AIR WRITING AND GESTURE RECOGNITION-THESIS

A THESIS REPORT
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE AWARD OF THE DEGREE
OF
BACHELOR OF TECHNOLOGY
IN
DIVISION OF COMPUTER ENGINEERING

Submitted by:

Ankush Grover (2017UCO1675)
Shorya Kumar Pradeep (2017UCO1680)
Anushka Jalan (2017UCO1696)
Akshay Chaturvedi (2017UCO1690)

Under the supervision of

Dr. Shampa Chakraverty

DIVISION OF COMPUTER ENGINEERING
NETAJI SUBHAS INSTITUTE OF TECHNOLOGY
UNIVERSITY OF DELHI

DECEMBER, 2020

Department of Computer Engineering
University of Delhi
Delhi-110007, India

CERTIFICATE OF ORIGINALITY

We, Ankush Grover (2017UCO1675), Shorya Kumar Pradeep (2017UCO1680), Anushka Jalan (2017UCO1696), and Akshay Chaturvedi (2017UCO1690), of B. Tech. Department of Computer Engineering, hereby declare that the Project-Thesis titled "Air Writing" which is submitted by us to the Department of Computer Engineering, Netaji Subhas Institute of Technology, Delhi (University of Delhi) in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is original and not copied from source without proper citation. The manuscript has been subjected to plagiarism check by Turnitin software. This work has not previously formed the basis for the award of any Degree.

Place: Delhi (Name and signature of student(s)) Date:

Department of Computer Engineering
Netaji Subhas University of Technology
Dwarka, Delhi-110078, India

CERTIFICATE OF DECLARATION

This is to certify that the Project-Thesis titled " Air Writing" which is being submitted by Ankush Grover (2017UCO1675), Shorya Kumar Pradeep (2017UCO1680), Anushka Jalan (2017UCO1696), and Akshay Chaturvedi (2017UCO1690) to the Department of Computer Engineering, Netaji Subhas Institute of Technology (university of Delhi) in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is a record of the thesis work carried out by the students under my supervision and guidance. The content of this thesis, in full or in parts, have not been submitted for any other degree or diploma.

Department of Computer Engineering
Netaji Subhas University of Technology
Dwarka, Delhi-110078, India

ACKNOWLEDGEMENT

Completing a goal is never easy , there will always be obstruction and hardships waiting for us on the way but it's possible to overcome all that but it does require a lot of efforts. The goal can be achieved in the end if one does his/her work with proper mindset ,dedication and obviously with the help of others.
This is to thank all those people who helped in completion of our project without any benefits for them in particular. We would like to extend our sincere thanks to all of them as it is as important as completing the report itself.
We are thankful to Dr. Shampa Chakraverty for her able guidance and useful suggestions, which helped us in completing the project work in time. She made sure our project got completed and kept a thorough check on its integrity. She was the one who inspired us with her dynamic energy and her proficiency.
Our thanks and appreciations also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

Ankush Grover (2017UCO1675)
Shorya Kumar Pradeep (2017UCO1680)
Anushka Jalan (2017UCO1696)
Akshay Chaturvedi (2017UCO1690)

ABSTRACT

In this project we aim to introduce a new way to communicate using air writing. Air writing refers to writing of linguistic characters or words in a free space by hand or finger movements. It is also believed to be one of the easiest and most efficient ways to communicate. In this project we implement air writing and also add the functionalities to write, clear or save using gesture recognition.

The system we have proposed has multiple applications. Our main motivation for this system was to help hyperlexic children and children in general, to learn the alphabet in a more intuitive and interactive way. This system can also be configured for pre existing systems in order to augment their functionalities. For example, an automobile can use this system to give various control options to the passengers in order to eliminate the need of pressing a button or rotating a dial, and make interactions more intriguing.

The rest of the thesis gives all the various details about how air writing with gesture control is implemented and the algorithms used as well.

INDEX

CERTIFICATE 2

CANDIDATE(S) DECLARATION 3

ACKNOWLEDGMENTS 4

ABSTRACT 5

INDEX 6

LIST OF FIGURES 10

CHAPTER 1 **12**
INTRODUCTION

1.1 MOTIVATION 13

1.2 KEY CHALLENGES 13

1.3 PROBLEM ADDRESSED IN THE THESIS 14

1.4 APPROACH TO THE PROBLEM AND ORGANIZATION OF THE THESIS 15

1.4.1 IMPLEMENT AIR WRITING

1.4.2 FIND THE NUMBER OF FINGERS A USER IS HOLDING UP

1.4.3 ORGANIZATION OF THE THESIS

CHAPTER 2 **16**
ALGORITHMS USED

2.1 METHODOLOGY OF PROJECT 16

2.2 IMAGE PROCESSING 16

2.3 IMAGE ENHANCEMENT 17

2.3.1 GAUSSIAN BLUR

2.4 MORPHOLOGICAL PROCESSING 19

2.4.1 EROSION

2.4.2 DILATION

2.5 OBJECT DETECTION AND RECOGNITION 21

2.5.1 COLOR SEPARATION

2.5.2 CONTOUR DETECTION

2.6 GESTURE RECOGNITION 22

2.6.1 COUNT FINGERS METHODOLOGY

2.6.2 CONVEX HULL

2.6.3 EUCLIDEAN DISTANCE

2.6.4 BITWISE AND

CHAPTER 3 **26**

IMPLEMENTATION

3.1 TOOLS USED 26

3.1.1 PYTHON

3.1.2 VISUAL STUDIO CODE

3.1.3 NUMPY

3.1.4 COLLECTIONS

3.1.5 MATHS

3.1.6 LINUX

3.1.7 OPENCV

3.2 DATA STRUCTURES USED 28

3.2.1 DEQUE

3.2.2 TUPLE

3.2.3 ARRAY

3.3 STEPS OF PROGRAM 29

3.3.1 CAPTURING FRAMES

3.3.2 CONVERTING THE FRAME

3.3.3 COLOUR SEPARATION

3.3.4 MORPHOLOGICAL PROCESSING AND IMAGE SMOOTHING

3.3.5 DETECTION OF HAND AND FINGERTIP

3.3.6 GESTURE RECOGNITION

3.3.7 INITIALIZING DEQUEUES

3.3.8 TRACKING AND TRACING

3.3.9 SCREENSHOT

CHAPTER 4 **38**

CONCLUSION AND FUTURE WORK

4.1 RESULTS OBTAINED 38

4.2 STEPS OF PROGRAM 40

4.2.1 EDUCATION PURPOSES

4.2.2 ANDROID APPS

4.2.3 MODES OF COMMUNICATION FOR PHYSICALLY DISABLED

4.2.4 HOME APPLIANCES

4.2.5 AUTONOMOUS VEHICLES

4.3 CONCLUSION 41

4.4 FUTURE WORK 42

4.4.1 RECOGNIZING LINGUISTIC CHARACTERS

4.4.2 PARTIAL ERASE

4.4.3 MORE FEATURES

4.4.4 VOICE COMMANDS

REFERENCES 43

PLAGIARISM CHECK 44

LIST OF FIGURES

Figure Title

Figure 1.1 Output Frame

Figure 2.1 Gaussian Distribution 1-D

Figure 2.2 Gaussian Distribution 2-D

Figure 2.3 Gaussian Distribution Graph

Figure 2.4 Gaussian Matrix

Figure 2.5 Morphological Processing

Figure 2.6 Erosion

Figure 2.7 Dilation

Fig 2.8 (a) Thresholding and Segmenting Hand region - Thresholded image

Fig 2.8 (b) Thresholding and Segmenting Hand region - All Contours

Fig 2.8 (c) Thresholding and Segmenting Hand region - Max Contour

Figure 2.9 Count Fingers Methodology

Figure 2.10 Convex Hull

Figure 2.11 Euclidean Distance

Figure 2.12 Bitwise AND

Figure 3.1 Python Logo

Figure 3.2 VS Code Logo

Figure 3.3 Numpy Logo

Figure 3.4 Linux Logo

Figure 3.5 Workflow

Figure 3.6 Capturing Frames 1

Figure 3.7 Capturing Frames 2

Figure 3.8 Frame Conversion

Figure 3.9 Color Separation

Figure 3.10 Morphological Processing and Image Smoothing

Figure 3.11 Hand and Fingertip Detection

Figure 3.12 count() function 1

Figure 3.13 count() function 2

Figure 3.14 count() function 3

Figure 3.15 Deque Initialization

Figure 3.16 Tracking and Tracing 1

Figure 3.17 Tracking and Tracing 2

Figure 3.18 Screenshot Flag

Figure 3.19 Screenshot Save

Figure 4.1 Frame - One Finger Gesture

Figure 4.2 Frame - Five Fingers Gesture

Figure 4.3 Frame - Two Fingers Gesture

CHAPTER 1: INTRODUCTION

Communication forms the basis of human nature. It involves exchanging ideas, no matter how simple or complex, between two or more people. The earliest form of communication is hand signs and speech. After this communicating through writing came into prominence. It can be clearly seen that a revolution in modes of communication, revolutionizes humankind. In this thesis, we propose a way to communicate using our fingers that requires minimal effort.

The objective of this project is to implement a system which allows the user to effectively draw or write using a fingertip. The system will track the movements of the user's fingertip in free space and display them on the computer screen. Not only this, the system is also able to save the canvas or clear it when the user performs a particular hand gesture.

The main difference between previous projects and this project is that the proposed system does not require any external hardware. Previous projects used sensors and data gloves to perform air writing and gesture recognition. The data glove based methods tracked the fingertip using various intricate sensors to perform air writing, which increased project cost exponentially and narrowed the field for widespread adoption.

Fig 1.1 Output Frame

1.1 MOTIVATION

- **Enable children to learn alphabets more intuitively**. Studies show that physically air writing alphabets creates muscle memory of what each letter 'feels' like, and this in turn helps in cementing the letter in a child's memory. It helps children remember letter formation in a way which actively involves muscle memory and movements. Air writing can also help children with hyperlexia. Hyperlexia is a condition where a child is able to read words very quickly, but struggles to comprehend the meaning of these words. Air writing can help hyperlexic children by creating a visual map of what each letter looks like which in turn increases comprehending ability.

- **Make communication between two or more people easier.** Air writing requires less effort compared to other ways of communication. Individuals can just write their ideas in free space, therefore reducing the time and effort required to communicate. Furthermore, air writing does not require a pen or pencil which improves usability. Also free space is available everywhere, this eliminates all the physical limitations such as lack of paper when writing.

- **Augment already existing digital systems.** Nowadays many vehicles have features which are based on gestures. For example, the radio volume of a car can be increased or decreased using certain hand gestures. The system proposed in this paper can augment these existing systems by giving the user the ability to perform hand gestures in free space. Also, the proposed system can be applied to household appliances. For example, the lights can be dimmed or lit using slight fingertip movements.

1.2 KEY CHALLENGES

- **Finger count matching:** The approach previously used to identify how many fingers the user holds up was based on the OpenCV *matchShapes()* function. The *matchShapes()* function compares two shapes or contours and returns a metric showing the similarity between these two shapes. This resulted in very high time complexity as in every frame of the live video, the *matchShapes()* function was executed. The classification results were also not satisfactory.

- **Hand detection:** In order to perform finger tracing, the first step is to detect the hand. The proposed system used skin based segmentation for hand detection.
  The challenge was to find the proper HSV mask values to detect skin color in environments with varying brightness. Finding the optimal upper and the lower limits for skin color based segmentation took a great number of trials.

- **Finger tracing:** Finger tracing is the action of drawing a trajectory of where the finger moves along a canvas. The number of points that need to be stored for drawing

the trajectory can be very huge and therefore a data structure which supports both fast insertion and deletion of elements must be used. The exact details about the approach used to tackle this challenge are provided in the subsequent chapters.

1.3 PROBLEM ADDRESSED IN THE THESIS

The main problem addressed in the thesis is how can an air writing system be implemented without using any external hardware. Another problem is using gesture recognition to count the number of fingers a user is holding up.

Air writing can be broken down into three main steps:

1. **Hand detection**.
2. **Fingertip detection**
3. **Fingertip tracing**.

There are three gestures that are used in the system:

- **One-Finger:** When a user holds up only **one** finger, air writing is started.
- **Two-Finger:** When a user holds up **two** fingers, the canvas is cleared and the user can start writing on a new canvas.
- **Five-Finger:** When a user holds up **Five** fingers, a screenshot of the canvas is saved.

1.4 APPROACH TO THE PROBLEM AND ORGANIZATION OF THE THESIS

1.4.1 IMPLEMENT AIR WRITING

- **Hand detection:** The hand of the user is detected using a HSV color mask that removes all the pixels of the live video frame that do not fall in the upper and lower range of skin color.
- **Fingertip detection:** Fingertip detection is performed only when the hand gesture for writing (One-Finger) is performed by the user. The fingertip of the user is detected by first detecting the biggest contour present in the frame and then classifying the highest point of the contour as the fingertip.
- **Fingertip tracing:** Fingertip tracing also happens only when the writing hand gesture (One-Finger) is performed by the user. A double ended queue is used to store all the trajectory points of the fingertip and then these points are drawn on the canvas.

1.4.2 FIND THE NUMBER OF FINGERS A USER IS HOLDING UP

- Find extreme points of the hand contour.
- Find the center of the hand contour.
- Draw a circle with the hand center as the circle center which passes through the extreme points detected earlier.
- Bitwise AND the hand contour and the circle to find the number of fingers a user is holding up.

1.4.3 ORGANIZATION OF THE THESIS

- **Chapter 2:** This chapter states and explains all the algorithms used to create this system.
- **Chapter 3:** This chapter states how the algorithms have been implemented and also provides the tools used for the project.
- **Chapter 4:** This chapter concludes the thesis and explains future work that can be done regarding air writing and gesture recognition.

At last, a list of all the resources utilized in this project are mentioned.

CHAPTER 2- ALGORITHMS USED

2.1 METHODOLOGY OF PROJECT

There are two main components of the project. The first is implementing **air writing**, and the second component is **counting the number of fingers** a user is holding up.

Air writing is implemented by first **detecting the hand** of the user. This is done by using skin color based segmentation, in which we specify the upper and the lower thresholds of skin HSV color value. After hand detection, **fingertip detection** is performed. This is done by first capturing the biggest contour (hand contour) from the video frame and then detecting the topmost point of the contour as the fingertip. The last and final step for air writing is **finger tracing**. Finger tracing is done by storing all the points visited by the fingertip in a double ended queue (deque) and then displaying these points.

The **number of fingers** a user is holding up is **counted** in order to add the functionalities of **writing**, **erasing** and **saving** to the system. There are three different gestures associated with the three functionalities. The number of fingers is counted by first drawing a circle around the biggest contour in the frame and then performing the bitwise AND operation between the circle and the biggest contour.

In the following sections, algorithms and functions used in this project are described in detail.

2.2 IMAGE PROCESSING

Image processing is the method of performing certain operations to get an enhanced image. In simple terms, it is the use of algorithms in order to extract useful information from an image.

It mainly includes 3 steps:

1. Image acquisition for importing an image
2. Manipulating the image
3. Output based on the analysis of image

The different phases of image processing we have used in our image are:

1. **Acquisition**: The main work in image acquisition involved scaling and color conversion.
2. **Image Enhancement**: We used image smoothing techniques such as Gaussian Blur and 2D Convolution.

3. **Morphological Processing**: Non-linear operations for extraction of image components related to the shape or features of an image. In our project, this was done through 2 main operations- Erosion and Dilation.
4. **Image segmentation**: Partitioning an image into constituent parts is part of segmentation. We have done this using image thresholding.
5. **Object detection and recognition**: It is the process of detecting an object and assigning a label to it based on its descriptor.

These steps have been explained in detail over the next sections from 2.3 to 2.5.

2.3 IMAGE ENHANCEMENT

Image enhancement is done to improve the quality and information content of original data before processing. Image Smoothing, also known as Low Pass Filtering is employed to remove high spatial frequency noise from a digital image. The low-pass filters work by employing a moving window operator that affects one pixel of the image at a time, changing its value by some function of a local region of pixels.

2.3.1 GAUSSIAN BLUR

We have used Gaussian Blur for image smoothing, which is a 2-D convolution operator that is used to blur images and remove detail and noise. It is very similar to the mean filter, but it uses a different kernel that represents a `bell-shaped' hump. It is done with the function, cv2.GaussianBlur().

In our project we have used 2D convolution using Gaussian Blur. In 2-D, a circularly symmetric Gaussian has the form:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Fig 2.2 Gaussian Distribution 2-D

This is shown in the image below:

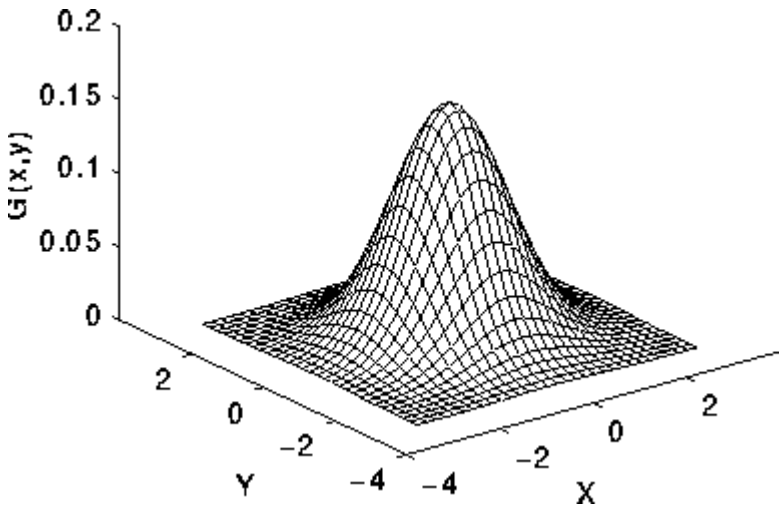

Fig 2.3 Gaussian Distribution Graph

In any blur, we consider a rectangular group of pixels surrounding the pixel to filter. This group of pixels is called the kernel and moves along with the pixel that is being filtered. This filtering is done to the current pixel by taking a weighted average of the the color values of the pixels in the kernel.

In a Gaussian blur, the pixels nearest the center of the kernel are given more weight than those far away from the center (as is clear from the 2D figure shown). This averaging is done on a window-by-window basis, and the average value of the window becomes the new value for the filtered pixel in our output image. Larger kernels have more values factored into the average, therefore a larger kernel blurs the image more than a smaller kernel.

For example, a discrete approximation of a Gaussian function with σ=1.0 would look like

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Fig 2.4 Gaussian Matrix

2.4 MORPHOLOGICAL PROCESSING

Binary images may often contain many imperfections. Once we have performed color space conversion, we have used morphological processing. This is because images can undergo distortion when simple thresholding is used to convert them into binary regions. Morphological operations are non-linear operations done to remove these imperfections by accounting for the form and structure of the image. These rely on the relative ordering of pixel values and not on their numerical values, therefore they are especially suited to the processing of binary images.

A structuring element called a kernel is used for probing of an image in morphological processing. This kernel is moved across all possible locations in the input image. This way, the pixel being processed is compared with its respective neighbourhood of pixels to remove noise.

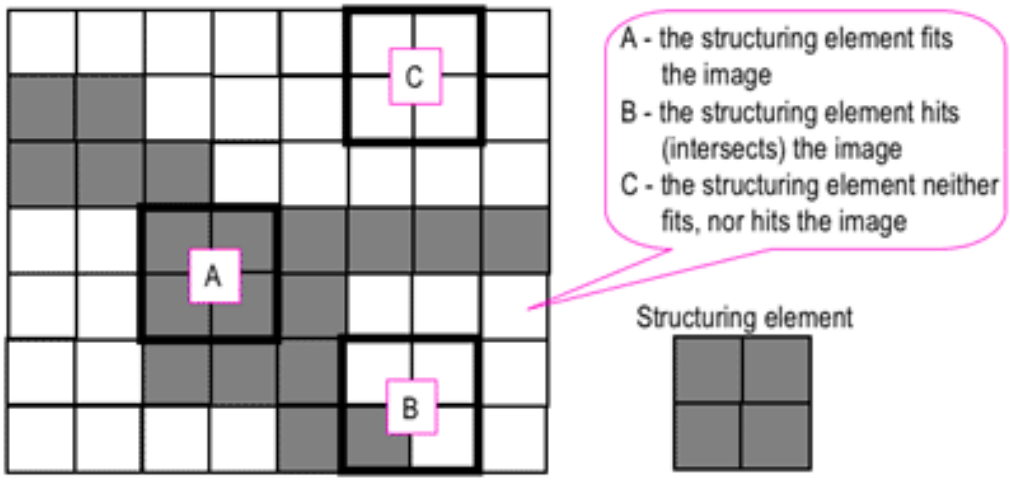

Fig 2.5 Morphological Processing

2.4.1 EROSION

In erosion, the kernel or structuring element is used to reduce the shapes present in the input image by behaving as a local minimum filter. It performs an opposite operation of dilation.

The structuring element works by shrinking the image pixels by removing a layer of pixels from the object boundaries. This helps in eliminating the holes and increasing the gaps between different objects.

The value of the output pixel is the minimum value of all pixels in the neighborhood. This implies that in the input image, a particular pixel is set to 0 if any of its neighboring pixels have the value 0. This operation works by removing islands and small noises in the binary image. Eventually, only substantial objects are left.

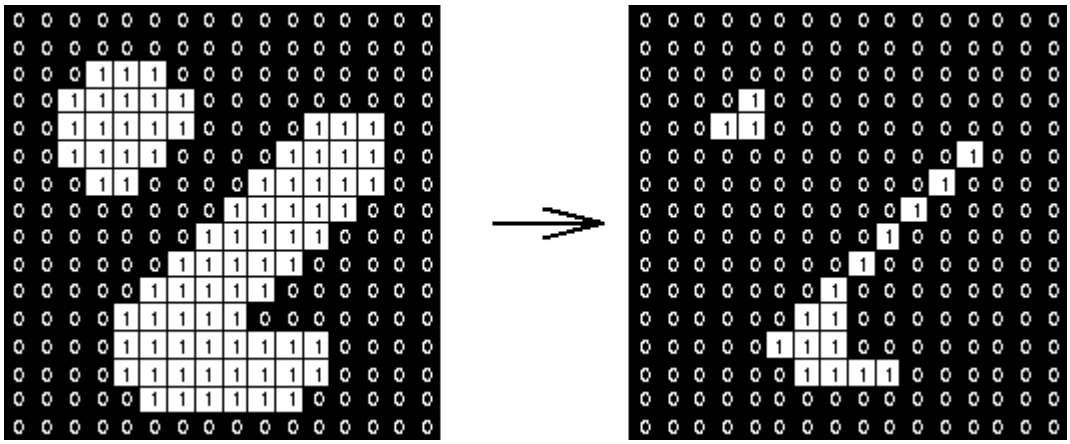

Fig 2.6 Erosion

2.4.2 DILATION

In dilation, the structuring element or kernel is used to expand the objects present in the image. It is the opposite of erosion and increases the white region in a binary image by filling in small holes.

It does so by acting as a local maximum filter. The value of the output pixel is the maximum value of all pixels in the neighborhood. Therefore, a pixel is set to 1 if atleast one of the neighboring pixels have the value 1. This way dilation accentuates features and increases object area.

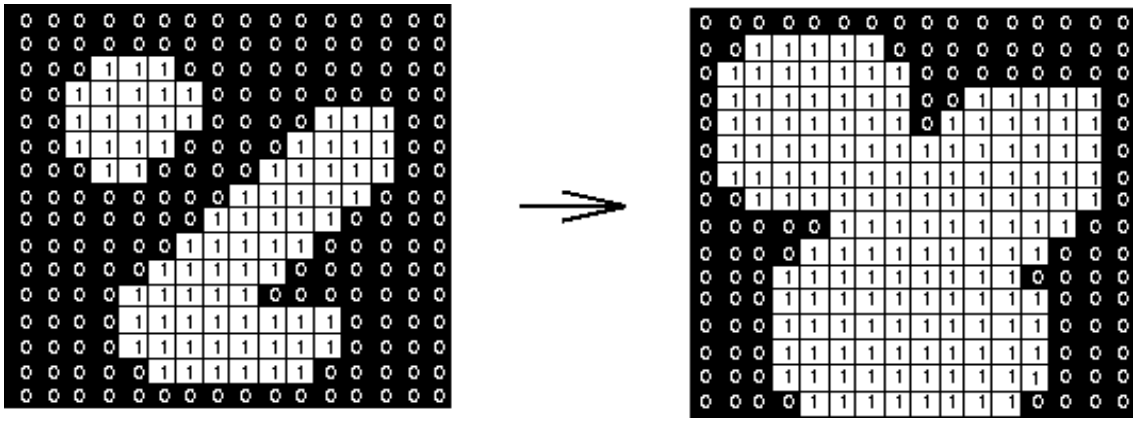

Fig 2.7 Dilation

2.5 OBJECT DETECTION AND RECOGNITION

Object detection and recognition is a part of image processing that deals with detecting instances of semantic objects of a certain class in digital images and videos. Various algorithms can be used for this purpose in computer vision. Our approach involves using skin

color based segmentation for foreground detection and capturing the biggest contour from the video frame as the hand contour. The topmost point of the contour is taken as the fingertip for air-writing purpose when the gesture for writing is used.

2.5.1 COLOR SEPARATION

Color processing is used as primary filtering to eliminate the unrelated colour or object in the image. Color space conversion is done for the purpose of converting an image that is represented in one color space to another color space, to make the translated image look as similar as possible to the original. There are multiple color spaces such as RGB, HSV, etc.

The algorithm for object tracking using color space conversion is:

1. Take each frame of the video
2. Convert from BGR to HSV color-space
3. We threshold the HSV image for a range of skin color to create a mask
4. We then extract the skin color object using a bitwise AND operator - to perform the operation between the original frame captured and the mask created for skin color ranges.

2.5.2 CONTOUR DETECTION

In simple terms, contour is a curve joining all the continuous points (along the boundary), which have the same color or intensity. The contours are a useful tool for object detection and recognition.

After applying skin color separation and morphological operation, we use the findContours() to find the biggest contour in our image and identify it as the hand. In OpenCV, this function involves finding white objects from black background so we first threshold the image into black and white pixels.

We see that there are three essential arguments in cv2.findContours() function. First one is source image, second is contour retrieval mode, third is contour approximation method and it outputs the image, contours, and hierarchy. ‘contours‘ is a Python list of all the contours in the image. Each individual contour is a Numpy array of (x, y) coordinates of boundary points of the object.

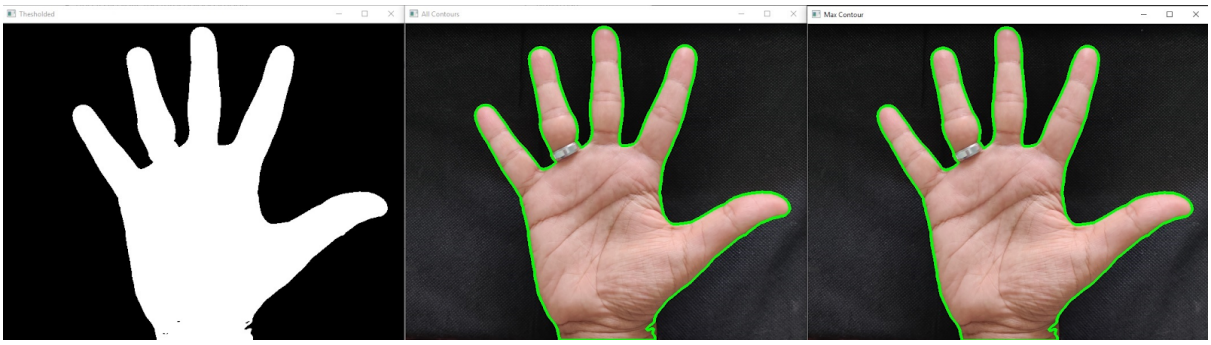

Fig 2.8 Thresholding and Segmenting Hand region
(a) Thresholded image (b) All Contours (c) Max Contour

2.6 GESTURE RECOGNITION

Apart from image processing algorithms, we have used an efficient to recognize gestures so appropriate actions (such as writing, clearing, saving) can be taken accordingly. The gesture recognition algorithm used in our project needs no training and hence requires very less computational time.

2.6.1 COUNT FINGERS METHODOLOGY

Count Fingers Method is an efficient algorithm for hand gesture recognition. It consists of 4 intermediate steps to count the fingers, given a segmented hand region.

1. Take the contour (segmented hand region) and find its convex hull and then compute the most extreme points in the resulting convex hull (Extreme Top, Extreme Bottom, Extreme Left, Extreme Right).

2. Using the extreme points extracted from the convex hull, find the center of the palm.

3. Construct a circle from the palm's center with radius as the maximum Euclidean distance between the palm's center and the extreme points.

4. By performing bitwise AND between the two- thresholded hand image and the circular region of interest, we can reveal the finger slices, which can then be used to count the number of fingers presented.

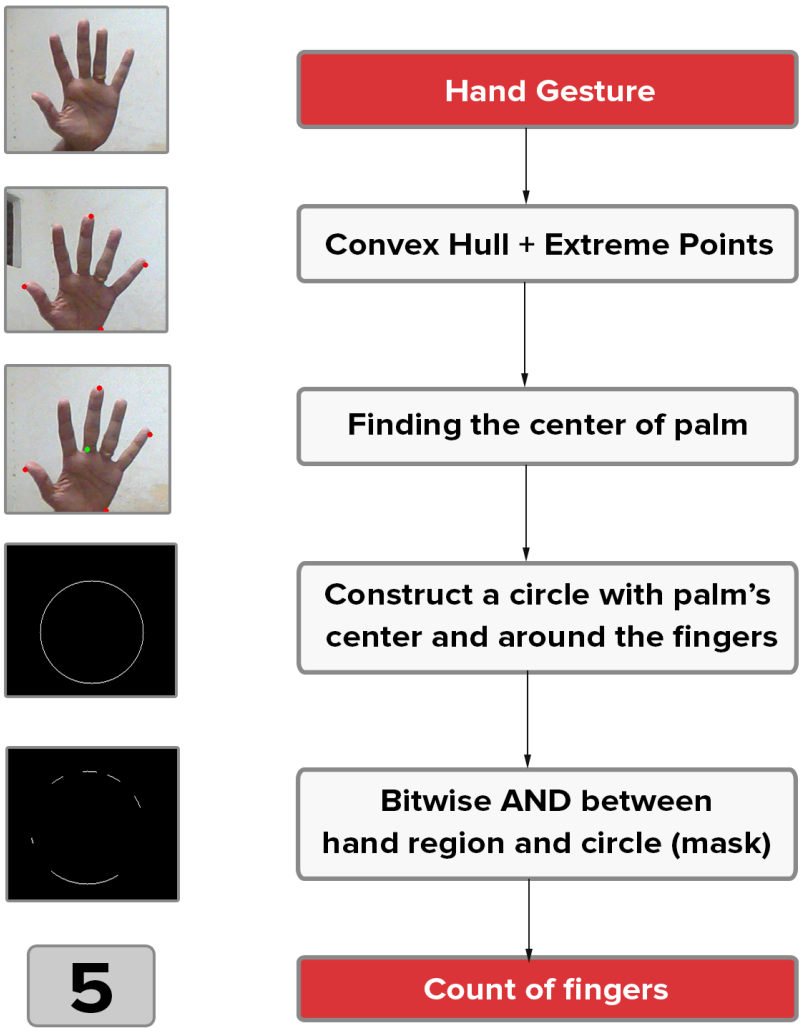

Fig 2.9 Count Finger Methodology

2.6.2 CONVEX HULL

The convex hull of the set is the smallest convex polygon that contains all the points of it. It can be thought of as an envelope that covers the entire region of interest- our hand region.

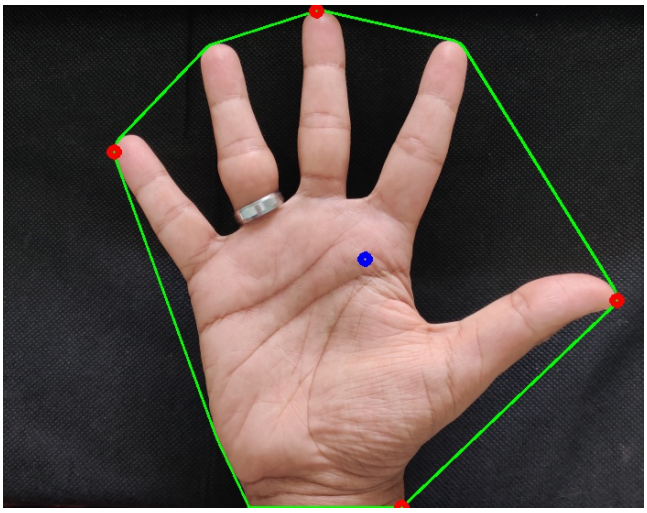

Fig 2.10 Convex Hull

cv2.convexHull() function can be used for the purpose of finding the convex hull. Using the convex hull, we found the most extreme points of the hand and thus the centre of the palm.

2.6.3 EUCLIDEAN DISTANCE

Euclidean distance between two points on a 2-D surface can be computed using the formula shown in Fig 2.11.

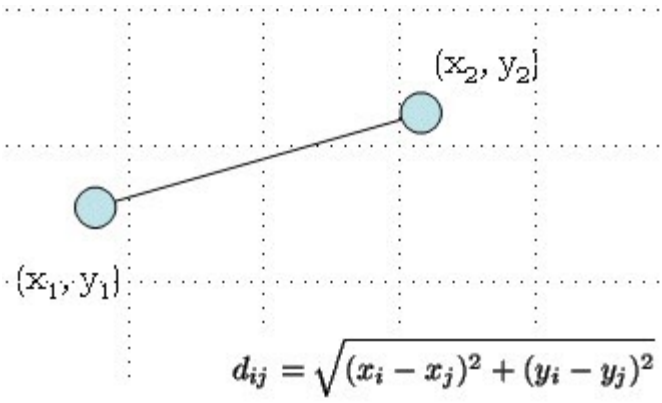

Fig 2.11 Euclidean Distance

It can be calculated from one point to any number of other points through the use of pairwise.euclidean_distances() function from Scikit-learn library. In our project, we have used it to find out the maximum Euclidean distance between the palm's center and the extreme points, which is further used to construct a circle with this distance.

2.6.3 BITWISE AND

Bitwise AND between any two given objects is taken by defining a mask and only extracting the parts of an image that lie under the predefined mask. We have used OpenCV

cv2.bitwise_and() function to take bitwise AND between the circular region of interest and the thresholded image of the hand. In our project, it displays the cuts in the fingers as is shown in figure 2.11. The slice that is covered by the wrist is then taken out of account by using a simple condition.

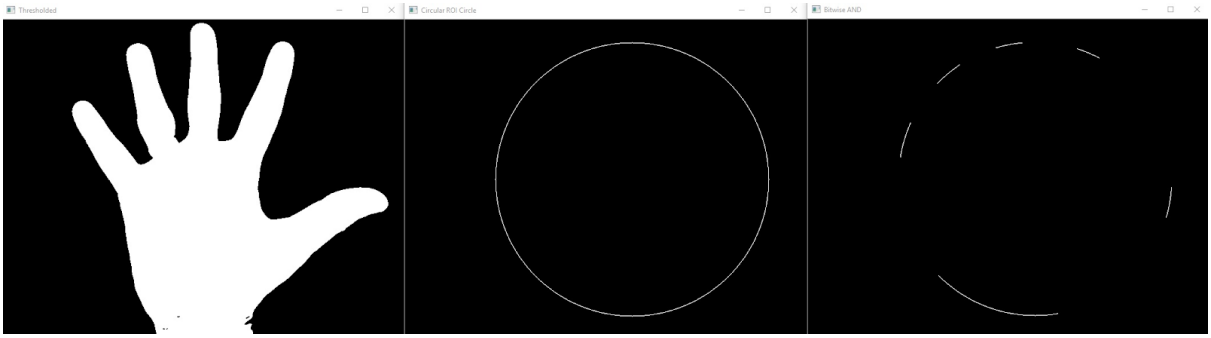

Figure 2.11 Bitwise AND

CHAPTER 3: IMPLEMENTATION

3.1 TOOLS USED

The tools used for completion of the project are briefed below.

3.1.1 PYTHON

We used the python programming language for the implementation of our project which was developed by Guido van Rossum in 1991. We used python 3.9.0 version. It consists of various in-built libraries that give flexibility to the user. Python is a high level language which is intuitively readable in nature with less syntactic constraints. It is also encoded, Interactive and object-oriented scripting language for high-level purpose. It also supports garbage collection and is easily integratable with Java, C++ and COBRA. Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick. Python can be treated in a procedural way, an object-oriented way or a functional way.

Fig 3.1: Python Logo

3.1.2 VISUAL STUDIO CODE

We have used Visual Studio Code for interactive user development (IDE). It is a software based interface. It is a free source code editor designed and developed by Microsoft for Windows, Linux and MacOS. It supports debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. We can highly modify it according to our needs with the help of various Extensions.

Fig 3.2: VS Code Logo

3.1.3 NUMPY

Numpy, created in 2005 by Travis Oliphant, stands for Numerical Python. It is a python library mainly used for tasks including arrays and matrices. It can work in domains of linear algebra, fourier transform, and matrices. It is an open source project which can be used freely. Arrays of Numpy are better than python lists in terms of space complexity, and are computationally faster also.

Fig 3.3: NumPy Logo

3.1.4 COLLECTIONS

Collections is pythons solution for providing various data structures templates for easier access and better use. It is a library in which there are containers used to store collections of data other than the built-in ones like list, dict, set, tuple etc. It was first introduced in the 2.4 version release.

Few container data types are:

- Counter
- Defaultdict
- OrderedDict
- Deque
- ChainMap

- Namedtuple

3.1.5 MATHS

Math is an inbuilt module in python which is used for performing mathematical calculations. In our project, we used various different mathematical formulas like finding the center for the circle, approximations, area percentages and lots more. All these could be easily implemented with the use of math modules.

3.1.6 LINUX

We have developed our project on Ubuntu and Kali versions of Linux OS. It is a free open source operating system with a very powerful support for development functionalities. Hence, it is very popular in the developer community. It is faster, smoother, and more secure than commonly used OS.

Fig 3.4: Linux Logo

3.2 DATA STRUCTURES USED

3.2.1 DEQUE

Deque (short for Dequeue) is a double ended queue data structure. Unlike normal queues, in which entry and exit is allowed from one end each only, deque allows both entry and exit from both ends making it a very flexible data structure with entry and exit time complexity of O(1). It can be implemented using a collection module or can be implemented manually for custom changes.

We have used Dequeue for storing coordinates of the screen for tracing letters or words.

3.2.2 TUPLE

A tuple is a collection which is ordered and unchangeable, used for storing multiple items in a single variable. Tuple is a built-in data type in Python.

We have used Tuples for finding extreme points in the convex hull for detecting fingertips.

3.2.3 ARRAY

We have used arrays to store the colour values of skin colour for detecting skin colour when we capture the frame. Arrays are collections of the same data type values that are stored in a continuous memory locations.

3.3 STEPS OF PROGRAM

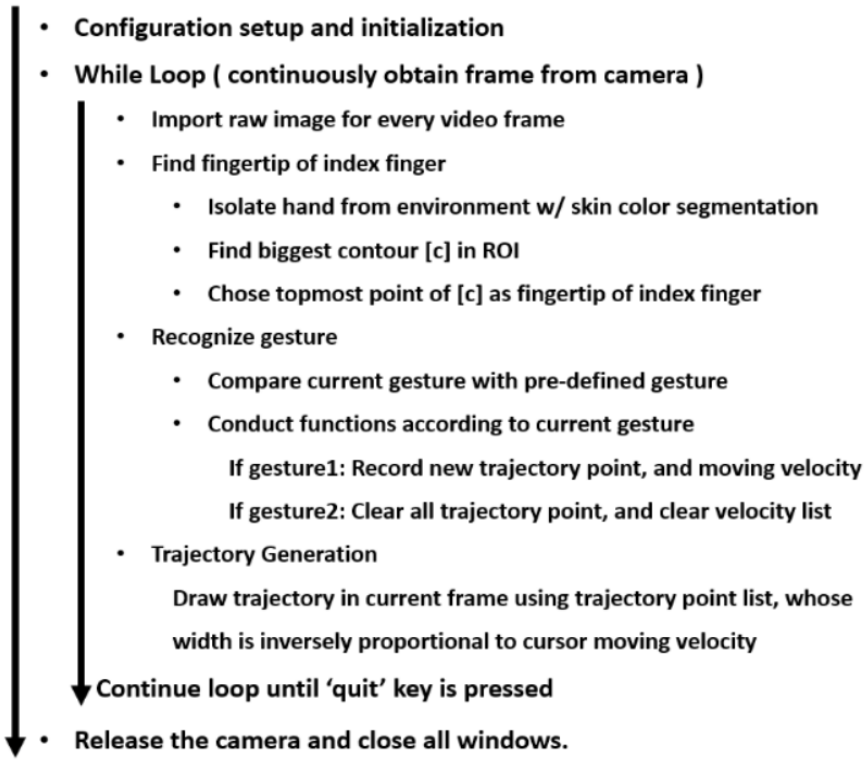

Fig 3.5 Workflow

3.3.1 CAPTURING FRAMES

We are capturing real time video frames using the in-built or external webcam as per the user requirement as we wanted to make this application as flexible as possible. We capture these video frames using videoCapture () function. In videoCapture function argument is passed as

device variable which refers to the webcam index in the computer (default saved as 0). The "ret, frame_raw" returns us the captured frame of the ongoing real time video.

```
cap = cv2.VideoCapture(device)

cap_height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
cap_width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
```

Fig 3.6 Capturing Frames 1

```
while(cap.isOpened()):
    ## Capture frame-by-frame
    ret, frame_raw = cap.read()
    while not ret:
        ret,frame_raw = cap.read()
```

Fig 3.7 Capturing Frames 2

3.3.2 CONVERTING THE FRAME

First, we flip the captured frame as the feed we receive is laterally inverted i.e, our left appears on the right and right on the left, which is not desirable as it leads to confusion. This is achieved using the flip() function of openCV.

Then we capture the Region of interest (ROI) in a different variable. The captured image is in BGR format. In BGR, the B, G, R co-relates to the colour luminance also known as the intensity of the colour and this cannot be separated from the colour information in BGR, so there is a need to convert the image into HSV format as it can be used according to the needed requirements and it makes it easier to deal with frames of the videos. This can be done by using the cvtColor() function in OpenCV.

```
# Flipping the frame to see same side of yours
frame_raw = cv2.flip(frame_raw,1)
frame = frame_raw[:round(cap_height),:round(cap_width)] #ROI of the image
# Color separation and noise cancellation at HSV color space
#  Converting to HSV color space
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

Fig 3.8 Frame Conversion

3.3.3 COLOR SEPARATION

The hand region is segmented using color based segmentation, in which we specify the upper and the lower thresholds of skin HSV color value. We then take a bitwise and between the captured frame and this skin color mask.

```
#  Converting to HSV color space
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

#  Making a mask for skin color ranges
mask = cv2.inRange(hsv, skin_min, skin_max)

#  Bitwise AND between the HSV frame and mask to segment hand
res = cv2.bitwise_and(hsv, hsv, mask= mask)
```

Fig 3.9 Color Separation

3.3.4 MORPHOLOGICAL OPERATIONS AND IMAGE SMOOTHING

Erosion and Dilation are performed on the image to remove noise using kernels defined in the program. Image smoothing techniques such as Gaussian Blur are then performed using cv2.GaussianBlur which takes as input the width and height of the kernel, as well as the standard deviation sigmaX and sigmaY. Gaussian filter is very effective in removing Gaussian noise from the captured frame.

```
# Erosion and Dilation for noise cancellation
res = cv2.erode(res, kernel1, iterations=1)
res = cv2.dilate(res, kernel1, iterations=1)

## defining frame img to show trajectory on white background
img = 255 * np.ones((round(cap_height),round(cap_width),3), np.uint8)

## Converting to grayscale
gray = cv2.cvtColor(rgb, cv2.COLOR_BGR2GRAY)

## 2D Convolution for image filtering
gray = cv2.filter2D(gray,-1,kernel2)

## Gaussian Blur for image smoothing
gray = cv2.GaussianBlur(gray, (11, 11), 0)
```

Fig 3.10 Morphological Processing and Image Smoothing

3.3.5 DETECTION OF HAND AND FINGERTIP

After image preprocessing, hand detection is done by first capturing the biggest contour (hand contour) from the video frame using the findContours() function in Opencv. The fingertip is then detected as the topmost point of the contour. Since findContours works by detecting white objects in black background, the image is first converted into white and black pixels.

```
## threshold image into black and white pixels
gray = cv2.threshold(gray,30, 255,cv2.THRESH_BINARY)[1]
cv2.imshow('gray',gray)

## main function: find finger cursor position & draw trajectory

# find all contours in the image
im2, contours, hierarchy = cv2.findContours(gray, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

# if any contour is found
if len(contours) !=0:
    # find biggest contour in the image
    c = max(contours, key = cv2.contourArea)

    if cv2.contourArea(c) > 1000:
        # consider the topmost point of the biggest contour as cursor
        topmost = tuple(c[c[:,:,1].argmin()][0])
```

Fig 3.11 Hand and Fingertip Detection

3.3.6 GESTURE RECOGNITION

The number of fingers a user is holding up is counted in order to add the functionalities of writing, erasing and saving to the system. There are three different gestures associated with the three functionalities. The number of fingers is counted by first drawing a circle around the biggest contour in the frame and then performing the bitwise AND operation between the circle and the biggest contour.

```
def count(thresholded, segmented):
    # find the convex hull of the segmented hand region
    chull = cv2.convexHull(segmented)

    # find the most extreme points in the convex hull
    extreme_top    = tuple(chull[chull[:, :, 1].argmin()][0])
    extreme_bottom = tuple(chull[chull[:, :, 1].argmax()][0])
    extreme_left   = tuple(chull[chull[:, :, 0].argmin()][0])
    extreme_right  = tuple(chull[chull[:, :, 0].argmax()][0])
```

Fig 3.12 count() function 1

```
# find the center of the palm
cX = int((extreme_left[0] + extreme_right[0]) / 2)
cY = int((extreme_top[1] + extreme_bottom[1]) / 2)

# find the maximum euclidean distance between the center of the palm
# and the most extreme points of the convex hull
distance = pairwise.euclidean_distances([(cX, cY)], Y=[extreme_left, extreme_right, extreme_top,
extreme_bottom])[0]
maximum_distance = distance[distance.argmax()]

# calculate the radius of the circle with 80% of the max euclidean distance obtained
radius = int(0.8 * maximum_distance)

# find the circumference of the circle
circumference = (2 * np.pi * radius)

# find the area of the circle
area = (np.pi * radius * radius)

# find the area of the segmented hand (contour)
carea = cv2.contourArea(segmented)

# if the hand is in a fist, return count of fingers as 0
if carea/area >= 0.9:
    return 0

# take out the circular region of interest which has the palm and the fingers
circular_roi = np.zeros(thresholded.shape[:2], dtype="uint8")

# draw the circular ROI
cv2.circle(circular_roi, (cX, cY), radius, 255, 1)

# take bitwise AND between thresholded hand using the circular ROI as the mask
# which gives the cuts obtained using mask on the thresholded hand image
circular_roi = cv2.bitwise_and(thresholded, thresholded, mask=circular_roi)

# compute the contours in the circular ROI
(_, cnts, _) = cv2.findContours(circular_roi.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
```

Fig 3.13 count() function 2

```
# initialize the finger count
count = 0

# loop through the contours found
for c in cnts:
    # compute the bounding box of the contour
    (x, y, w, h) = cv2.boundingRect(c)

    # increment the count of fingers only if -
    # 1. The contour region is not the wrist (bottom area)
    # 2. The number of points along the contour does not exceed
    #    25% of the circumference of the circular ROI
    if ((cY + (cY * 0.25)) > (y + h)) and ((circumference * 0.25) > c.shape[0]):
        count += 1

return count
```

Fig 3.14 count() function 3

3.3.7 INITIALIZING DEQUEUES

Two deques are used for the purpose of this project. The first is named traj and is used to store trajectory points when the detected gesture is one-fingered(for writing).

Another low pass filter is used for gesture recognition to increase the accuracy of predicting the gesture. It has a size of 10 and the gesture index that has maximum occurrences in this deque at every frame is considered to be the gesture index in that frame, which is then inserted in the deque.

```
## trajectory drawing initialization
#  an array that will hold the points of trajectory
traj = [deque(maxlen = 1024)]

## gesture_index low_pass filter
#  this increases accuracy for gesture recognition
gesture_filter_size = 10
gesture_matching_filter = deque([0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0], gesture_filter_size )
```

Fig 3.15 Deque Initialization

3.3.8 TRACKING AND TRACING

Appropriate step is taken after gesture recognition. If the two fingered gesture is detected, the screen is cleared by making the traj deque empty.

If one fingered gesture is detected, it indicates that the user wants to write. The last step for air writing is finger tracing. Finger tracing is done by storing all the points visited by the fingertip in a double ended queue (deque) and then displaying these points.

If no fingers are detected in the frame, we append the next deque in traj to enable non continuous writing.

```
if gesture_index >= 0:  # filter big position change of cursor
    try:
        ## if two fingered gesture then clear frame
        if gesture_index == 2:
            traj = [deque(maxlen = 512)]
            index = 0
            pass

        ## if three fingered gesture then take screenshot
        elif gesture_index == 3:
            screenshot = 1

        ## if one fingered gesture then store trajectory point and draw
        elif gesture_index == 1:
            traj[index].appendleft(topmost)

        ## if fist gesture then do nothing and append the next deque
        elif gesture_index == 0:
            traj.append(deque(maxlen = 512))
            index += 1
    except:
        print('error')
        pass
```

Fig 3.16 Tracking and Tracing 1

```
## draw lines on frame from stored trajectory points
for i in range(len(traj)):
  for j in range(1, len(traj[i])):
    thickness = 10
    if traj[i][j-1] is None or traj[i][j] is None:
      continue
    cv2.line(frame, tuple(traj[i][j-1]), tuple(traj[i][j]), blue , thickness)
    cv2.line(img, tuple(traj[i][j-1]), tuple(traj[i][j]), blue , thickness)
```

Fig 3.17 Tracking and Tracing 2

3.3.9 SCREENSHOT

In case of the three fingered gesture, a screenshot of the written trajectory is saved. For this purpose, a variable called last_screenshot is maintained to avoid repetition of the same trajectory from different frames. To screenshot the frame, cv2.imwrite() is used to save it as a .jpg image.

```
## if three fingered gesture then take screenshot
elif gesture_index == 3:
  screenshot = 1
```

Fig3.18 Screenshot Flag

```
## if screenshot gesture and no repetition then save screenshot as jpg image
if screenshot==1 and last_screenshot==0:
  s+=1
  cv2.imwrite('screenshot' + str(s) + '.jpg', img)
  last_screenshot = 1
elif screenshot==0:
  last_screenshot = 0
```

Fig3.19 Screenshot Save

CHAPTER 4: CONCLUSION AND FUTURE WORK

4.1 RESULTS OBTAINED

So we have tested our project over 100 times with the help of friends and family as everyone has some different style of writing.Our project was able to detect whatever was written by the user almost every time .
For the Gesture Recognition part,we have tested our project on around 100 plus samples and obtained an accuracy of above 90%.It was noticed that images taken under insufficient lighting and a bad quality camera led to incorrect results.Algorithm works fine unless there are too many pixels in background with skin-like colors.

1. One fingered gesture: Writing

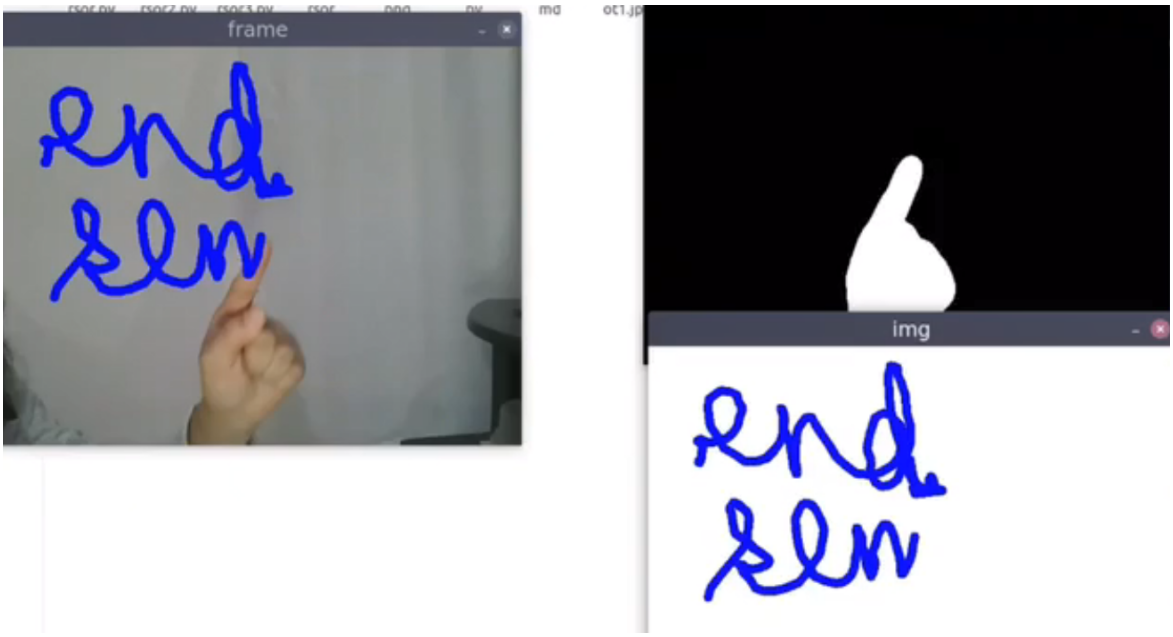

Fig 4.1 Frame - One Finger Gesture

2. Five-fingered gesture: saving image

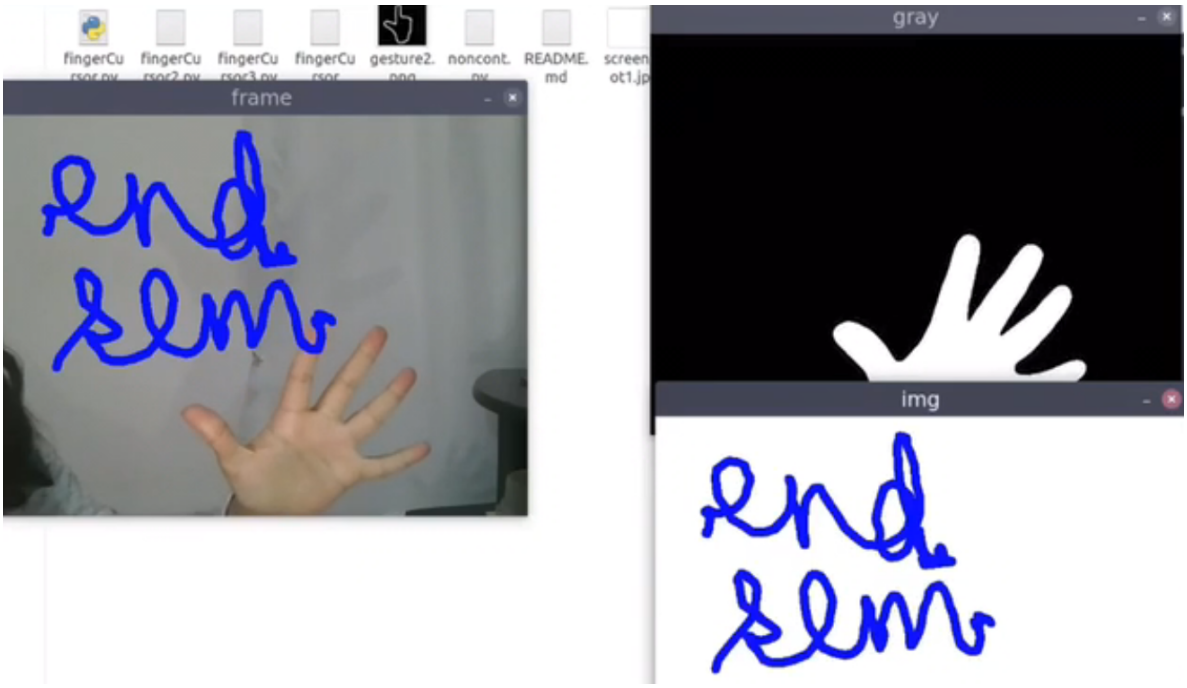

Fig 4.2 Frame - Five Fingers Gesture

3. Two- fingered gesture: clear screen

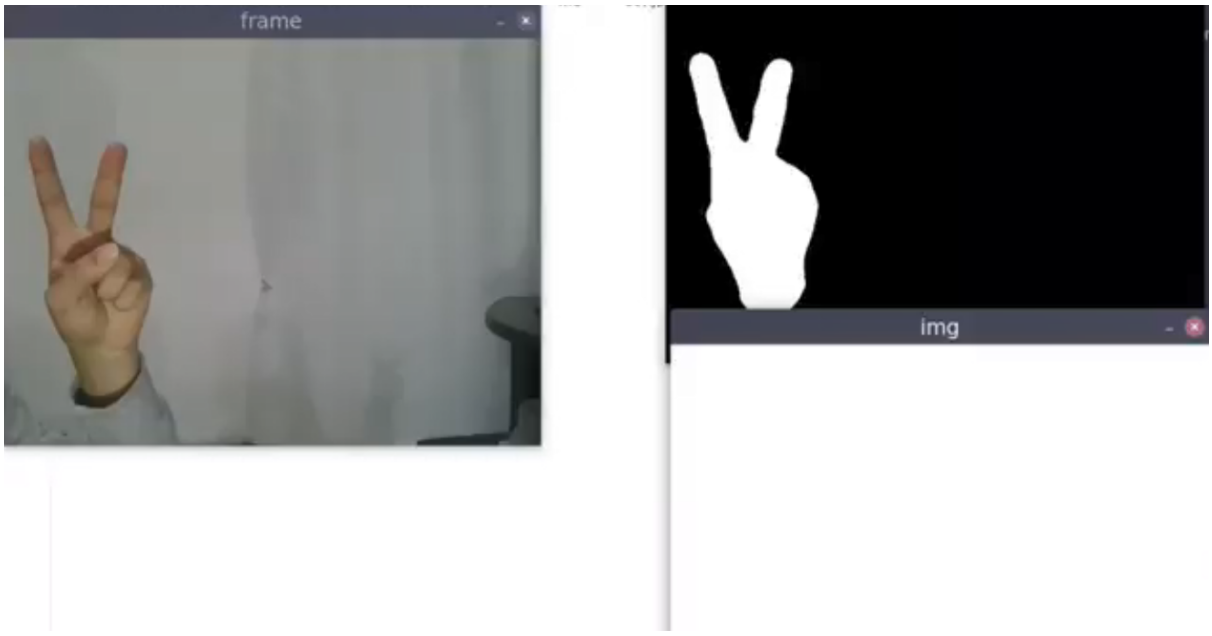

Fig 4.3 Frame - Two Fingers Gesture

4.2 APPLICATIONS

This project has a lot of scope and can be used in various applications and environment.Some of them are as follows:

4.2.1 EDUCATION PURPOSES

We all have heard the saying “Picture tells a thousand word “.Using Air writing children can be taught to learn alphabets of language easily and it will make the learning more interactive and fun for them.
Images taken from Air Writing can ease up the work of teachers and students. By using Air writing teachers don’t have to use chalk and board where the board has to be cleaned with a duster and cleaning duster afterwards making all that dust.Air writing also ease up the work of students and make the learning process more fun. As students we all at some point were not able to make notes of whatever we were taught in class.This problem can be solved using Air writing.Students can just take screenshots of anything written by their teachers by using the gesture feature.
Air writing can help save the environment by reducing the use of paper. It is estimated that 23 trees are used to make 1 ton of paper and almost 2,000,000,000 trees are chopped down just to make paper.Air writing does not involve use of any paper,therefore no trees needed to be chopped.

4.2.2 ANDROID APPS

In today's world technology is everything and it is advancing really fast.A lot of apps can use the air writing to make their work more interactive and easy to use to attract more audience. For example just waving your hands in the air to browse through the app or in music apps to go to the next track or previous track etc.

4.2.3 MODE OF COMMUNICATION FOR PHYSICALLY DISABLED

Air writing is an effective method for communication for individuals who cannot write properly due to some deformity or accident.As air writing does not involve use of any pen and can be done by using any finger and is really easy to use.

4.2.4 HOME APPLIANCES

Air writing can be used in many home appliances to make their use easier and improve their working. For example we can use gesture recognition and air writing in switching of lights and fans by just simply waving our hands in the air.

4.2.5 AUTONOMOUS VEHICLES

Automobiles these days use the latest technology to make driving more enjoyable and comfortable.Air writing can be used in vehicles for giving commands to it and making it easy to use.

4.3 CONCLUSION

In this project we combined Air writing and gesture recognition to make something which can be used in our everyday lives without any prior experience or any expensive equipments.By doing so it opened a whole lot of topics where this project can be used to make things a whole lot easier and better.
In our project we were able to develop a code that is able to track your finger tip and trace its movement and therefore displaying what one wants to write or draw and added some features like taking a screenshot,clearing the screen etc by using a gesture recognition algorithm. It can be used by any device with the only requirement of a camera in it or else earlier methods which perform similar tasks used sensors which limited the use to specific devices and also making it very expensive.

4.4 FUTURE WORK

In our project we have achieved air tracking and tracing depending on the user's will.User can write in a non continuous manner which gives the user a feeling of writing using pen and paper.We have added features like clearing the screen and taking screenshots of the screen by using some predefined gestures.
There is a lot of scope of improvement in certain areas such as :

4.4.1 RECOGNIZING LINGUISTIC CHARACTERS

In our project, if we write suppose a character “A” , it keeps the tracing as it is even if it is a very sloppily written linguistic character. So for this we can use **Optical Character Recognition(OCR)** , it will convert the sloppily written characters into proper characters. This will be very helpful for teachers and students.When teachers write something on the board,that is not clear to understand because of time issues or handwriting flaws. So if we can detect what's written by the user,it will greatly benefit the user as he/she did not have to worry about sloppiness.

4.4.2 PARTIAL ERASE

In our current code we have the feature of completely erasing our ongoing work.To improve we could add the concept of partial erase i.e. erase only a partial of our ongoing work with the help of some predefined gesture for this purpose.

4.4.3 MORE FEATURES

In our project we currently have features like completely erasing the screen ,taking a screenshot of the screen. We can enhance it by adding features like redo and undo.

4.4.4 VOICE COMMANDS

In our project we can use voice commands to control functionalities.For example if the user wants to take the screenshot of the screen,he/she can just say take screenshot or to erase the screen.For this we can use **Long short-term memory (LTSM).**

REFERENCES

[1] J. Davis and M. Shah "Visual Gesture Recognition", IEE Proc.-Vis. Image Signal Process., Vol. 141, No.2, April 1994.

[2] A Malima, E Ozgur, and M Cetin “A Fast Algorithm For Vision Based Hand Gesture Recognition For Robot Control” 2019, Faculty of Engineering and Natural Sciences, Sabancı University, Tuzla, İstanbul, Turkey

[3] Nithya.S, Chandru, Krishnakanth.G, Pavithran.P, “An Optimized Approach For Deaf and Dumb People Using Air Writing”, 2018, IRJET

[4] R.Meena Prakash, T.Deepa, T.Gunasundari, N.Kasthuri, “Gestures Recognition and Fingertip Detection for Human Computer Interaction”, 2017, ICIIECS

PLAGIARISM CHECK

Air Writing Project Thesis

ORIGINALITY REPORT

%16	%11	%7	%14
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Netaji Subhas Institute of Technology Student Paper	%3
2	Submitted to University of Nottingham Student Paper	%1
3	datacarpentry.org Internet Source	%1
4	medium.com Internet Source	%1
5	www.geeksforgeeks.org Internet Source	%1
6	www.w3schools.com Internet Source	%1
7	mmeysenburg.github.io Internet Source	%1
8	Submitted to Higher Education Commission Pakistan Student Paper	%1

9	Submitted to Universiti Teknologi Petronas Student Paper	<%1
10	Submitted to The University of the South Pacific Student Paper	<%1
11	Riaz, Muhammad, An Youngeun, and Park Jongan. "Feature Vector Extraction System Based on Adaptive Segmentation of HSV Information Space", 2009 11th International Conference on Computer Modelling and Simulation, 2009. Publication	<%1
12	lrd.yahooapis.com Internet Source	<%1
13	Jiayi Yuan, Hongye Li, Meng Wang, Ruyang Liu, Chuanyou Li, Beilun Wang. "An OpenCV-based Framework for Table Information Extraction", 2020 IEEE International Conference on Knowledge Graph (ICKG), 2020 Publication	<%1
14	Submitted to University of London External System Student Paper	<%1
15	dar.aucegypt.edu Internet Source	<%1
16	Submitted to University of Wolverhampton Student Paper	<%1

17	Jorge Pena Queralta, Jussi Taipalmaa, Bilge Can Pullinen, Victor Kathan Sarker et al. "Collaborative Multi-Robot Search and Rescue: Planning, Coordination, Perception, and Active Vision", IEEE Access, 2020 Publication	<%1
18	Submitted to Vel Tech University Student Paper	<%1
19	Submitted to University of Zululand Student Paper	<%1
20	www.irjet.net Internet Source	<%1
21	Submitted to Taibah University Student Paper	<%1
22	docplayer.net Internet Source	<%1
23	Submitted to University of Bath Student Paper	<%1
24	Submitted to Rajarambapu Institute of Technology Student Paper	<%1
25	shahaab-co.ir Internet Source	<%1
26	slidelegend.com Internet Source	

		<%1
27	Submitted to Kensington College of Business Student Paper	<%1
28	Submitted to The University of Manchester Student Paper	<%1
29	V. Muthukumar, D.V. Rao. "Image processing algorithms on reconfigurable architecture using HandelC", Euromicro Symposium on Digital System Design, 2004. DSD 2004., 2004 Publication	<%1
30	www.ijert.org Internet Source	<%1
31	Submitted to IIT Delhi Student Paper	<%1
32	acikerisim.deu.edu.tr Internet Source	<%1
33	Ahmed Kadem Hamed AlSaedi, Abbas H. Hassin AlAsadi. "An efficient hand gestures recognition system", IOP Conference Series: Materials Science and Engineering, 2020 Publication	<%1
34	Mie Mie Aung. "Study for License Plate Detection", International Journal of Image, Graphics and Signal Processing, 2019	<%1

Publication

EXCLUDE QUOTES	ON	EXCLUDE MATCHES	< 6 WORDS
EXCLUDE BIBLIOGRAPHY	ON		