## Mini Project Report On

**AIR CANVAS USING PYTHON**

Submitted in partial fulfillment of the requirements for the award of the degree of

# BACHELOR OF TECHNOLOGY

In

#### COMPUTER SCIENCE AND ENGINEERING

Submitted By:

##### K.SHRAVANI 22UP1A0592

**Project Coordinator**

**Mrs.G.Ramya**

Assistant Professor

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

### VIGNAN’S INSTITUTE OF MANAGEMENT AND TECHNOLOGY FOR WOMEN

**(Affiliated to JNTUH, Hyderabad, Accredited by NBA)**

**Kondapur(v), Ghatkesar(M), Medchal-Malkajgiri(D)-501301 [2022-2026]**

#### 

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# CERTIFICATE

This is to certify that the Project work titled **“Air Canvas Using Python”** submitted by **K.SHRAVANI(22UP1A0592)** of B-Tech in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineeringto the Vignan’s Institute of Management and Technology for Women is a record of bona-fide work carried out by them under my guidance and supervision. The results embodied in this project report have not been submitted to any university for the award of any degree and the results are achieved satisfactorily.

**Project Coordinator HEAD OF THE DEPARTMENT**

##### Mrs.G.Ramya Mrs.M.PARIMALA

(Assistant Professor) (Associate Professor)

##### (External Examiner)

##### 

# DECLARATION

I, hereby declare that the results embodied in this dissertation entitled “**Air Canvas Using Python**” is carried out by us during the year 2022-2026 in partial fulfillment of the award of Bachelor of Technology in Computer Science and Engineering from VIGNAN’S INSTITUTE OF MANAGEMENT AND TECHNOLOGY FOR WOMEN is an authentic record of our work carried under the guidance of Mrs.G.Ramya(Assistant Professor), Department of Computer Science and Engineering. We have not submitted the same to any other university or organization for the award of any other degree.

**K.SHRAVANI(22UP1A0592)**

**ACKNOWLEDGEMENT**

We would like to express sincere gratitude to **Dr. G. AppaRao Naidu, Principal**, **Vignan’s Institute of Management and Technology for Women** for his timely suggestions which helped us to complete the project in time.

We would also like to thank our madam **Mrs. M Parimala, Head of the Department, Computer Science and Engineering,** for providing us with constant encouragement and resources which helped us to complete the project in time.

We would like to thank our project guide**, Mrs.G.Ramya, Assistant Professor, Computer Science and Engineering,** for his timely cooperation and valuable suggestions throughout the project. We are indebted to his for the opportunity given to work under his guidance.

Our sincere thanks to all the teaching and non-teaching staff of the Department of Computer Science and Engineering for their support throughout our project work.

**K.SHRAVANI(22UP1A0592)**

## 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 motionto-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.

KEYWORDS: Gesture recognition, Air writing, Wearable devices.

| CONTENTS | INDEX | PAGE NO |
| --- | --- | --- |
| 1 | Introduction | 9 |
| 1.2 | Proposed System | 11 |
| 2 | System Requirements |  |
| 2.1 | Hardware requirements | 13 |
| 2.2 | Software Requirements | 13 |
| 3 | System Design | 15 |
| 3.1 | System Architecture | 16 |
| 3.2 | Methodologies | 17 |
| 4 | UML Diagrams | 18 |
| 4.1 | Use Case Diagram | 18 |
| 4.2 | Flow Diagrams | 19 |
| 4.3 | Data flow Diagram | 20 |
| 4.4 | Collaboration Diagram | 21 |
| 4.5 | Sequence Diagram | 21 |
| 4.6 | Class Diagram | 22 |
| 5 | Software Environment | 25 |
| 5.1 | Air Canvas Using Python | 25 |
| 5.2 | Working Of Air Canvas Using Python | 25 |
| 5.3 | Flow Chart Of Air Canvas Using Python | 26 |
| 5.4 | Camera Used IN The Air Canvas | 26 |
| 5.5 | Capturing The Video And Processing | 27 |
| 5.6 | Methodology | 28 |
| 5.7 | Advantages | 29 |
| 5.8 | Disadvantages | 30 |
| 5.9 | Modules Used In This Project | 31 |
| 6 | Implementation | 34 |
| 6.1 | SDLC To Be Used Planning | 34 |
| 7 | Source Code | 35 |
| 8 | Results | 44 |
| 8.1 | User Interface Representation | 45 |
| 9 | Conclusion | 48 |
| 10 | Future Scope/Enhancement | 49 |

# LIST OF FIGURES

| **Fig No** | **Name** | **PageNo** |
| --- | --- | --- |
| 2.2 | Mediapipeline | 14 |
| 3.1.5 | User Interface Diagram | 16 |
| 3.2 | System Architecture | 17 |
| 3.3 | Methodology | 17 |
| 4.1 | Use Case Diagram | 18 |
| 4.2 | Flow Diagram | 19 |
| 4.3 | Data Flow Diagram | 20 |
| 4.4 | Collobration Diagram | 21 |
| 4.5 | Sequence Diagram | 22 |
| 4.6 | Class Diagram | 23 |
| 5.3 | Flow Chart Of Air Canvas | 26 |
| 5.4 | Capturing Video | 26 |
| 5.5 | Selecting An Option | 27 |
| 5.6 | Figure Gesture Recognization | 29 |
| 5.9 | Frame Processing | 32 |
| 6.1 | System Design Diagram | 35 |
| 7.3 | Hand Landmark Model | 44 |
| 8.1 | Result | 46-47 |

#### 1. INTRODUCTION

Writing has evolved majorly over the years. Writing was first invented in 2000 BC by Neolithic people. They first started writing on walls then it was replaced by stones. Stones were replaced by cloth and presently we use paper for communication. With the help of QWERTY keyboards, we are moving towards a more digitalized form of writing. These electronic devices are slowly taking the place of traditional forms of writing with pen and paper. The need to develop human machine interactions is rapidly growing with the surge in the usage of augmented and virtual reality. Applications using hand gestures have gained popularity over the coming years. Automotive interfaces (Ohn-Bar and Trivedi, 2014), Economical Air Writing system(Pavithra and Prabhu, 2016) and Handwriting recognition in Free Space(Shashidhar, Kim and Chai, 2015) have developed systems for hand gestures recognition. However hand gestures recognition is not enough for writing in air. It also involves fingertip detection, tracking and tracing of it. Fingertip detection or LED light detection system is developed by [pavithra], where they first detect LED light and then capture the movement and by using Optical Character Recognition (OCR) and display the alphabet on the screen. However these methods which have the usage of devices have some limitations. The proposed system eliminates the usage of cell phones for taking notes. Fingertip detection and finger movement techniques are used to develop the system. Using Python OpenCV and CNN techniques fingertip is first detected and then the trajectory of fingertip is traced and displayed on the screen. Python programming is used for developing the virtual mouse system, and also, the library for computer vision that is OpenCV is used in the AI virtual mouse system. In the proposed system, the model uses the media Pipe package for the tracking of the hands and for tracking the tip of the fingers, also, Pynput, Autopy, and Py Auto GUI packages are used for moving around the window screen of the computer for performing various functions such as left click, right click, and scrolling functions. The results of the proposed AI virtual mouse model showed very high accuracy level, and the model can work very well in real-world application with the use of a CPU without the use of a GPU. With increasing technology each sector need to be modernized. With the improvement of clever gadgets, the system can be now controlled virtually with aid of using human gestures. While using paint, sometimes we feel difficult to draw and feel like drawing our imagination just by waving our hand. The Project Air Canvas makes a speciality of growing a motion-to-textual converter. This project works on hand tracking system development which aims to track the hand which acts as pen and functioning as pen to create or draw different shapes and also as an eraser using Open Computer Vision Library(OpenCV) and Mediapipe. The existing project which allows us to draw just by waving hand uses technology or methodology which takes a lot of process and time. Avoiding or decreasing these limitations we came up with this project that uses new technologies and easy methodologies. System Camera is used to track the hand and create drawings. This also helps to annotate pdf just by waving hands. Digital Art and Traditional art are interrelated and interdependent. Social improvement is certainly not a people's will, yet at the same the At any rate, requirements of human existence are the vitally main impetus. The same circumstance occurs in Art. In the current conditions, digital Art and traditional art are comprehensive of the advantageous state, so we really want to efficiently comprehend the essential knowledge on the structure between digital art and traditional art. The fundamental point of advanced workmanship is of building hand signal acknowledgment framework to write design

#### 

#### 1.2 PROPOSED SYSTEM

In this proposed framework, we are going to utilize camera and the screen for the reading inputs and displaying outputs. We are using our hand fingers to drawing required shapes on the output screen. We have to be on safe distance where our hand can be fully visible in camera hence it is reading our input by recognizing movements of our fingers tip. Some other hand sign for selecting shapes and draw going to use as per given in used modules and libraries. Ever wanted to draw your imagination by just waiving your finger in air. In this post we will learn to build an Air Canvas which can draw anything on it by just capturing the motion of a coloured marker with camera. Here a colored object at tip of finger is used as the marker**.** We will be using the computer vision techniques of OpenCV to build this project. The preffered 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 Colour Detection and tracking is used in order to achieve the objective. The colour marker in 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. Imagine an artist standing before an invisible canvas, where each movement becomes a stroke of pure imagination, untethered by physical tools. This is the captivating realm of “Virtual Brushes in the Wind,” a fusion of OpenCV and Air Canvas that’s redefining art’s boundaries in a magical way.

**Air Canvas: Where Whispers Shape Art**

Enter a world where art transcends the visible. Air Canvas lets artists sculpt their creations using the language of motion and breath. A wave of a hand transforms into a graceful line, a gentle exhale adds depth to a stroke. It’s art formed by the wind’s gentle caress—a symphony of movement and creativity.

**OpenCV: The Tech Maestro Behind the Curtain**

Behind this enchantment lies OpenCV, a technical marvel translating an artist’s gestures into a dance of colors and shapes. Imagine painting on an unseen canvas, with OpenCV as the interpreter. Its algorithms work like magic, ensuring each flicker of movement turns into a stroke of genius, seamlessly blending imagination and reality

**Colors of Freedom and Creativity** “Virtual Brushes in the Wind” celebrates freedom in creativity. It’s like setting imagination free in an infinite playground. There are no bounds to the canvas, no limits to the colors. Artists can let their ideas flow without worrying about supplies or space. It’s an arena where the wind carries their vision into reality.

**Air Canvas Python Project: A Fusion of Art and Technology**

“Virtual Brushes in the Wind” is more than a concept; it’s an Air Canvas project driven by OpenCV and Python. Collaborators in this venture explore the fusion of artistic expression and technology, giving rise to an innovative way of creating art. The marriage of Air Canvas and OpenCV within a Python project opens new doors for artists to explore and experiment.

**Art as a Glimpse into the Soul**

Art bares the artist’s soul, revealing thoughts and emotions through strokes. With this innovation, it’s as if we’re gazing through a window into their mind. The way they move, the rhythm of their gestures it all contributes to the bigger picture. Each stroke becomes a diary entry, each motion a chapter in their story.

**The Dance of Art and Tech**

“Virtual Brushes in the Wind” is a harmonious tango of human creativity and technological brilliance. Artists and tech experts unite to craft a canvas that’s part imagination, part code. It’s a collaborative effort where art inspires technology, and technology ignites new dimensions of creativity. In this world, Virtual Brushes lead the dance on the canvas of Air, choreographed by OpenCV. Here, creativity and innovation waltz hand in hand. The result is a masterpiece that speaks to the heart, dances with the wind, and invites everyone to join the symphony of limitless artistry

#### 

#### 

#### 2. SYSTEM REQUIREMENTS

**2.1 Hardware Requirements:**

* System : Minimum i3.
* Hard Disk : 40 GB.
* Ram**:** 4 GB.

**2.2 Software Requirements:**

* Pycharm Community Edition
* OpenCV
* Numpy
* Mediapipe

**2.2.1.** **PyCharm:**

* PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.
* PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quickfixes, along with automated code refactoring and rich navigation capabilities.

**2.2.2. OpenCv:**

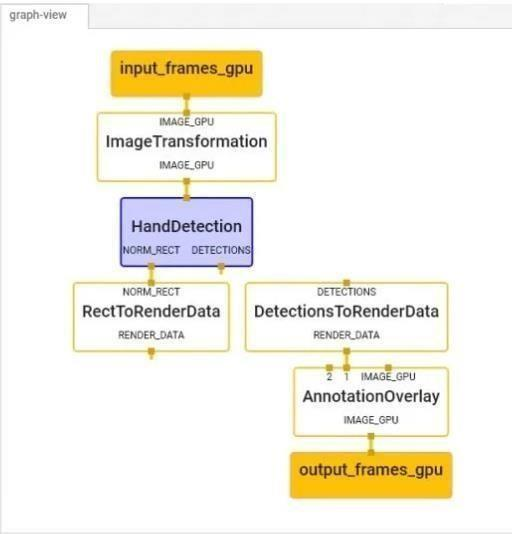
* OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today’s systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

**2.2.3. NumPy:**

* NumPy. The collection of identically categorized things is described. A zero-based index The N-dimensional array type known as ndarray is the most significant object defined in can be used to access items in the collection. A ndarray’s items all take up the same amount of space as a memory block. Every item in ndarray is a data-type object (called dtype). A Python object of one of the array scalar types represents each item that is retrieved from and array object (via slicing).

**2.2.4. Media Pipeline:**

MediaPipe is a cross-platform pipeline framework to build custom machine learningsolutions for live and streaming media. The framework was open-sourced by Googleand is currently in the alpha stage. In computer vision pipelines, those components include model inference, media processing algorithms, data transformations, etc. Sensory data such as video streams enter the graph, and perceived description such as object-localization or face-keypoint streams exit the graph.



**Fig 2.2.4:-MediaPipeline**

**3. SYSTEM DESIGN**

**Design Approach**

System design is the solution of a “how to approach to the creation of the new system. It is composed of several steps. It facilitates the understanding and provides the procedural details necessary for implementation of the system recommended in the feasibility study. Emphasis is given on translating the performance requirements into design specification. Design goesthrough logical and physical stages of development. The very first problem to solve to achieve our target is to receive the image from outer world and process the image so that we are able to extract only desired color that this system requires. There are different strategies to distinguish a static gesture; however, the primary situation is to manage dynamic motions that shows dynamic changes with time. The procedure to select rely upon the client and the situations utilized as need to decide the gesture. The nature of image that we capture is colorful so we have to separate the colour bands by using Python OpenCV library and the have to extract the color band used on fingers with help of colour tapes. In this process we have to train the model

and the system must work in real time to identify different gestures made by finger.

**Detailed Design**

Hand tracking is the process in which a computer uses computer vision to detect a hand from an input image and keeps focus on the hand’s movement and orientation. Hand tracking allows us to develop numerous programs that use hand movement and orientation as their input.We tend to write the same code in different projects to perform hand tracking as part of our program. Creating a hand tracking module solves this problem since we write the code once. We then convert this piece of code into a module. We can import this module into any python project that we are working on and it will perform hand tracking. Before we jump into coding, let us discuss how MediaPipe performs hand tracking. Hand tracking using MediaPipe involves two stages:

**Palm detection**

MediaPipe works on the complete input image and provides a cropped image of the hand.

**Hand landmarks identification**

MediaPipe finds the 21 hand landmarks on the cropped image of the hand.

**User Interface Design**

After The Successful completion of our hand tracking module the next step was to create the user interface where the user will interact with the machine to achieve desired result. We created a simple output using webcam with various options for the user to select the desired colour. The position of the colour box with placed. After selecting the colour an automatic list is created with the coordinates where ever our hand is moving in the screen, it gets stored in list to give a collections of pixels as generated output.



**Fig 3.1.5. User Interface System**

**3.1. SYSTEM ARCHITECTURE**

#### 

#### 

#### Fig 3.1. :-System Architecture

**3.2.Methodologies:**

The system mainly consists of three parts:

**A. Write in air:**

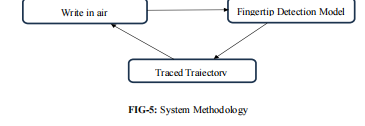
Writing in air is the first step where user starts the application, wears the cap on finger and starts drawing or writing in air so that the path is traced on the canvas. This can be treated as an input for our application.

**B. Fingertip Detection Model:**

The fingertip detection model is the most crucial part of the system architecture because it stores the coordinates of the movement of cap in a queue and then these coordinates are used for tracing the path.

**C. Traced Trajectory:**

Each pixel is then colored in a first in first out manner that are stored in the queue by the fingertip detection model. The stored coordinates are then colored pixel by pixel in a FIFO i.e.first in first out manner since they are stored in a queue and we get a traced trajectory of the finger.

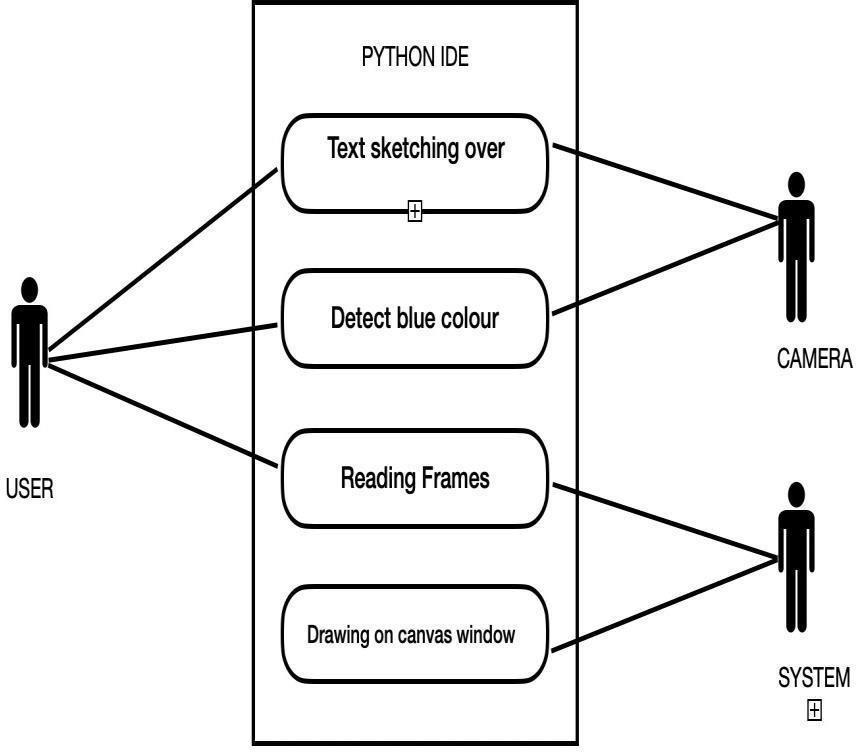


**4. UML DIAGRAMS**

UML is a way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. Today, UML is accepted by the Object Management Group (OMG) as the standard for modeling software development.

#### 4.1.USE CASE DIAGRAM

It represents the functionality of a system by utilizing actors and use cases. It encapsulates the functional requirement of a system and its association with actors. It portrays the use case view of a system.Use case diagrams have use cases, actors, system, communication link, generalization.

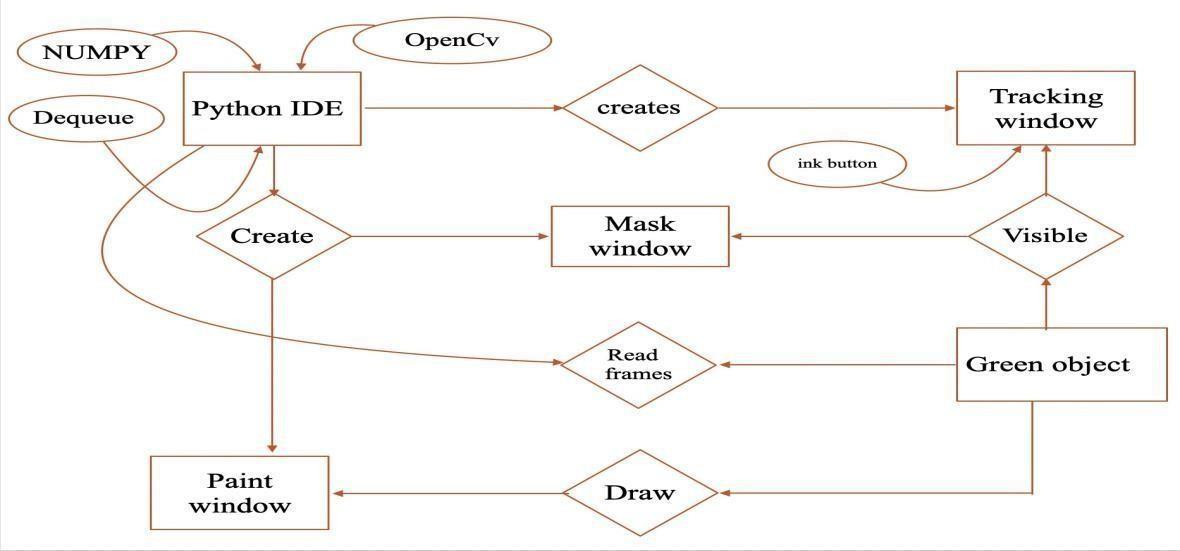


**Fig 4.1:-Use case diagram**

In Figure describes high-level functions and the scope of the system. It also shows the

interaction between the system and the users. Here the user acts as an actor and python IDE acts like a system.

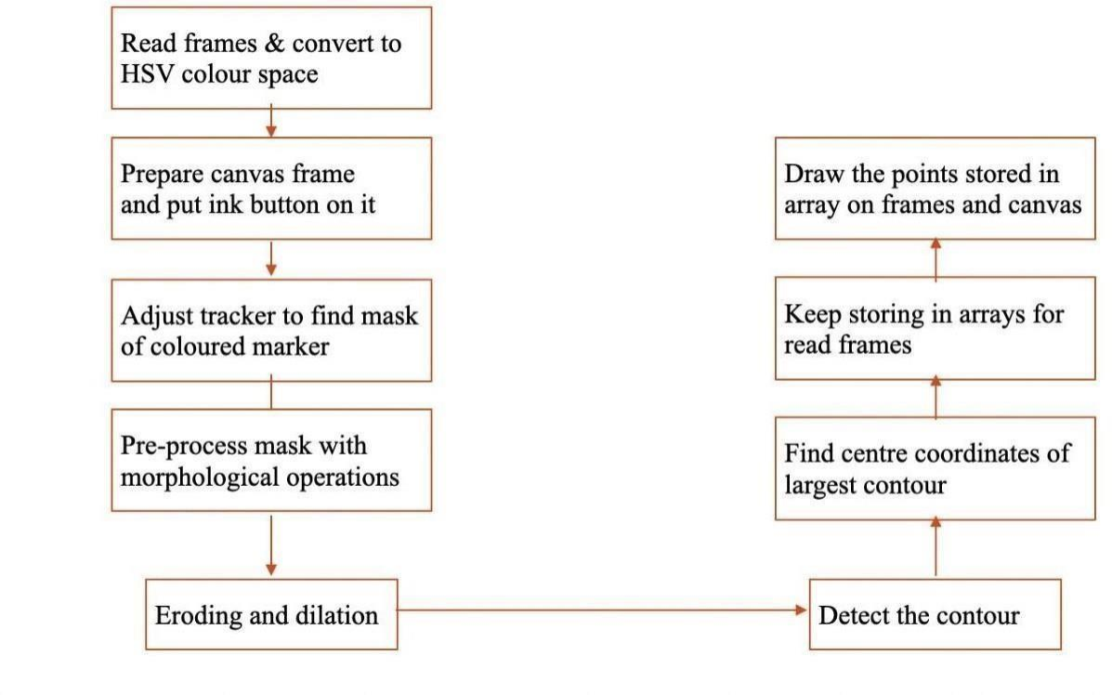
**4.2.FLOW DIAGRAMS MODULE DESCRIPTION**

****

**Fig 4.2:-Architecture Diagram**

In Figure, we are installing Open-CV, Numpy, Dequeue from collections in Python IDE. Then we need to create the live frame and add Colour buttons to it. After that we need to create a Mask window for the object detection, here we are using any blue color object for detecting. Simultaneously we can create a painted window to draw on it. Now the movement of the greenobject will help in reading the frames by the software and drawing on the Paint window. We can change the drawing color also. In reading frames, we are importing necessary packages like OpenCV and NumPy in Python IDE. Creating the trackbars needed for adjusting the marker color. These trackbars will be used for setting the upper and lower ranges of the HSV required for particular colors giving different arrays to handle input color Points of a different color. These arrays will hold the points of a particular color in the array which will further be used to draw on the canvas.

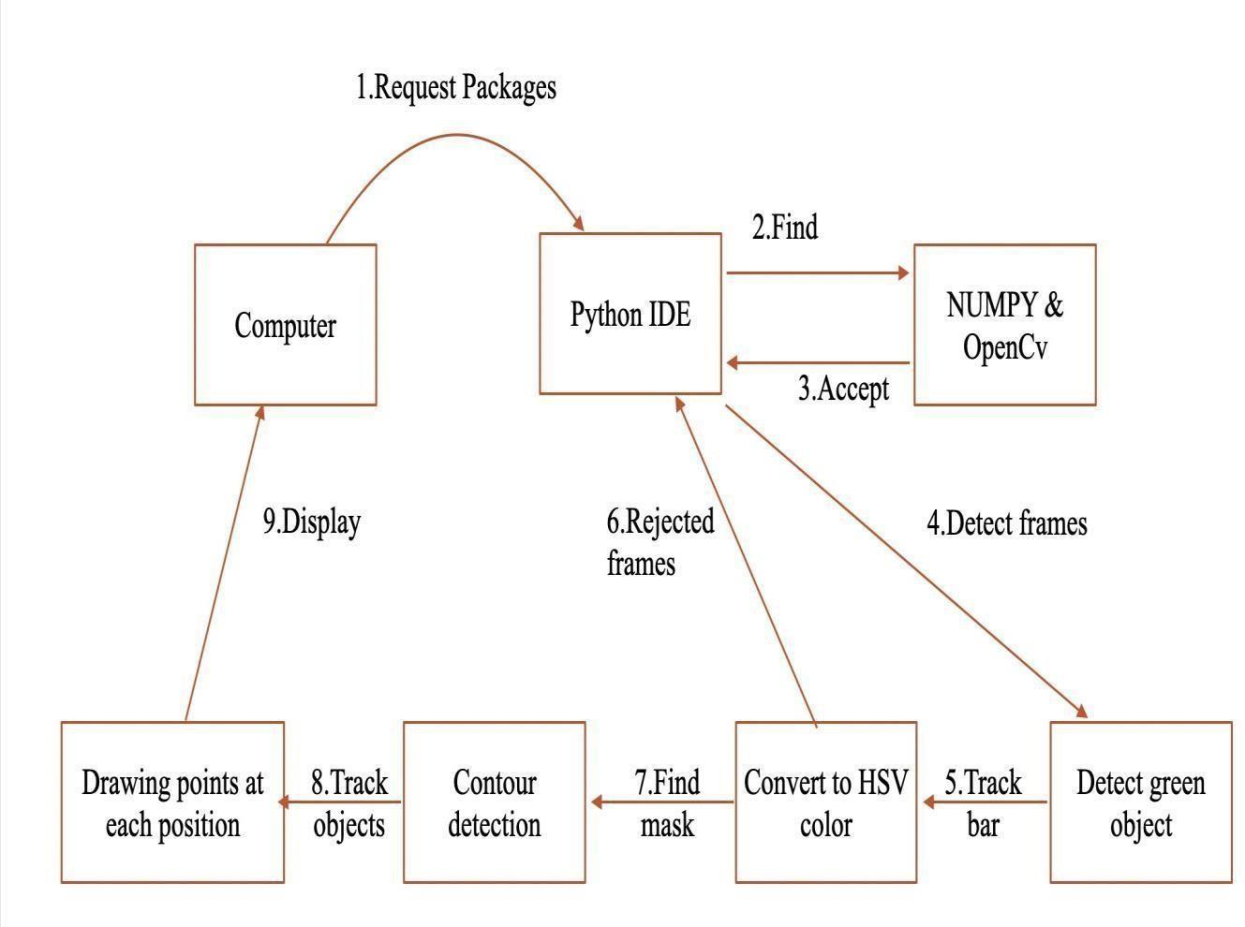
**4.3.Data flow diagram**

****

**Fig 4.3:-Data Flow Diagram**

In Figure, we are initially reading the frames of the object which we want to use as a marker on the Canvas window. After reading the frames convert them into HSV color space. Now prepare a canvas window along with a live frame and put ink buttons on it. Adjust the tracker to find the mask of the colored marker. After this preprocess the mask with morphological operations such as eroding and dilation. To the detected mask we need to find the contour. Find the centercoordinates of the contour and draw an enclosing circle. Keep storing the coordinates in arrays. Based on the coordinates of the color dequeues change t he frames of the screen to the respective colors. In the creation of the canvas window, we create a zero matrix using the np. zeros() function. Set some color buttons on top of the window to change the pencil color. These indexes will be used to mark the position of pointers in the color array. The kernel is to be used for dilation purposes. The colors will be used as ink for the drawing purpose. Setting up the canvas window and loading the default webcam of the PC.

**4.4.Collaboration Diagram**

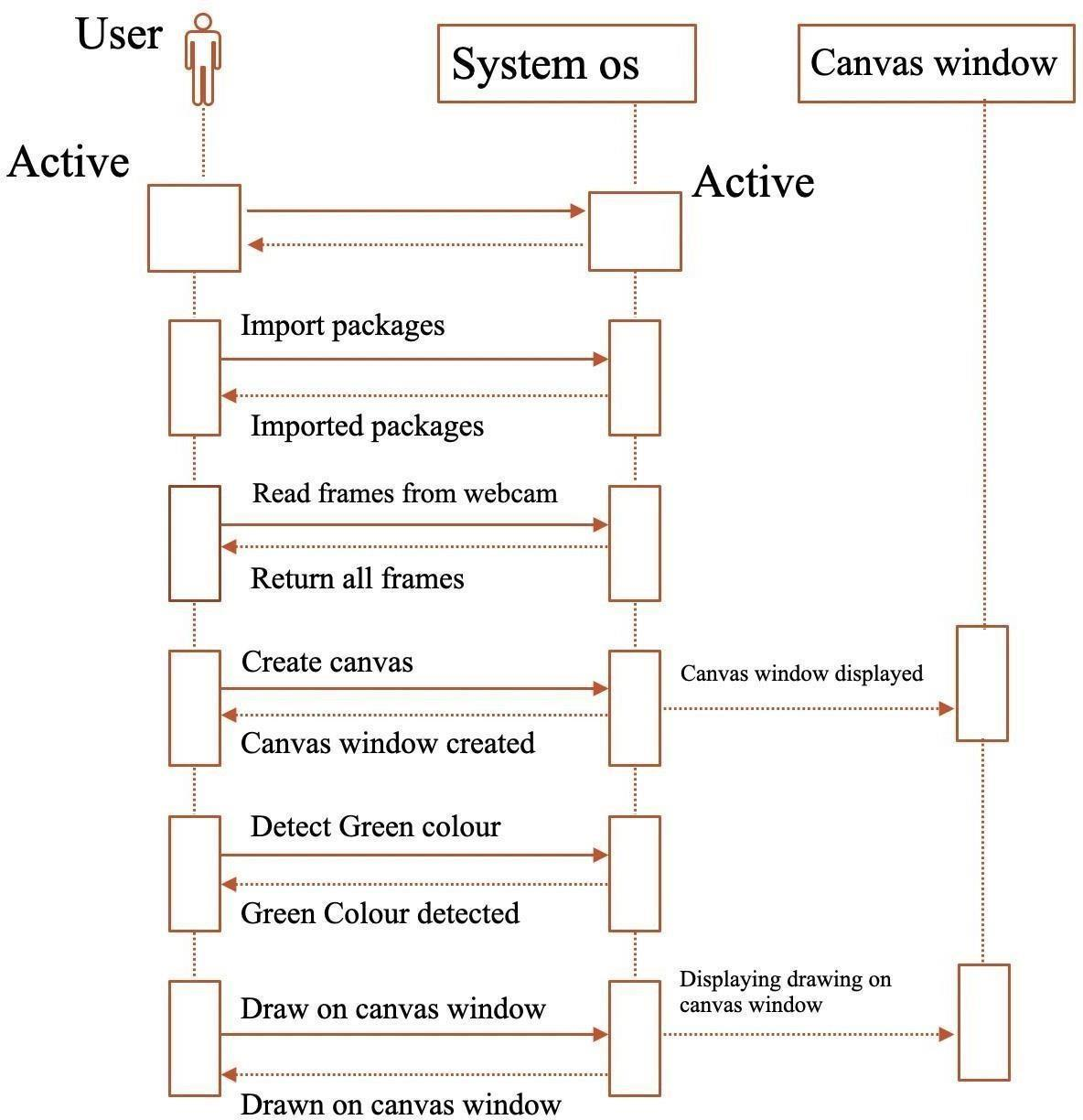
****

**Fig 4.4.:-Collaboration Diagram**

In Figure, we are requesting the packages such as Numpy, OpenCV to python IDE. We are using a green color object for object detection. So Python IDE detects the frames and sets a track bar for detecting blue color. Then read frames are converted into HSV color space. Now find the mask of the object. For tracking the movements of the object we have to detect the contour of the mask. Now track the movements of the contour. With this help, we can draw on the canvas just by moving the blue object in the air in front of the computer screen. Drawing

on the Canvas window is visible

**4.5.Sequence diagram**

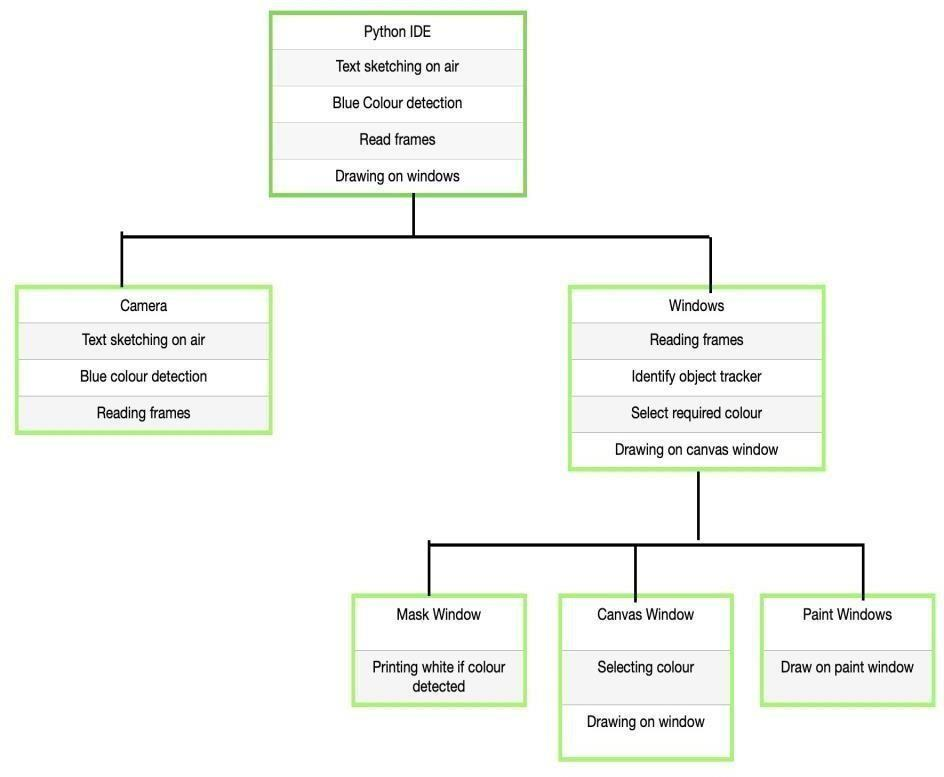
****

**Fig 4.5:-Sequence diagram**

In Figure, we are initially checking whether computer and system OS are in an active state or not. After checking the user have to install the necessary packages. Now read frames from the webcam and create the canvas window. Check whether the canvas window is created or not. After the creation of the window, detect the green color from the webcam. Now return or draw the frames on the canvas window and display the canvas window on the screen.

**4.6.CLASS DIAGRAM**

A class diagram in UML (Unified Modeling Language) is a visual representation of the structure and relationships within a system. It depicts classes, their attributes, methods, and the associations between them. Class diagrams are crucial for understanding the architecture of a system, facilitating communication among stakeholders, and guiding the implementation process in software development. They provide a blueprint for developers to design and implement software systems effectively



**Fig 4.6. Class Diagram**

Figure is a static diagram. The class diagrams are widely used in the modeling of object-oriented systems because there are the only UML diagrams. In this class diagram python, IDE is the main object that consists of two sub-objects those are camera and windows. The object python IDE consists of attributes that are text sketching on-air, green color detection, reading frames, and drawing on windows. The window object consists of three main windows those are Mask window, Canvas window, and Paint window.

**5.SOFTWARE ENVIRONMENT:**

**5.1.AIR CANVAS USING PYTHON**

This computer vision experiment uses an Air canvas, which allows you to draw on a screen by waving a finger equipped with a colourful tip or a basic coloured cap. These computer vision projects would not have been possible without OpenCV's help. There are no keypads, styluses, pens, or gloves needed for character input in the suggested technique.

**5.2.WORKING OF AIR CANVAS USING PYTHON**

Based on the web camera frames that were captured, a virtual paint programme was offered.

The web camera sends the system the frames that it has received. Until the application is finished, the approach uses a web camera to collect each frame.

In the below flow chart, the working of air canvas is explained. There are 5 steps. They are

1. Acquisition of videostream from WEBCAM

2. Extracting frames

3. Detect Hand and Handtips using Mediapipe and OpenCV

4. Draw the Land Handmarks

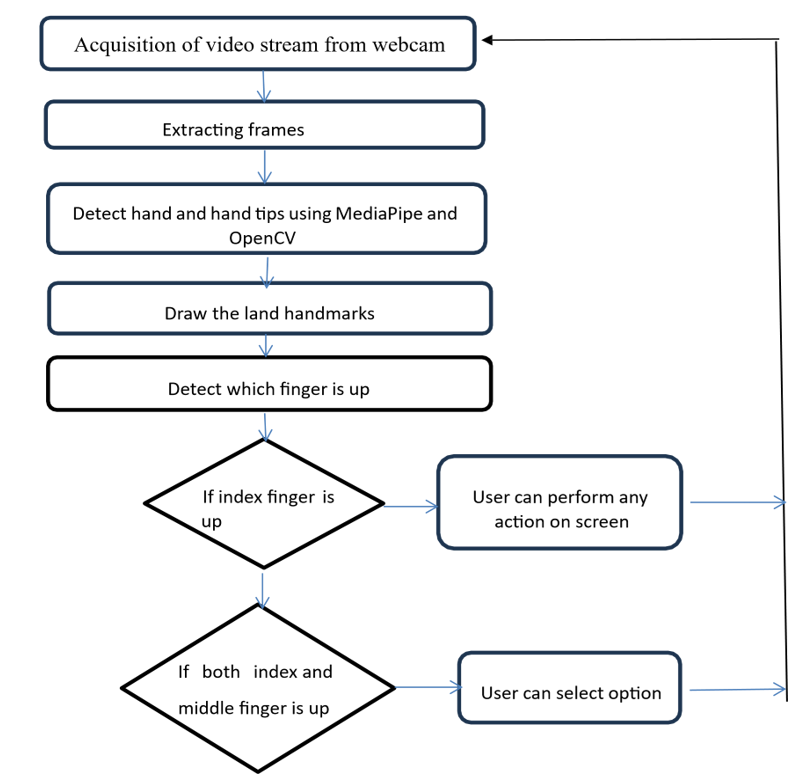
5. Detect which finger is UP**:**

a. If index finger is up, then user can perform any action that is writing or drawing.

b. If index and middle finger are up, then can select option from toolblox.If all fingers are up, then it clears the screen.

c. If all fingers are up, then it clears the screen

**5.3.FLOW CHART OF AIR CANVAS**

****

**Fig5.3:-flow chart of air canvas using python**

**5.4.THE CAMERA USED IN THE AIR CANVAS**

The frames that have been recorded by a laptop or PC's webcam serve as the foundation for the proposed AIR CANVAS. As seen in figure-4, the web camera will begin recording video after the video capture object is created using the Python computer vision package OpenCV. The virtual system receives frames from the web camera and processes them.



**Fig5.4:-Capturing video from webcam(computer vision)**

**5.5.CAPTURING THE VIDEO AND PROCESSING**

The Web cam is used by the AIR CANVAS system, and every frame is recorded up until the end of the application. As illustrated in the accompanying code, the video frames are converted from BGR to RGB in order to find the hands in the video frame by frame.

Def findHands(self, img, draw= True):

imgRGB=cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

self.results=self.hands.process(imgRGB)



**Fig 5.5:-Selecting an option**

**DETECTING WHICH FINGER IS UP AND PERFORMING THE PARTICULAR FUNCTION**

Using the tip Id of the specific finger that we located using the MediaPipe and the corresponding coordinates of the fingers that are up, as shown in Figure 6, we are able to determine which finger is up at this point. Then, in accordance with that determination, the specific mouse function is carried out.

The methodologies or the stages of the proposed system are discussed below:

1. Run or Execute the Code Execute the code once all the libraries are installed, this leads to turning the camera on automatically and the opencv frame with buttons displaying various shapes, colors, size, save, clear, erase etc.

2. Webcam starts recording and converts the video each frame and sends the frame to hand tracker class to track or detect the positions of finger. Figure 2 displays the buttons and frame which records video.

3. Detects Hand Landmarks Each frame received is compared with mediapipe hand landmarks i.e., positions of finger are found using getPositions() and which finger is opened using getUpFingers() functions of hand tracker class. These 2 functions are part of the Hand tracking module.4. Perform actions according to button Different buttons are choosing with the help of index finger hover over those buttons. In this way different colours are choose and also among them is the option of clear the screen.

**FOR THE MOUSE TO PERFORM ACTION**

The pynput Python module is used to programme the computer to execute the right mouse button click if the middle finger with tip Id=2 and the index finger with tip Id= 1 are both up and the distance between the two fingers is less than 40 px, as illustrated in figure 10.



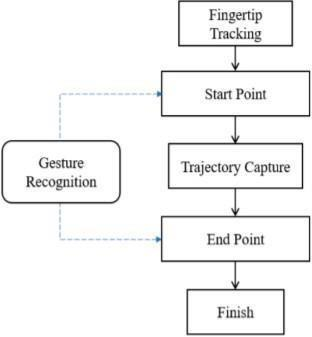
**Fig5.7.Performing an action**

**5.6. METHODOLOGY**

The objective is to create a free space where one can draw in air freely. The RGB camera detects the fingertip and tracks its motion through out the screen. Whenever the hand comes in front of the camera, the initial thing to do is detect the fingertip. There are various ways of fingertip detection.

**5.6.1. Fingertip Detection**

We are aiming to develop a system which can accurately detect the fingertips. First we will detect the whole hand and then the region segmentation is done. Region segmentation is a two step approach which includes skin segmentation and background subtraction. This system will work accurately in real time. For background subtraction we may use faster RCNN methods. Determining the center of gravity is important as it is used to detect some particular hand gestures for operations to be done. The proposed system aims to use two algorithms for centroid calculation and then take the average value of both as the final result. Distance transformation is the algorithm used and the pixel with the highest intensity is the center of gravity



**5.6.1.FigGesture Recognition Fingertip Tracking**

After successive detection of hand region and center of gravity the next step is to track the fingertip movement on the screen. According to the previous work, faster R-CNN handheld detector is intensive and the frames produced are below real-time performance. Thus we are aiming to use KCF tracting algorithm. The algorithm converts the detected fingertip into HSV color space. After the mask is detected in the air, the system will do some morphological operation to remove the impurities from the masked image. After detecting the contours drawing the line is the most important step. After this a python deque is created. `The deque will memorize the position of the outline in each subsequent frame, and we will use these accumulated points to create a line using OpenCV’s drawing capabilities.

**5.7.Advantages Of Air Canvas Using Python**

 **Contactless Interaction**:

* **Hygiene**: No need to touch surfaces, making it ideal for environments

requiring high hygiene standards.

* **Accessibility**: Useful for individuals with mobility impairments who might find traditional drawing tools challenging to use.

 **Innovative and Engaging**:

* **Interactivity**: Provides a novel way of interacting with technology, which can be engaging and fun.
* **Creativity**: Offers a new medium for artists and creators to explore their creativity.
* **Versatility**:
* **Multiple Applications**: Can be used in various fields such as education, gaming, art, and presentations.
* **Integration**: Can be integrated with other technologies like augmented reality (AR) and virtual reality (VR).

 **Python Ecosystem**:

* **Libraries and Frameworks**: Python has a rich set of libraries such as OpenCV, TensorFlow, and Mediapipe that facilitate the development of computer vision and machine learning applications.
* **Community Support**: A large community of developers and abundant resources make troubleshooting and development easier.

**5.8.Disadvantages Of Air Canvas Using Python**

 **Technical Challenges**:

* **Accuracy**: Ensuring precise tracking of hand movements can be challenging and may require sophisticated algorithms.
* **Latency**: Real-time processing demands significant computational power, which can lead to latency issues if not optimized properly.

 **Environmental Limitations**:

* **Lighting Conditions**: Performance can be affected by changes in lighting, background, or other environmental factors.
* **Camera Quality**: High-quality cameras may be required for better accuracy, which can increase costs.

 **User Fatigue**:

* **Physical Strain**: Prolonged use may lead to arm and hand fatigue, especially if used for extended periods.
* **Learning Curve**: Users may need time to get accustomed to the system and learn how to use it effectively.

 **Hardware Dependence**:

* **Device Requirements**: Requires devices with cameras and sufficient processing power, which may not be available in all settings.
* **Maintenance**: Regular calibration and maintenance of hardware components might be necessary to ensure consistent performance.

 **Privacy Concerns**:

* **Data Security**: Handling video data can raise privacy concerns, necessitating robust data security measures to protect user information.

**5.9.MODULES USED IN PROJECT**

**5.9.1.Color Tracking**

Understanding the HSV(Hue, Saturation, Value) shading space for Color Tracking.

Furthermore, following the little hued object at fingertip. The approaching picture from the webcam is to be changed over to the HSV shading space for recognizing the hued object at the tip of the finger.

**5.9.2.**  **Trackbars**

When the trackbars are arrangement, we will get the real time esteem from the trackbars and make range. This reach is a numpy structure which is utilized to be passed in the capacity cv2.inrange(). Thi capacity returns the Mask on the hued object. This Mask is a high contrast picture with white pixels at the situation of the ideal tone.

**5.9.3.Contour Detection**

Recognizing the position of the Colored item at fingertip and shaping a circle over it. We are playing out some morphological procedure on the Mask, to make it liberated from contaminations and to distinguish shape without any problem. That is Contour Detection.

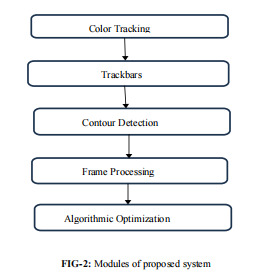
**5.9.4.Frame Processing**

Following the fingertip and drawing focuses at each position for air material impact. That is Frame Processing.

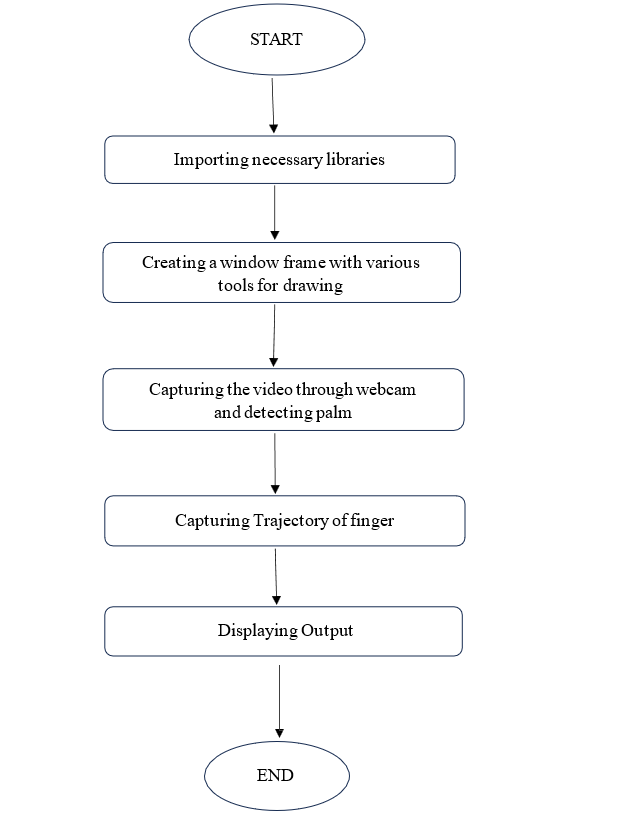
**5.9.5.Algorithmic Optimization**

Making the code efficient to work the program without a hitch. That is Algorithmic

Optimization.



**5.9.6.ARCHITECTURE:**



**Fig 5.9.6:-Architecture of Air Canvas.**

**START**

In this Start module, the initiation takes place. At this place, the system will start and air canvas process takes place.

**IMPORTING NECESSARY LIBRARIES**

At this stage, we import necessary libraries like numpy, opencv, mediapipe. We can import this libraries after downloading python. We import libraries from python and these libraries play a main role in this air canvas project.

**CREATING A WINDOW FRAME USING VARIOUS TOOLS FOR DRAWING**

In this stage of architecture, we write a code using python and a window frame is created by running that code. In that frame, we also attached some other tools like colours which are blue, yellow, red, green and also a clear option. Like this, a window frame is created using various tools for drawing.

**CAPTURING THE VIDEO THROUGH WEBCAM AND DETECTING PALM**

Then a window frame is appeared, now just by using only webcam and our palm we can draw whatever we want on that window frame. Here we can see landmarks on our palm on the screen. These landmarks are detected using python library mediapipe. Landmarks are the marks which we seen in our palm there are 21 landmarks.

**CAPTURING TRAJECTORY OF FINGER:**

In this stage of architecture, the trajectory of finger capturing takes place. Trajectory means curve or path with a specific angle. The webcam starts capturing the path of the finger tips mainly fore finger. The webcam captures those path and store those in the form of frames.

**DISPLAYING THE OUTPUT:**

The frames which are formed in the above stage are stored in an array, then these frames are displayed in the form of output. Then the action we performed are displayed on the screen.

**END:**

This is the last stage of the air canvas project. In this stage, the process is completed and the output screen is shown with the action performed by us. Then we can close or end or terminate the screen

**6. IMPLEMENTATION**

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

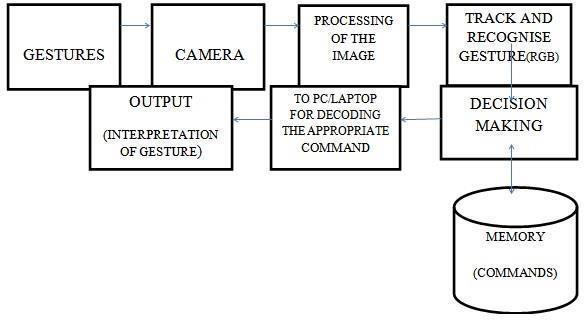
* Prepare the canvas frame and put the respective ink buttons on it. Adjust the track bar values for finding the mask of colored marker.
* Preprocess the mask with morphological operations. (Erosion and dilation)
* Detect the contours, find the center coordinates of largest contour and keep storing them in the array for successive frames .(Arrays for drawing points on canvas)
* Finally draw the points stored in array on the frames and canvas.
* Requirements: python3, NumPy, OpenCV installed on your system.

**6.1. SDLC TO BE USED Planning**

We had set a user-friendly application which everyone can learn to use without any prior knowledge.

**6.1.1. Analysis**

Analysing and thinking where we got our ideas from about this particular application is the need of papers, extra softwares during classes,designing.

****

**Fig 6.1.1:-System Design Flow Diagram**

**6.1.2.Testing**

We are constantly testing our application to make it free from any crashing and potential bugs. For testing we explore more and more by inputting as many values as possible to find the potential bugs.

**6.1.3.Maintenance**

We try to update our application from time to time. Our main focus is on improving the user interface and remove all logical problems from our application. More over the leveraged technological tools used in our project are updating time to time. Thus, it becomes absolute necessary to keep up to the mark with tools and code behind it as well.

**7. 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 specific 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 # Red

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], colors[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()

**SO WHAT IS COLOR DETECTION AND SEGMENTATION IN IMAGE PROCESSING**

We know that artists create paintings on a canvas. But what if we can paint on air just by waving out hands. So, in this project, we are going to build an air canvas using OpenCV and Python. OpenCV is an opensource computer vision library for performing various advanced image processing tasks. We’ll use color detection and segmentation techniques to achieve this objective. Here, we’re going to use a green object to simulate a pencil for the canvas. 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. Project Prerequisites:

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

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

3. Numpy – 1.20.1

**STEPS TO DEVELOP AIR CANVAS PROJECT USING OPENCV**

1. Import necessary packages.

2. Read frames from a webcam

3. Create the canvas window

4. Detect the green color

5. Draw on the canvas

**Step 1 – Import necessary packages and pre-define some settings:**

To build this project, we need only two packages, OpenCV and Numpy. So first we’ve to import

these.

**Step 2 – Read frames from the Webcam**

* + First, we created a VideoCapture object and defined the object as cap.
  + cap.read() function reads each frame from the webcam.
  + cv2.flip() flips the frame. The arguments are input image/frame and axis. 0 is for flip vertically, and 1 for flip horizontally.
  + cv2.imshow() function shows frame in a new window.
  + cv2.waitKey(1) == ord(‘q’) keep open the window until ‘q’ is pressed.

**Step 3 – Create the canvas window:**

* + First, we define some colors that we’ll be using during this project.
  + Then we create an empty blank canvas the same size as the camera frame.
  + np.zeros() create a matrix containing all 0’s.

We want to change the pencil color during painting, That’s why we have to create some buttons in the frame. Using those buttons we can select the pencil color during painting in real time

* Using cv2.rectangle(), we draw some rectangles of different colors that we’ve defined before.
* cv2.putText() draws texts on a frame. Using this function we put each of the names of the colors to their corresponding rectangle box.

**Step 4 – Detect the green color:**

Here we’re using a green object to simulate the pencil of our canvas. OpenCV reads frames as BGR color space. But to detect colors, we have to convert the frame to the HSV color space.

**But what is HSV color-space?**

HSV stands for HUE, SATURATION, and VALUE (or brightness). It is basically a cylindrical color space.

* **HUE–** The hue encodes color information in angular dimension.
* **SATURATION** – Saturation encodes the intensity of color.
* **VALUE** – Value represents the brightness of the color.

So let’s detect the green object:

* **cv2.cvtCOLOR()** function converts the color-space of an image.
* **cv2.inRange()** function returns a binary segmented mask containing a white region where the color is detected and the rest of the region as black

.

* **Cv2.dilate()** increases the area of the segmented white region.
* **cv2.findContours()** finds all the continuous points along the boundary having the same intensity or color. In our case, we are finding points along the segmented white region of the mask. Now, let’s find the center of the detected object. The center point will be the reference point with which we’ll be painting something on the canvas.

**Step 5 – Draw on the canvas:**

* First, we check if the previous\_center\_point is 0, which means nothing is detected in the previous frame.
* But in the current frame, if something is detected, we already know the coordinates of the color selection buttons. So we checked in which button the center point of the contour is detected first. And then set the color accordingly.
* The first button is for clearing the canvas, that’s why if the center point is detected in the first button, then we reset the canvas to a blank canvas using np.zeros() function.
* Now we check if the previous\_center\_point is not 0, which means a contour is detected in the previous frame, then we draw a line between the previous\_center\_point and the current center point.
* After that, we update the previous\_center\_point to the current center point. But we want to show the drawings in the original frame. So let’s see how we can do thatalso.
* Using cv2.cvtCOLOR() function we convert the canvas to grayscale image.
* Then using cv2.threshold() we create a binary mask of the canvas.
* Then again we convert the binary mask to BGR color space. Because in order to add two images, the shape of the two images needs to be the same.
* cv2.bitwise\_and() is true if both pixels are greater than zero.
* cv2.bitwise\_or() is true if one of the pixels is greater than zero.

**LIBRARIES USED FOR HAND LANDMARK MODEL**

* + Mediapipe
  + Hand Landmark Model
  + Numpy

Hand tracking is the process in which a computer uses computer vision to detect a hand from an input image and keeps focus on the hand’s movement and orientation. Hand tracking allows us to develop numerous programs that use hand movement and orientation as their input. We tend to write the same code in different projects to perform hand tracking as part of our program.

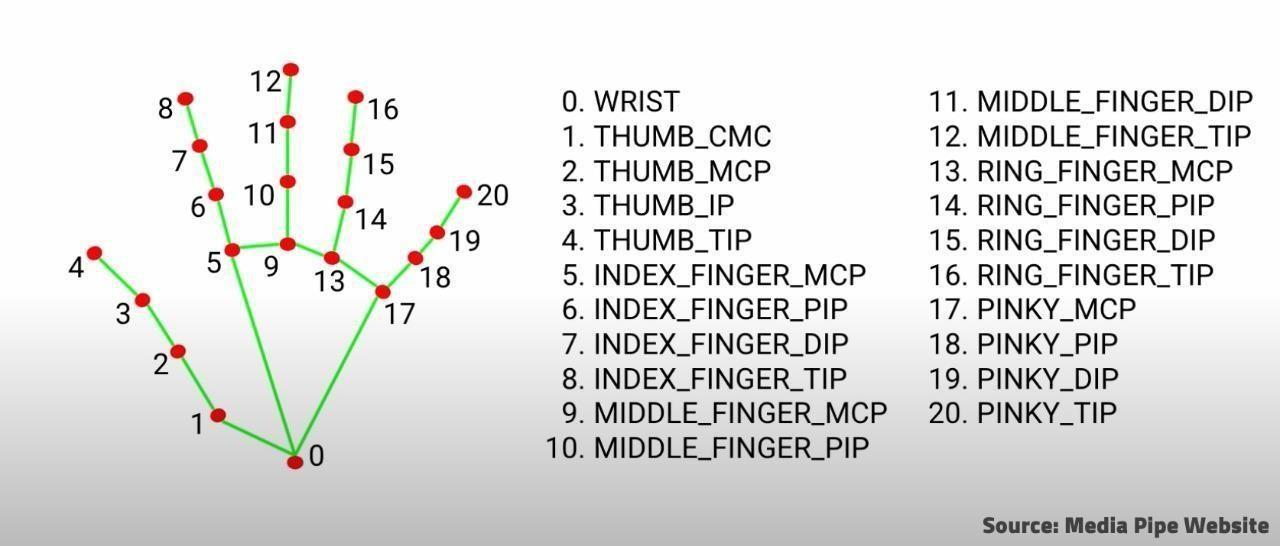
Creating a hand tracking module solves this problem since we write the code once.

We then convert this piece of code into a module. We can import this module into any python project that we are working on and it will perform hand tracking.

Before we jump into coding, let us discuss how MediaPipe performs hand tracking.Hand tracking using MediaPipe involves two stages:

* **Palm detection** - MediaPipe works on the complete input image and provides a cropped image of the hand.
* **Hand landmarks identification** - MediaPipe finds the 21 hand landmarks on the cropped image of the hand.

The 21 hand points that MediaPipe identifies are shown in the image below:



**Fig:-Hand Landmark Model**

**8.RESULTS:**

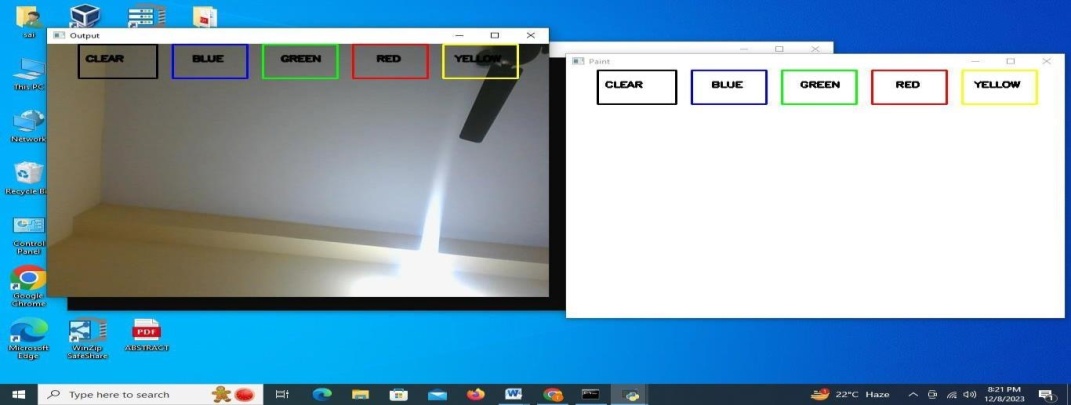
Python, OpenCV, and other python modules are the foundation of this project. We learn how to choose an element and use that choice to initiate an action. Fresh ideas and techniques for solving problems. When developing the hand tracking module, we learn how to use OpenCV concepts like limiting the active hand detection and assigning the action to the fingers. We acquire the project deployment techniques. We use modern applications like PyCharm, Visual Studio Code, and Figma. We have the opportunity to deal with several platforms, languages, and technologies by working on this project.

**8.1. USER INTERFACE REPRESENTATION**

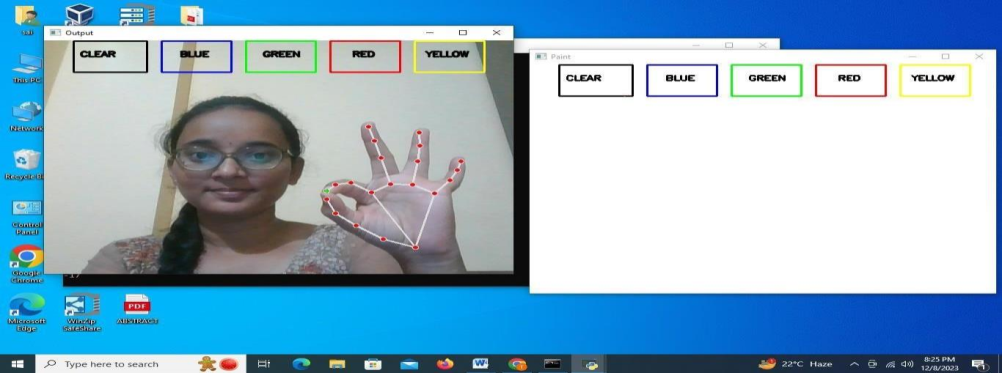
We created a simple output using web-cam with various options for the user to select the desired colour. The position of the colour box with placed. After selecting the colour an automatic list is created with the coordinates where ever our hand is moving in the screen, it gets stored in list to give a collections of pixels as generated output.

The output generates 2 screens one to help web cam to gather information about the movement of our hand and subsequent tracking and the other one is a blank white screen

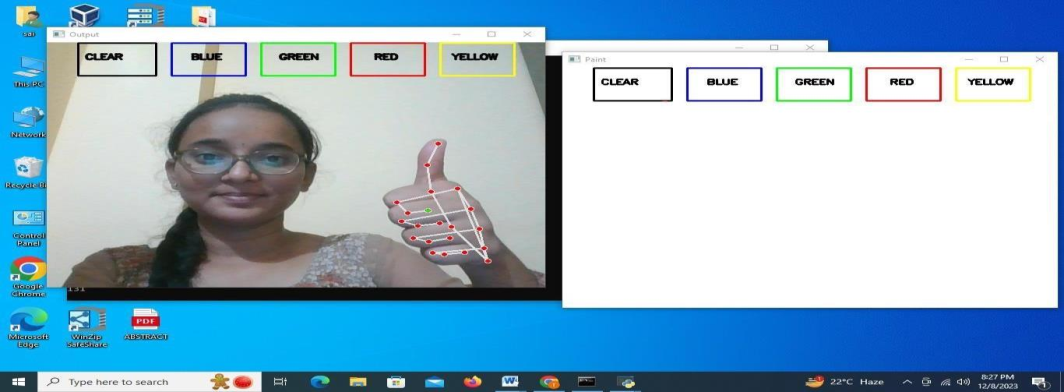
to print that movement.



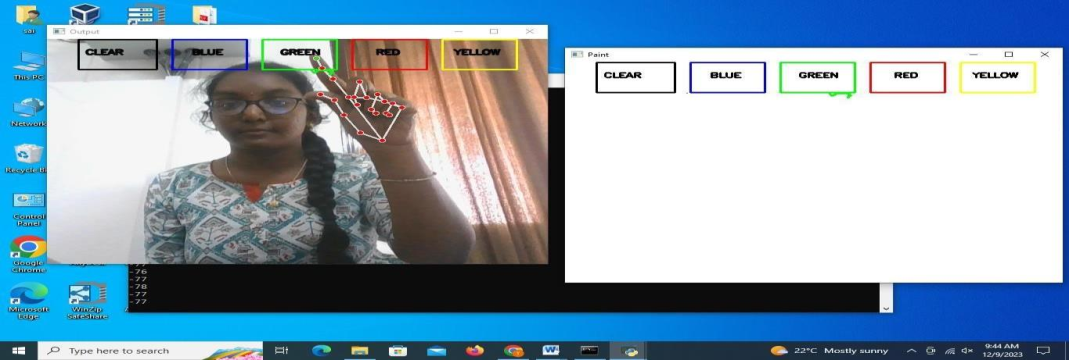
**Snapshots of GUI**



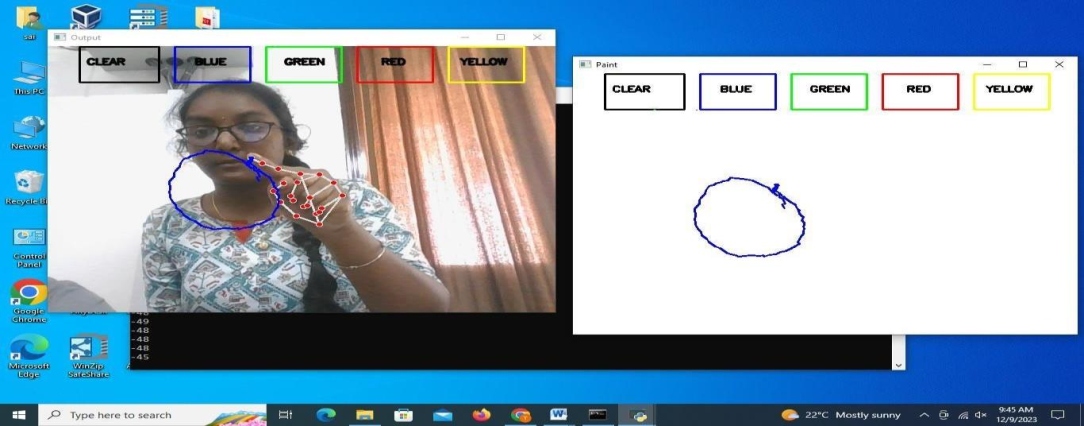
**Fingers Tracking with Hand Landmarks**



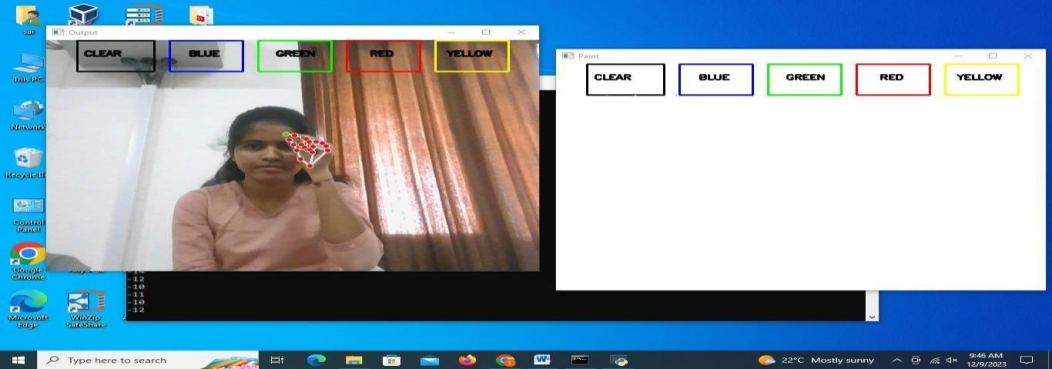
**Thumb detection while all the landmarks not visible still detectable**



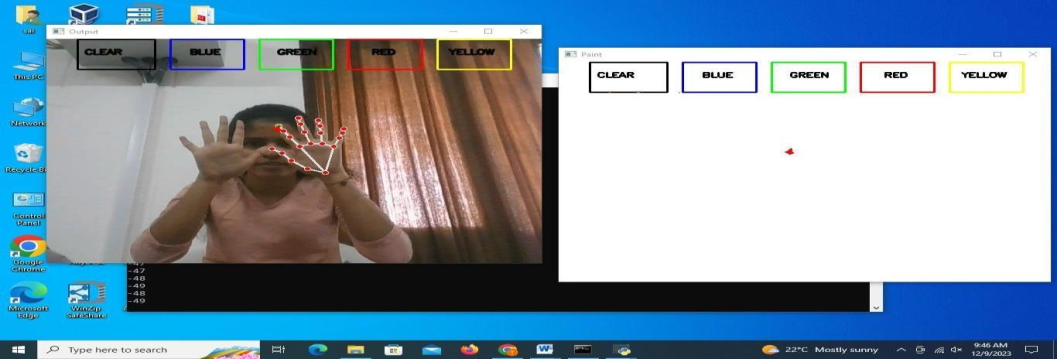
**Selection of color green**



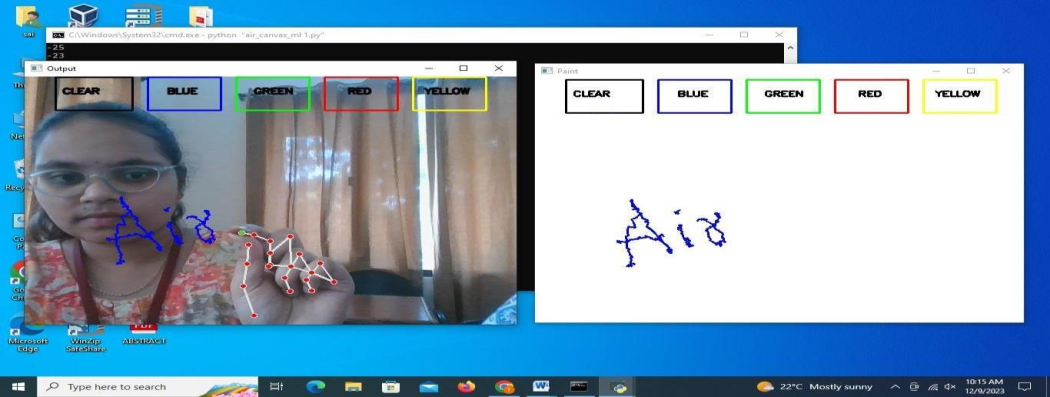
**Drawing a circle**

****

**Detection blocking with thumb finger**



**Multi Detection at a Time**

****

**Writing in the air**

****

**Drawing a rectangle**

#### 9. CONCLUSION

This program has the potential to challenge traditional writing methods. Eliminates the need to carry a cell phone in hand to take notes, to give an easy way on the go to do the same. It will again work towards a greater purpose in helping especially those who know them to communicate easily. Even adults who find it difficult to use the key board can easily use the program. Expanding functionality, this program can also be used to control IoT devices soon. Air painting can also be made happen. This program will be very good smart clothing software using which people can work better with the digital world. The unpopular reality of tax payers we see can make the text come alive. Wind-writing programs should listen only to their master's control touch and should not be misled by people all around. Such discovery algorithms are as follows as YOLO v3 can improve fingerprint recognition accuracy and speed. In the future, progression Artificial Intelligence will improve the efficiency of writing in the air. This program has the potential to challenge traditional writing methods. Eliminates the need to carry a cell phone in hand to take notes, to give an easy way on the go to do the same. It will again work towards a greater purpose in helping especially those who know them to communicate easily. Even adults who find it difficult to use the keyboard can easily use the program. Expanding functionality, this program can also be used to control IoT devices soon. Air painting can also be made happen. This program will be very good smart clothing software using which people can work better with the digital world. The unpopular reality of taxpayers we see can make the text come alive. Wind-writing programs should listen only to their master's control touch and should not be misled by people all around. Such discovery algorithms are as follows as YOLO v3 can improve fingerprint recognition accuracy and speed. In the future, progress on Artificial Intelligence will improve the efficiency of writing in the air. 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.

#### 10. FUTURE SCOPE/ENHANCEMENT

* Air Canvas has immense future potential as a platform for artistic expression, digital design, and interactive experiences.
* Its accurate hand-tracking capabilities using OpenCV and MediaPipe can be further enhanced to include gesture recognition, allowing users to perform specific actions by making predefined hand gestures.
* The system can be integrated with augmented reality (AR) and virtual reality (VR)

technologies, enabling users to create immersive 3D artworks in a virtual space.

* Expanding the system's compatibility with various devices and operating systems would broaden its accessibility and user base.
* Ongoing research and development can focus on refining the system's performance under challenging conditions and exploring additional features to enhance the overall user experience.

If we had more time to devote to this endeavor, we would enhance hand contour recognition, investigate our initial Air Canvas objectives, and make an effort to comprehend the multicore module. We would need to go further into OpenCV in order to improve hand gesture tracking. There are other ways to analyze contours, but for this particular procedure, it would be beneficial to look at the color histogram that was used to draw the contours in question. Additionally, we could test out various interpolation techniques. PyGame has a line drawing technique(pygame.draw.line ()) that might be helpful for creating lines that are smoother and cleaner. In the same line, adding different brush types, textures, and perhaps a rubber to Air Canvas will strengthen its artistic capabilities. Unique features that imitate actual creativity software could also include letting the user save their finished product or watching their drawing

process as an animation. There might even be a way to link Air Canvas with real digital drawing applications like Adobe Photoshop, Clip Studio Paint, or GIMP! Finally, by understanding how multicore processing interacts with in-order information processing, we could make significant progress.