes the system to work.

CHAPTER 4

OCR MODULE

4.1 COMPONENTS OF OCR MODULE

The OCR module can be classified into two:

- In-built unit
- Extension unit

The classification is based on the camera hardware used by the raspberry pi. In the case of the in-built unit, the raspberry pi uses the webcam which is connected with the raspberry pi. For the extension unit the camera used is of higher pixel density – this is particularly needed for increasing the accuracy of the complete single page or double page OCR functions.

The software used for the complete OCR module includes:

- Python 3.4
- Tesseract
- Numpy
- Scipy
- OpenCV

The softwares are used to perform transitions from the input obtained from the camera feed to a suitable image which can be fed to the Tesseract for conversion into digital text. This involves the pre-processing steps associated with image processing, to binarize the input images and make them Tesseract readable.

4.2 IN-BUILT UNIT

The in-built unit makes use of the webcam connected to the raspberry pi, the input resolution of the camera feed is 640x480. This is accessed by moving

the finger pointer to the OCR label and clicking. Figure 4.1 shows an image of how the input from the camera looks like.

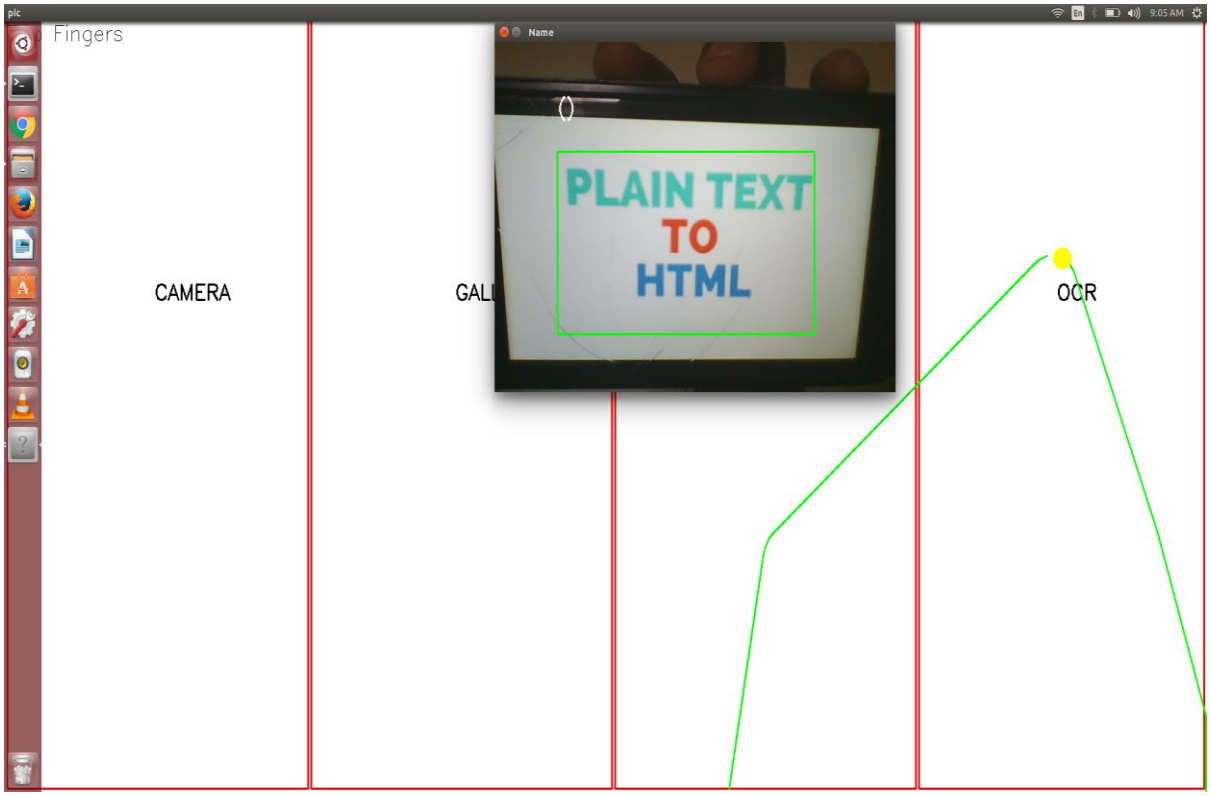

Fig 4.1 Input Feed

From the figure 4.1, it can be inferred that the input feed consists of a green hollow rectangle along with the video feed streamed in by the webcam. The purpose of the green rectangle is to show the limits within which the desired text to be converted has to be placed. There is a click function which enables us to capture the particular frame once the user has positioned it correctly. Once the user clicks the frame is captured and undergoes pre-processing steps. The steps performed include:

- Cropping to region of interest
- Gaussian blurring (reduce noise)
- Local adaptive thresholding (to convert RBG to BW)

After these pre-processing steps are performed, the image looks as shown below in the figure 4.2.

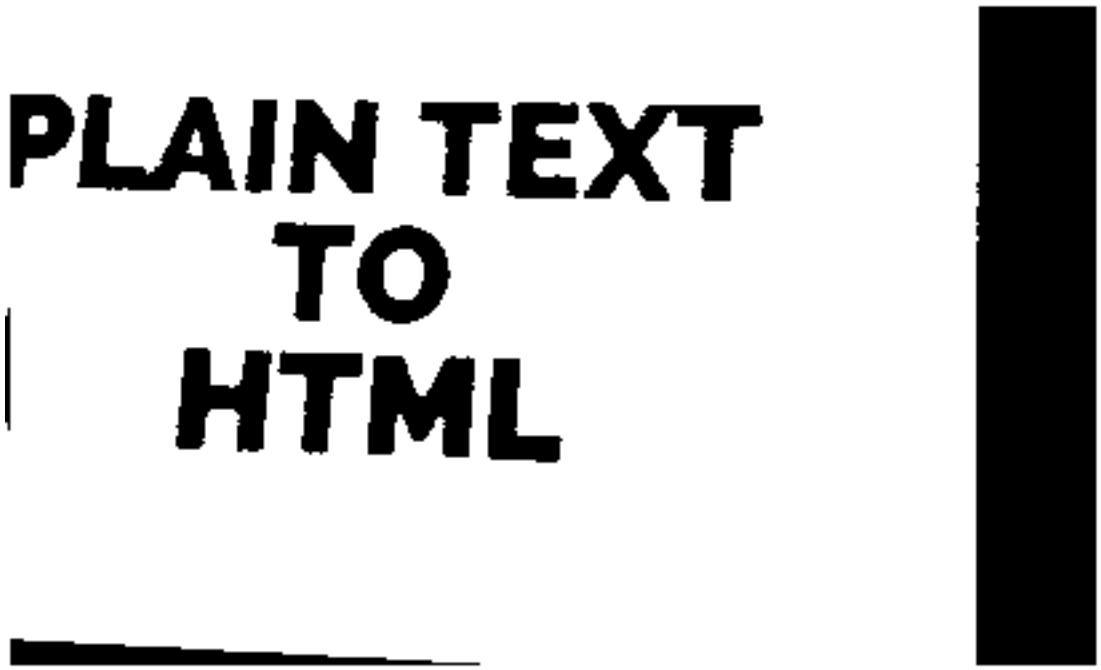

Fig 4.2 Preprocessed Image

It is observed that the words are in black and the background colour is predominantly white. The figure 4.2 shows the standard input that must be passed on to the Tesseract for it to perform OCR with optimal accuracy.

The second part of the in-built unit would be to use the digital text outputted by the Tesseract to perform useful actions. The digital text which is obtained is sent to a python program which surfs the YouTube database and helps to play the first search result which is the most relevant one according to the query given. Thus it provides the ability to simply capture a picture of a text and allows playing a YouTube video relevant to it from the internet. The feature is handy as it is automated and allows the user to access information easily without having to search from the internet manually.

4.3. Extension Unit

The extension unit comprises of two divisions, they are:

- Single page OCR
- Double page OCR

The need for this division is due to the difference in the pre-processing steps associated with them. The single page OCR works simply by identifying the edges of the paper from the background after which perspective changes are applied to get best possible straight image. In the case of double page OCR the approach is radically different as the edges of a book pose a limitation and cannot be approximated into a rectangle contour to perform perspective transform. This case using dilation the regions of text are found and pre-processing is done separately on each region to get text.

4.3.1 Single page OCR

A picture of the page which is to be converted is sent for processing. The steps involved in the pre-processing are as follows:

- RGB to BW conversion
- Canny edge detection
- Contour approximation
- Perspective transform
- Local adaptive thresholding

The output of the pre-processing is a binary image which is fed to the OCR and the text is obtained and stored in a folder.

The image is first converted into BW to speed up the process as the main objective is only to find the edges of the paper. The Gaussian filter is used to smooth the images so the noise is reduced. The canny edge detection performed

helps to extract the edges present in the image. The output of the canny edge detection is as shown in the figure 4.3.

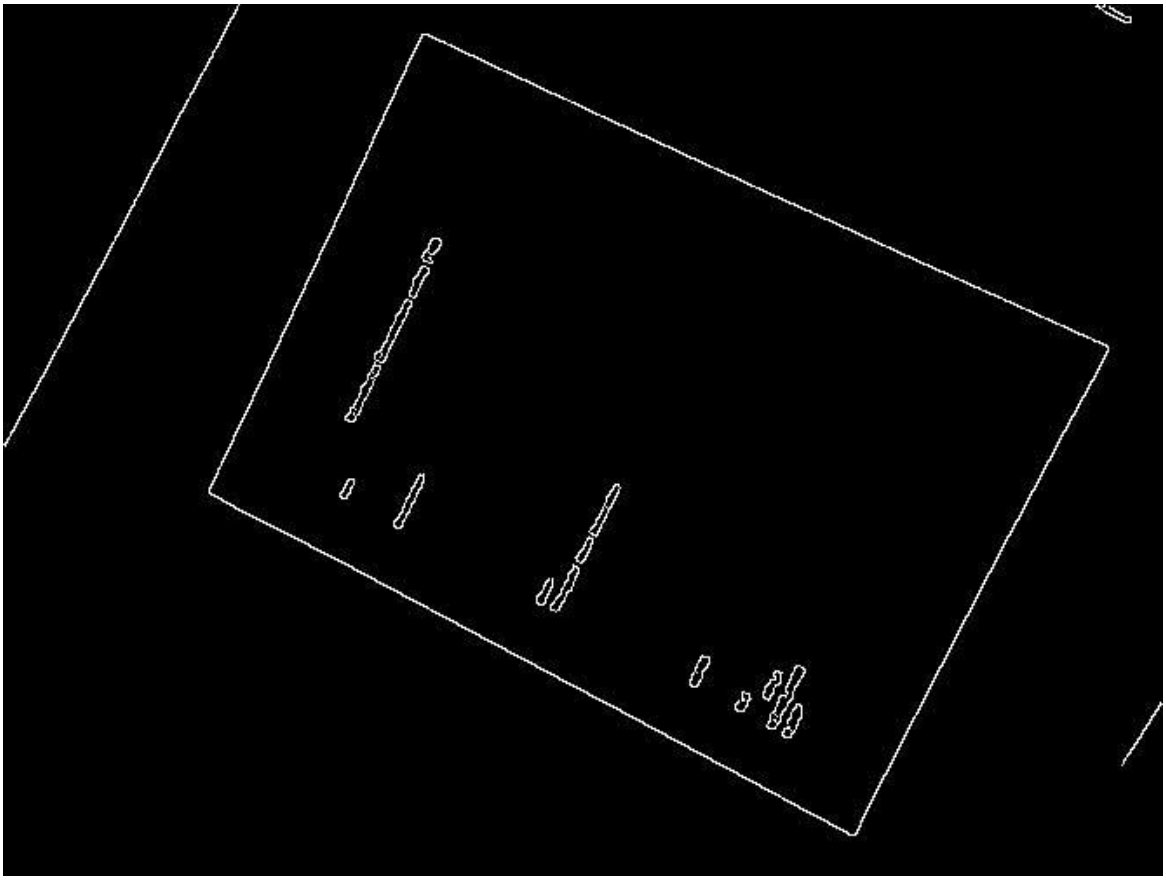

Fig 4.3 Canny Edged Image

The output image is used as the base for finding contours. A contour is a bounded region. Many contours are detected in the image and only the one with the largest area and one which can be approximated into 4 points is chosen. This contour represents the edges of the page. Then based on the angle and translation of the contour perspective transform is performed. The figures 4.4 and 4.5 show the contour and the binarized perspective transform.

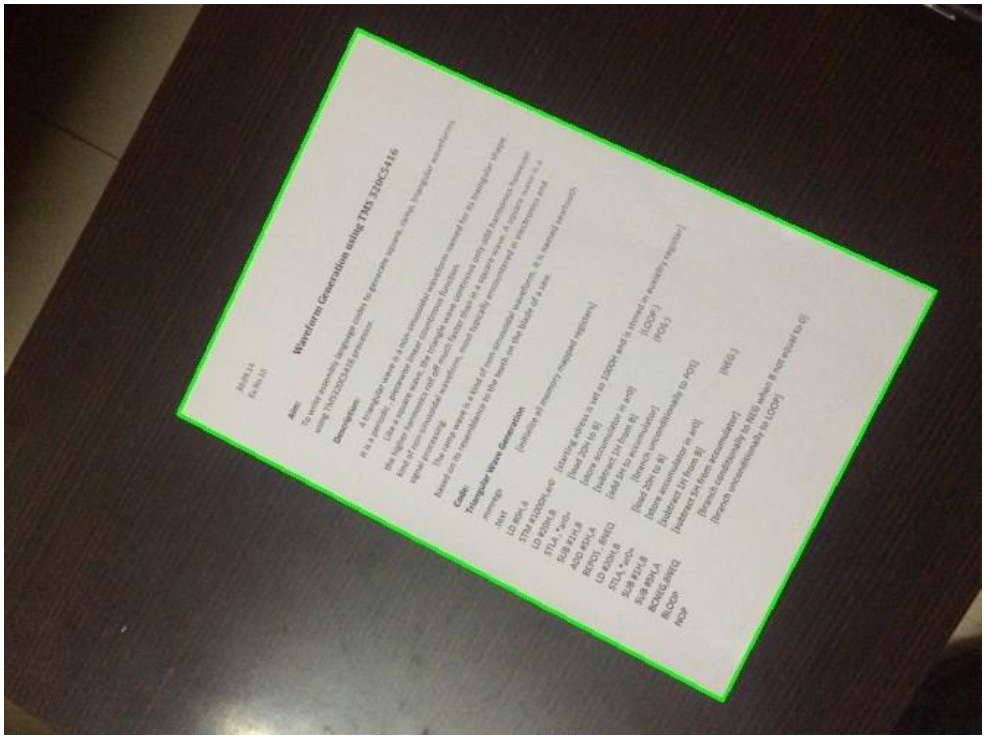

Fig. 4.4 Contour

30.09.14
Ex.No.10

Waveform Generation using TMS 320C5416

Aim:

To write assembly language codes to generate square, ramp, triangular waveforms using TMS320C5416 processor.

Description:

A triangular wave is a non-sinusoidal waveform named for its traingular shape . It is a periodic , piecewise linear countinous function.

Like a square wave, the triangle wave continous only odd harmonics however the higher harmonics roll off much faster than in a square wave. A square wave is a kind of non-sinusoidal waveform, most typically encountered in electronics and signal processing.

The ramp wave is a kind of non-sinusoidal waveform. It is named sawtooth based on its resemblance to the teeth on the blade of a saw.

Code:

Triangular Wave Generation

```
.mmregs          [initialize all memory mapped registers]
.text
LD #0H,A
STM #1000H,ar0   [starting adress is set as 1000H and is stored in auxiallry register]
LD #20H,B        [load 20H to B]                    {LOOP:}
STLA ,*ar0+      [store accumulator in ar0]          {POS:}
SUB #1H,B        [subtract 1H from B]
ADD #5H,A        [add 5H to accumulator]
BEPOS , BNEQ          [branch unconditionally to POS]
LD #20H,B        [load 20H to B]
STLA,*ar0+       [store accumulator in ar0]         {NEG:}
SUB #1H,B        [subtract 1H from B]
SUB #5H,A        [subtract 5H from accumulator]
BCNEG,BNEQ            [branch conditionally to NEG when B not equal to 0]
BLOOP                 [branch unconditionally to LOOP]
NOP
```

Fig. 4.5 Final Image

4.3.1.1 Canny Edge Detection Algorithm

The algorithm runs in 5 separate steps:

1. **Smoothing**: Blurring of the image to remove noise.
2. **Finding gradients**: The edges should be marked where the gradients of the image has large magnitudes.
3. **Non-maximum suppression**: Only local maxima should be marked as edges.
4. **Double thresholding**: Potential edges are determined by thresholding.
5. **Edge tracking by hysteresis**: Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge.

4.3.1.1a Smoothing

It is inevitable that all images taken from a camera will contain some amount of noise. To prevent that noise is mistaken for edges, noise must be reduced. Therefore the image is first smoothed by applying a Gaussian filter. The kernel of a Gaussian filter with a standard deviation of $\sigma = 1.4$ is shown in Equation (4.1).

$$B = \frac{1}{159} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{4.1}$$

4.3.1.1b Finding gradients

The Canny algorithm basically finds edges where the grayscale intensity of the image changes the most. These areas are found by determining gradients of the image. Gradients at each pixel in the smoothed image are determined by applying what is known as the Sobel-operator. First step is to approximate the gradient in the x- and y-direction respectively by applying the kernels shown in

Equation (4.2).

$$K_{GX} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$
$$K_{GY} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (4.2)$$

The gradient magnitudes (also known as the edge strengths) can then be determined as an Euclidean distance measure by applying the law of Pythagoras as shown in Equation (4.3). It is sometimes simplified by applying Manhattan distance measure as shown in Equation (4.4) to reduce the computational complexity. The Euclidean distance measure has been applied to the test image.

$$|G| = \sqrt{G_x^2 + G_y^2} \quad (4.3)$$
$$|G| = |G_x| + |G_y| \quad (4.4)$$

where: Gx and Gy are the gradients in the x- and y-directions respectively.

It is obvious from Figure 3, that an image of the gradient magnitudes often indicate the edges quite clearly. However, the edges are typically broad and thus do not indicate exactly where the edges are. To make it possible to determine this, the direction of the edges must be determined and stored as shown in Equation (4.5).

$$\theta = \arctan\left(\frac{|G_y|}{|G_x|}\right) \quad (4.5)$$

4.3.1.1c Non-maximum suppression

The purpose of this step is to convert the “blurred” edges in the image of the gradient magnitudes to “sharp” edges. Basically this is done by preserving all local maxima in the gradient image, and deleting everything else.

The algorithm is for each pixel in the gradient image:

1. Round the gradient direction θ to nearest 45◦, corresponding to the use of an 8-connected neighbourhood.
2. Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient direction. I.e. if the gradient direction is north (theta = 90◦), compare with the pixels to the north and south.
3. If the edge strength of the current pixel is largest; preserve the value of the edge strength. If not, suppress (i.e. remove) the value.

A simple example of non-maximum suppression is shown in Figure 4.6 Almost all pixels have gradient directions pointing north. They are therefore compared with the pixels above and below. The pixels that turn out to be maximal in this comparison are marked with white borders. All other pixels will be suppressed.

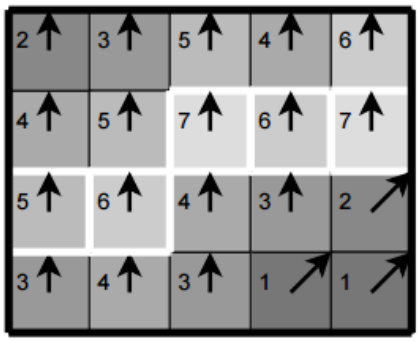

Fig. 4.6 Non-maximum suppression

4.3.1.1d Double thresholding

The edge-pixels remaining after the non-maximum suppression step are (still) marked with their strength pixel-by-pixel. Many of these will probably be true edges in the image, but some may be caused by noise or colour variations for instance due to rough surfaces. The simplest way to discern between these would be to use a threshold, so that only edges stronger that a certain value would be preserved. The Canny edge detection algorithm uses double

thresholding. Edge pixels stronger than the high threshold are marked as strong; edge pixels weaker than the low threshold are suppressed and edge pixels between the two thresholds are marked as weak.

4.3.1.1e Edge tracking by hysteresis

Strong edges are interpreted as "certain edges", and can immediately be included in the final edge image. Weak edges are included if and only if they are connected to strong edges. The logic is of course that noise and other small variations are unlikely to result in a strong edge (with proper adjustment of the threshold levels). Thus strong edges will (almost) only be due to true edges in the original image. The weak edges can either be due to true edges or noise/colour variations. The latter type will probably be distributed independently of edges on the entire image, and thus only a small amount will be located adjacent to strong edges. Weak edges due to true edges are much more likely to be connected directly to strong edges. Edge tracking can be implemented by BLOB-analysis (Binary Large OBject). The edge pixels are divided into connected BLOB's using 8-connected neighbourhood. BLOB's containing at least one strong edge pixel are then preserved, while other BLOB's are suppressed.

4.3.1 1f Necessity for Canny Edge Detection

Compared to other methods of edge detection the canny method detects edges more efficiently even in noisy images and improves the signal and noise ratio at a comparatively better rate.

Advantages

- The smoothing concept has been applied in this Gaussian operation, so the finding of errors is effective by using the probability.
- The next advantage is improving the signal with respect to the noise ratio and this is established by non-maxima suppression method as it results in

one pixel wide ridges as the output.

- The third advantage is better detection of edges especially in noise state with the help of thresholding method.

Disadvantages

- A major disadvantage is the computation of Gradient calculation for generating the angle of suppression. This consumes a lot of time due to complex computation.

4.3.1.2 Perspective transforms

From the contoured image seen in figure 4.4 it is an approximated rectangle contour. The contour values are used to find the 4 vertices of the contour and which of these points contribute to the points in the final image. The top-left, top-right, bottom-left and bottom-right points are obtained by performing simple addition and subtraction.

- The top-left point will have the smallest sum.
- The bottom-right point will have the biggest sum.
- The top-right point will have the smallest difference.
- The bottom-left point will have the largest difference.

Before the final perspective transformed image is obtained the dimensions of the image need to be found out. This is done by using the 4 points obtained above which define the ends of the image. The width is taken as the max of bottom-width (bottom-left to bottom-right) and top-width (top-left to top-right). Similarly the height of the image is taken as the max of the left-height (top-left to bottom-right) and right-height (top-right to bottom-right). The formulas below describe this in detail:

```
widthA = np.sqrt(((br[0] - bl[0]) ** 2) + ((br[1] - bl[1]) ** 2))        (4.6)
```

widthB = np.sqrt(((tr[0] - tl[0]) ** 2) + ((tr[1] - tl[1]) ** 2)) (4.7)

maxWidth = max(int(widthA), int(widthB)) (4.8)

heightA = np.sqrt(((tr[0] - br[0]) ** 2) + ((tr[1] - br[1]) ** 2)) (4.9)

heightB = np.sqrt(((tl[0] - bl[0]) ** 2) + ((tl[1] - bl[1]) ** 2)) (4.10)

maxHeight = max(int(heightA), int(heightB)) (4.11)

where,

bl = bottom-left,

br = bottom-right,

tl = top-left,

tr = top-right.

Once the dimensions for the image are decided, the perspective matrix has to be determined to map the points from the src (source image) to the dst (destination image).The perspective transform does not change the image content but deforms the pixel grid and maps this deformed grid to the destination image. To avoid sampling artifacts, the mapping is done in the reverse order, from destination to the source. That is, for each pixel (x, y) of the destination image, the functions compute coordinates of the corresponding "donor" pixel in the source image and copy the pixel value:

$$dst(x,y)= src(fx(x,y), fy(x,y)) \quad (4.12)$$

Interpolation of pixel values. Usually fx(x,y) and fy(x,y) are floating-point numbers. This means that <fx, fy> can be either an affine or perspective transformation, or radial lens distortion correction, and so on. So, a pixel value at fractional coordinates needs to be retrieved. In the simplest case, the coordinates can be just rounded to the nearest integer coordinates and the corresponding pixel can be used. This is called a nearest-neighbour interpolation. However, a better result can be achieved by using more

sophisticated interpolation methods , where a polynomial function is fit into some neighborhood of the computed pixel (fx(x,y), fy(x,y)) , and then the value of the polynomial at (fx(x,y), fy(x,y)) is taken as the interpolated pixel value.

A perspective transform is calculated from the four pairs of points which represent the vertices of the contour as given below:

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = \texttt{map\_matrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{4.13}$$

$$\texttt{dst}(i) = (x'_i, y'_i), \texttt{src}(i) = (x_i, y_i), i = 0, 1, 2, 3 \tag{4.14}$$

The Equation (4.13) gives us M the map_matrix which helps to wrap the image from the source to the destination image to give a bird's eye of the selected page. For the wrapping function the source image is given as input along with M and the max width and height. This wrapping function gives us the final image which is seen in the figure 4.5.

$$\texttt{dst}(x, y) = \texttt{src}\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right) \tag{4.15}$$

The equation (4.15) is used to transform the perspective transformation to the image. This finally gives us the image on which the adaptive thresholding is performed and it can be fed into the Tesseract for outputting the required digital text.

4.3.1.3 Adaptive thresholding

Adaptive thresholding involves selecting a good greyscale at which to re-quantize an image to a small number of greyscales. Adaptive thresholding algorithms operate either globally or locally. A global thresholding algorithm sets a single threshold for the entire page. A local algorithm varies the threshold

across the page; often by partitioning the page into a set of non-overlapping tiles. The threshold is then held constant within each tile. Using a global value as threshold value is not good in all the conditions where image has different lighting conditions in different areas. In that case, we go for adaptive thresholding. In this, the algorithm calculates the threshold for small regions of the image. So we get different thresholds for different regions of the same image and it gives us better results for images with varying illumination.

Threshold value is the weighted sum of neighborhood values where weights are a Gaussian window. The function transforms a grayscale image to a binary image according to the formulae:

$$\texttt{dst}(x,y) = \begin{cases} \texttt{maxValue} & \text{if } \texttt{src}(x,y) > T(x,y) \\ 0 & \text{otherwise} \end{cases} \qquad \text{Equation (4.16)}$$

Where $T(x,y)$ is a threshold calculated individually for each pixel. The threshold value is a weighted sum (cross-correlation with a Gaussian window) of the neighborhood of minus C. The default sigma (standard deviation) is used for the specified block Size.

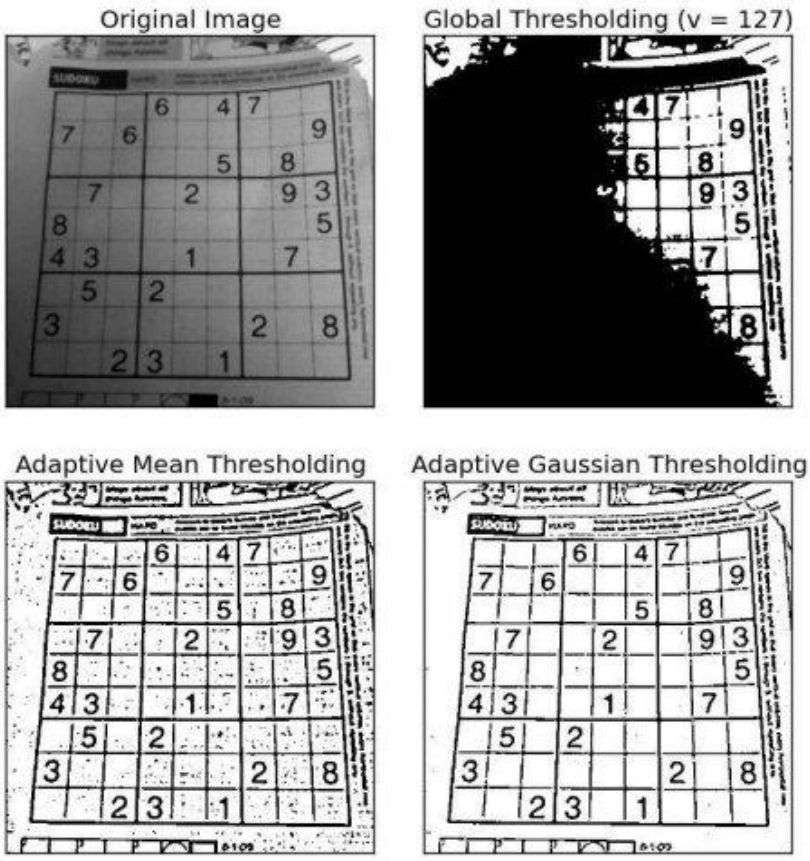

Fig. 4.7 Examples of Thresholding

4.3.1.3a Necessity of Adaptive Thresholding in OCR

Not every document in the office is an immaculate original printed on snow-white paper. The kinds of documents where adaptive thresholding can help are:

- Originals printed with very fine strokes. The thin strokes combine with the limited resolution of the scanner to produce pixels which can easily disappear when thresholded.
- Badly reproduced copies. Lightening of text can have the same effect as thin strokes. Darkening can have a corresponding effect on the gaps between strokes.
- Text printed on a colored or half toned background, or under a coffee stain. Such text can easily be lost by thresholding. Even if the text is retained, allowing halftone dots through can cause havoc with the OCR

system.

4.3.2 Double page OCR

The double page OCR uses the following pre-processing steps:

- Extract region of interest
- Crop to region of interest
- Adaptive thresholding
- Inverse and apply dilation
- Compare areas and choose text regions
- Crop regions and perform OCR

The steps used here are different from the previous one page pre-processing. The limitation is that it's not possible to perform perspective transform. The curved text near the centre of the book makes it hard to extract the text from those curved words. This type of pre-processing is necessary or else the Tesseract will read the lines from both pages continuously and it doesn't make sense. Also this approach works when text is separated in boxes. The different regions are separately binarized and sent to the Tesseract.

The figure 4.8 shows the region of interest from the input image. This is obtained using canny edge filter similar to the previous single page method and it is approximated into a rectangle contour. This contour is extracted and the subsequent pre-processing steps are performed on them.

The figure 4.9 shows the regions of text which are extracted after the inverse and dilation steps of the pre-processing is performed. The figure 4.10 shows the dilation performed on the inverse thresholded image. The regions of text are white and they occupy an area. The area limits are set in such a way that the white regions which are not text are omitted. This makes sure that only text is extracted and noise is not taken into consideration.

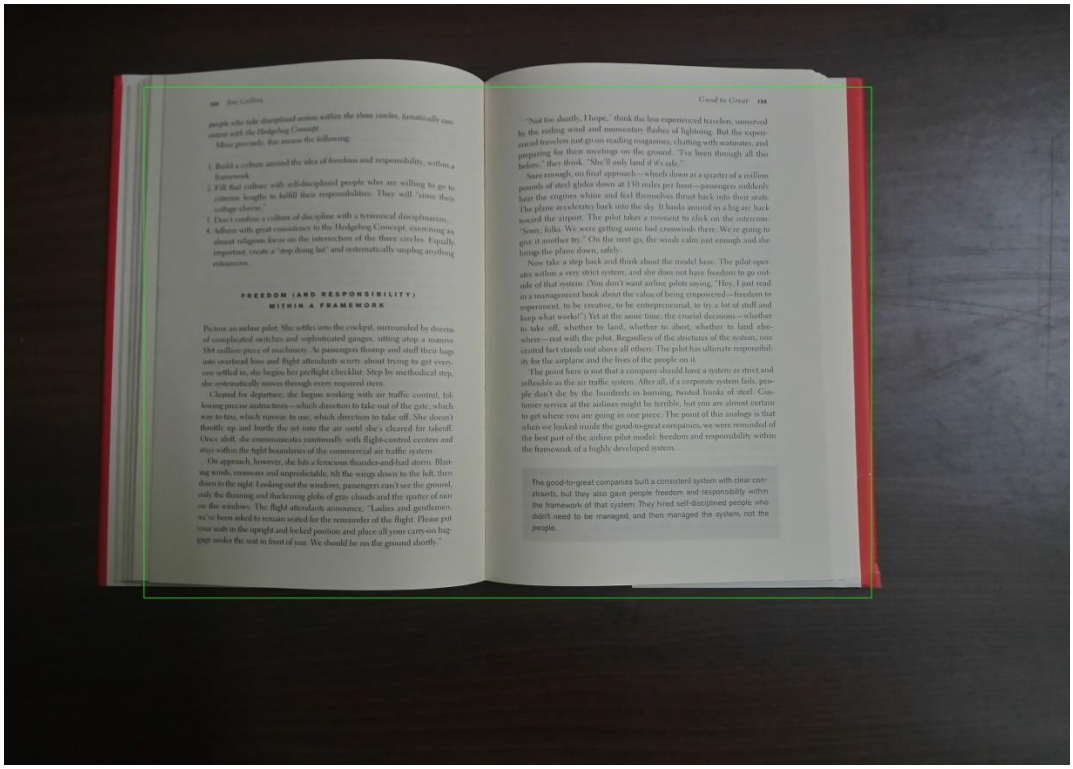

Fig 4.8 Region of Interest of the Input Image

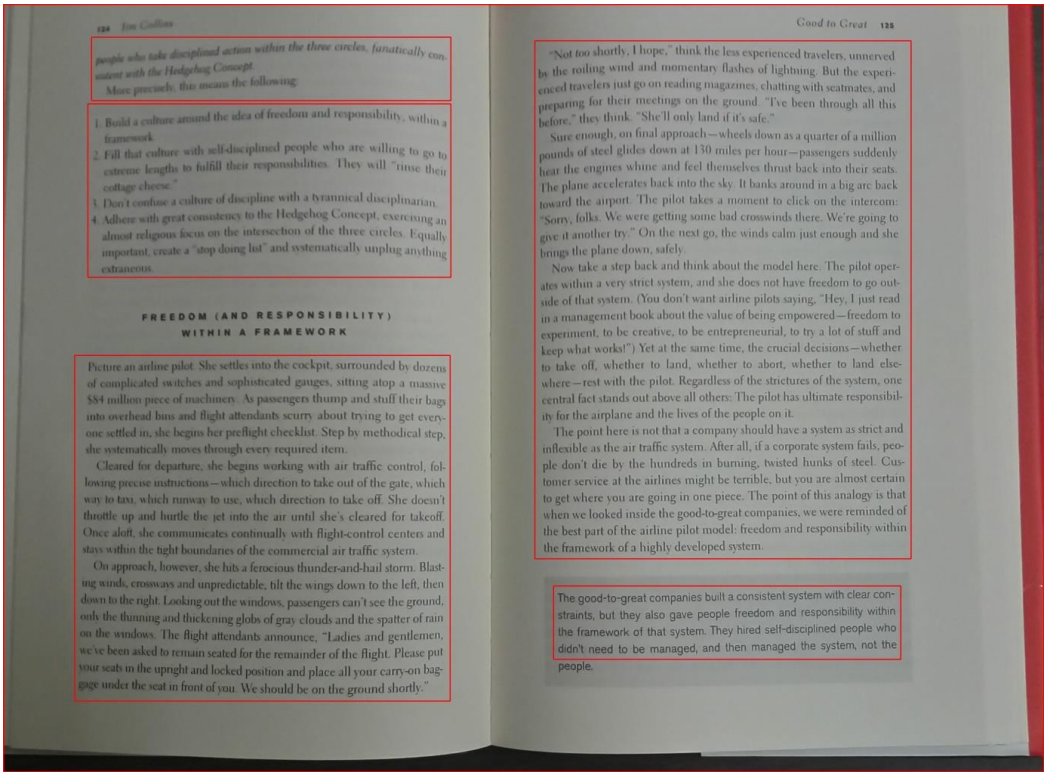

Fig. 4.9 Extracted Regions of text

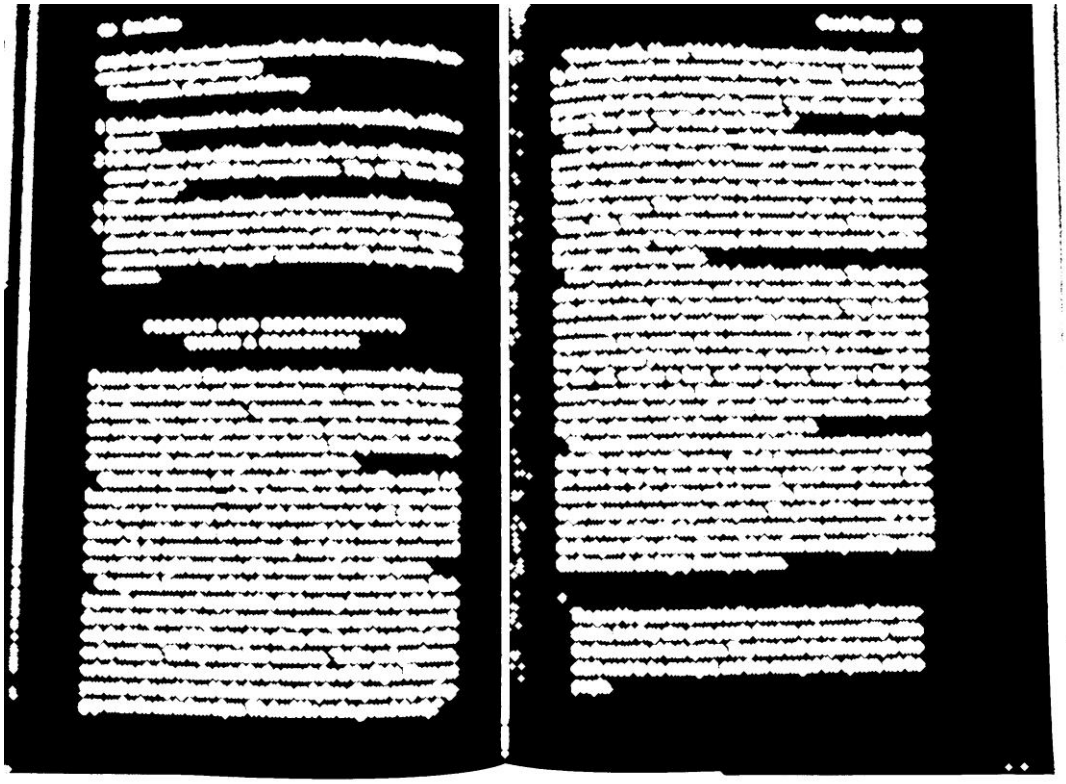

Fig. 4.10 Dilation applied on Inverse Threshold Image

It is observed in figure 4.9 that by setting proper limits the white regions in the centre and the sides are not selected in the final text regions. The limit parameters have some limitations in the sense that under certain conditions the page numbers aren't recognized and very small texts are not included. The separate contours are cropped binarized and sent to the Tesseract for processing. The end result is stored in memory which can be referenced later.

CHAPTER 5

RESULTS AND DISCUSSION

5.1MODULES

5.2 CONCLUSION

The device can be seen as an example how gesture-based control is an efficient user interface. The gesture-based control makes the user interaction with digital devices more user-friendly when compared to traditional hardware keypads and buttons. Gesture based control need not only be used for the above mentioned applications but can be extended to control a lot more devices. The OCR text may be used to in web search. A lot more gestures can be incorporated. The keypad need not be restricted just to the number pad but can be extended to include letters and other special symbols. The concept can also

be made of to control domestic appliances like lights, fans, etc.

Limitations

- The hand detection algorithm works when the hand is over a plain white background with no other object in the field of view of the camera.
- The OCR function can detect large sized fonts with the current camera quality.

Future extensions

- High quality camera can be used to achieve double page OCR detection with small fonts as well.
- Apart from YouTube videos the OCR text can be used to search and view anything on the web.

APPENDIX 1

A1.1 RASPBERRY PI SPECFICATIONS

- Broadcom BCM2835 SoC CPU
- 700MHz Low Power ARM1176JZFS Processor
- 512MB SDRAM
- Boots from Micro SD card, running a version of Linux OS.
- 85x56x17mm Dimensions.
- 10/100 Base-T Ethernet socket
- HDMI A/V output
- Micro USB socket 5V,2A

A1.2 CAMERA SPECIFICATIONS

Focus Range	30 - 150 cm

Focus Type	Fixed
Additional Features	Automatic Image Adjustment With Manual Override
Built In Microphone	Noise cancelling microphone
Connectivity	USB 2.0
Brand	Microsoft
Is HD	Yes
Sensor Type	CMOS
Video Capture Resolution	1280 x 720
Frame Rate	30 fps
Image Capture Resolution	1280 x 800
Memory	1 GB RAM

A1.3 PROJECTOR SPECIFICATION

- Ultra Bright 25 Lumens, HD Resolution
- Mini-HDMI, 16:9 Compatible ratio, Built-in Speakers, Micro SD, A/V connection, USB reader
- Portable 80 Minute built-in battery.
- Up to 60 inch Image.

A1.4 WIRELESS TRANSMITTER

Tenda TE-W311M

W311M is a wireless high gain USB Adapter with wireless transmission rate 3 times faster than 802.11g devices. It connects via USB port to connect the raspberry pi to wireless networks for internet access and file sharing. Capable of speeds up to 150 Mbps and also has 64/128-bit WEP, WPA and WPA2.

- Standard & Protocol IEEE 802.11b/g/n

- Interface USB 2.0
- Antenna 2dBi fixed antenna* 1 (internal PCB);Frequency: 2.4GHz
- Button WPS
- Dimension 38.4mm×17.2mm×7.9mm
- LED 1* Link/Act
- Wireless Speed Up to 150Mbps over 11n
- Frequency 2.4GHZ
- Channel 1~13
- Transmit Power 17dBm (Max)

REFERENCES

[1] http://opencv.org/documentation.html - OpenCV functions that are most commonly used for performing Image Processing.

[2] https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf - Usage of Python syntax with OpenCV library functions.

[3] http://scikit-image.org/

[4] https://pyautogui.readthedocs.org/en/latest/ - Moving and Clicking Operation of the Mouse Cursor.