In-air Drawing using OpenCV & Python

A PROJECT REPORT

Submitted by

RAVIBHASKAR JHA (19BCS2270**)**

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

November 2022

BONAFIDE CERTIFICATE

Certified that this project report **"IN AIR DRAWING USING PYTHON & OPENCV"** is the bonafide work of "**RAVIBHASKAR JHA"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
NAVPREET KAUR	DR. Mohammad Shuaib Khan
HEAD OF THE DEPARTMENT	**SUPERVISIOR**

Submitted for the project viva-voce examination held on 18NOV2022.

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

ACKNOWLEDGMENTS

We would like to express my deep gratitude to Professor **DR. Mohammad Shuaib Khan Sir**, our project supervisor, for his patient guidance, enthusiastic encouragement and useful critiques of this project work. we would also like to thank him for his advice and assistance in keeping our progress on schedule. Ourgrateful thanks are also extended to all external examiners for their help in doingthis project.

We would also like to extend my thanks to the Chandigarh University and our department for their help in offering us all the help and the resources in developing this project.

Finally, we wish to thank our parents for their support and encouragement throughout our studies.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Writing in air has been one of the most fascinating and challenging research areas in field of image processing and pattern recognition in the recent years. It contributes immensely to the advancement of an automation process and can improve the interface between man and machine in numerous applications. Several research works have been focusing on new techniques and methods that would reduce the processing time while providing higher recognition accuracy. Object tracking is considered as an important task within the field of Computer Vision. The invention of faster computers, availability of inexpensive and good quality video cameras and demands of automated video analysis has given popularity to object tracking techniques. Generally, video analysis procedure has three major steps: firstly, detecting of the object, secondly tracking its movement from frame to frame and lastly analysing the behaviour of that object. For object tracking, four different issues are taken into account; selection of suitable object representation, feature selection for tracking, object detection and object tracking. In real world, Object tracking algorithms are the primarily part of different applications such as: automatic surveillance, video indexing and vehicle navigation etc. The project takes advantage of this gap and focuses on developing a motion-to-text converter that can potentially serve as software for intelligent wearable devices for writing from the air. This project is a reporter of occasional gestures. It will use computer vision to trace the path of the finger. The generated text can also be used for various purposes, such as sending messages, emails, etc. It will be a powerful means of communication for the deaf. It is an effective communication method that reduces mobile and laptop usage by eliminating the need to write.

GRAPHICAL ABSTRACT

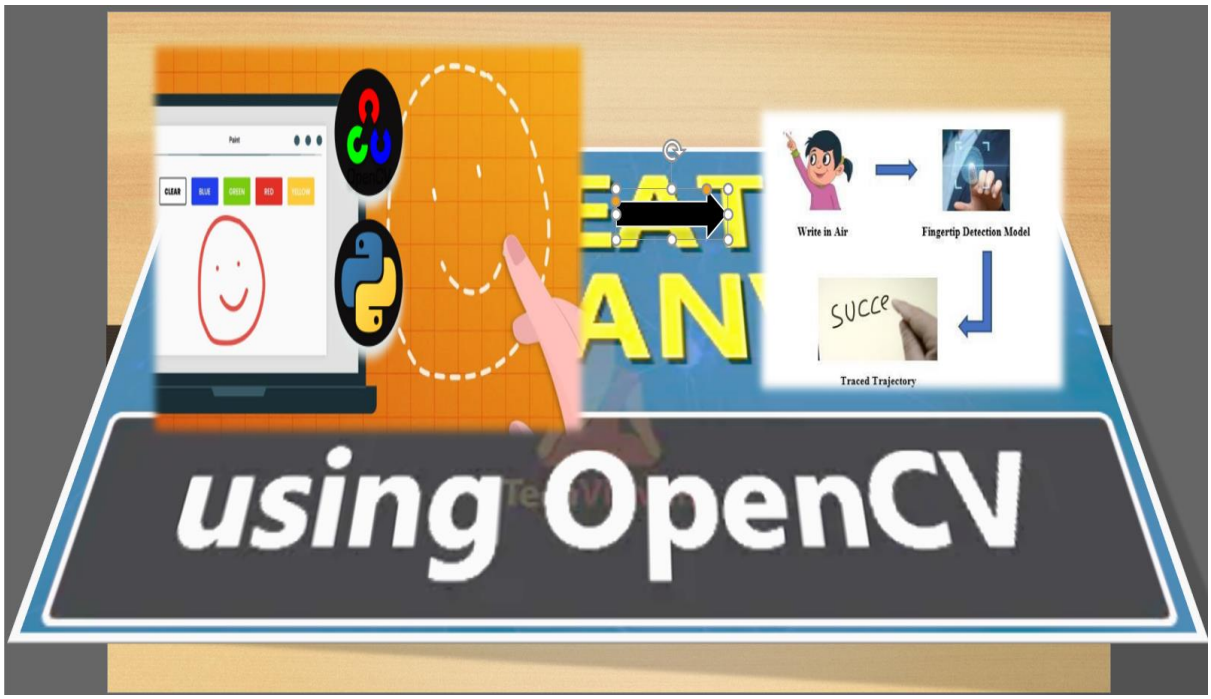
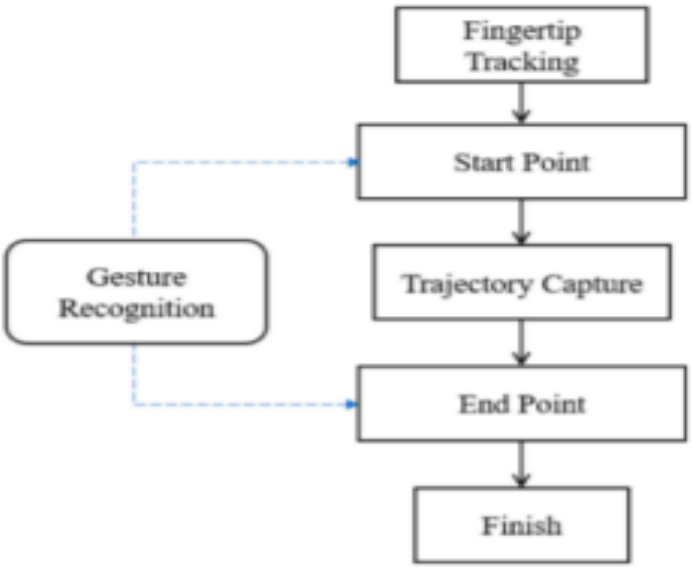


Figure 1. Graphical Abstract



ABBREVIATIONS

Abbreviation	Meaning	Page No.
AW	Air-Writing	10
LED	Light Emission Diode	10
HI	Hearing Impairment	20
CNT	Contour	35
ND	National Democratic	47
HCI	Human – computer interaction	10
SH	syntax highlighting	69

CHAPTER 1
INTRODUCTION

In the era of digital world, traditional art of writing is being replaced by digital art. Digital art refers to forms of expression and transmission of art form with digital form. Relying on modern science and technology is the distinctive characteristics of the digital manifestation. Traditional art refers to the art form which is created before the digital art. From the recipient to analyse, it can simply be divided into visual art, audio art, audio-visual art and audio-visual imaginary art, which includes literature, painting, sculpture, architecture, music, dance, drama and other works of art. Digital art and traditional art are interrelated and interdependent. Social development is not a people's will, but the needs of human life are the main driving force anyway. The same situation happens in art. In the present circumstances, digital art and traditional art are inclusive of the symbiotic state, so we need to systematically understand the basic knowledge of the form between digital art and traditional art. The traditional way includes pen and paper, chalk and board method of writing. The essential aim of digital art is of building hand gesture recognition system to write digitally. Digital art includes many ways of writing like by using keyboard, touch-screen surface, digital pen, stylus, using electronic hand gloves, etc. But in this system, we are using hand gesture recognition with the use of machine learning algorithm by using python programming, which creates natural interaction between man and machine. With the advancement in technology, the need of development of natural ‘ (HCI) systems to replace traditional systems is increasing rapidly.

Object tracking is considered as an important task within the field of Computer Vision. The invention of faster computers, availability of inexpensive and good quality video cameras and demands of automated video analysis has given popularity to object tracking techniques. Generally, video analysis procedure has three major steps: firstly, detecting of the object, secondly tracking its movement from frame to frame and lastly analysing the behaviour of that object. For object

tracking, four different issues are taken into account; selection of suitable object representation, feature selection for tracking, object detection and object tracking. In real world, Object tracking algorithms are the primarily part of different applications such as: automatic surveillance, video indexing and vehicle navigation etc.

Another application of object tracking is human computer interaction. Different researchers proposed many algorithms which are categorically divided into two main approaches: image-

based approach and Glove based approach. Image based approach requires images as input in order to recognize the hand (object) movements. On the other hand, Glove based approach require specific hardware which includes special sensors etc. Such applications are beneficial for disabled people. In this paper, a real time fast video-based fingertip tracking and recognizing algorithm is presented. The proposed algorithm has two major tasks: first it detects the motion of coloured finger in video sequence and then applies OCR (Optical Character Reorganization) in order to recognize the plotted image. Proposed method is software-based approach while in literature, almost all existing finger tracking based character recognition system require extra hard ware e.g., (LED) pen, Leap Motion controller device etc. Furthermore, for recognition they perform comparison operation in order to recognize the input character but for our proposed system we apply OCR for character recognition. As a result of which our computational time is much reduce than.

Basically AW is a form of input that helps a person to write or sketch in free space using their fingertips. AW, like motion gestures, is recorded as a regular stream of information including a series of hand and finger motions. Air-writing differs from traditional writing in that the latter employs a paper pen-based writing system, while the former uses an abstract board in the air. Unlike handwriting characters, those created in three-dimensional space lack the pen-up and quill detail that turns intra letters into sunstrokes. According to a review, air-drawn letter recognition is as effective as expression. The identification of airdrawn letters is not a simple sweep or rotating hand gesture but can represent different symbols. Writing in the air is sometimes used as a means of communication in this situation, particularly when interacting with those that are far away.

“Many utilizing various types of sensors to reliably detect free-air motions in three-dimensional space. Yanmei and Chen et al" A.’s Actual Dynamic Handwriting Recognition Framework Using Sensor Module" is one example. Hand detection, data analysis, model training, and gesture classification are all part of this real-time Kinect-based dynamic HGR framework. As recognition algorithms, the Hidden Markov Model (HMM) and Support Vector Machine (SVM) are used. The identification rate for gestures 6, C, O, and V was less than 50%, bringing the overall average rate down to 82.86 percent of HMM. SVM, on the other hand, graded almost all of the movements as 100%, but 60% on gesture O, resulting in a 95.42 percent overall annual average”. Tsuchida et alreport, "Written letter identification in the Air by Using Leap way of Controller," recognizes 46 Japanese Katakana symbols and 26 alphabets with an overall

recognition rate of 86.7 percent, with 84.54 percent precision for recognizing the English alphabet. Besides, in 2015, Jayesh Kumar Sharma et al. published "Numeral Gesture Recognition with Leap Motion Sensor," and was using the Mathematical Face Recognition device and the Leap Cellular Modem to track free air digit gestures from 0-9. Position spotting, feature recognition, gesture tracking, preparation, and classification were the three phases of the method. The algorithm's drawback, on the other hand, is that it fails to recognize expressions with an open curve. As a result, the system was only able to reach a classification rate of 70.2 percent. The main issue in previous studies was lettered with curves, especially open curves. Other recent experiments have issues with the letter's size and form. The system's consistency rating continues to drop as a result of misclassified letters.

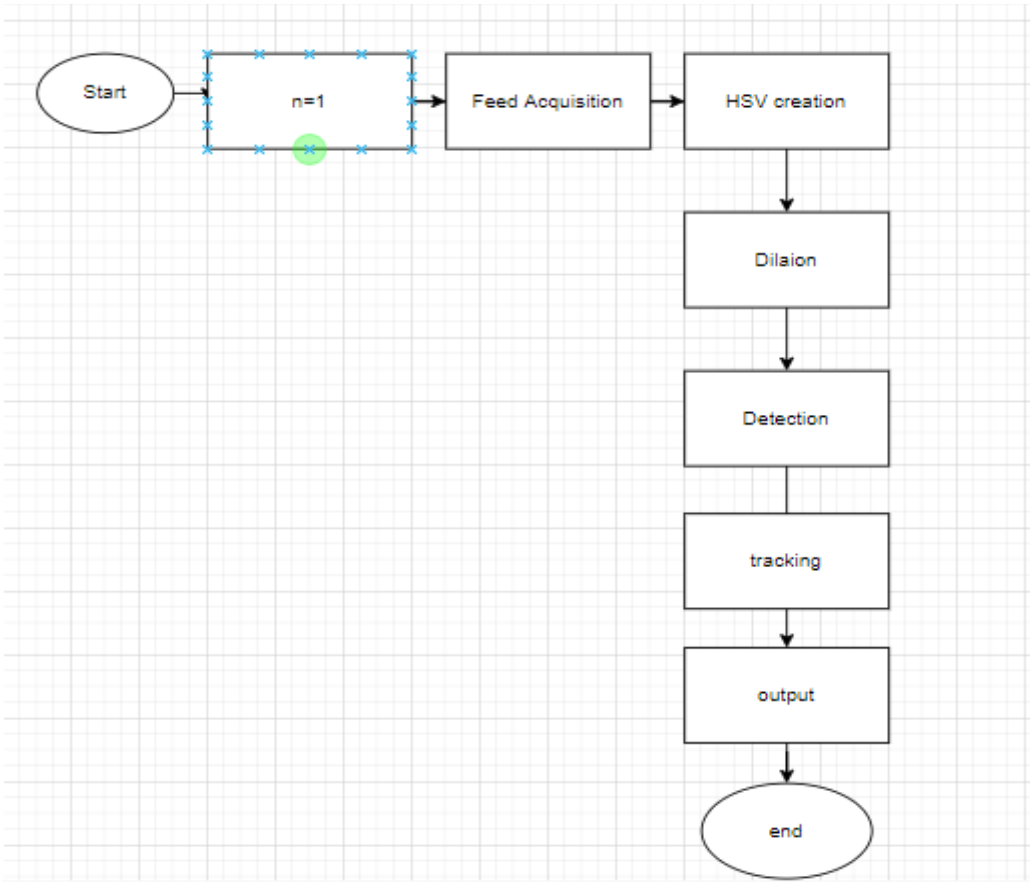


Figure2. Flowchart

The researchers suggest a recognition scheme based on the Slope Orientation Sequence Matching Algorithm to address an issue that has been identified in previous studies. The entire function of the machine is shown in the system block diagram. It contains portable air as the

system input, acquisition of human hand distal phalange joint coordinates, generation of slopes and angles created, slope orientation determination, slope orientation matching, and graded numeric values as the system output. One of the most cutting-edge technologies in the area of personal communication is air-writing technical environmental. As a result, the research would be very useful in the scholarly and medical fields. The suggested algorithm also introduces a novel idea of complex sign language recognition, which can be used in future research.

1.1 Client identification

A. People hearing impairment: Although we take hearing and listening for granted, they communicate using sign languages. Most of the world can't understand their feeling, their emotions without a translator in between.

B. Overuse of Smartphones: They cause accidents, depression, distractions, and other illnesses that we humans can still discover. Although its portability, ease to use is profoundly admired, the negatives include life-threatening events.

C. Paper wastage is not scarce news. We waste a lot of paper in scribbling, writing, drawing, etc.… Some basic facts include - 5 litres of water on average are required to make one A4 size paper, 93% of writing is from trees, 50% of business waste is paper, 25% landfill is paper, and the list goes on. Paper wastage is harming the environment by using water and trees and creates tons of garbage. Air Writing can quickly solve these issues. It will act as a communication tool for people with hearing impairment. Their air-written text can be presented using AR or converted to speech. One can quickly write in the air and continue with your work without much distraction. Additionally, writing in the air does not require paper. Everything is stored electronically.

1.2 Identification of Problem

A. Fingertip detection The existing system only works with your fingers, and there are no highlighters, paints, or relatives. Identifying and characterizing an object such as a finger from an RGB image without a depth sensor is a great challenge.

B. Lack of pen up and pen down motion The system uses a single RGB camera to write from above. Since depth sensing is not possible, up and down pen movements cannot be followed. Therefore, the fingertip's entire trajectory is traced, and the resulting image would be absurd and not recognized by the model. The difference between hand written and air written 'G' is shown in Figure 1.

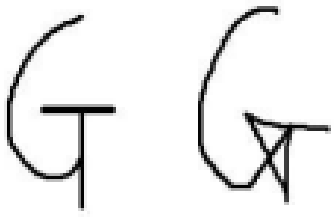


Figure 3. Actual Character vs Trajectory

C. Controlling the real-time system Using real-time hand gestures to change the system from one state to another requires a lot of code care. Also, the user must know many movements to control his plan adequately

1.3 Identification of Task

The project focuses on solving some major societal problems –

1. People hearing impairment: Although we take hearing and listening for granted, they communicate using sign languages. Most of the world can't understand their feeling, their emotions without a translator in between.

2. Overuse of Smartphones: They cause accidents, depression, distractions, and other illnesses that we humans can still discover. Although its portability, ease to use is profoundly admired, the negatives include life-threatening events.

3. Paper wastage is not scarce news. We waste a lot of paper in scribbling, writing, drawing, etc.… Some basic facts include - 5 litres of water on average are required to make one A4 size paper, 93% of writing is from trees, 50% of business waste is paper, 25% landfill is paper, and the list goes on. Paper wastage is harming the environment by using water and trees and creates tons of garbage. Air Writing can quickly solve these issues. It will act as a communication tool for people with hearing impairment. Their air-written text can be presented using AR or

converted to speech. One can quickly write in the air and continue with your work without much distraction. Additionally, writing in the air does not require paper. Everything is stored electronically.

1.4 Timeline

Task	Deadline (starts 5th Oct)
Requirement analysis	1 week
GUI Design	4-7 days
Implementation	2 weeks
Smart-contract development	3 weeks
Testing - Deployment	3-4 days

Figure 2. Project Timeline

At its core, a project timeline is an overview of a project's deliverables laid out in chronological order. It maps out what needs to be completed before a new task can commence and keeps everything ticking along nicely.

Usually, it comes in a visual format, which means stakeholders and team members can get a quick overview at a glance. This allows them to picture the roadmap of a project, including the key milestones, manage individual tasks that need to be completed, dependencies, and those all-important delivery dates.

Benefits of Project timeline:

Every good project manager knows that a project timeline is their trusty companion when starting out on a new adventure. Not only does it help everyone involved visualize the steps in a project, but it helps keep tasks on track to impress stakeholders.

If we dig a bit deeper, we're greeted by a lot more benefits. Here are a few:

Reveals a clear path forward. It's easy to see what step to take first and what needs to happen after that (this is particularly helpful in an overly complex project that has multiple steps and multiple layers).

Maintains the big picture. It's hard to see the end of a project when there are so many little steps needed to get there. A project timeline helps you visualize everything in a single view so you can focus on the bigger picture.

Ensures everyone has the same goal. You need your entire team on board for a project to be successful. Project timelines help everyone recognize how they slot in and what role they play in the overall success of a project.

Keeps everyone in the loop. A project timeline helps everyone involved track the progress of a project, which builds trust with stakeholders and makes communication far easier.

Tracks what happens and when. Knowing the hierarchy of tasks and what needs to happen before something else can take place is a crucial part of project management. A project timeline helps you lay this out clearly and concisely.

Prevents bottlenecks and hold-ups. Discover potential dependencies to avoid bottlenecks that delay a project and frustrate stakeholders.

Make changes easy. Projects don't always follow a linear route. Something might crop up halfway through those needs addressing. A project timeline makes it easy to actively change elements of a project and switch up tasks without leaving anyone in the dark.

1.5 Task/Objective

The objective/Task of this system is mainly used as a powerful means of communication for the deaf, which means implementing this project can help in:

1. An effective communication method that reduces mobile and laptop usage by eliminating the need to write.

2. It helps people with hearing impairments to communicate well.

3. Various purposes, such as sending messages, e-mails, etc., as the generated text can also be used for that.

1.6 Organization of Report

Chapter 1: Introduction (Including Identification of client & need, Relevant contemporary issues, Problem Identification, Task Identification, Timeline, organization of the report)

Chapter 2: Literature survey Timeline of the reported problem as investigated throughout the world, bibliometric analysis, proposed solutions by different researchers, summary linking literature review with the project, Problem Definition, Goals and Objectives.

Chapter 3: Design flow/Process Concept Generation, Evaluation & Selection of Specifications/Features, Design Constraints– Regulations, Economic, Environmental, Health, manufacturability, Safety, Professional, Ethical, Social & Political Issues considered in design, Analysis and Feature finalization subject to constraints, Design Flow (at least 2 alternative designs to make the project), Best Design selection (supported with comparison and reason) and Implementation plan ((Flowchart /algorithm/ detailed block diagram).

Chapter 4: Results analysis and validation Implementation of design using Modern Engineering tools in analysis, design drawings/schematics/ solid models, report preparation, project management, and communication, Testing/characterization/interpretation/data validation.

Chapter 5: Conclusion and future work deviation from expected
results and way ahead References
Appendix
User manual (Complete step by step instructions along with pictures necessary to run the project)
Achievements

1.7 Features Of In-Air Drawing

1. Can track any specific-coloured pointer.
2. User can draw in four different colours and even change them without any Hussle.
3. Able to rub the board with a single location at the top of the screen.
4. No need to touch the computer once the program is run

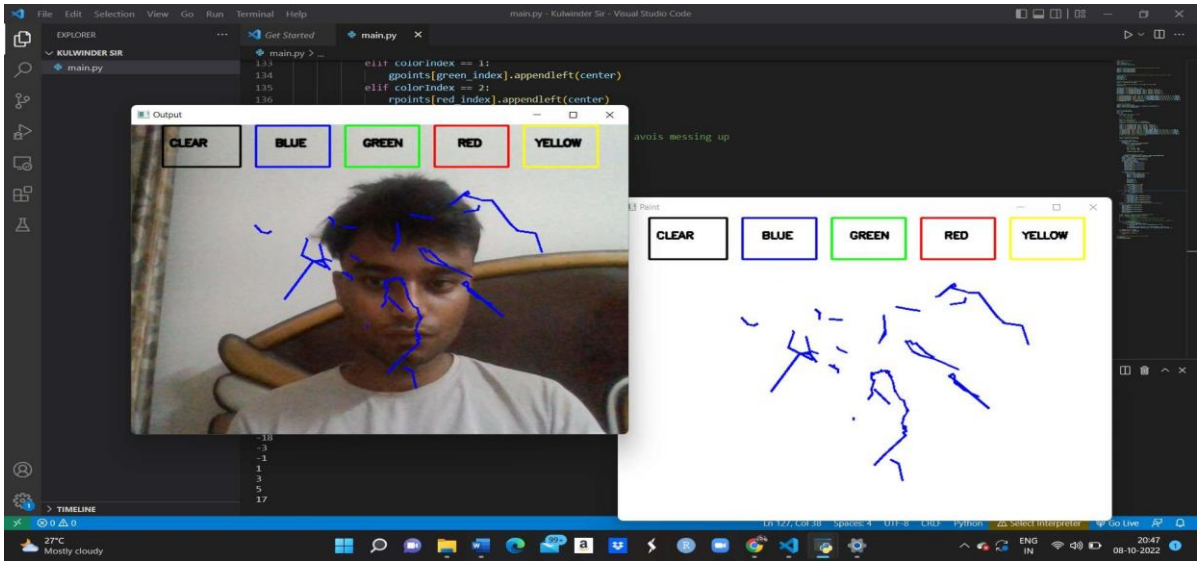


Figure 4. Air canvas

1.8 Relevant Contemporary Issues

If we use the system which uses the depth and colour information from the Kinect sensor to detect the hand shape. As for gesture recognition, even with the Kinect sensor. It is still a very challenging problem. The resolution of this Kinect sensor is only 640×480. It works well to track a large object, e.g., the human body. But following a tiny thing like a finger is complex.

If we use Augmented segmented desk interface approach for interaction This system makes use and a video projector and charge-coupled device (CCD) camera so that using the fingertip; users can operate desktop applications. In this system, each hand performs different tasks. The left hand is used to select radial menus, whereas the right hand is used for selecting objects to be manipulated. It achieves this by using an infrared camera. Determining the fingertip is computationally expensive, so this system defines search windows for fingertips.

1.9 Summary

The aim of the project has been well established by now that we wanted to create a virtual canvas to draw in air without having to touch the computer screen or computer connected devices. This system or project aims to help people with HI and also helps to reduces looking at screen and giving less damage to eyesight. Also, this system aims to reduce the wastage of the paper which is a factor that is harming our precious environment.

CHAPTER 2

LITERATURE SURVEY

A. Robust Hand Recognition with Kinect Sensor In [4], the system proposed used the depth and colour information from the Kinect sensor to detect the hand shape. As for gesture recognition, even with the Kinect sensor. It is still a very challenging problem. The resolution of this Kinect sensor is only 640×480. It works well to track a large object, e.g., the human body. But following a tiny thing like a finger is complex.

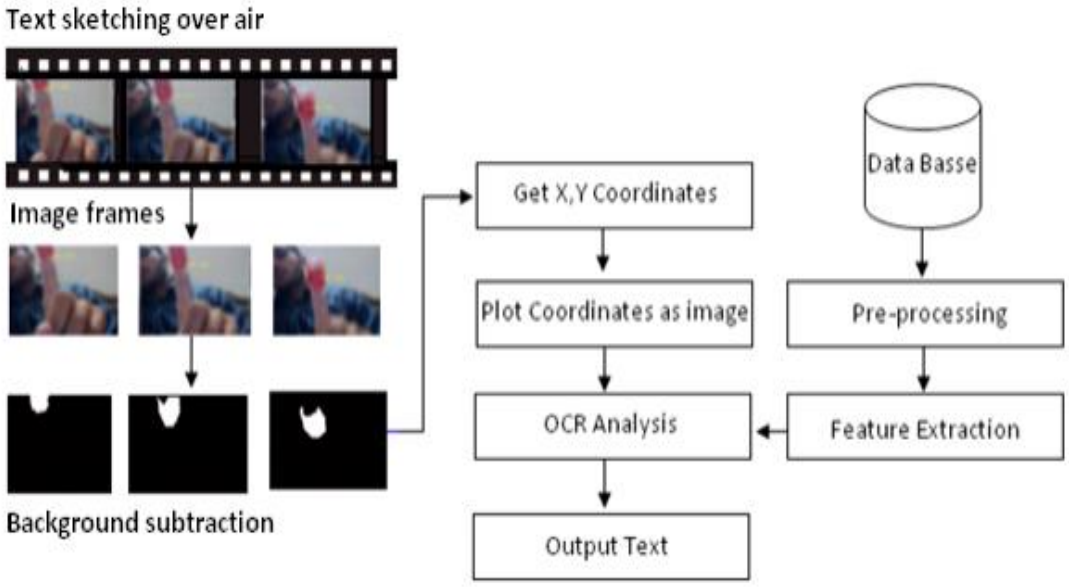


Figure 5. Design

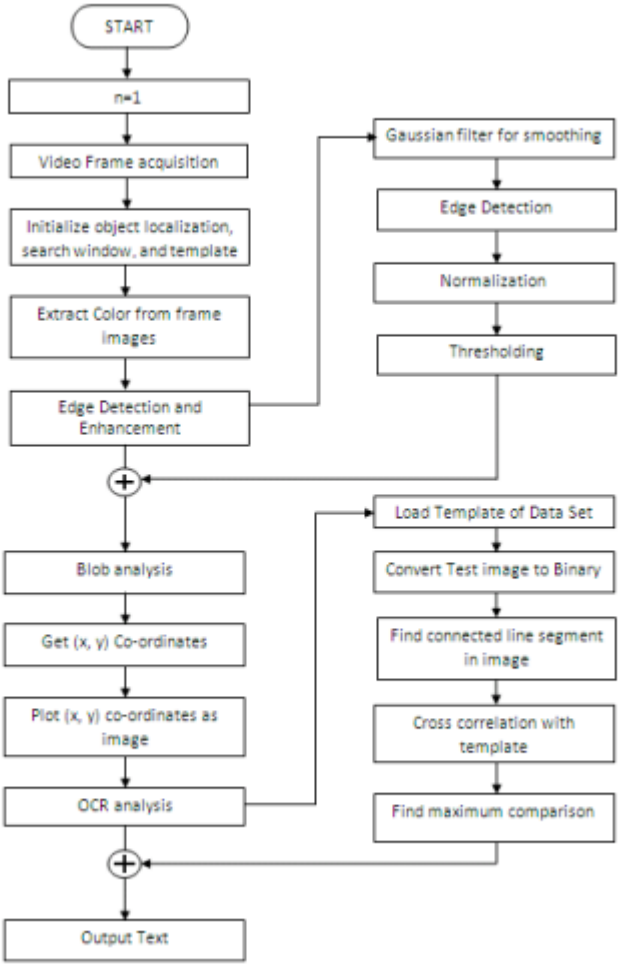
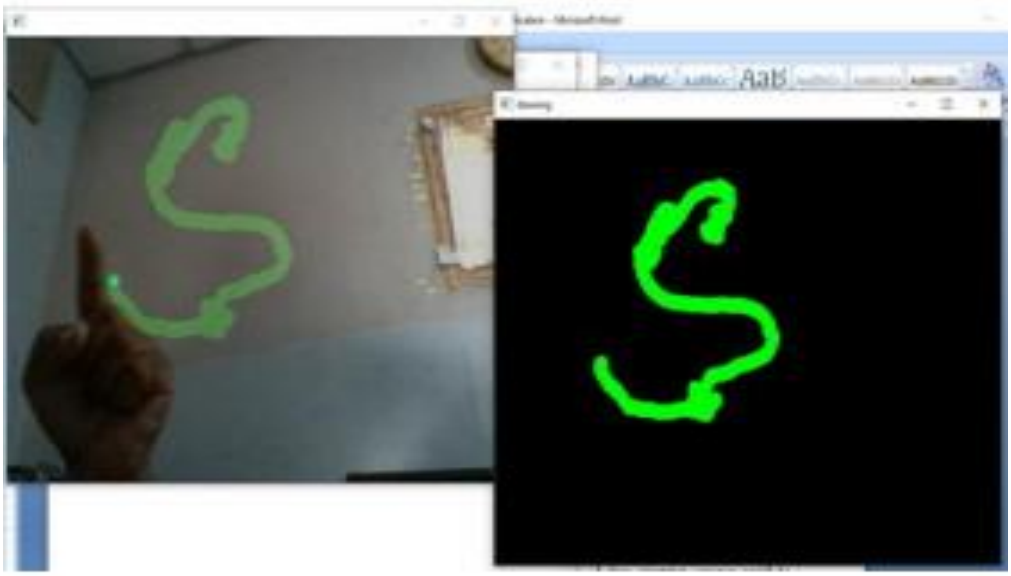


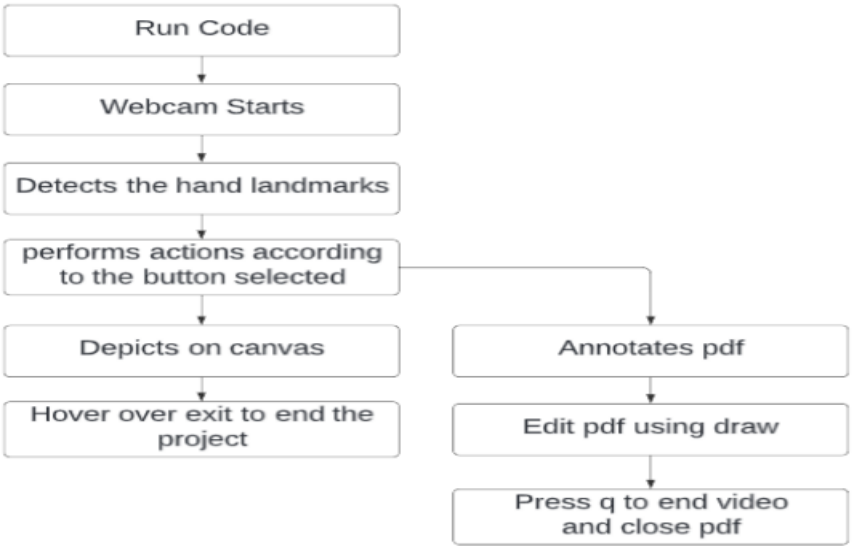
Figure 6. Flowchart

B. LED fitted finger movements Authors in [5] suggested a method in which an LED is mounted on the user's finger, and the web camera is used to track the finger. The character drawn is compared with that present in the database. It returns the alphabet that matches the pattern drawn. It requires a redcoloured LED pointed light source is attached to the finger. Also, it is assumed that there is no red-coloured object other than the LED light within the web camera's focus.

C. Augmented Desk Interface In [6] Augmented segmented desk interface approach for interaction was proposed. This system makes use and a video projector and charge-coupled device (CCD) camera so that using the fingertip; users can operate desktop applications. In this system, each hand performs different tasks. The left hand is used to select radial menus, whereas the right hand is used for selecting objects to be manipulated. It achieves this by

using an infrared camera. Determining the fingertip is computationally expensive, so this system defines search windows for fingertips.





Recognizing multi thumb is a difficult and open topic in computer display due to the different issues addressed in the previous segment. The list consists of previous research that is relevant to this project: Plane recognition fingertip detection and monitoring, as well as existing hand tools related to air-drawing information acknowledgment Fingertip Detection method Based In the fields of human-computer connectivity and virtual reality, fingertip detection, and tracking use color images cameras has been a hot topic (AR). As a preliminary phase for fingertip detection, a model-based technique four for 3Dtechnic hand tracking (de La Gorce et al. Tang

et al) used however, these approaches are computationally expensive and necessitate a large volume of training data. As a consequence, our real-time programming is incompatible with them. After the hand is subdivided using color, working depth, or movement cues, the tips are viewed from the derived binary hand shield in the version scheme. Liang and his colleagues [2] used a range metric from either the back of the hand to both the contour's furthermost points to find candidate fingertip marks. Krejov and Bowden [3] generalized the distance principle by using a geodesic distance for the normal structure of the side. In hand setups where previous techniques failed, this increased the localization of fingertips. Since fingertips have a higher curvature than other parts of the hand, the authors of (Lee and Hollered, 2007) used the contour curvature as a signal to spot fingertips. Inside the device's interaction field, the Motion Sensor has a skeletal tracking function that depends on individual fingertips. As a result, the finger's skeletal joint coordinates can be obtained. As a result, the distal phalange joint coordinates of the human hand are acquired using the Leap Motion Controller sensor. The device can map points and gain coordinates when the user enters an alphanumeric character in free space. "There are many uses for automatic object recognition, including machine learning and social contact. Objects can range from a text to a person being that needs to be monitored. The monitoring algorithm has been implemented in a variety of ways in the literature. Another number of experts used it to decode Sign Languages another for hand gesture recognition, and yet another for text translation and detection, tracing complete body movement of items for virtual reality, and yet another for fingerprint tracking dependent character recognition". Neumann and colleagues devised a system for text identification and localization in real-world images. They used a hypothesis system in their paper that can process several text lines. They also train the algorithm with synthetic scripts, and they take advantage of the Maximally Legitimate Power Regions (MSERs), which provide reliability to geometric but important for quality [3]. Wang et al. have explored a color-based body movement method for both interior and exterior settings in [1]. To detect the item, they used a recording device and a colored shirt in their proposed process. The results of their proposed approach suggest that it can be used in augmented reality implementations. Objects can range from a text to a person being that needs to be monitored. The monitoring algorithm has been implemented in a variety of ways in the literature. Another number of experts used it to decode Sign Languages, another for hand gesture recognition, and yet another for text translation and detection, tracing complete body movement of items for virtual reality [1], and yet another for fingerprint tracking dependent character recognition. The writers of talk about a visual guy that can recognize Japanese

katakana characters in the breeze. They used a TV camera and a Light Emission Diode (LED) pen to monitor the hand gesture. They translate pen action into path codes. To remove the impact of writing speed, the codes are normalized to 100 data pieces, with 46 Japanese characters specified. They reach 92.9 percent character recognition accuracy with a single camera and 9 percent gesture path accuracy with multiple cameras. As previously said, our suggested approach works on the color portion of the frame. It also monitors the whole area for it. Two characters could be distinguished by using tape only on the frontal portion of the right thumb, and after writing every letter, simply rotate the finger (where no tape is present) and write the next letter. When the proposed procedure fails to locate a colored item, it is concluded that one of the characters has been achieved. Another option is to add a delay until each character is written. The waiting time will be m seconds, resulting in duplicate frames of the same features. As previously said, we quantify the difference between two images using the previous frame as a reference image. The smaller the discrepancy value, which in the case of reproduced frames (coordinates of x, y), is normally equivalent to zero, the more accurate one letter is. Both statements are undeniably right.

2.1 FUNCTIONAL SPECIFICATIONS

This system needs a Computer with built-in camera and software like python 3.x and also OpenCV i.e., open-source computer vision library and other required libraries. There are algorithms like object tracking and contour detection in OpenCV which we are using in our project which is expense free and easy to use. We are taking feed from the camera dilating it using dilation function of OpenCV in order to remove impurities from the feed and the we are using masking in order to detect the contour and then generating out output on the screen. Our project canvas is visible in the image below:

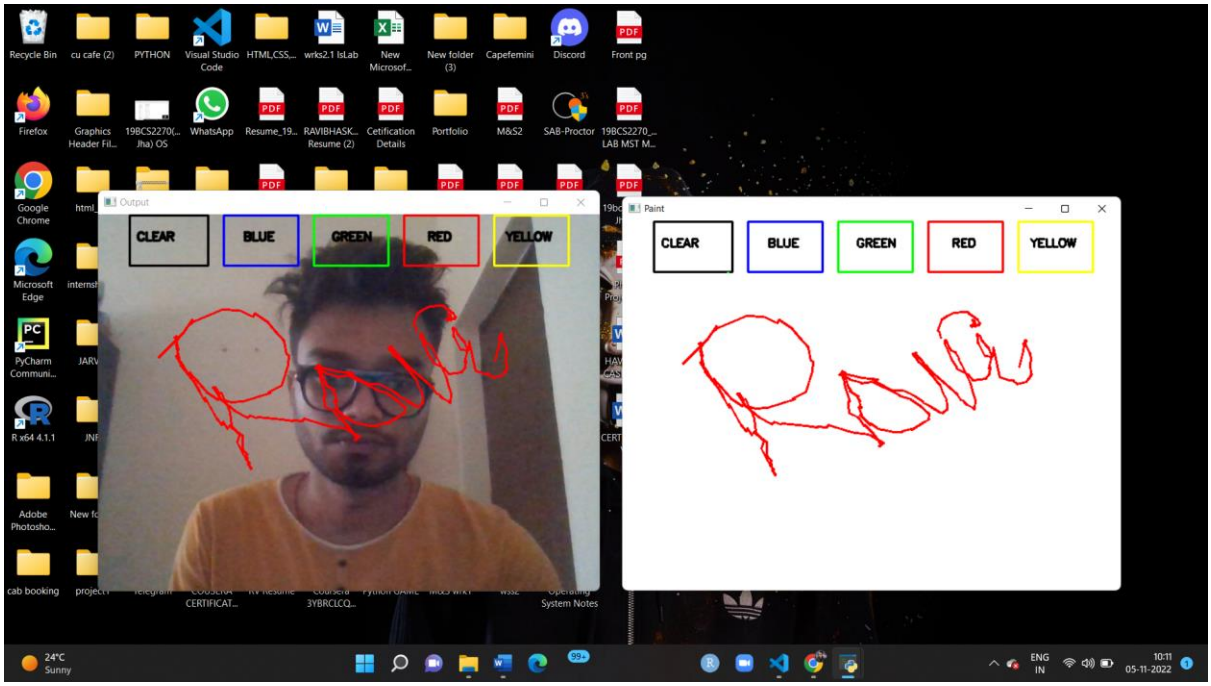
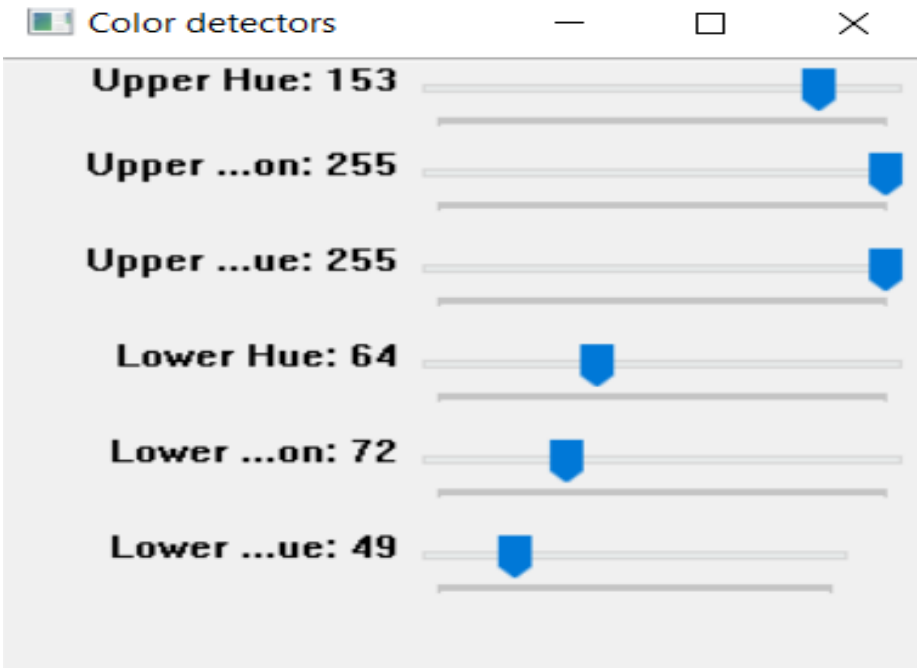


Figure 7. Drawn Canvas



We also have functionalities like colour tracker which provide trackbars or slide bars in order to change colour values for the contour detection.

Comparison:

If we compare all the above mentioned methods/systems, why we choose the third one that we are proposing is because:

1) Our method is not Costly.
2) The use of external hardware is not there in our proposed system.
3) We don't need any kind of LED's in order to generate the output.
4) The time complexity of our project is minimum as compared to the two others listed above.

2.2 SOFTWARE TOOLS SPECIFICATIONS

1. **Operating system – Windows 10**

   Air Canvas is a hands-free digital drawing canvas that utilizes a Raspberry Pi, a PiCamera, and OpenCV to recognize and map hand gestures onto a PiTFT screen. The user's "brush" can be modified in size and color by using built-in buttons. The direction of the brush is controlled completely using open source OpenCV software and modified to map the pointer finger onto the screen using Pygame following a calibration screen to measure and record the color of the user's hand. The idea for Air Canvas was a result of our interest in digital drawing and smart photo recognition software.

. It is comparatively easier to use.

. We don't need any specialized computer education or training to use Windows.

. Wide support for games. Almost all games run on Windows.

. It can run on most of the processors.

. The collection of software programs, utilities and games is relatively much larger for Windows.

2. **Programming Language python** –

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s as a successor

to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects [magic methods]). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It uses dynamic name resolution (late binding), which binds method and variable names during program execution.

Its design offers some support for functional programming in the Lisp tradition. It has filter , map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (Itertools and functools) that implement functional tools borrowed from Haskell and Standard ML

Its core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:[

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than building all of its functionality into its core, Python was designed to be highly extensible via modules. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, wrote: "To describe something as 'clever' is *not* considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C; or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter

VS code –

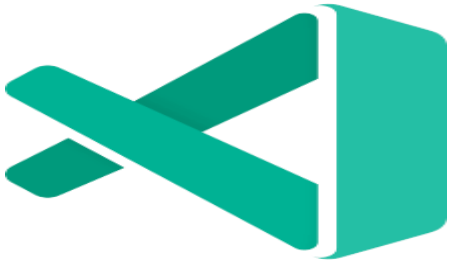
Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include supportfor debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

In the Stack Overflow 2021 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 70% of 82,000 respondents reporting that they use it.

Visual Studio Code is a source-code editor that can be used with a variety of programminglanguages,including Java, JavaScript, Go, Node.js, Python, C++ and Fortran. It is based on the Electron framework, which is used to develop Node.js Web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code

Marketplace.



Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports many programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette.

Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new languages, themes, and debuggers, perform static code analysis, and add code linters using the Language Server Protocol.

Source control is a built-in feature of Visual Studio Code. It has a dedicated tab inside of the menu bar where you can access version control settings and view changes made to the current project. To use the feature you must link Visual Studio Code to any supported version control system (Git, Apache Subversion, Perforce, etc.). This allows you to create repositories as well as to make push and pull requests directly from the Visual Studio Code program.

Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

Visual Studio Code allows users to set the code page in which the active document is saved, the newline character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language.

Visual Studio Code collects usage data and sends it to Microsoft, although this can be disabled. Due to the open-source nature of the application, the telemetry code is accessible to the public, who can see exactly what is collected.

We will utilize the PC vision strategies of OpenCV to assemble this venture. The favoured

language is Python because of its comprehensive libraries and simple to utilize grammar however understanding the fundamentals it tends to be executed in any OpenCV upheld language.

Here Color Detection and following are utilized to accomplish the goal. The variety marker is distinguished and a veil is delivered. It remembers the further strides of morphological activities for the veil delivered which are Erosion and Dilation. Disintegration lessens the pollutants present in the cover and expansion further re-establishes the dissolved fundamental veil.

2.3 Algorithm:

1. Start reading the frames and convert the captured frames to HSV color space (Easy for color detection).

2. Prepare the canvas frame and put the respective ink buttons on it.

3. Adjust the track bar values for finding the mask of the colored marker.

4. Preprocess the mask with morphological operations (Eroding and dilation).

5. Detect the contours, find the center coordinates of largest contour and keep storing them in the array for successive frames (Arrays for drawing points on canvas).

6. Finally draw the points stored in an array on the frames and canvas.

2.4 REQUIREMENTS

1.Python – 3.x (We used 3.8.8 for this project):

Python 3.0 is end-of-lifed with the release of Python 3.1. All users of Python 3.0. x should upgrade to the most recent version of Python 3; please see the downloads pages. Python 3.0 has been replaced by a newer bugfix release of Python.It supports functional and structured programming methods as well as OOP. It can be used as a scripting language or can be compiled to byte-code for building large applications. It provides very high-level dynamic data types and supports dynamic type checking.The preferred language is Python due to its

exhaustive libraries and easy to use syntax but understanding the basics it can be implemented in any OpenCV supported language. Here Color Detection and tracking are used in order to achieve the objective. The color marker is detected and a mask is produced. It includes the further steps of morphological operations on the mask produced which are Erosion and Dilation. Erosion reduces the impurities present in the mask and dilation further restores the eroded main mask.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Characteristics of Python

Following are important characteristics of python −

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.

- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is one of the most widely used language over the web. I'm going to list few of them here:

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

2. OpenCV – 4.4. Run “pip install opencv-python opencv_contrib-python” to install the package.

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your

Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Learning OpenCV will teach any developer or hobbyist to use the framework quickly with the help of hands-on exercises in each chapter.

OpenCV is a great tool for image processing and performing computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more. It supports multiple languages including python, java C++.

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies. Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and

New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

3.Numpy – 1.20.1-

Numpy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project. NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents due to the absence of compiler optimization. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

2.5 OBJECTIVES

1. To create a simple prototype for a drawing tool that uses hand gesture recognition software
2. Using OpenCV to recognize the contour.
3. Mapping counts of contour extracted from hand canvas to tracking algorithm to produce a drawing.
4. Implementing additional features such as color change etc.

Using contour detection, we can detect the borders of objects, and localize them easily in an image. It is often the first step for many interesting applications, such as image-foreground extraction, simple-image segmentation, detection and recognition.

Colour Tracking of Object at fingertip. First of all, The incoming image from the webcam is to be converted to the HSV colour space for detecting the colored object at the tip of finger. The below code snippet converts the incoming image to the HSV space, which is very suitable and perfect color space for Color tracking.Now, We will make the Trackbars to arrange the HSV values to the required range of color of the colored object that we have placed at our finger.ow, When the trackbars are setup, we will get the realtime value from the trackbars and create range. This range is a numpy structure which is used to be passed in the function cv2.inrange( ). This function returns the Mask on the colored object. This Mask is a black and white image with white pixels at the position of the desired color. 2. Contour Detection of the Mask of Color Object Now, After detecting the Mask in Air Canvas, Now is the time to locate its center position for drawing the Line. Here, In the below Snippet of Code, We are performing some morphological operations on the Mask, to make it free of impurities and to detect contour easily. Drawing the Line using the position of Contour Now Comes the real logic behind this Computer Vision project, We will form a python deque (A data Structure). The deque will store the position of the contour on each successive frame and we will use these stored points to

make a line using OpenCV drawing functions. Now, we will use the position of the contour to make decision, if we want to click on a button or we want to draw on the sheet. We have arranged some of the buttons on the top of Canvas, if the pointer comes into their area, we will trigger their method. We have four buttons on the canvas, drawn using OpenCV. Clear : Which clears the screen by emptying the deques. Red : Changes the marker to red color using color array. Green : Changes the marker to Green color using color array. Yellow : Changes the marker to Yellow color using color array. Blue : Changes the marker to Blue color using color array. Also, to avoid drawing when contour is not present, We will Put a else condition which will capture that instant.

2.6 GOALS

The basic goal of Air Canvas is to map the coordinates of the user's pointer finger to the screen, where colored circles are drawn and connected to simulate a crude brush stroke.

Air Canvas is a contribution to the education industry also making an impact on reduce, recuse and recycle chain by introducing a method of teaching which just requires a teacher to teach. No more use of plastic made markers or environment harming chalks. You got everything at your fingertips.

The main goal of this project is that there will be no use of paper or any other thing. The wastage of paper would be less.

1. People hearing impairment: Although we take hearing and listening for granted, they communicate using sign languages. Most of the world can't understand their feeling, their emotions without a translator in between.

2. Overuse of Smartphones: They cause accidents, depression, distractions, and other illnesses that we humans can still discover. Although its portability, ease to use is profoundly admired, the negatives include life threatening events.

3. Paper wastage is not scarce news. We waste a lot of paper in scribbling, writing, drawing, etc.… Some basic facts include - 5 liters of water on average are required to make one A4 size paper, 93% of writing is from trees, 50% of business waste is paper, 25% landfill is paper, and the list goes on. Paper wastage is harming the environment by using water and trees and creates tons of garbage. Air Writing can quickly solve these issues. It will act as a communication tool for people with hearing impairment. Their air-written text can be presented using AR or converted to speech. One can quickly write in the air and continue with your work without much distraction. Additionally, writing in the air does not require paper. Everything is stored electronically.

CHAPTER 3

DESIGN FLOW/PROCESS

3.1 Concept Generation

Air Canvas is a hands-free digital drawing canvas that utilizes a Camera, and OpenCV to recognize and map hand gestures onto a screen. The user's "brush" or we can say object/Contour can be modified in size and colour by using built-in buttons. The direction of the brush is controlled completely using open source OpenCV software and modified to map the pointer finger onto the screen using OpenCV library algorithms following a calibration screen to measure and record the colour of the user's hand/Contour. The idea for Air Canvas was a result of human's interest in digital drawing and smart photo recognition software.

3.2 Evaluation & Selection of Specifications/Features

1. Can track any specific-coloured pointer.
2. User can draw in four different colours and even change them without any Hussle.
3. Able to rub the board with a single location at the top of the screen.
4. No need to touch the computer once the program is run

3.3 Design

Design Overview

The basic goal of Air Canvas is to map the coordinates of the user's pointer finger to the screen, where coloured circles are drawn and connected to simulate a crude brush stroke. Our project consisted of three hardware devices: a Camera, a display screen, and a Contour. The Screen is connected to the camera and the camera is taking the feed and then our programme is processing it and displaying the output on the screen .We designated physical buttons as program controls on the canvas for colour change and also to clear the screen:

A basic visual of Air Canvas can be seen in Figure below:

Figure 5. Air Canvas

3.4 Design Flow

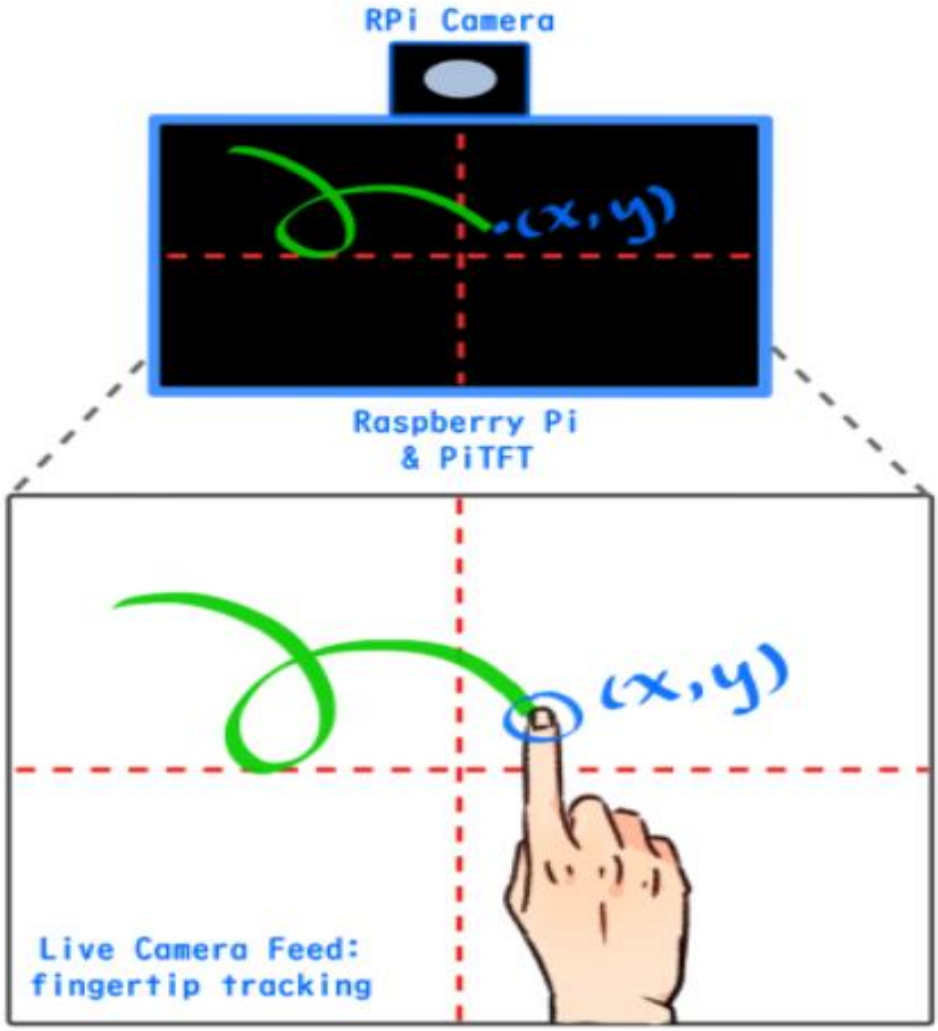
Alternative Designs to make the project

1. The first one is using different kind of hardware's which is a little costly and require two to three hardware equipment's eg.:

This approach of the project consists of three hardware devices: a Raspberry Pi board, a PiTFT display screen, and a Raspberry Pi Camera. The PiTFT screen is connected to the RPi's 40-pin connector, and the camera module is attached directly into the camera port. We designated each of the PiTFT's physical buttons as program controls:

- Pin 17: color change between red, green, and blue
- Pin 22: increase brush size
- Pin 23: decrease brush size
- Pin 27: quit/end program

A basic visual of Air Canvas can be se



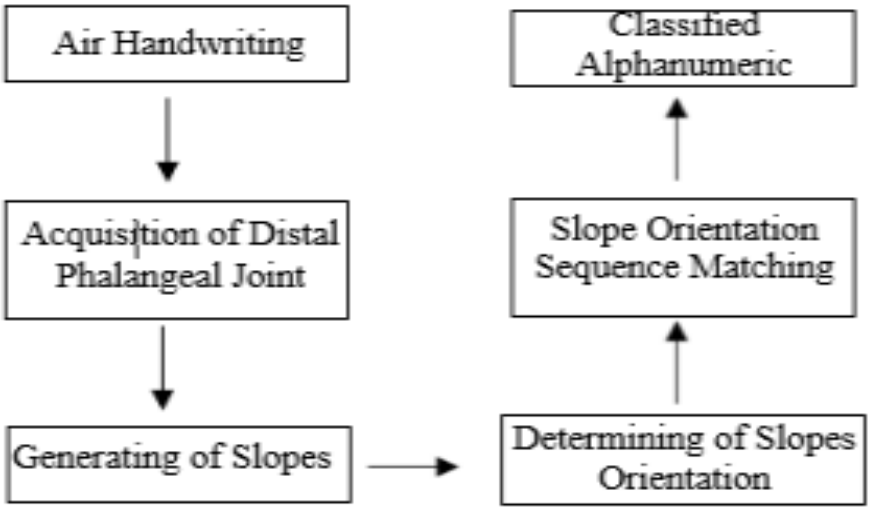
en in Figure 1:

Figure 6. Hardware Implemented Design

2. The Second alternative approach is using Slope detection to gain direction method eg**.**

In this the slope formula is applied to the two consecutive main points, which are made up of x and y coordinates. The value of change in x-coordinates and change in y-coordinates determines the slope of a line. The horizontal length of the line formed by two consecutive major points is defined by the change in x-coordinates, while the vertical length is determined by the change in y-coordinates. The slope orientation is

determined by the acquired slope, which is classified into eight categories: Upward (U) Right (R) Downward (D) Left (L) RightUp (RU), Absolutely correct (RD), Decided to leave (LD), and ND (LU) (LU). Straight lines are difficult to input while using a leap motion sensor. As a result, the researchers looked at the average variety of angles used by five users when drawing straight lines - up, right, down, and left - in five trials each.

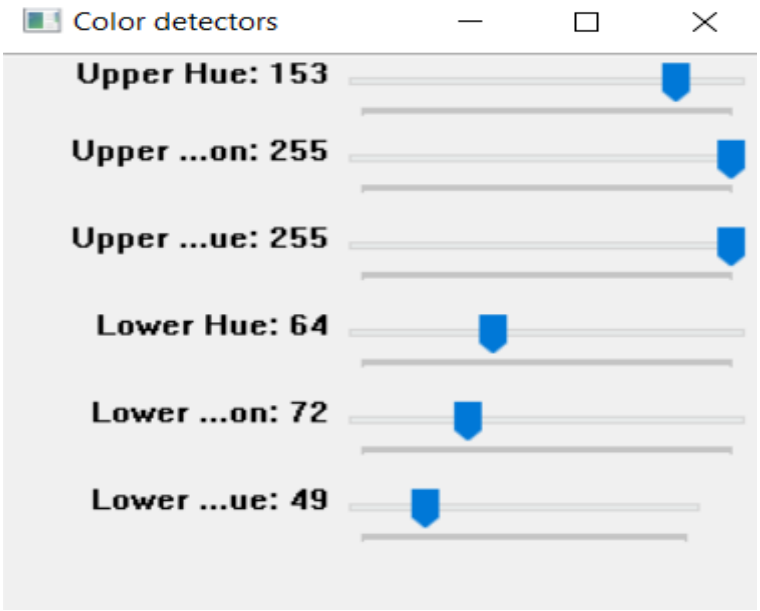


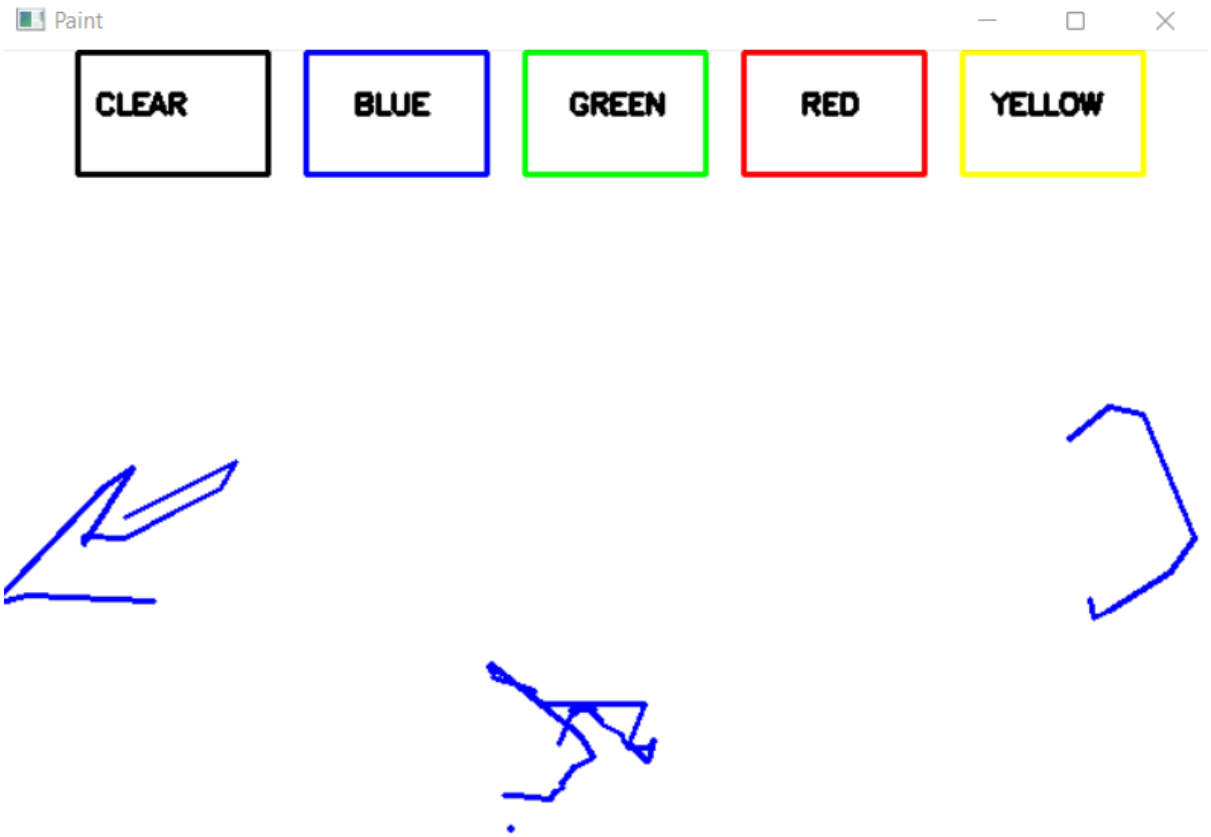
You'll see a text that says "Draw: s" in the first detection panel. The number "s" is created at random, and you must draw it on the canvas. Since the 'object' is a green bottle-cap, a text will appear at the top of the webcam that says, "Object Found." Another text would appear in the lower region of the detection screen, indicating the object's area. The size of the pen will be proportional to the size of the area. To change the pen height, move the environmentally friendly object nearer to the camera (to increase) or further away from the webcam (to decrease).

3. The Third approach is using contour detection method and object tracking which is the method that we have chosen eg.:

This system needs a laptop with built-in camera and software like python 3.x and also OpenCV i.e., open-source computer vision library and other required libraries. There are algorithms like object tracking and contour detection in OpenCV which we are using in our project which is expense free and easy to use. We are taking feed from the camera dilating it using dilation function of OpenCV in order to remove impurities from

the feed and the we are using masking in order to detect the contour and then generating out output on the screen. Our project canvas is visible in the image below:





We also have functionalities like colour tracker which provide trackbars or slide bars in order to change colour values for the contour detection.

Comparison:

If we compare all the above three mentioned methods/systems why we choose the third one that we are proposing is because:

5) There is no expense at all.
6) The use of external hardware is not there in our proposed system.
7) We don't need any kind of Datasets in order to compare our input to generate the output.
8) There is a limited use of mathematical formula's which are time consuming.
9) The time complexity of our project is minimum as compared to the two others listed above.

3.5 Design Issues

OpenCV Trial and Error

Our first and foremost challenge was understanding OpenCV. At first, while experimenting with various open-source programs, we were using an older version of OpenCV. To our dismay, heavy image processing dropped our frame rate to around 4 frames per second, even when we only applied a few basic filters, and the lag on the camera feed was unbearably slow. Additionally, our desired open-source program by Pandy was only compatible with the newest version of OpenCV. We decided to upgrade our OpenCV, and doing so increased our frame rate dramatically. While our program still showed considerable lag, it was fast enough to be considered functional.

The Jitters and the Lab Gloves

We encountered difficulties with hand contour detection, which in turn affected our ability to draw smoothly on the PiTFT. As mentioned in the design description, occasional erroneous calculations caused the farthest point coordinate to leap erratically to the edges of the screen.

The addition of threshold conditions and usage of blue gloves helped reduce the causes of detection jitters, but we also discovered another bug in the process. Since the algorithm looks for the farthest point from the center of the contour, a wrist-length glove triggers the program to register the cuff of the glove as the farthest point instead of the finger. This was discovered during much trial and error, when the highlighted current point frequently bounced to the wrist area. There was not much code correction we could do for this, so we decided on simply rolling up the cuff to limit the glove to only the fingers and the palm. Doing so solved the rest of our jitter problems, but this specific bug opens up room for improvement of the hand detection algorithm in OpenCV.

Other Issues

In this section, we discuss issues that arise in tracking objects in realistic scenarios. These include locating objects as they undergo occlusion and keeping unique tracks of objects as they are viewed through multiple cameras.

Occlusion can be classified into three categories: self occlusion, interobject occlusion, and occlusion by the background scene structure. Self occlusion occurs when one part of the object occludes another. This situation most frequently arises while tracking articulated objects. Interobject occlusion occurs when two objects being tracked occlude each other. Similarly, occlusion by the background occurs when a structure in the background occludes the tracked objects. Generally, for interobject occlusion, the multiobject trackers like MacCormick and Blake [2000] and Elgammal et al. [2002] can exploit the knowledge of the position and the appearance of the occluder and occludee to detect and resolve occlusion. Partial occlusion of an object by a scene structure is hard to detect since it is difficult to differentiate between the object changing its shape and the object getting occluded. A common approach to handle complete occlusion during tracking is to model the object motion by linear dynamic models or by nonlinear dynamics and, in the case of occlusion, to keep on predicting the object location until the object reappears. For example, a linear velocity model is used in Beymer and Konolige [1999] and a Kalman filter is used for estimating the location and motion of objects. A nonlinear dynamic model is used in Isard and MacCormick [2001] and a particle filter employed for state estimation. Researchers have also utilized other features to resolve occlusion, for example, silhouette projections [Haritaoglu et al. 2000] (to locate persons' heads during partial occlusion), and optical flow [Dockstader and Tekalp 2001b] (assuming that two

objects move in opposite directions). Occlusion can also be implicitly resolved during generation of object tracks. In 2004, Sato and Aggarwal fit a slant cylindrical to the TSV (see Section 5.2 for more details) in the spatio-temporal space to recover from occlusions. In particular, two continuous slant cylinders, which are fit to the tracked object silhouette before the occlusion and after the occlusion, will intersect with each other for the frames whre the occlusion occurred. This intersection provides continuous trajectory before, during, and after the occlusion. Free-form object contour trackers employ a different occlusion resolution approach. These methods usually address occlusion by using shape priors which are either built ahead of time [Cremers et al. 2002] or built online [Yilmaz et al. 2004]. In particular, Cremers et al. [2002] build a shape model from subspace analysis (PCA) of possible object shapes to fill in missing contour parts. Yilmaz et al. [2004] build online shape priors using a mixture model based on the level set contour representation. Their approach is able to handle complete object occlusion. The chance of occlusion can be reduced by an appropriate selection of camera positions. For instance, if the cameras are mounted on airborne vehicles, that is, when a birds-eye view of the scene is available, occlusions between objects on the ground do not occur. However, oblique view cameras are likely to encounter multiple object occlusions and require occlusion handling mechanisms. Multiple cameras viewing the same scene can also be used to resolve object occlusions during tracking [Dockstader and Tekalp 2001a; Mittal and Davis 2003].

3.7 Implementation Plan

In this project, we will learn to build an Air Canvas which can draw anything on it by just capturing the motion of a colored object with a camera. Here a colored object at the tip of the finger is used as the marker.

We will be using the computer vision techniques of OpenCV to build this project. The preferred language is Python due to its exhaustive libraries and easy to use syntax but understanding the basics it can be implemented in any OpenCV supported language.

Here Color Detection and tracking are used in order to achieve the objective. The color marker is detected and a mask is produced. It includes the further steps of morphological

operations on the mask produced which are Erosion and Dilation. Erosion reduces the impurities present in the mask and dilation furtherrestores the eroded main mask.

Algorithm:

1. Start reading the frames and convert the captured frames to HSV color space (Easy for color detection).
2. Prepare the canvas frame and put the respective ink buttons on it.
3. Adjust the track bar values for finding the mask of the colored marker.
4. Preprocess the mask with morphological operations (Eroding and dilation).
5. Detect the contours, find the center coordinates of largest contour and keep storing them in the array for successive frames (Arrays for drawing points on canvas).
6. Finally draw the points stored in an array on the frames and canvas.

FLOWCHARTS:

A flowchart is a picture of the separate steps of a process in sequential order. It is a generic tool that can be adapted for a wide variety of purposes, and can be used to describe various processes, such as a manufacturing process, an administrative or service process, or a project plan, Flowchart for our Proposed System:

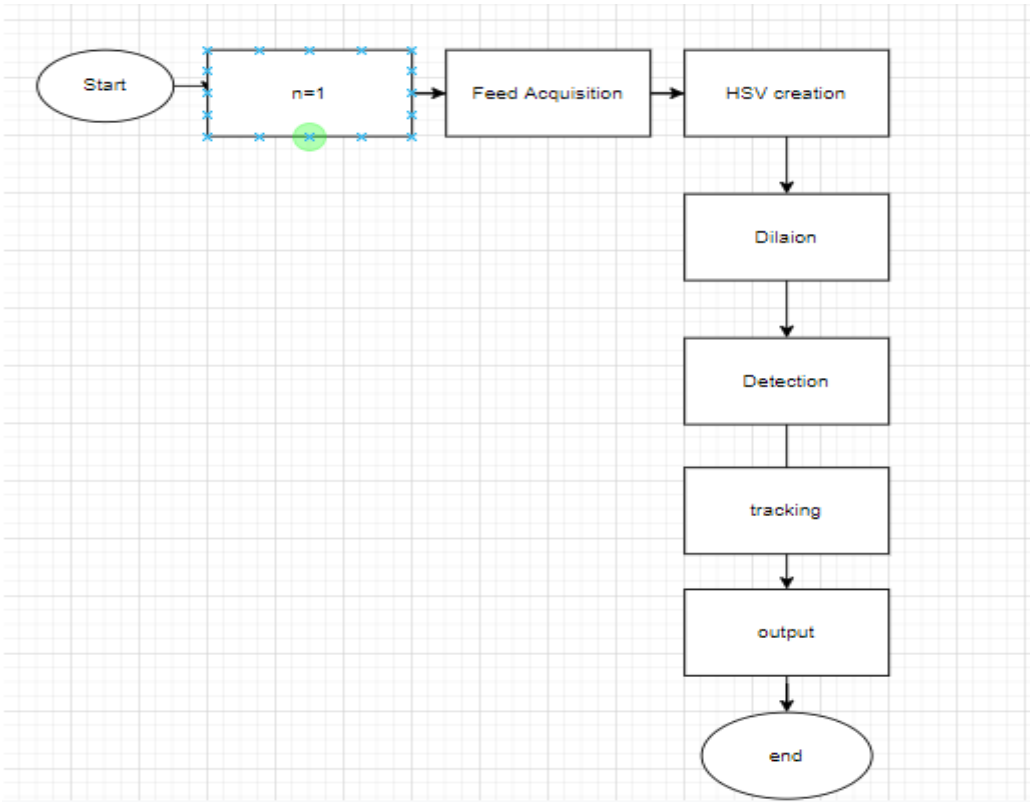


Figure. Proposed System

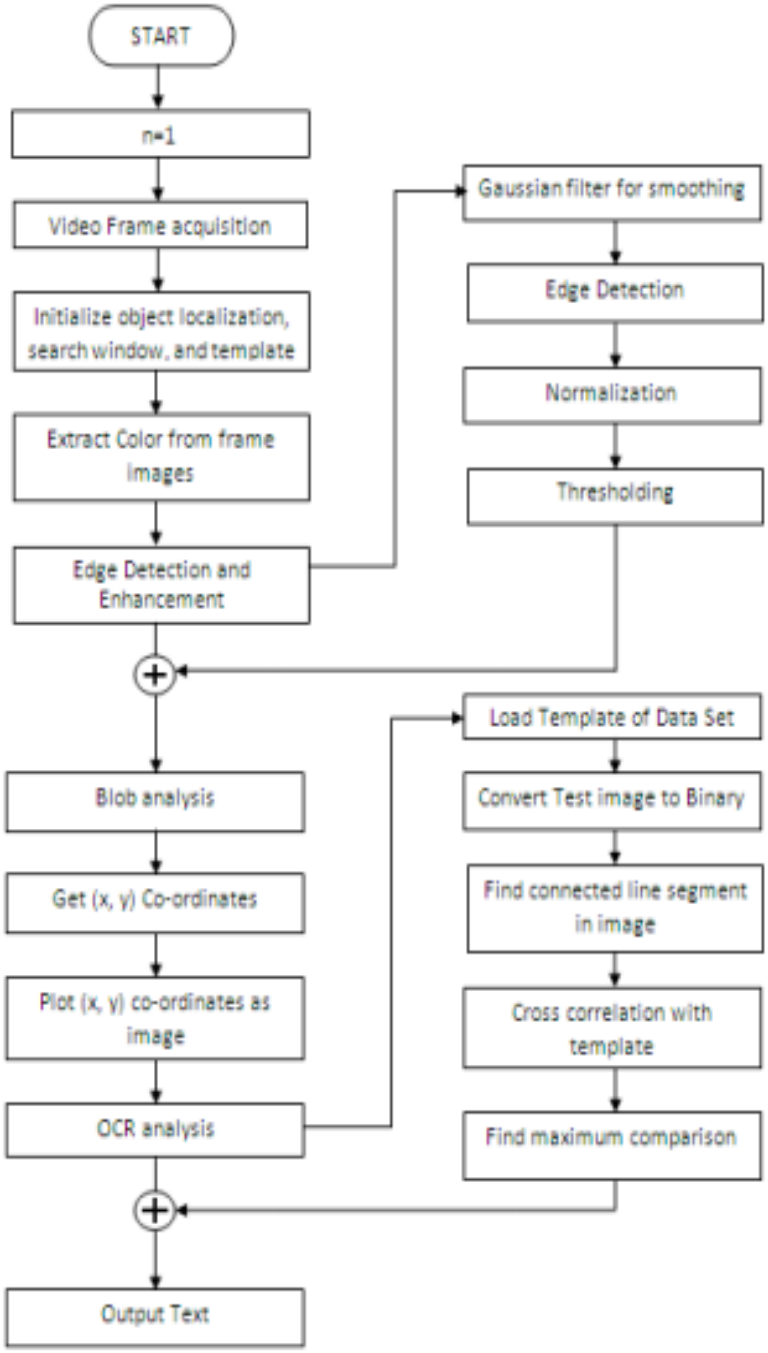


Figure 8. Alternative System Approach.

METHODOLOGY:

1.Colour Tracking of Object at fingertip.

First of all, The incoming image from the webcam is to be converted to the HSV color space for detecting the colored object at the tip of finger. The source code converts the incoming image from the feed to the HSV space, which is very suitable and perfect color space for color tracking. Now, we'll make the Trackbars to rearrange the HSV values to the specified range of color of the colored object that we've placed at our finger. Now, When the trackbars are setup, we'll get the Realtime value from the trackbars and make range. This range may be a NumPy structure which is employed to be passed within the function cv2.inrange( ). This function gives us back the Mask on or of the colored object. This Mask may be a black and white image with white pixels at the position of the desired color.

2. Drawing the road using the position of Contour

Now Comes the important logic behind this Computer Vision project, we'll form a python deque (A data Structure). The deque will store the position of the contour on each successive frame and that we will use these stored points to form a line using OpenCV drawing functions. Now, we'll use the position of the contour to form decision, if we want to click on a button or we would like to draw on the sheet. We have arranged a number of the buttons on the highest of Canvas, if the pointer comes into their area, we'll trigger their method.

Clear sc : Clears the screen by emptying deques.
Red: Changes the marker to Red using array.
Green: Changes the marker to Green using array.
Yellow: Changes the marker to Yellow using array.

SOURCE CODE:

```
# All the imports go here
import cv2
import numpy as np
import mediapipe as mp
from collections import deque


# Giving different arrays to handle colour points of different colour
```

```
bpoints = [deque(maxlen=1024)]
gpoints = [deque(maxlen=1024)]
rpoints = [deque(maxlen=1024)]
ypoints = [deque(maxlen=1024)]


# These indexes will be used to mark the points in particular arrays of spe-
cific colour
blue_index = 0
green_index = 0
red_index = 0
yellow_index = 0

#The kernel to be used for dilation purpose
kernel = np.ones((5,5),np.uint8)

colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (0, 255, 255)]
colorIndex = 0

# Here is code for Canvas setup
paintWindow = np.zeros((471,636,3)) + 255
paintWindow = cv2.rectangle(paintWindow, (40,1), (140,65), (0,0,0), 2)
paintWindow = cv2.rectangle(paintWindow, (160,1), (255,65), (255,0,0), 2)
paintWindow = cv2.rectangle(paintWindow, (275,1), (370,65), (0,255,0), 2)
paintWindow = cv2.rectangle(paintWindow, (390,1), (485,65), (0,0,255), 2)
paintWindow = cv2.rectangle(paintWindow, (505,1), (600,65), (0,255,255), 2)

cv2.putText(paintWindow, "CLEAR", (49, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,
0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "BLUE", (185, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,
0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "GREEN", (298, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "RED", (420, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,
0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "YELLOW", (520, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(0, 0, 0), 2, cv2.LINE_AA)
cv2.namedWindow('Paint', cv2.WINDOW_AUTOSIZE)


# initialize mediapipe
mpHands = mp.solutions.hands
hands = mpHands.Hands(max_num_hands=1, min_detection_confidence=0.7)
mpDraw = mp.solutions.drawing_utils


# Initialize the webcam
cap = cv2.VideoCapture(0)
ret = True
```

```
while ret:
    # Read each frame from the webcam
    ret, frame = cap.read()

    x, y, c = frame.shape

    # Flip the frame vertically
    frame = cv2.flip(frame, 1)
    #hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    frame = cv2.rectangle(frame, (40,1), (140,65), (0,0,0), 2)
    frame = cv2.rectangle(frame, (160,1), (255,65), (255,0,0), 2)
    frame = cv2.rectangle(frame, (275,1), (370,65), (0,255,0), 2)
    frame = cv2.rectangle(frame, (390,1), (485,65), (0,0,255), 2)
    frame = cv2.rectangle(frame, (505,1), (600,65), (0,255,255), 2)
    cv2.putText(frame, "CLEAR", (49, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,
0, 0), 2, cv2.LINE_AA)
    cv2.putText(frame, "BLUE", (185, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,
0, 0), 2, cv2.LINE_AA)
    cv2.putText(frame, "GREEN", (298, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,
0, 0), 2, cv2.LINE_AA)
    cv2.putText(frame, "RED", (420, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0,
0), 2, cv2.LINE_AA)
    cv2.putText(frame, "YELLOW", (520, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,
0, 0), 2, cv2.LINE_AA)
    #frame = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)

    # Get hand landmark prediction
    result = hands.process(framergb)

    # post process the result
    if result.multi_hand_landmarks:
        landmarks = []
        for handslms in result.multi_hand_landmarks:
            for lm in handslms.landmark:
                # # print(id, lm)
                # print(lm.x)
                # print(lm.y)
                lmx = int(lm.x * 640)
                lmy = int(lm.y * 480)

                landmarks.append([lmx, lmy])


            # Drawing landmarks on frames
            mpDraw.draw_landmarks(frame, handslms, mpHands.HAND_CONNECTIONS)
        fore_finger = (landmarks[8][0],landmarks[8][1])
```

```
        center = fore_finger
        thumb = (landmarks[4][0],landmarks[4][1])
        cv2.circle(frame, center, 3, (0,255,0),-1)
        print(center[1]-thumb[1])
        if (thumb[1]-center[1]<30):
            bpoints.append(deque(maxlen=512))
            blue_index += 1
            gpoints.append(deque(maxlen=512))
            green_index += 1
            rpoints.append(deque(maxlen=512))
            red_index += 1
            ypoints.append(deque(maxlen=512))
            yellow_index += 1

        elif center[1] <= 65:
            if 40 <= center[0] <= 140: # Clear Button
                bpoints = [deque(maxlen=512)]
                gpoints = [deque(maxlen=512)]
                rpoints = [deque(maxlen=512)]
                ypoints = [deque(maxlen=512)]

                blue_index = 0
                green_index = 0
                red_index = 0
                yellow_index = 0

                paintWindow[67:,:,:] = 255
            elif 160 <= center[0] <= 255:
                    colorIndex = 0 # Blue
            elif 275 <= center[0] <= 370:
                    colorIndex = 1 # Green
            elif 390 <= center[0] <= 485:
                    colorIndex = 2 #
            elif 505 <= center[0] <= 600:
                    colorIndex = 3 # Yellow
        else :
            if colorIndex == 0:
                bpoints[blue_index].appendleft(center)
            elif colorIndex == 1:
                gpoints[green_index].appendleft(center)
            elif colorIndex == 2:
                rpoints[red_index].appendleft(center)
            elif colorIndex == 3:
                ypoints[yellow_index].appendleft(center)
    # Append the next deques when nothing is detected to avois messing up
    else:
        bpoints.append(deque(maxlen=512))
        blue_index += 1
```

```
        gpoints.append(deque(maxlen=512))
        green_index += 1
        rpoints.append(deque(maxlen=512))
        red_index += 1
        ypoints.append(deque(maxlen=512))
        yellow_index += 1

    # Draw lines of all the colors on the canvas and frame
    points = [bpoints, gpoints, rpoints, ypoints]
    # for j in range(len(points[0])):
    #         for k in range(1, len(points[0][j])):
    #             if points[0][j][k - 1] is None or points[0][j][k] is None:
    #                 continue
    #             cv2.line(paintWindow, points[0][j][k - 1], points[0][j][k],
colors[0], 2)
    for i in range(len(points)):
        for j in range(len(points[i])):
            for k in range(1, len(points[i][j])):
                if points[i][j][k - 1] is None or points[i][j][k] is None:
                    continue
                cv2.line(frame, points[i][j][k - 1], points[i][j][k], col-
ors[i], 2)
                cv2.line(paintWindow, points[i][j][k - 1], points[i][j][k],
colors[i], 2)

    cv2.imshow("Output", frame)
    cv2.imshow("Paint", paintWindow)

    if cv2.waitKey(1) == ord('q'):
        break

# release the webcam and destroy all active windows
cap.release()
cv2.destroyAllWindows()
```

Screenshots/Output:

Figure Air Canvas

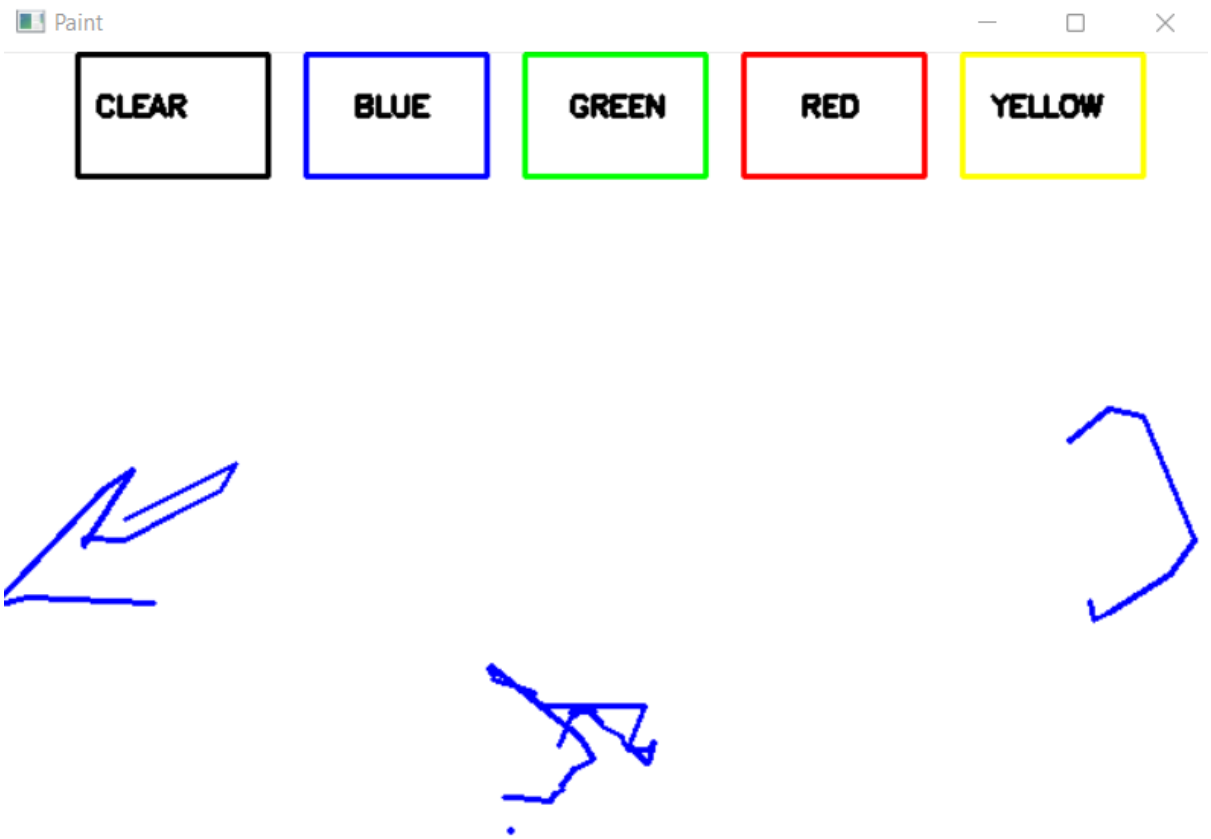


Figure. Tracking

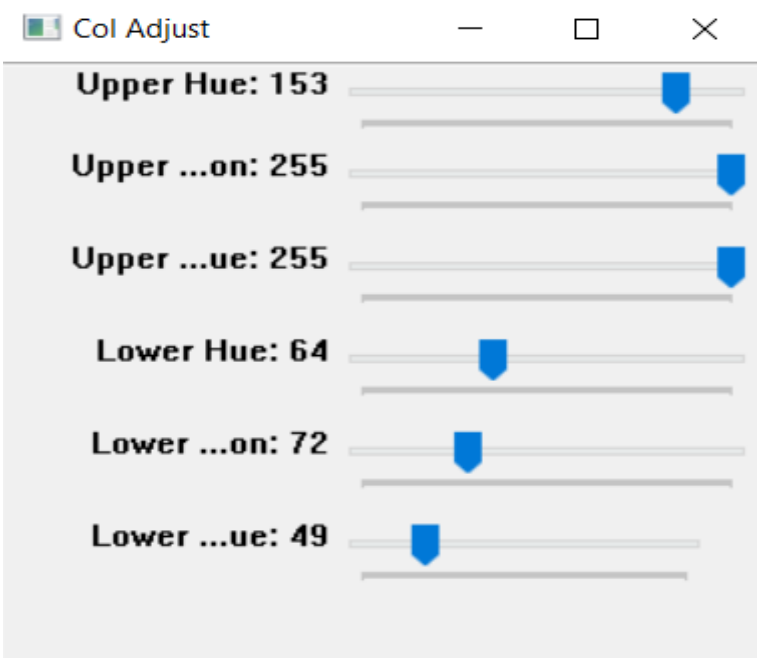


Figure. Colour Adjustor

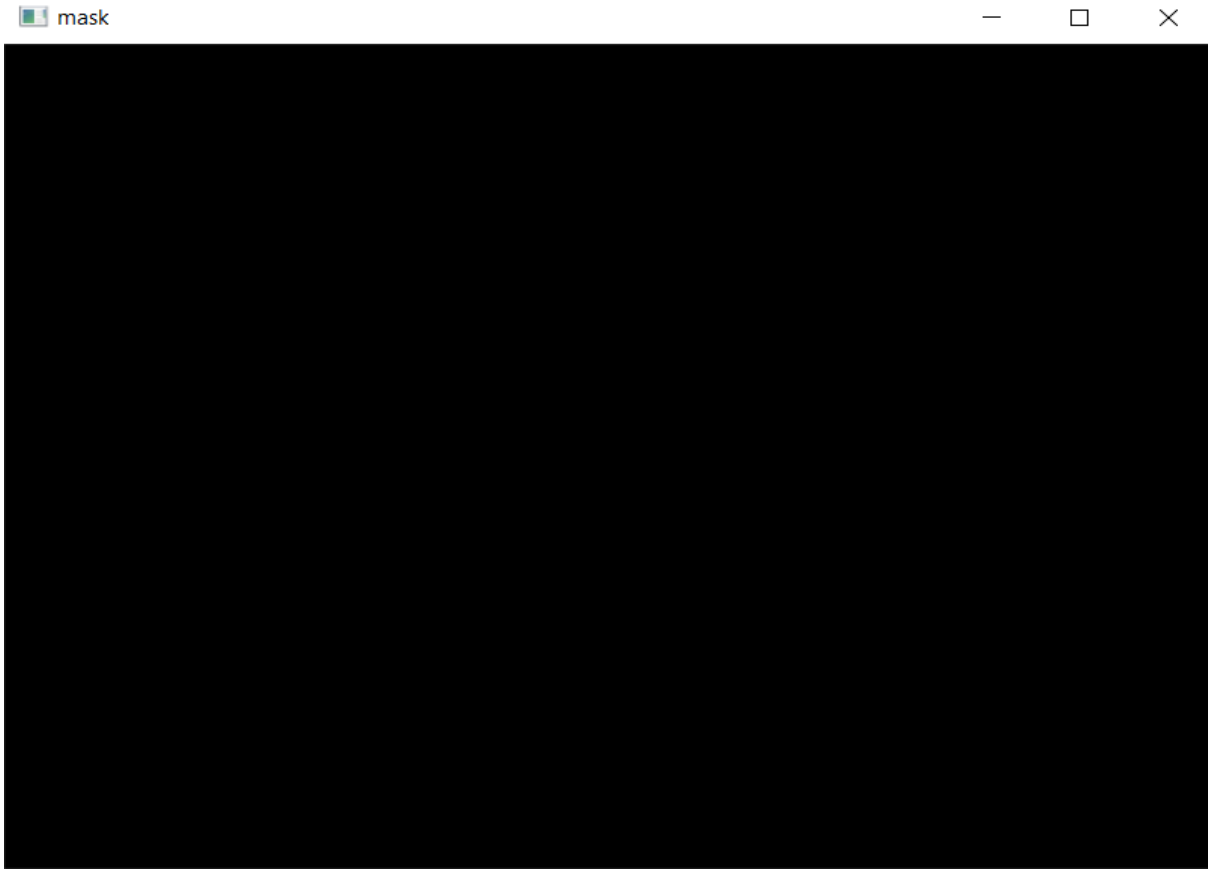


Figure. Masking

Figure. contour Detection

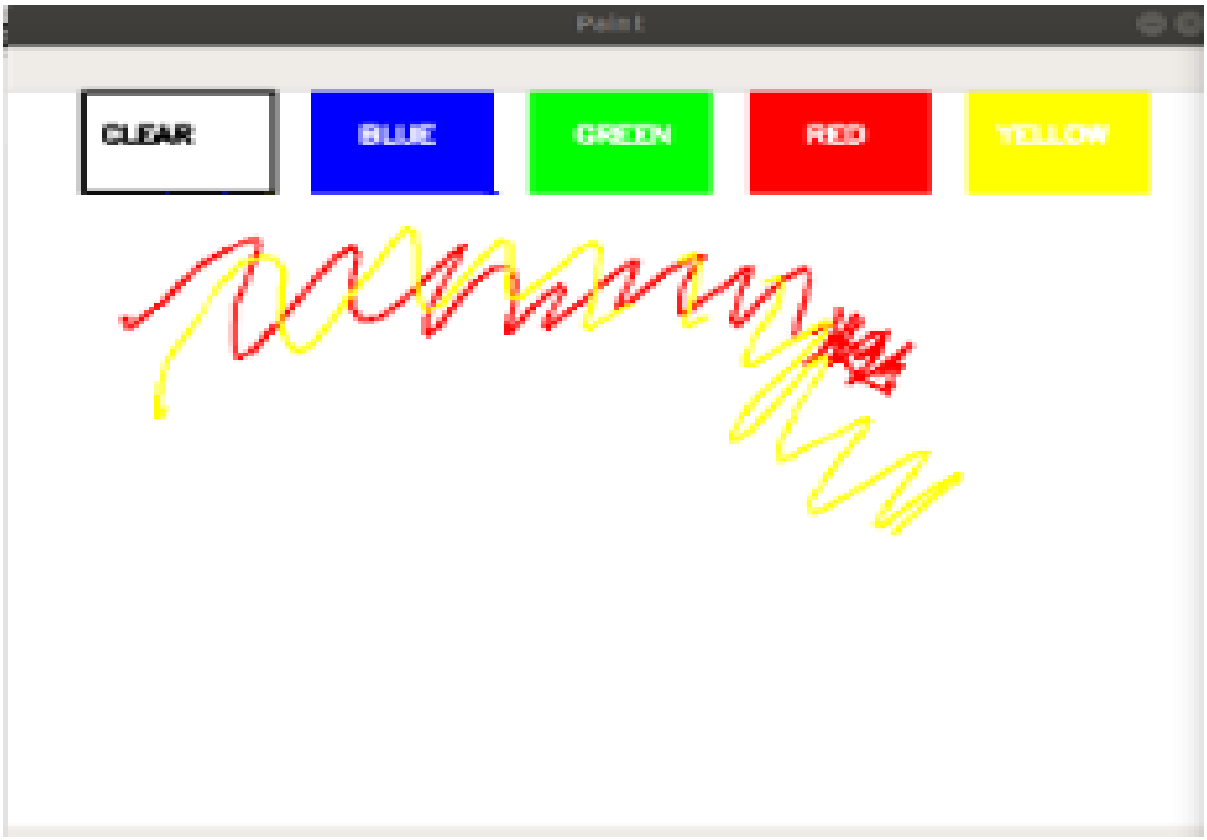


Figure. Canvas Drawing

BLOCK DIAGRAM:

A block diagram is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks. They are heavily used in engineering in hardware design, electronic design, software design, and process flow diagrams.

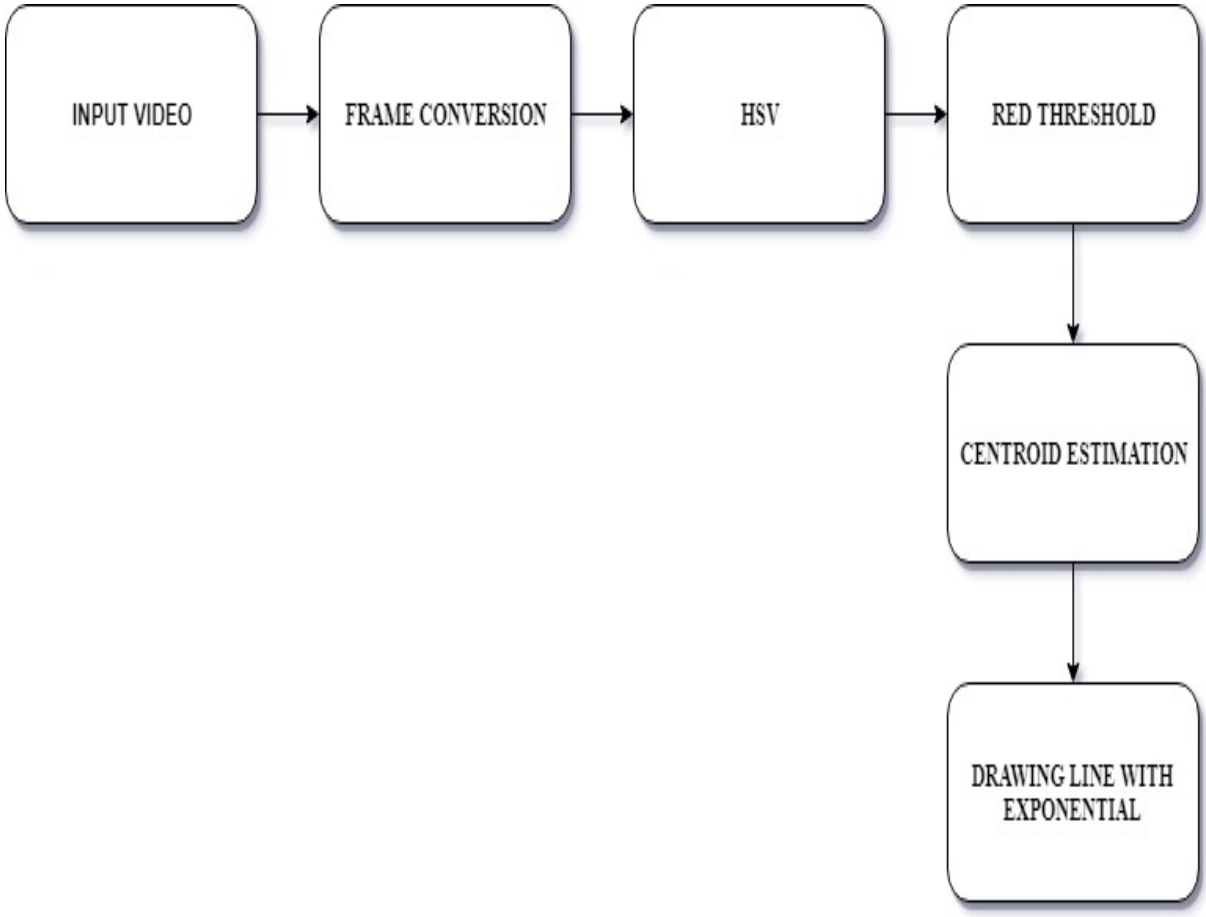


Figure. Block Diagram

CHAPTER 4

RESULTS ANALYSIS AND VALIDATION

4.1 Implementation of design using Modern Engineering tools in analysis

We are using different type of engineering tools and technologies to develop this project all of them (i.e., VScode, Python, OpenCV) are stated below briefly.

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, SH, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

In the Stack Overflow 2021 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool, with 70% of 82,000 respondents reporting that they use it.

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python, C++ and Fortran. It is based on the Electron framework, which is used to develop Node.js Web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services). Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace.

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports many programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette.

Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new languages, themes, and debuggers, perform static code analysis, and add code linters using the Language Server Protocol.

Source control is a built-in feature of Visual Studio Code. It has a dedicated tab inside of the menu bar where you can access version control settings and view changes made to the current project. To use the feature you must link Visual Studio Code to any supported version control system (Git, Apache Subversion, Perforce, etc.). This allows you to create repositories as well as to make push and pull requests directly from the Visual Studio Code program.

Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

Visual Studio Code allows users to set the code page in which the active document is saved, the newline character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language.

Visual Studio Code collects usage data and sends it to Microsoft, although this can be disabled. Due to the open-source nature of the application, the telemetry code is accessible to the public, who can see exactly what is collected. We will utilize the PC vision strategies of OpenCV to assemble this venture. The favoured language is Python because of its

comprehensive libraries and simple to utilize grammar however understanding the fundamentals it tends to be executed in any OpenCV upheld language.

Here Color Detection and following are utilized to accomplish the goal. The variety marker is distinguished and a veil is delivered. It remembers the further strides of morphological activities for the veil delivered which are Erosion and Dilation. Disintegration lessens the pollutants present in the cover and expansion further re-establishes the dissolved fundamental veil.

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects [magic methods]). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It uses dynamic name resolution (late binding), which binds method and variable names during program execution.

Its design offers some support for functional programming in the Lisp tradition. It has filter , map and reduce functions; list comprehensions, dictionaries, sets,

and generator expressions. The standard library has two modules (Itertools and functools) that implement functional tools borrowed from Haskell and Standard ML

Its core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:[

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than building all of its functionality into its core, Python was designed to be highly extensible via modules. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, wrote: "To describe something as 'clever' is *not* considered a compliment in the Python culture."

> Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C; or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. Air Canvas is a hands-free digital drawing canvas that utilizes a Raspberry Pi, a PiCamera, and OpenCV to recognize and map hand gestures onto a PiTFT screen. The user's "brush" can be modified in size and color by using built-in buttons. The direction of the brush is controlled completely using open source OpenCV software and modified to map the pointer finger onto the screen using Pygame following a calibration screen to measure and record the color of the

user’s hand. The idea for Air Canvas was a result of our interest in digital drawing and smart photo recognition software.

. It is comparatively easier to use.

. We don't need any specialized computer education or training to use Windows.

. Wide support for games. Almost all games run on Windows.

. It can run on most of the processors.

. The collection of software programs, utilities and games is relatively much larger for Windows.

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV’s deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at

Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

We began our project by searching for open-source hand gesture recognition software that utilized OpenCV in combination with Python. In doing so, our project's design changed as we discovered different image processing algorithms. Our primitive implementation sought to use hand gestures to control our color and size variables. To do so, we first set out to create an image mask that would separate the hand from the background. With some trial and error using OpenCV, we successfully captured an image, Gaussian blurred it, and applied a binary mask to starkly contrast the hand shape from the background. This is a method obtained from Izane's Finger Detection tutorial1, chosen because of its use of convexity detection; in other words, determining the valleys between the fingers. However, we discovered that the camera's sensitivity to the lab room's lighting made this a difficult endeavor, and we often encountered extraneous silhouettes in our processed image.

4.2 CHARACTERISATION

1. To create a simple prototype for a drawing tool that uses hand gesture recognition software
2. Using OpenCV to recognize the contour.
3. Mapping counts of contour extracted from hand canvas to tracking algorithm to produce a drawing.
4. Implementing additional features such as color change etc.

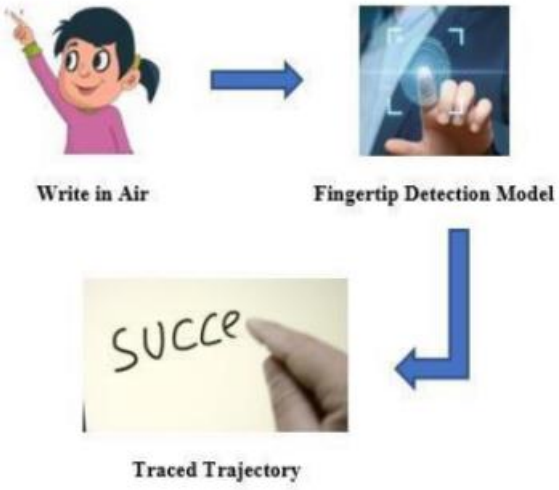
Colour Tracking of Object at fingertip. First of all, The incoming image from the webcam is to

be converted to the HSV colour space for detecting the colored object at the tip of finger. The below code snippet converts the incoming image to the HSV space, which is very suitable and perfect color space for Color tracking.Now, We will make the Trackbars to arrange the HSV values to the required range of color of the colored object that we have placed at our finger.ow, When the trackbars are setup, we will get the realtime value from the trackbars and create range. This range is a numpy structure which is used to be passed in the function cv2.inrange( ). This function returns the Mask on the colored object. This Mask is a black and white image with white pixels at the position of the desired color. 2. Contour Detection of the Mask of Color Object Now, After detecting the Mask in Air Canvas, Now is the time to locate its center position for drawing the Line. Here, In the below Snippet of Code, We are performing some morphological operations on the Mask, to make it free of impurities and to detect contour easily. Drawing the Line using the position of Contour Now Comes the real logic behind this Computer Vision project, We will form a python deque (A data Structure). The deque will store the position of the contour on each successive frame and we will use these stored points to make a line using OpenCV drawing functions. Now, we will use the position of the contour to make decision, if we want to click on a button or we want to draw on the sheet. We have arranged some of the buttons on the top of Canvas, if the pointer comes into their area, we will trigger their method. We have four buttons on the canvas, drawn using OpenCV. Clear : Which clears the screen by emptying the deques. Red : Changes the marker to red color using color array. Green : Changes the marker to Green color using color array. Yellow : Changes the marker to Yellow color using color array. Blue : Changes the marker to Blue color using color array. Also, to avoid drawing when contour is not present, We will Put a else condition which will capture that instant.

Can track any specific colored pointer . User can draw in four different colors and even change them without any hussle. Able to rub the board with a single location at the top of the screen. No need to touch the computer once the program is run

The initial motivation was a need for a dustless class room for the students to study in**.** I know that there are many ways like touch screens and more but what about the schools which can’t afford it to buy such huge large screens and teach on them like a T.V. So, I thought why not can a finger be tracked, but that too at a initial level without deep learning. Hence it was OpenCV which came to the rescue for these computer vision projects.

4.3 DESIGN DRAWINGS



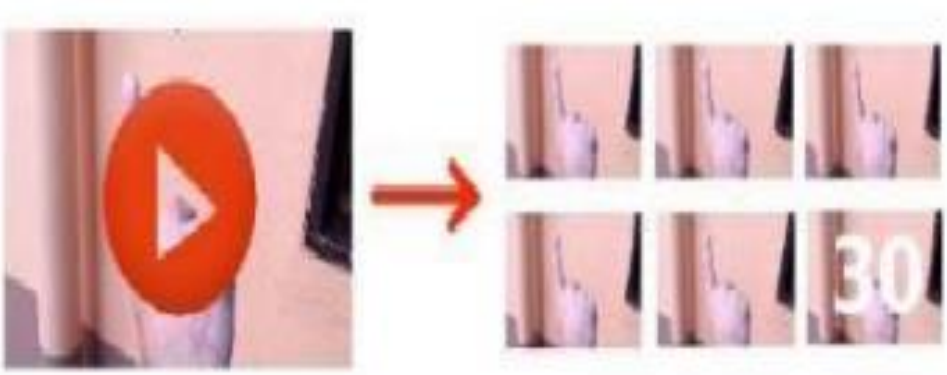
A. Fingertip Detection Model:

Air writing can be merely achieved using a stylus or air-pens that have a unique colour. The system, though, makes use of fingertip. We believe people should be able to write in the air without the pain of carrying a stylus. We have used Deep Learning algorithms to detect fingertip in every frame, generating a list of coordinates.

B. Techniques of Fingertip Recognition Dataset Creation:

a. Video to Images: In this approach, two-second videos of a person's hand motion were captured in different environments. These videos were then broken into 30 separate images, as shown in Figure . We collected 2000

images in total. This dataset was labeled manually using LabelImg. The best model trained on this dataset yielded an accuracy of 99%. However, since the generated 30 images were from the same video and the same environment, the dataset was monotonous. Hence, the model didn't work well for discrete backgrounds from the ones in the dataset.



b. Take Pictures in Distinct Backgrounds: To overcome the drawback caused by the lack of diversity in the previous method, we created a new dataset. This time, we were aware that we needed some gestures to control the system. So, we collected the four distinct hand poses, . The idea was to make the model capable of efficiently recognizing the fingertips of all four fingers. This would allow the user to control the system using the number of fingers he shows. He or she could now - promptly write by showing one index finger, convert this writing motion to e-text by offering two fingers, add space by showing three fingers, hit backspace by showing five fingers, inter prediction mode by showing four fingers, and then the show 1,2,3 fingers to select the 1st, 2nd or 3rd prediction respectively. To get out of prediction mode, show five fingers. This dataset consisted of 1800 images. Using a script, the previously trained model was made to auto-label this dataset. Then we corrected the mislabelled images and introduced another model. A 94% accuracy was achieved. Contrary to the former one, this model worked well in different backgrounds.

C. Fingertip Recognition Model Training:

Once the dataset was ready and labeled, it is divided into train and dev sets (85%-15%). We used Single Shot Detector (SSD) and Faster RCNN pre-trained models to train our dataset. Faster RCNN was much better in terms of accuracy as compared to SSD. Please refer to the Results Section for more information. SSDs combine two standard object detection

modules – one which proposes regions and the other which classifies them. This speeds up the performance as objects are detected in a single shot. It is commonly used for real-time object detections. Faster RCNN uses an output feature map from Fast RCNN to compute region proposals. They are evaluated by a Region Proposal Network and passed to a Region of Interest pooling layer. The result is finally given to two fully connected layers for classification and bounding box regression. We tuned the last fully connected layer of Faster RCNN to recognize the fingertip in the image.

4.4 INTERPRETATION

We wanted to draw our imagination by just waving our finger in the air. In this, we will learn to build an Air Canvas which can draw anything on it by just capturing the motion of a colored marker with a camera. Here a colored object at the tip of the finger is used as the marker.

We will be using the computer vision techniques of OpenCV to build this project. The preferred language is Python due to its exhaustive libraries and easy to use syntax but understanding the basics it can be implemented in any OpenCV supported language.

Air Canvas is a hands-free digital drawing canvas that utilizes a Raspberry Pi, a PiCamera, and OpenCV to recognize and map hand gestures onto a PiTFT screen. The user's "brush" can be modified in size and color by using built-in buttons. The direction of the brush is controlled completely using open source OpenCV software and modified to map the pointer finger onto the screen using Pygame following a calibration screen to measure and record the color of the user's hand. The idea for Air Canvas was a result of our interest in digital drawing and smart photo recognition software.

Object tracking is considered as an important task within the field of Computer Vision. The invention of faster computers, availability of inexpensive and good quality video cameras and

demands of automated video analysis has given popularity to object tracking techniques. Generally video analysis procedure has three major steps: firstly detecting of the object, secondly tracking its movement from frame to frame and lastly analyzing the behavior of that object. For object tracking, four different issues are taken into account; selection of suitable object representation, feature selection for tracking, object detection and object tracking [1]. In real world, Object tracking algorithms are the primarily part of different applications such as: automatic surveillance, video indexing and vehicle navigation etc.
Here Color Detection and tracking are used in order to achieve the objective. The color marker is detected and a mask is produced. It includes the further steps of morphological operations on the mask produced which are Erosion and Dilation. Erosion reduces the impurities present in the mask and dilation further restores the eroded main mask.

So, in this project, we are going to build an air canvas using OpenCV and Python.

OpenCV is an open-source computer vision library for performing various advanced image processing tasks.

We'll use color detection and segmentation techniques to achieve this objective.

- Color detection is an image processing technique where we can detect any color in a given range of HSV color space.
- Image segmentation is the process of labeling every pixel in an image, where each pixel shares the same certain characteristics.

ALGORITHMIC WORKFLOW:

1 .Start reading the frames and convert the captured frames to HSV colour space.(Easy for colour detection) 2.Prepare the canvas frame and put the respective ink buttons on it. 3.. Adjust the trackbar values for finding the mask of coloured marker. 4.Preprocess the mask with morphological operations.(Erotion and dilation) 5.Detect the contours, find the center coordinates of largest contour and keep storing them in the array for successive frames .(Arrays for drawing points on canvas) 6.Finally draw the points stored in array on the frames and canvas .

For making this project we, Antriksh and Anshuman used opencv, numpy and python.

So what is Opencv:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

OpenCV is a great tool for image processing and performing computer vision tasks. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more. It supports multiple languages including python, java C++.

And for about numpy and python,
We all know about python language because it is quite famous and is used in opencv because Python is well-suited for implementing new features. Libraries like OpenCV are written in C++ and make Python have slower runtime as it will still call C/C++ libraries. This means you will have the development advantage from Python while you can have performance optimization from C++
And about numpy, very few know about this.

Well, NumPy stands for Numerical Python and it is a core scientific computing library in Python. It provides efficient multi-dimensional array objects and various operations to work with these array objects.

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices

and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original. The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory.

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

And last python:

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific

problems. Python is a general-purpose coding language—which means that, unlike HTML, CSS, and JavaScript, it can be used for other types of programming and software development besides web development. Python can be used for things like: Back end (or server-side) web and mobile app development.

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human

Python consistently ranks as one of the most popular programming languages.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects [magic methods]). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It uses dynamic name resolution (late binding), which binds method and variable names during program execution.

Its design offers some support for functional programming in the Lisp tradition. It has filter , map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (Itertools and functools) that implement functional tools borrowed from Haskell and Standard ML

Its core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:[

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than building all of its functionality into its core, Python was designed to be highly extensible via modules. This compact modularity has made it particularly popular as a

means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, wrote: "To describe something as 'clever' is *not* considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C; or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

The framework can possibly challenge customary composing techniques. It annihilates the need to convey a portable telephone close by to write down notes, giving a straightforward onthe-go method for doing likewise. It will likewise serve an extraordinary reason in aiding particularly abled individuals impart without any problem. Indeed, even senior residents or individuals who think that it is troublesome to utilize consoles will ready to easily utilize framework. Broadening the usefulness, framework can likewise be utilized to control IoT gadgets presently. Attracting the air can likewise be made conceivable. The framework will be a magnificent programming for savvy wearables utilizing which individuals could better communicate with the computerized world. Increased Reality can make text wake up. There are a few impediments of the framework which can be worked on from here on out. First and foremost, utilizing a penmanship recognizer instead of a person recognizer will permit the client to compose word by word, making composing quicker. Also, hand-signals with a respite can be utilized to control the constant framework as done by as opposed to utilizing the quantity of fingertips. Thirdly, our framework in some cases perceives fingertips in the foundation and impacts their state. Air-composing frameworks ought to just submit to their lord's control motions furthermore, ought not be deceived by individuals around. Likewise, we utilized the EMNIST dataset, which is certifiably not an appropriate air-character dataset. Impending item recognition calculations, for example, Consequences be damned v3 can further develop fingertip acknowledgment precision and speed. Later on, progresses in Artificial Intelligence will upgrade the proficiency of air-composing. To avoid the use of mouse and difficulty to draw using it in the existing systems , this project Air Canvas helps us a lot. We can easily draw or present our imagination just by waving our hand . This uses the easy methods or libraries like mediapipe making the project efficient than existing one. In this system we have implemented an air canvas system using mediapipe which is efficient way to track hand positions . Mediapipe also helps us to reduce the process of image processing to detect the positions of fingers . This can be used in different aspects like teaching, drawing etc. This helps us to reduce the use of hardware components like mouse, touch screen etc. This can also be used as base project for various system that require hand tracking. The project discussed in this

paper also helps to improve creativity in people. This helps us to teach and draw easily than earlier. In future, we can also use this project as base project for many other hand tracking projects. we can also use this in sign language detection ,virtual mouse etc.

5.2 Future Work

We want to automate stuff with this means if we use gesture in this then the app will open after this for which we have selected the gesture .In this modern time, we have already witnessing that use technology on every work is increasing day by day. So in the upcoming future we'll see that there will be very less use of hands to work, everyone will apply the use of technology. There will be some advantages also like:

People hearing impairment: Although we take hearing and listening for granted, they communicate using sign languages. Most of the world can't understand their feeling, their emotions without a translator in between.

Paper wastage is not scarce news. We waste a lot of paper in scribbling, writing, drawing, etc.… Some basic facts include - 5 liters of water on average are required to make one A4 size paper, 93% of writing is from trees, 50% of business waste is paper, 25% landfill is paper, and the list goes on. Paper wastage is harming the environment by using water and trees and creates tons of garbage. Air Writing can quickly solve these issues. It will act as a communication tool for people with hearing impairment. Their air-written text can be presented using AR or converted to speech. One can quickly write in the air and continue with your work without much distraction. Additionally, writing in the air does not require paper. Everything is stored electronically.

Overuse of Smartphones: They cause accidents, depression, distractions, and other illnesses that we humans can still discover. Although its portability, ease to use is profoundly admired, the negatives include life threatening events.

The initial motivation was a need for a dustless class room for the students to study in. I know that there are many ways like touch screens and more but what about the schools which can't afford it to buy such huge large screens and teach on them like a T.V. So, I thought why not can a finger be tracked, but that too at a initial level without deep learning. Hence it was OpenCV which came to the rescue for these computer vision projects.

Significant progress has been made in object tracking during the last few years. Several robust trackers have been developed which can track objects in real time in simple scenarios. However, it is clear from the papers reviewed in this survey that the assumptions used to make the tracking problem tractable, for example, smoothness of motion, minimal amount of

occlusion, illumination constancy, high contrast with respect to background, etc., are violated in many realistic scenarios and therefore limit a tracker's usefulness in applications like automated surveillance, human computer interaction, video retrieval, traffic monitoring, and vehicle navigation. Thus, tracking and associated problems of feature selection, object representation, dynamic shape, and motion estimation are very active areas of research and new solutions are continuously being proposed. One challenge in tracking is to develop algorithms for tracking objects in unconstrained videos, for example, videos obtained from broadcast news networks or home videos. These videos are noisy, compressed, unstructured, and typically contain edited clips acquired by moving cameras from multiple views. Another related video domain is of formal and informal meetings. These videos usually contain multiple people in a small field of view. Thus, there is severe occlusion, and people are only partially visible. One interesting solution in this context is to employ audio in addition to video for object tracking. There are some methods being developed for estimating the point of location of audio source, for example, a person's mouth, based on four or six microphones. This audio-based localization of the speaker provides additional information which then can be used in conjunction with a video-based tracker to solve problems like severe occlusion In general, an important issue that has been neglected in the development of tracking algorithms is integration of contextual information. For example, in a vehicle tracking application, the location of vehicles should be constrained to paths on the ground as opposed to vertical walls or the sky. Recent work in the area of object recognition [Torralba 2003; Kumar and Hebert 2003] has shown that exploiting contextual information is helpful in recognition. In addition, advances in classifiers [Friedman et al. 2000; Tipping 2001] have made accurate detection of scene context possible, for example, man made structures, paths of movement, class of objects, etc. A tracker that takes advantage of contextual information to incorporate general constraints on the shape and motion of objects will usually perform better than one that does not exploit this information. This is because a tracker designed to give the best average performance in a variety of scenarios can be less accurate for a particular scene than a tracker that is attuned (by exploiting context) to the characteristics of that scene. The use of a particular feature set for tracking can also greatly affect the performance. Generally, the features that best discriminate between multiple objects and, between the object and background are also best for tracking the object. Many tracking algorithms use a weighted combination of multiple features assuming that a combination of preselected features will be discriminative. A wide range of feature selection algorithms have been investigated in the machine learning and pattern recognition

communities. However, these algorithms require offline training information about the target and/or the background. Such information is not always available. Moreover, as the object appearance or background varies, the discriminative features also vary. Thus, there is a need for online selection of discriminative features. Some work has been done in this area for online selection of individual features [Collins and Liu 2003; Stern and Efros 2002]. However, the problem of efficient online estimation of discriminative feature sets remains unresolved. One promising direction to achieve this goal is the use of the online boosting methods [Oza 2002] for feature selection. In a similar vein, most tracking algorithms use prespecified models for object representation. The capability to learn object models online will greatly increase the applicability of a tracker. Motion-based segmentation [Vidal and Ma 2004; Black and Anandan 1996; Wang and Adelson 1994] and multibody factorization [Costeira and Kanade 1998; Gear 1998] methods have been used to learn models for multiple objects moving in a scene. However, these approaches assume rigid body motion. Unsupervised learning of object models for multiple nonrigid moving objects from a single camera remains an unsolved problem. One interesting direction that has largely been unexplored is the use of semisupervised learning techniques for modeling objects. These techniques (cotraining [Levin et al. 2003; Blum and Mitchell 1998], transductive SVMs [Joachims 1999], constrained graph cuts [Yu and Shi 2004]) do not require prohibitive amounts of training data. Moreover, they can not only learn nonrigid shapes and/or appearance, but they can also encode the knowledge of the background in the form of negative training data. Probabilistic state-space methods including Kalman Filters [Bar-Shalom and Foreman 1988], JPDAFs [Cox 1993], HMMs [Rabiner 1989], and Dynamic Bayesian Networks (DBNs) [Jensen 2001] have been extensively used to estimate object motion parameters. Among these methods, DBNs are probably the most general method for representation of conditional dependencies between multiple variables and/or image observations. They also provide a principled framework for fusing information from different sources. However, there is a need for more efficient solutions for inference before DBNs are more commonly used in tracking applications. Overall, we believe that additional sources of information, in particular prior and contextual information, should be exploited whenever possible to attune the tracker to the particular scenario in which it is used. A principled approach to integrate these disparate sources of information will result in a general tracker that can be employed with success in a variety of applications.

5.3 Results

Overall, we achieved the goals we set out to accomplish and learned a considerable amount

about OpenCV, multicore processing. We use opencv, python and numpy in this project. We have done finger tracking, contour detection, Colour Tracking of Object at fingertip.

5.4 Appendix

Project Members

Antriksh Thakur: CSE Undergraduate of batch of 2018

The idea for using OpenCV was driven by his interest in the topic!

- Project conception and design
- Connecting OpenCV information to his knowledge of python, and most other implementation

Anshuman: CSE undergraduate of batch of 2018

Air Canvas was inspired by his love of digital art!

- Writing additional functionality for colours and size, and most other software implementation
- Project conception and other implementation

5.5 References

[1]. H.M. Cooper, “Sign Language Recognition:Generalising to More Complex Corpora”, PhThesis, Centre for Vision, Speech and Signal Processing Faculty of Engineering and Physical Sciences, University of Surrey, UK, 2012

[2]. Yusuke Araga, Makoto Shirabayashi, KeishiKaida, Hiroomi Hikawa, “Real Time GestureRecognition System Using Posture Classifierand Jordan Recurrent Neural Network”, IEEEWorld Congress on Computational Intelligence, Brisbane, Australia, 2012

[3]. Ruiduo Yang, Sudeep Sarkar, “Coupledgrouping and matching for sign and gesturerecognition”, Computer Vision and Image Understanding, Elsevier, 2008

[4] Saira Beg, M. Fahad Khan and Faisal Baig, "TextWriting in Air," Journal of Information

[5] Alper Yilmaz, Omar Javed, Mubarak Shah, "ObjectTracking: A Survey", ACM Computer Survey. Vol. 38, Issue. 4, Article 13, Pp. 1-45, 2006

[6] Yuan-Hsiang Chang, Chen-Ming Chang, "Automatic Hand-Pose Trajectory Tracking System Using Video Sequences", INTECH, pp. 132- 152, Croatia, 2010.

[7] Kenji Oka, Yoichi Sato, and Hideki Koike, "Real-Time Fingertip Tracking and Gesture Recognition," IEEE Computer Graphics and Applications, 2002, pp.64-71.

[8]Guo-Zhen Wang, Yi-Pai Huang, Tian-Sheeran Chang, and Tsu-Han Chen, "Bare Finger 3D AirTouch System Using an Embedded Optical Sensor Array for Mobile Displays", Journal Of Display Technology, VOL. 10, NO. 1, JANUARY 2014, pp.13-18

[9]. Erik B. Sudderth, Michael I. Mandel, William T. Freeman, Alan S. Willsky, “Visual Hand Tracking Using Nonparametric Belief Propagation”, Mit Laboratory For Information & Decision Systems Technical Report P-2603, Presented at IEEE CVPR Workshop On Generative Model Based Vision, Pp. 1-9, 2004

[10]. Robert Y. Wang, Jovan Popovi´c, “Real-Time Hand-Tracking with a Color Glove”, 2008

[11]. T. A. C. Bragatto, G. I. S. Ruas, M. V. Lamar, “Real-time Video Based Finger Spelling Recognition System Using Low Computational Complexity Artificial Neural Networks”, IEEE ITS, pp. 393-397, 2006

[12]. Kim, Sung Deuk, Jaeyoun Yi, Hyun Mun Kim, and Jong Beom Ra. "A deblocking filter with two separate modes in block-based video coding." Circuits and Systems for Video Technology, IEEE Transactions on 9, no. 1 (1999): 156-160.

[13]. Buckingham, Michael J., Broderick V. Berknout, and Stewart AL Glegg. "Imaging the ocean with ambient noise." Nature 356, no. 6367 (1992): 327-329.

[14]. Vaseghi, Saeed V. Advanced digital signal processing and noise reduction. Wiley, 2008.

[15]. Plataniotis, Konstantinos N., and Anastasios N. Venetsanopoulos. Color image processing and applications. Springer, 2000.

[16] Napa Sae-Bae, Kowsar Ahmed, Katherine Isbister, NasirMemon, "Biometric-rich gestures: a novel approach to authentication on multi-touch devices," Proc. SIGCHI Conference on Human Factors in Computing System,2005, pp.977-986

[17] W. Makela, "Working 3D Meshes and Particles with Finger Tips, towards an Immersive Artists' Interface," Proc. IEEE Virtual Reality Workshop, pp. 77-80, 2005.

[18] A.D. Gregory, S.A. Ehmann, and M.C. Lin, "inTouch: Interactive Multiresolution Modeling and 3D Painting with a Haptic Interface," Proc. IEEE Virtual Reality (VR' 02), pp. 45-52, 2000.

[19] W. C. Westerman, H. Lamiraux, and M. E. Dreisbach, “Swipe gestures for touch screen keyboards,” Nov. 15 2011, US Patent 8,059,101

[20] S. Vikram, L. Li, and S. Russell, "Handwriting and gestures in the air, recognizing on the fly," in Proceedings of the CHI, vol. 13, 2013, pp. 1179–1184.

[21] X. Liu, Y. Huang, X. Zhang, and L. Jin. "Fingertip in the eye: A cascaded CNN pipeline for the real-time fingertip detection in egocentric videos," CoRR, abs/1511.02282, 2015.

[22]H.M. Cooper, “Sign Language Recognition: Generalising to More Complex Corpora”, PhD Thesis, Centre for Vision, Speech and Signal Processing Faculty of Engineering and Physical Sciences, University of Surrey, UK, 2012

[23] Robert Wang, Sylvain Paris, Jovan Popovi´c, “Practical Color-Based Motion Capture”, Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, 2011

[24] Kim, Sung Deuk, Jaeyoun Yi, Hyun Mun Kim, and Jong Beom Ra. "A deblocking filter

with two separate modes in block-based video coding." Circuits and Systems for Video Technology, IEEE Transactions on 9, no. 1 (1999): 156-160.

[25] Vladimir I. Pavlovic, Rajeev Sharma, and Thomas S. Huang, "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," IEEE Transactions on Pattern Analysis and Machine Intelligence, VOL. 19, NO. 7, JULY 1997, pp.677-695

[26] Maryam Khosravi Nahouji, "2D Finger Motion Tracking, Implementation For Android Based Smartphones", Master's Thesis, CHALMERS Applied Information Technology,2012, pp 1-48

[27] EshedOhn-Bar, Mohan ManubhaiTrivedi, "Hand Gesture Recognition In Real-Time For Automotive Interfaces," IEEE Transactions on Intelligent Transportation Systems, VOL. 15, NO. 6, December 2014, pp 2368-2377

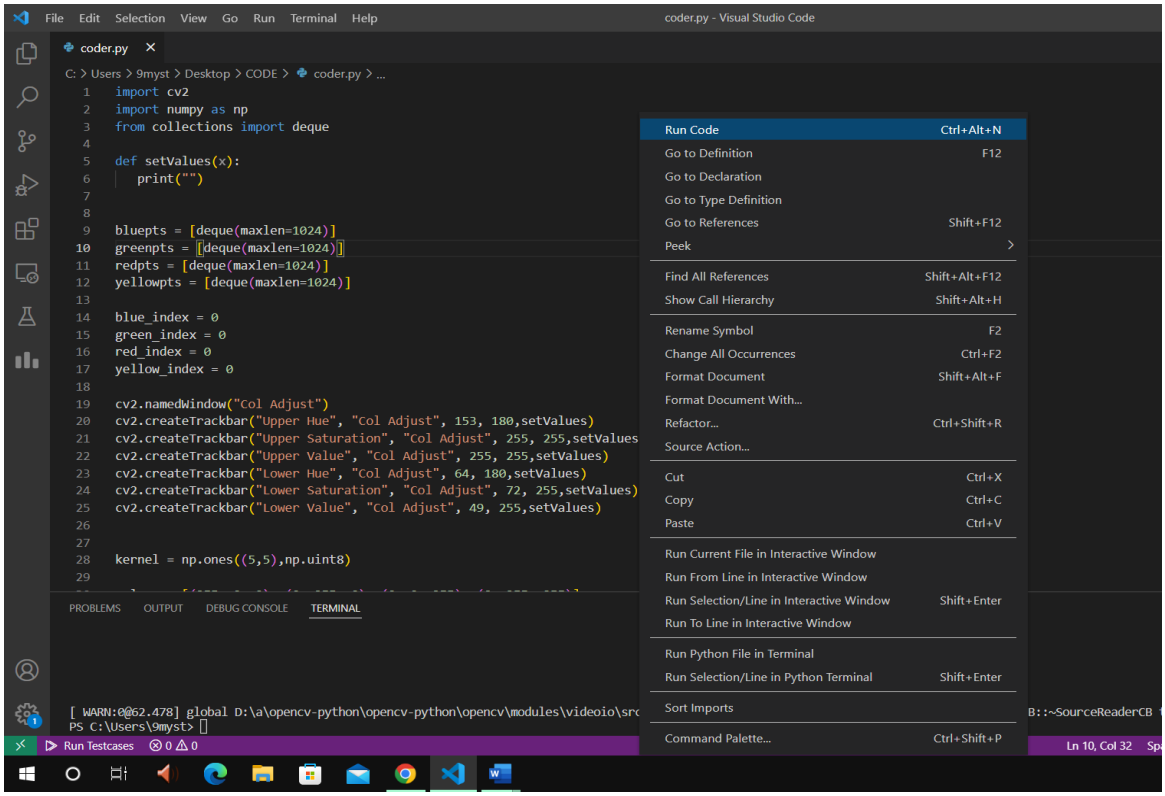
5.6 PROJECT SCOPE

The scope of this system is mainly used as a powerful means of communication for the deaf, which means implementing this project can help in:

1. An effective communication method that reduces mobile and laptop usage by eliminating the need to write.
2. It helps people with hearing impairments to communicate well.
3. Various purposes, such as sending messages, e-mails, etc., as the generated text can also be used for that.

5.7 User Manual

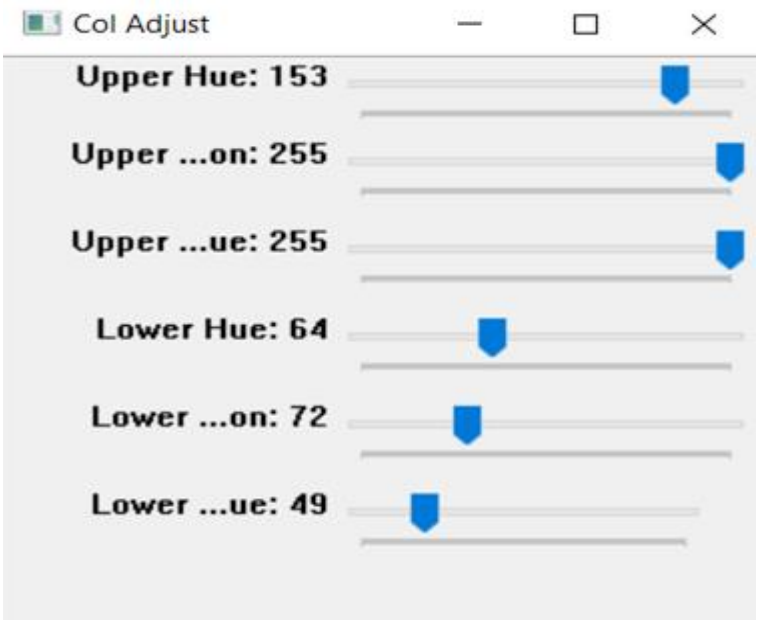
Step 1: Run the python Script By using any IDE for coding.Eg.



Step 2: Now The pop up windows will appear in the screen as follow:

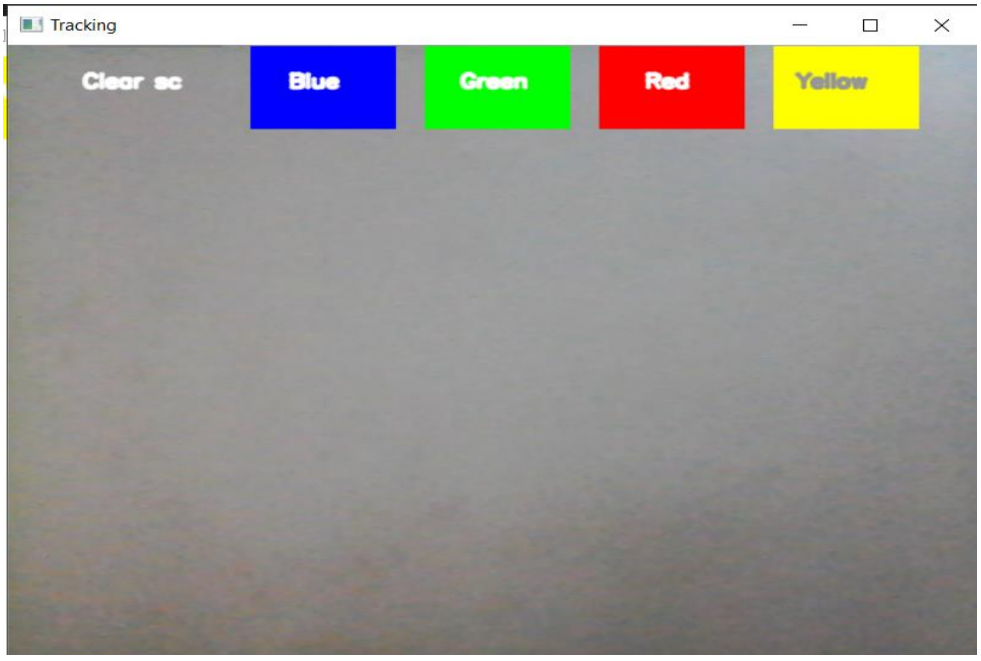
Window 1 should be a masking window.

Window 2 should be a colour adjustor window.



Window 3 will be a canvas for drawing.

Window 4 will be a camera feed showing canvas.



Step 3: Now place a Contour or use it in front of the camera so it can be detected.

Step 4: Now, after detecting the contour, draw in the air and the same will be printed on the screen.