International Institute of Professional Studies Devi Ahiliya Vishwavidyalaya, Indore.

PROJECT REPORT

ON

Air Canva on White board

Dissertation submitted in fulfilment of the requirement

for the award of BCA (3 yrs)

VI Semester

Under The Guidance of:

Dr. Kirti Mathur

Submitted By:

Paritosh Verma(IC-2k20-51)

Paragi Joshi(IC-2k20-51)

DECLARATION

We hereby for the declare that the project entitled "**Air Canva on White board**" which is submitted by us for the fulfilment of the requirement for the award of Bachelor of computer application VI Semester to International Institute of Professional Studies, DAVV Indore. This work has not been submitted anywhere else and comprises of our own work and due acknowledgement has been made in text and to all other materials used

Signature of the students:

Date :

DISSERTATION APPROVAL SHEET

The dissertation entitled “**Air Canva on White board**” being submitted by Paragi Joshi(IC-2k20-50) and Paritosh Verma(IC-2k20-51**)** in fulfillment of the requirement for the award of Bachelor of Computer Application VI semester to International Institute of Professional Studies, DAVV, Indore is satisfactory and approved.

Internal Examiner	**External Examiner**
Signature:	**Signature:**

ACKNOWLEDGEMENT

We acknowledge our sincere thanks to those who have contribute significantly to this project. It is pleasure to extend deep gratitude to our internal guide Dr. Kirti Mathur, for her valuable guidance and support and to continuously prompt us for the progress of our project. We thank her for her valuable suggestion towards project, which helped us in making this project more efficient and user friendly.

We are also thankful to all others who have directly or indirectly helped us to carry out this work.

ABSTRACT

This project presents a real time video based pointing method which allows sketching and writing text over air in front of camera. Air Canvas is developed for communicating a concept or presenting an idea in the real-world space. Proposed method track the colored object in the camera frames and here color detection and tracking is 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 and with that whatever you will draw in air will reflect on the whiteboard.

The proposed project is built in Python 3.6 and it utilizes the exceptionally well known OpenCV and numpy library.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Ever wanted to draw your imagination by just waving your hand in air. 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 marker with camera. Here a colored object is used as the marker.

Project will be exciting and fun to build. We will be working with colors and you will get to learn about many concepts throughout this project. Color detection is necessary to recognize objects; it is also used as a tool in various image editing and drawing apps.

Color detection is the process of detecting the name of any color. Simple isn't it? Well, for humans this is an extremely easy task but for computers, it is not straightforward. Human eyes and brains work together to translate light into color.

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

The OpenCV library is one of the Assembly's favorite toolkits, with its easy-to-use processing capabilities for real-time computer vision. Working seamlessly with Python, the open-source library has been very useful for processing live video capture on the fly with little overhead, delivering impressive results with minimal code.

I. Air Canvas: - Air Canvas is a without hands computerized drawing material that uses, a camera and OpenCV to perceive and plan hand signals onto a screen.

II. Contour: - Shapes may be clarified simply as a bend becoming member of all of the consistent a points (alongside the limit), having equal tone or power. The forms are a valuable instrument for shape analysis and item location and granting.

1.1 Motivation

The underlying inspiration was a requirement for modern day teaching purposes, organizational purposes and for physically challenged people .we realizes that there are numerous ways like touch screens and then some yet what might be said about the schools which cannot bear the cost of it to purchase such gigantic huge screens and instruct on them like a T.V. Also physically challenged people can express their imagination using our tool without dependency on any input devices. Along these lines, we thought why not a hand can be followed. Consequently it was OpenCV which acted the hero for these PC vision projects.

1.2 Objective

I. General objective

The main objective of the project is to achieve an air canvas so that anyone can draw in the air.

II. Specific objectives

In order to attain the general objective, the following are the list of specific objectives:

- To achieve the air canvas that reflects on the whiteboard you need to have open CV library in your system.
- Import all the modules that come in use.
- We will create track bars for adjusting color of the marker.
- We will use kernel for dilation purposes.
- To interact with the user we need to load the webcam of the system.
- The webcam will detect the movement of marker and store the data.
- We will draw a whiteboard for implementing the air canvas.
- You can adjust the color of the marker by adjusting track bars.
- The project will be implemented and you will able to draw on the whiteboard by waving your hand in air.

- ## 1.3 Scope and limitations of the project

1.3.1 Scope of the project

There are several motivations to develop the air canvas on the whiteboard:

- The proposed project will have several modern scopes in the world of Artificial intelligence and machine learning.
- The project can interact with the user that wants to draw anything in air and wants it to show on the whiteboard.
- This project is useful for modern day teaching purposes, children nowadays need attractive way of teaching and also it helps them to grab faster.
- It is also useful for many organizations for presenting anything through the webcam of your system.
- Also the color detection sensor market was valued at USD 1.77 billion in 2020 and is expected to reach USD 2.91 billion by 2026 and work at a CAGR of 8.68% over the forecast period (2021-2026).

1.3.2 Limitations of the project

When there are many advantages of a system there are also many problems with the system. In performing this project user may face difficulty in precise detection of the object.

- When webcam captures the marker it sometimes captures more than
One point because of the same color that is of the marker in the background.
- Sometimes dilation doesn’t occur and impurities are reflected on the whiteboard.
- Lack of proper knowledge how to use the marker.
- Lack of proper background.
- Lack of good quality of the web camera.
- Sometimes the module that has been responsible for working of the project doesn’t get installed in other systems.

1.4 Approach used

It specifies the approach and technology used to develop the whole artificial intelligence based project such as, the methods used to gather data, approach used to design the front end, and software and hardware requirements used to implement the project.

The proposed method provides a natural human-system interaction in such way that it do not require keypad, stylus, pen or glove etc for character input. It was OpenCV which came to the rescue for these computer vision projects

Chapter 2
Requirement

The project team met with the users who took interest in the idea of our project to outline each requirement in detail. The requirements are captured by using clear, simple language easy to understand by any user.

Clearly articulating the customer needs & wants

We will be understanding and articulating the customers need in this .As we can not speculate on this. By surveying through various customers we got some of the common customer needs types that are:

- **Functionality** — Functional needs to be able to solve user's problem or desire.
- **Price** — user wants the product to be budget friendly.
- **Customer Experience** — user's experience needs to be easy to use.
- **Design** — user wants Product or system design to be easy and intuitive to use
- **Performance** — user wants the Product or system to perform correctly as per his goal.
- **Accessibility** — user wants to access product or service by multiple channels.

Once the requirements were clearly articulated and defined, we conducted a requirements walk-through with our team members . There was a shared understanding across the team and the user, there was a inconsistent interpretations
Between the team but we came across a solution to solve that.
And by keeping all this users requirements in our mind we took the test and results are:

- In our project the user will have a **full functionality** to every feature of the tool we have built.
- **Yes**, when launched on a big platform our system will be **budget friendly and cost effective** as it doesn't need any huge setup or network
- As the name suggests "Air Canva" it is so **easy to use** by the users be it an adult, a child, physically challenged people, old people.

- System design of our project is **precise and more effective** as we do not have a lengthy source code we have used the openCV module in python solely and it is an efficient yet small and convenient design.
- Yes, our product will provide the user a **great performance** and the goal will be achieved by the user.
- The user will be **able to access our product** when we launch it on a big platform there will be easy access provided by us.

Requirements may normally change over time, as more information and knowledge is captured (particularly during the solution deep dive) and whilst working closely with customers. Customers can change their mind around specific requirements. The trick is in knowing how to manage the change and ensuring when to lock down the requirements.

Therefore to keep track of changed user requirements we will be conducting multiple surveys in specific amount of time ,we will brainstorm ideas by interacting with different kind of users .

We want our project to satisfy future needs of the users and upcoming customers preferably over anything else.

2.2 Use Case Diagram

2.2.1 Main user

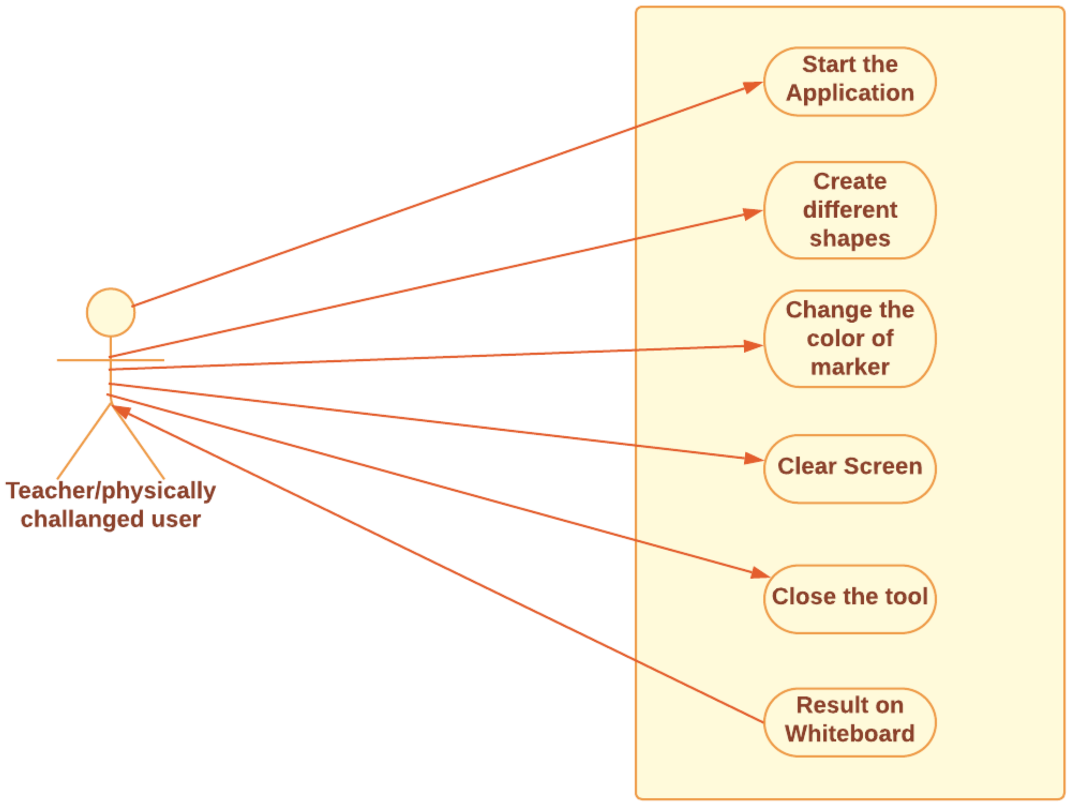


Fig 1: use case diagram of main user

2.2.2 Viewer

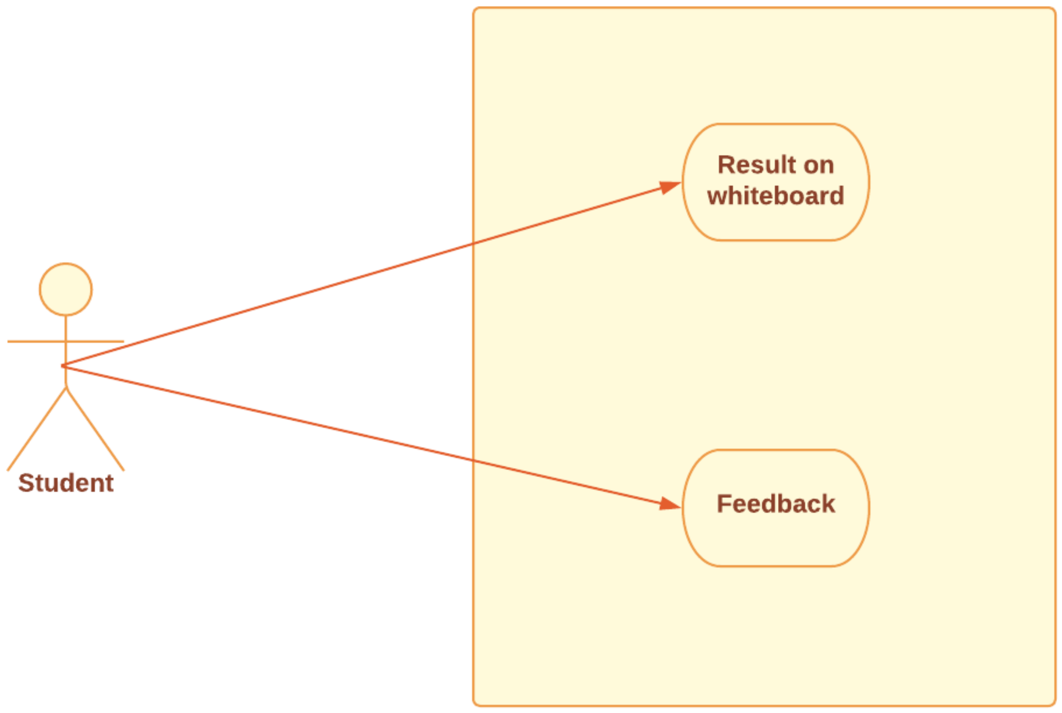


Fig 2 : use case diagram of viewer

2.2.3 Overall Use case diagram

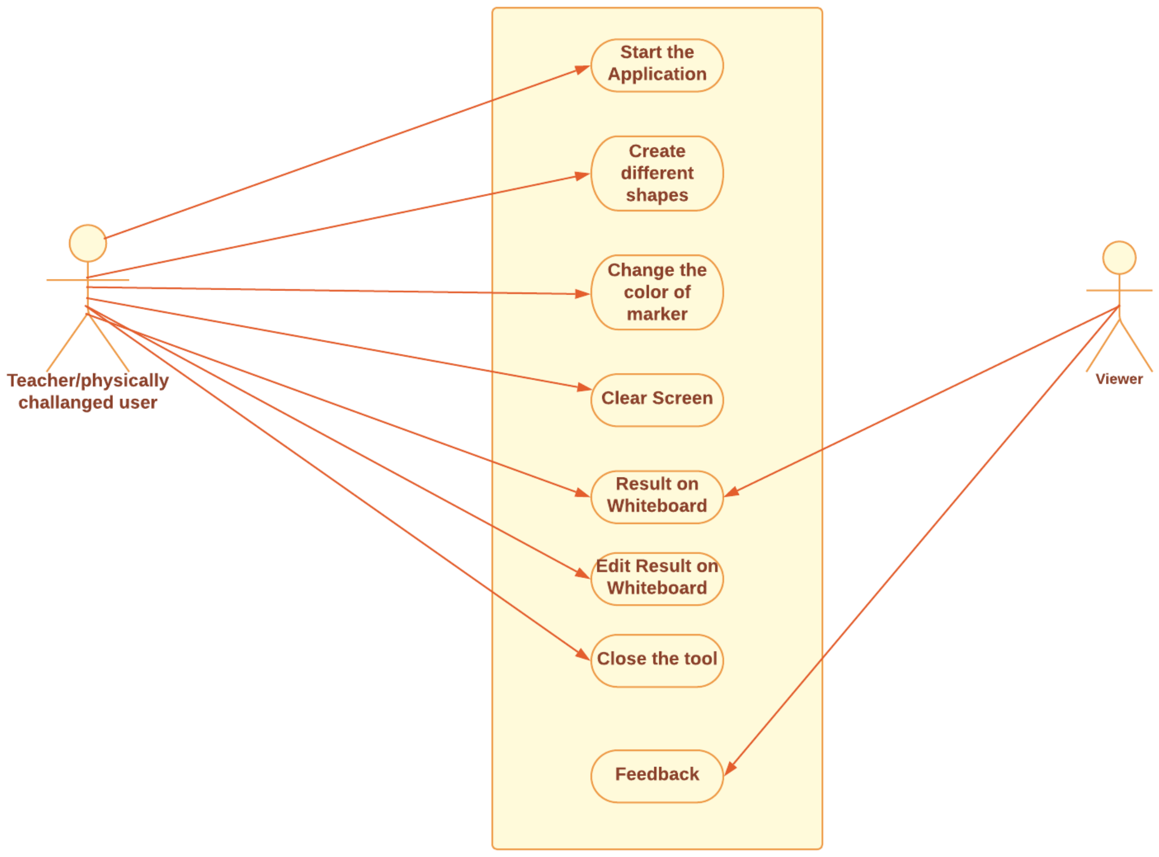


Fig 3 : Overall use case diagram

2.3 Existing System

The superior pen contains of a tri pivotal accelerometer, a microcontroller, and a RF remote transmission module for detecting and amassing gelocity will increase of hand composing and movement directions. Our implanted project first concentrates the time-and recurrence area highlights from the rate boom indicators and, then, at that point, sends the symptoms with the aid of using making use of RF transmitter. In beneficiary section RF symptoms may be gotten with the aid of using RF recipient and given to microcontroller. The regulator procedures the records finally then results may be proven on Graphical LCD.

2.4 Problem Definition

Item following is considered as a significant undertaking inside the field of Computer Vision. The development of quicker PCs, accessibility of cheap and great quality camcorders and requests of robotized video investigation has given ubiquity to protest following methods. For the most part video investigation system has three significant stages: initially distinguishing of the item, besides following its development from one casing to another and ultimately breaking down the conduct of that article. Who don’t needs to simply move their fingers in air and get their ideal picture.

2.5 Proposed System

In this proposed framework, going to utilize webcam and show unit (monitor screen). Here, will be utilizing pen or hand for drawing attractive images in front of the camera then we will attract those images, it will be shown the presentation unit. Our mounted on Framework is suit for decoding time-collection pace boom alerts into extensive thing vectors. Users can make use of the pen to compose digits or make hand motions and so on may be proven at the presentation unit.

2.6 Architecture of the Proposed System

The architecture of the proposed system has three components:

Webcam, system software and hardware, output Interface.

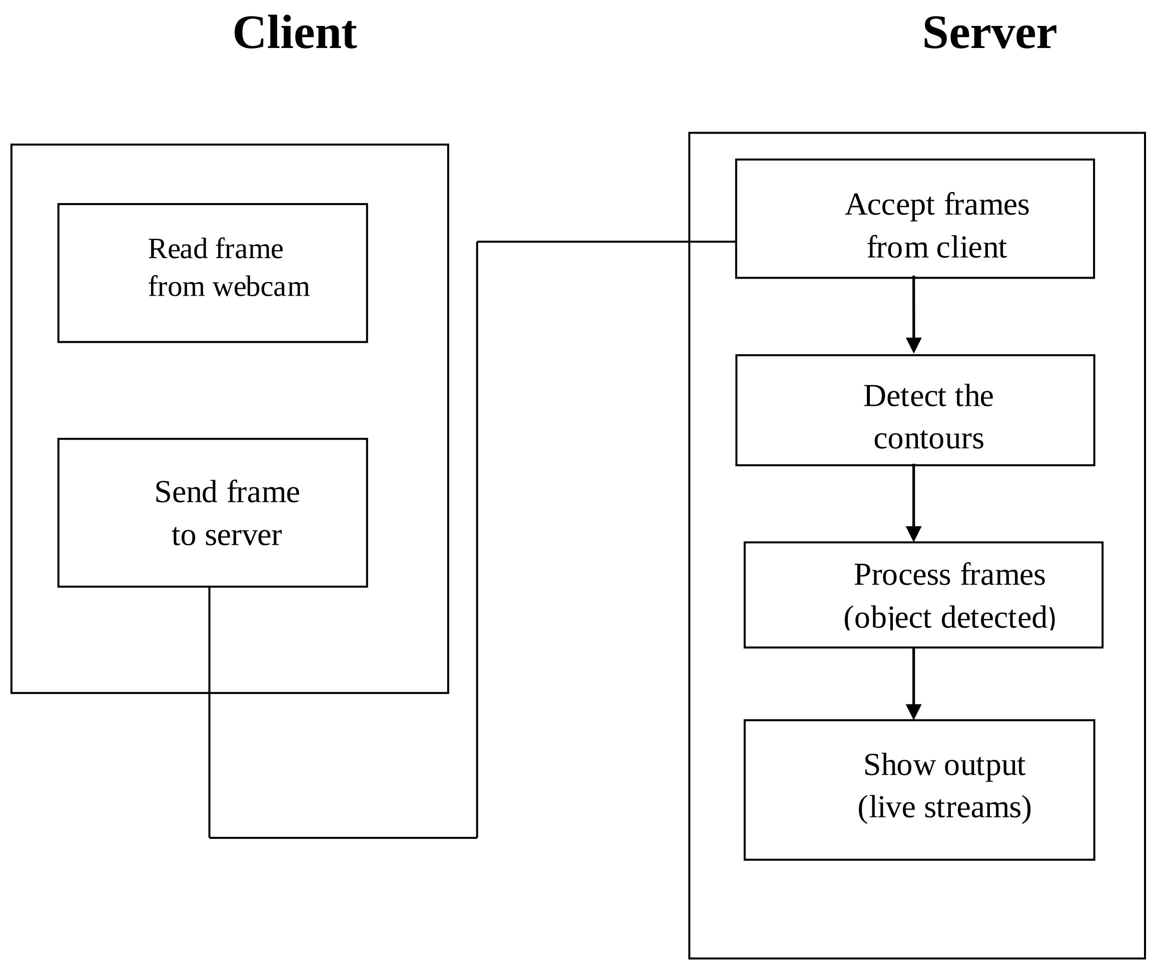

Client
Server
Read frame from webcam
Send frame to server
Accept frames from client
Detect the contours
Process frames (object detected)
Show output (live streams)

Fig 4: architecture of proposed system

2.7. Feasibility Study

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it's worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts in feasibility study.

Technical Feasibility

The project to be developed by using hundred percent technologically used backend language python .group members has enough capability to develop the project. Our focus is to develop well organized air canvas on a whiteboard that is technically efficient and effective for modern day teaching and effective presentations. Therefore, it can be concluding that the system is technically feasible.

Economic Feasibility

The project to be developed is economically feasible and the benefit is outweighing the cost. Since this project already computerizes the existing system and more advanced than the current system reduces and change the labor force to computerize system. Reduces the cost of the material used.

Behavioral Feasibility

The system is free from any legal and contractual risks. The proposed system is free from any harm and there is sufficient support for the user.

Chapter 3
System analysis

3.1 SYSTEM SPECIFICATION

3.1.1 Hardware Requirements:

1. Processor – Intel(R) Core(TM) i5

2. RAM – 8 GB

3. Hard Disk – 1 TB

4. Processor Speed – 2.5GHZ

3.1.2 Software Requirements:

1. Operating System – Microsoft Windows 10

2. Front-End – Python

3. Back-End – Python

4. Tool – VS studio, Browser.

3.1.3 Object requirement:

1. A blue colored pointed pen or marker

3.1.4 Display Mode:

1. Color Quality – Highest [32 bit]

2. Screen Resolution – 1024 by 768 Pixels

Chapter 4
System design

4.1 PROJECT DESIGN

Here Color Detection and tracking is used in order to achieve the objective. The color 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.

STEPS IN DETAIL:

1. Color Tracking of Object. First of all, the incoming image from the webcam is to be converted to the HSV color space for detecting the marker. 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 Track bars to arrange the HSV values to the required range of color of the colored object that we have placed at our fingerbowl, when the track bars are setup, we will get the real-time value from the track bars 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.

3. 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 seven buttons on the canvas, drawn using OpenCV.

- Clear: This clears the screen by emptying the deques.
- Black: Changes the marker to black color using color array.
- Green: Changes the marker to Green color using color array.
- Red: Changes the marker to red color using color array.
- Yellow: Changes the marker to yellow color using color array.
- Pink: Changes the marker to pink color using color array.
- Close: It will close the whiteboard, webcam, mask, track bars.

Or you can press “q” to close all the tabs.

Also, to avoid drawing when contour is not present, we will put else condition which will capture that instant.

Drawing the points now we will draw all the points on the positions stored in the deques, with respective color.

4.2 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 colored 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.

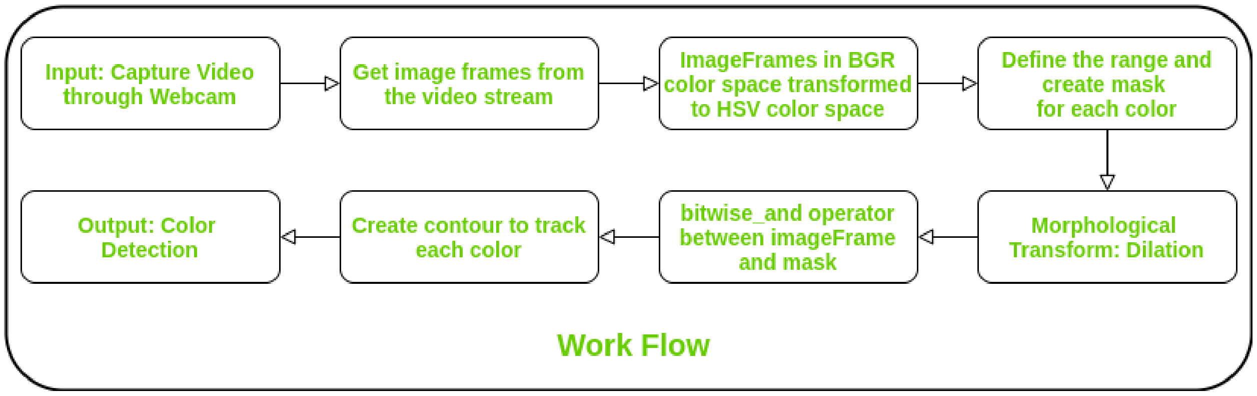

Input: Capture Video through Webcam
Get image frames from the video stream
ImageFrames in BGR color space transformed to HSV color space
Define the range and create mask for each color
Morphological Transform: Dilation
bitwise_and operator between imageFrame and mask
Create contour to track each color
Output: Color Detection
Work Flow

Fig 5: work flow diagram

Chapter 5
Coding

```
import numpy as np
import cv2
from collections import deque

#default called track bar function
def setValues(x):
   print("")


# Creating the trackbars needed for adjusting the marker colour
cv2.namedWindow("Color detectors")
cv2.createTrackbar("Upper Hue", "Color detectors", 153,
180,setValues)
cv2.createTrackbar("Upper Saturation", "Color detectors", 255,
255,setValues)
cv2.createTrackbar("Upper Value", "Color detectors", 255,
255,setValues)
cv2.createTrackbar("Lower Hue", "Color detectors", 64,
180,setValues)
cv2.createTrackbar("Lower Saturation", "Color detectors", 72,
255,setValues)
cv2.createTrackbar("Lower Value", "Color detectors", 49,
255,setValues)



# 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)]
ppoints = [deque(maxlen=1024)]
bpoints = [deque(maxlen=1030)]

# These indexes will be used to mark the points in particular arrays
of specific colour
blue_index = 0
green_index = 0
red_index = 0
```

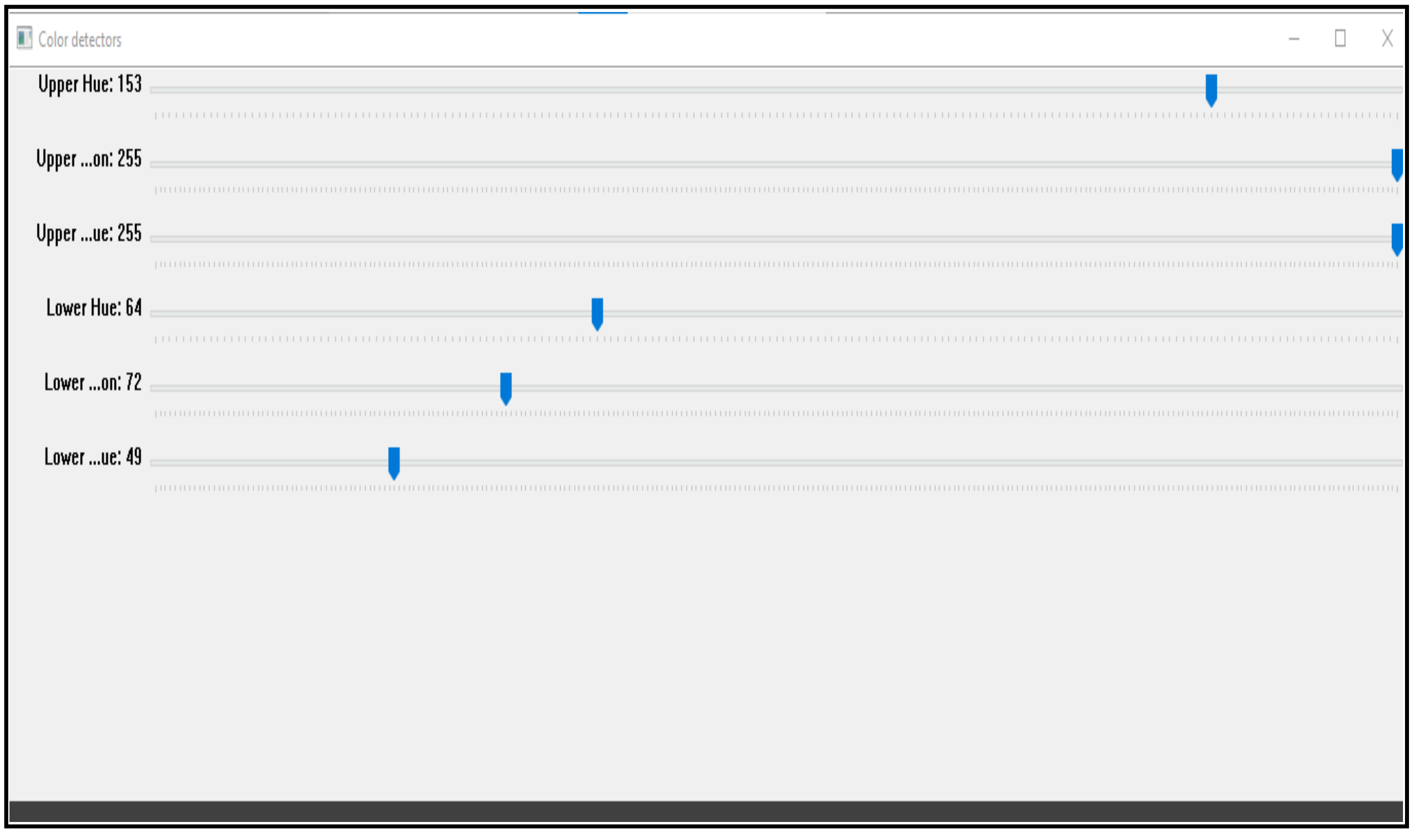
```
yellow_index = 0
pink_index = 0
black_index = 1
#The kernel to be used for dilation purpose
kernel = np.ones((5,5),np.uint8)

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

# Here is code for Canvas setup
paintWindow = np.zeros((640,900,3)) + 255
paintWindow = cv2.rectangle(paintWindow, (40,1), (140,65), (0,0,0), 
2)
paintWindow = cv2.rectangle(paintWindow, (160,1), (255,65), 
colors[0], -1)
paintWindow = cv2.rectangle(paintWindow, (275,1), (370,65), 
colors[1], -1)
paintWindow = cv2.rectangle(paintWindow, (390,1), (485,65), 
colors[2], -1)
paintWindow = cv2.rectangle(paintWindow, (505,1), (600,65), 
colors[3], -1)
paintWindow = cv2.rectangle(paintWindow, (620,1), (715,65), 
colors[4], -1)
paintWindow = cv2.rectangle(paintWindow, (735,1), (830,65), 
colors[5], -1)

cv2.putText(paintWindow, "CLEAR", (49, 33), 
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "BLACK", (185, 33), 
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "GREEN", (298, 33), 
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "RED", (420, 33), cv2.FONT_HERSHEY_SIMPLEX, 
0.5, (255, 255, 255), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "YELLOW", (520, 33), 
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (150,150,150), 2, cv2.LINE_AA)
cv2.putText(paintWindow, "PINK",(650,33),cv2.FONT_HERSHEY_SIMPLEX,
0.5,(0,0,0),2,cv2.LINE_AA)
```

```
cv2.putText(paintWindow,"CLOSE",(762,33),cv2.FONT_HERSHEY_SIMPLEX,
0.5,(255,255,255),2,cv2.LINE_AA)

cv2.namedWindow('Paint', cv2.WINDOW_NORMAL)
cv2.resizeWindow('Paint',700,800)



# Loading the default webcam of PC.
cap = cv2.VideoCapture(0)
def change_res(width,height):
     cap.set(3,width)
     cap.set(4,height)

def rescale_frame(frame,percent=150):
     percent = 150
     width=int(frame.shape[1]*percent/100)
     height=int(frame.shape[0]*percent/100)
     dim=(width,height)
     return cv2.resize(frame,dim,interpolation =cv2.INTER_AREA)
# Keep looping
while True:
    # Reading the frame from the camera
    ret, frame = cap.read()
    frame=rescale_frame(frame,percent=150)
    #Flipping the frame to see same side of yours
    frame = cv2.flip(frame, 1)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    u_hue = cv2.getTrackbarPos("Upper Hue", "Color detectors")
    u_saturation = cv2.getTrackbarPos("Upper Saturation", "Color
detectors")
    u_value = cv2.getTrackbarPos("Upper Value", "Color detectors")
    l_hue = cv2.getTrackbarPos("Lower Hue", "Color detectors")
    l_saturation = cv2.getTrackbarPos("Lower Saturation", "Color
detectors")
    l_value = cv2.getTrackbarPos("Lower Value", "Color detectors")
    Upper_hsv = np.array([u_hue,u_saturation,u_value])
    Lower_hsv = np.array([l_hue,l_saturation,l_value])
    # Adding the colour buttons to the live frame for colour access
    frame = cv2.rectangle(frame, (40,1), (140,65), (122,122,122), -
1)
    frame = cv2.rectangle(frame, (160,1), (255,65), colors[0], -1)
    frame = cv2.rectangle(frame, (275,1), (370,65), colors[1], -1)
    frame = cv2.rectangle(frame, (390,1), (485,65), colors[2], -1)
    frame = cv2.rectangle(frame, (505,1), (600,65), colors[3], -1)
    frame = cv2.rectangle(frame, (620,1), (715,65), colors[4], -1)
    frame = cv2.rectangle(frame, (735,1), (830,65), colors[5], -1)

    cv2.putText(frame, "CLEAR ALL", (49, 33),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
    cv2.putText(frame, "BLACK", (185, 33), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (255, 255, 255), 2, cv2.LINE_AA)
    cv2.putText(frame, "GREEN", (298, 33), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (255, 255, 255), 2, cv2.LINE_AA)
    cv2.putText(frame, "RED", (420, 33), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (255, 255, 255), 2, cv2.LINE_AA)
    cv2.putText(frame, "YELLOW", (520, 33),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (150,150,150), 2, cv2.LINE_AA)
    cv2.putText(frame, "PINK",(650,33),cv2.FONT_HERSHEY_SIMPLEX,
0.5,(255,255,255),2,cv2.LINE_AA)
    cv2.putText(frame,"CLOSE",(762,33),cv2.FONT_HERSHEY_SIMPLEX,
0.5,(255,255,255),2,cv2.LINE_AA)

    # Identifying the pointer by making its mask
    Mask = cv2.inRange(hsv, Lower_hsv, Upper_hsv)
    Mask = cv2.erode(Mask, kernel, iterations=1)
    Mask = cv2.morphologyEx(Mask, cv2.MORPH_OPEN, kernel)
    Mask = cv2.dilate(Mask, kernel, iterations=1)

    # Find contours for the pointer after idetifying it
    cnts,_ = cv2.findContours(Mask.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
    center = None

    # Ifthe contours are formed
    if len(cnts) > 0:
        # sorting the contours to find biggest
        cnt = sorted(cnts, key = cv2.contourArea, reverse = True)[0]
        # Get the radius of the enclosing circle around the found
contour
        ((x, y), radius) = cv2.minEnclosingCircle(cnt)
        # Draw the circle around the contour
        cv2.circle(frame, (int(x), int(y)), int(radius), (0, 255,
255), 2)
        # Calculating the center of the detected contour
        M = cv2.moments(cnt)
        center = (int(M['m10'] / M['m00']), int(M['m01'] /
M['m00']))

        # Now checking if the user wants to click on any button
above the screen
        if 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)]
                ppoints = [deque(maxlen=512)]
                bpoints = [deque(maxlen=520)]
                blue_index = 0
                green_index = 0
                red_index = 0
                yellow_index = 0
                pink_index = 0
                black_index = 1

                paintWindow[67:,:,:] = 255
            elif 140 <= center[0] <= 255:
                    colorIndex = 0 # black
            elif 275 <= center[0] <= 370:
                    colorIndex = 1 # Green
            elif 390 <= center[0] <= 485:
                    colorIndex = 2 # Red
            elif 505 <= center[0] <= 600:
                    colorIndex = 3 # Yellow
            elif 620 <= center[0] <=715:
                 colorIndex = 4 # pink
            elif 735 <= center[0] <=830:
                  colorIndex = 5 #close
        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)
            elif colorIndex ==4:
                 ppoints[pink_index].appendleft(center)
            elif colorIndex == 5:
                 bpoints[black_index].appendleft(center)
    # Append the next deques when nothing is detected to avoid
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
        ppoints.append(deque(maxlen=512))
        pink_index +=1
        bpoints.append(deque(maxlen=520))
        black_index += 2

    # Draw lines of all the colors on the canvas and frame
    points = [bpoints, gpoints, rpoints, ypoints, ppoints,bpoints]
    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)

    # Show all the windows

    cv2.imshow("Tracking", frame)
    cv2.imshow("Paint", paintWindow)
    cv2.imshow("mask",Mask)


    # If the 'q' key is pressed then stop the application
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

# Release the camera and all resources
cap.release()
cv2.destroyAllWindows()
```

Chapter 6
User Interface

- **Color detector track bars**



Color detectors

Upper Hue: 153

Upper ...on: 255

Upper ...ue: 255

Lower Hue: 64

Lower ...on: 72

Lower ...ue: 49

Fig 6: color detector track bars

- **Mask**

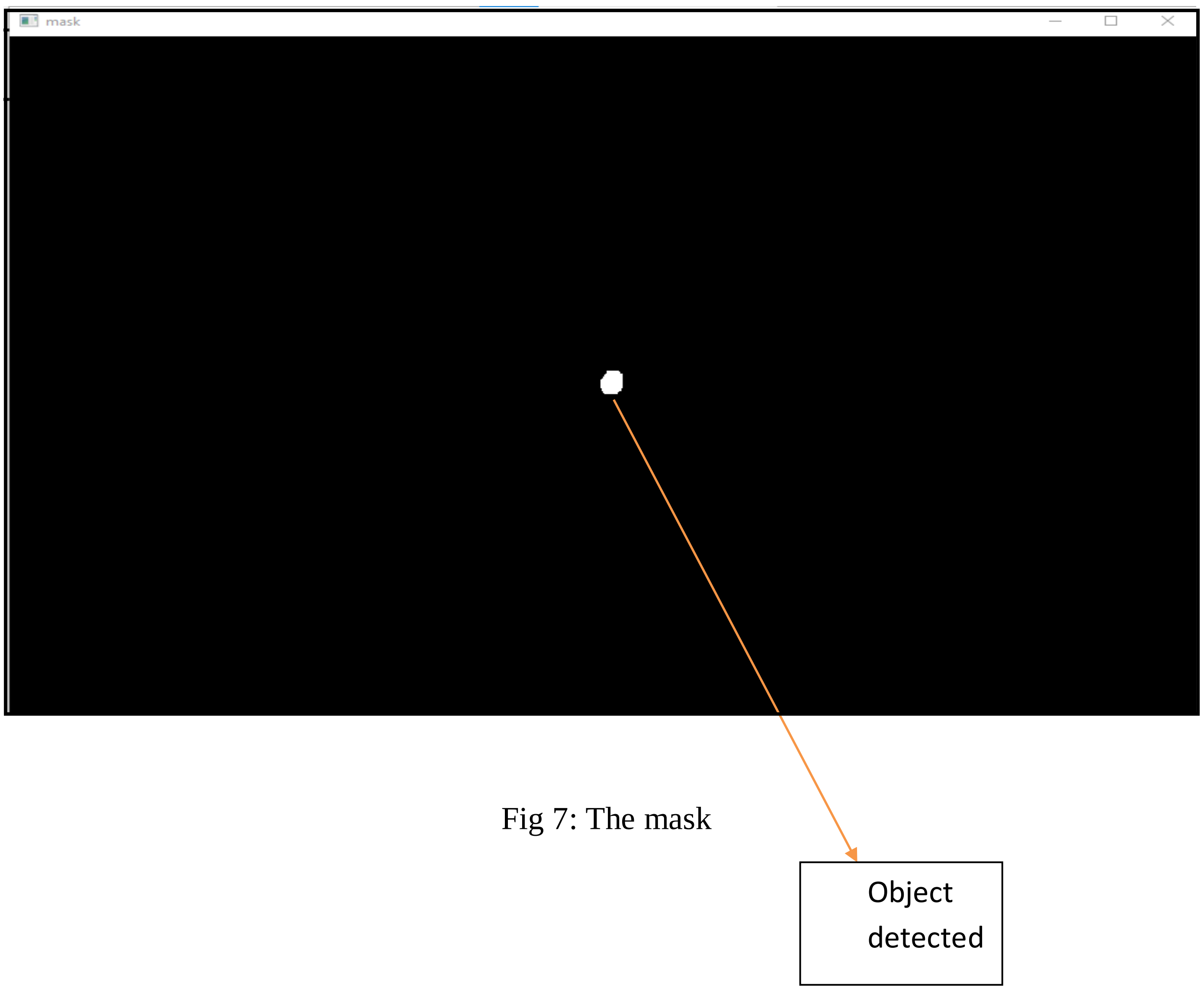


Fig 7: The mask

- **Tracking by webcam**

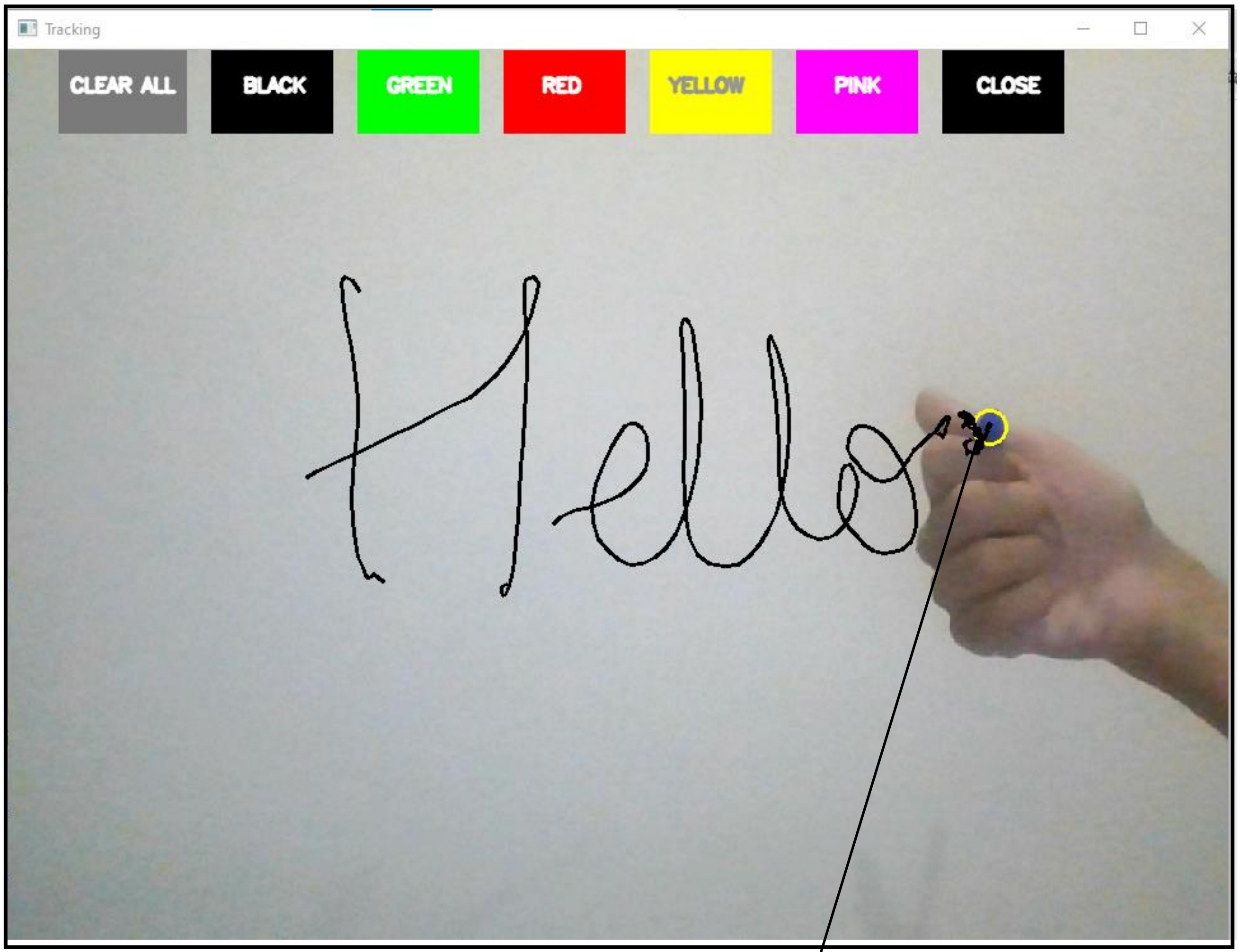


Fig 8: tracking by webcam

Blue coloured object detected by webcam by identifying it as a contour and surrounding it by yellow circle.

- **The whiteboard**

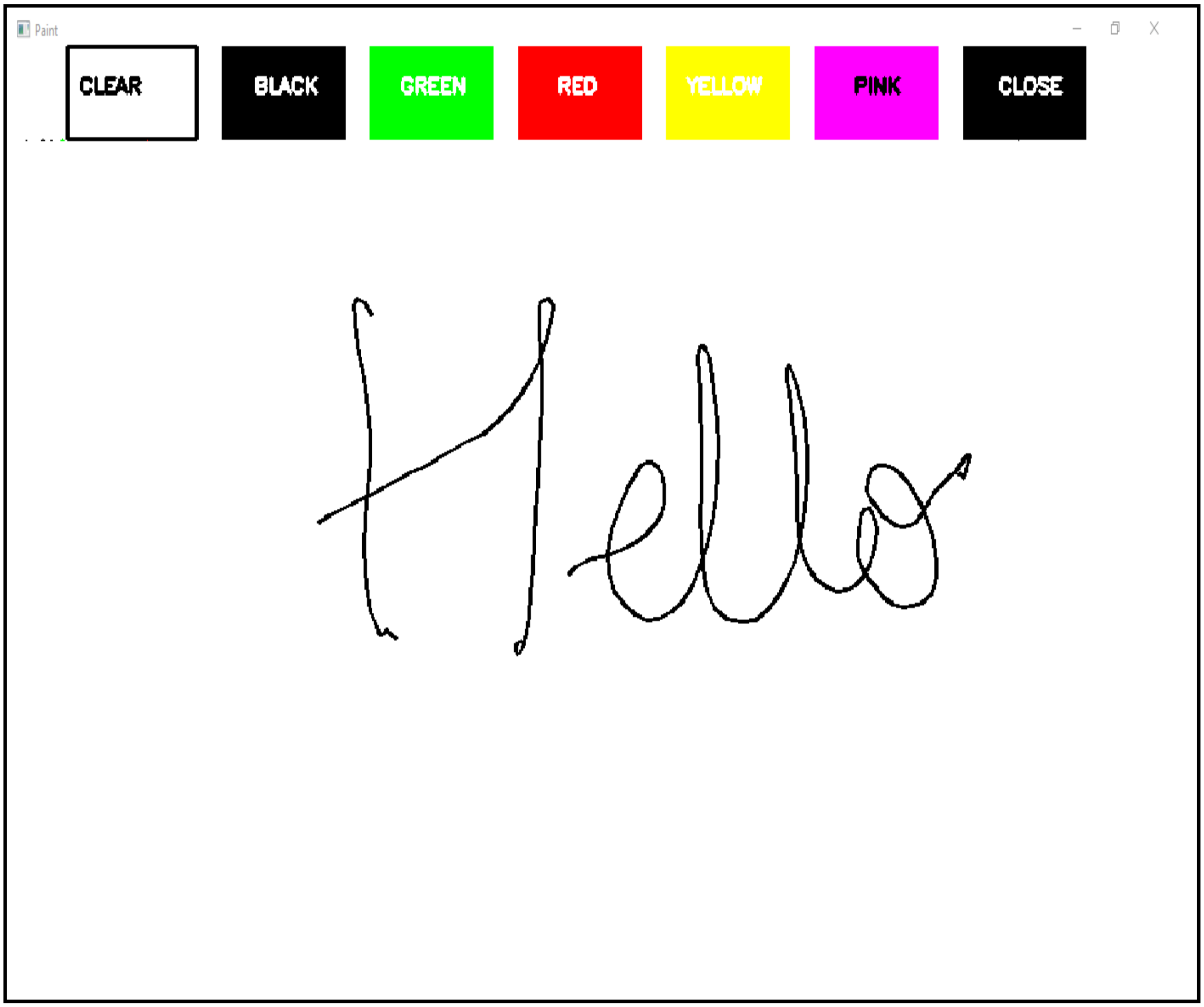


Fig 9: whiteboard

- **Using clear button to clear the window**

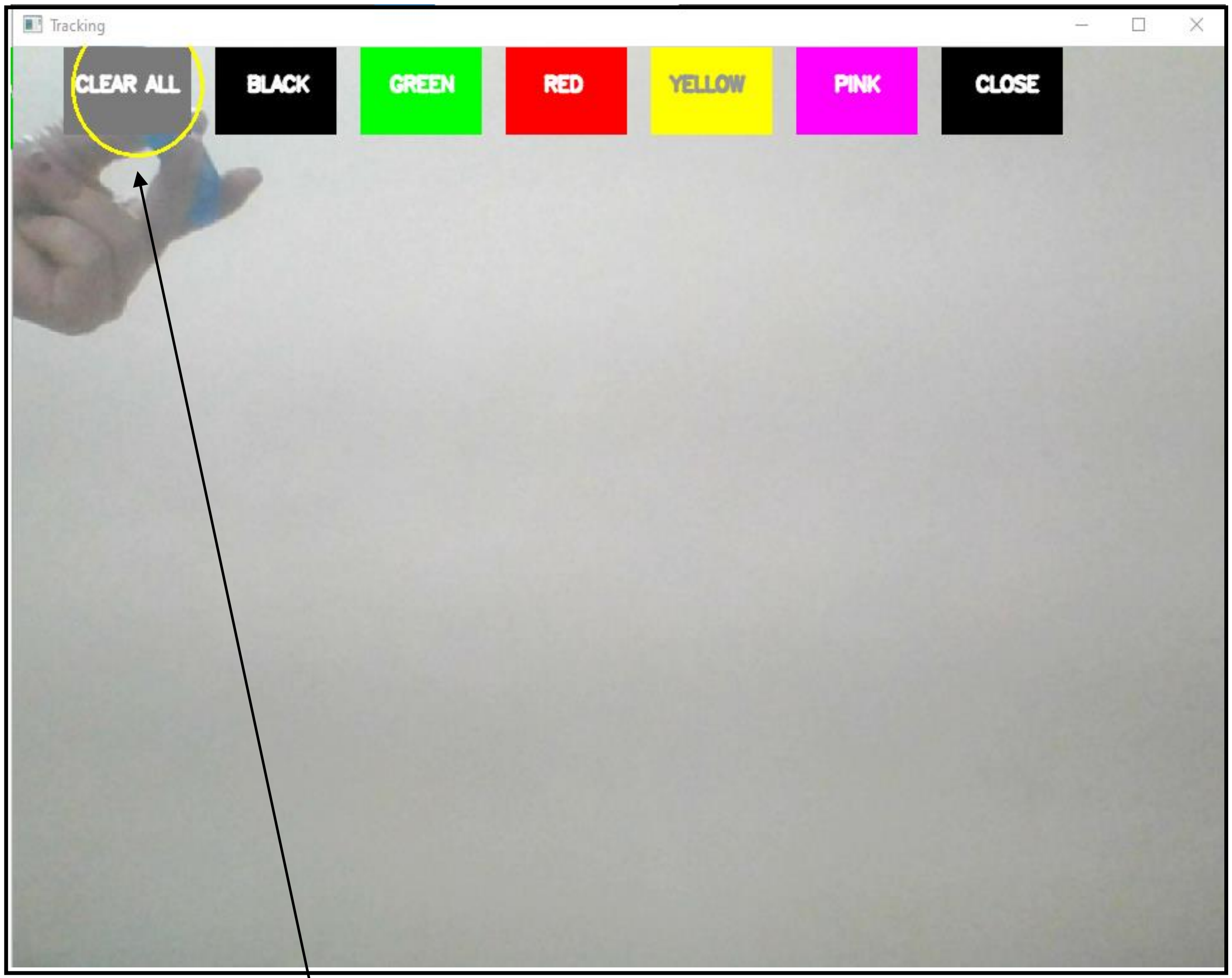


Fig 10: using clear button

By hovering over CLEAR ALL Tab user can clear the whole screen on White Board

- **The Whiteboard gets cleared**

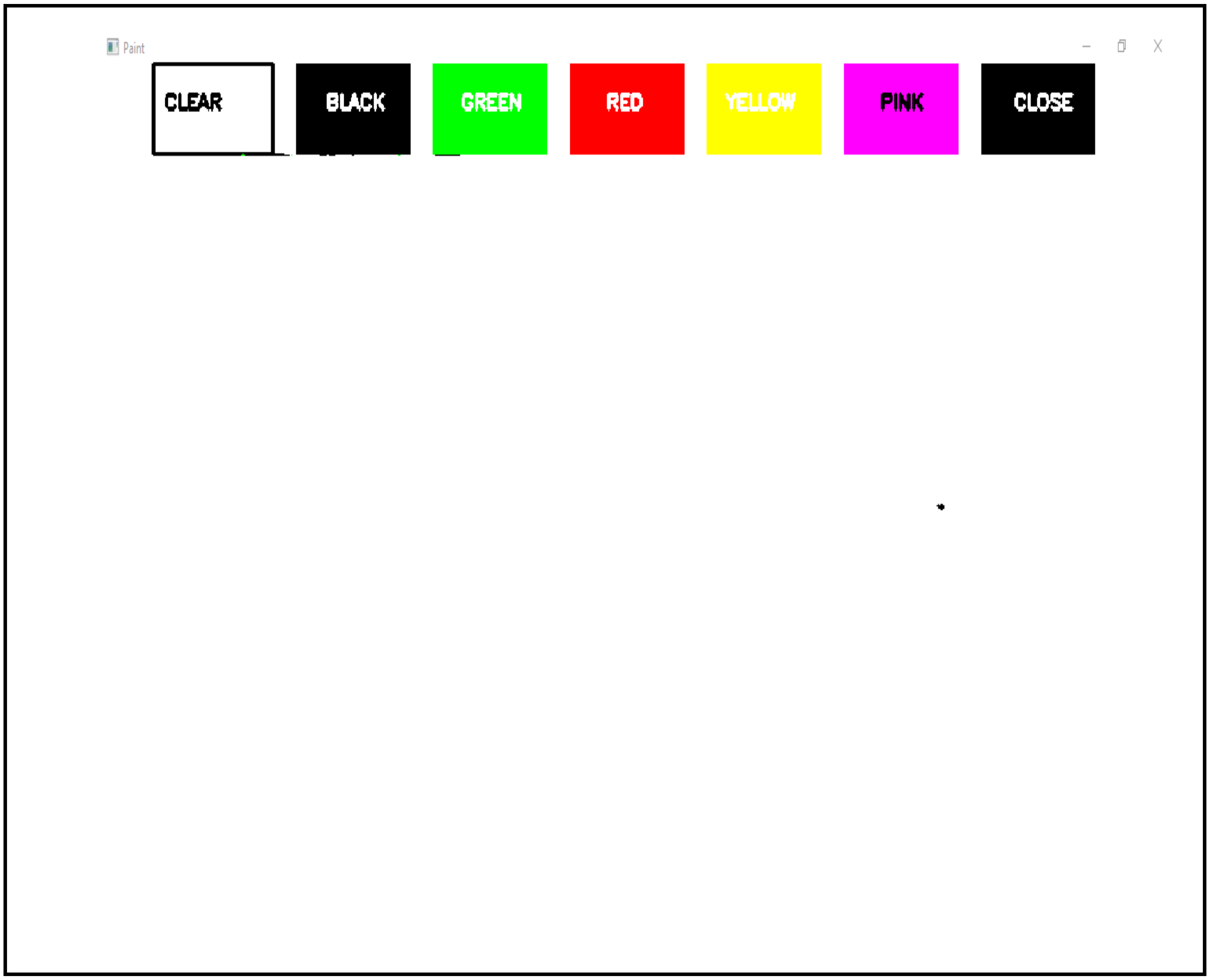


Fig 11: whiteboard cleared

- **Using different colors from color palette**.

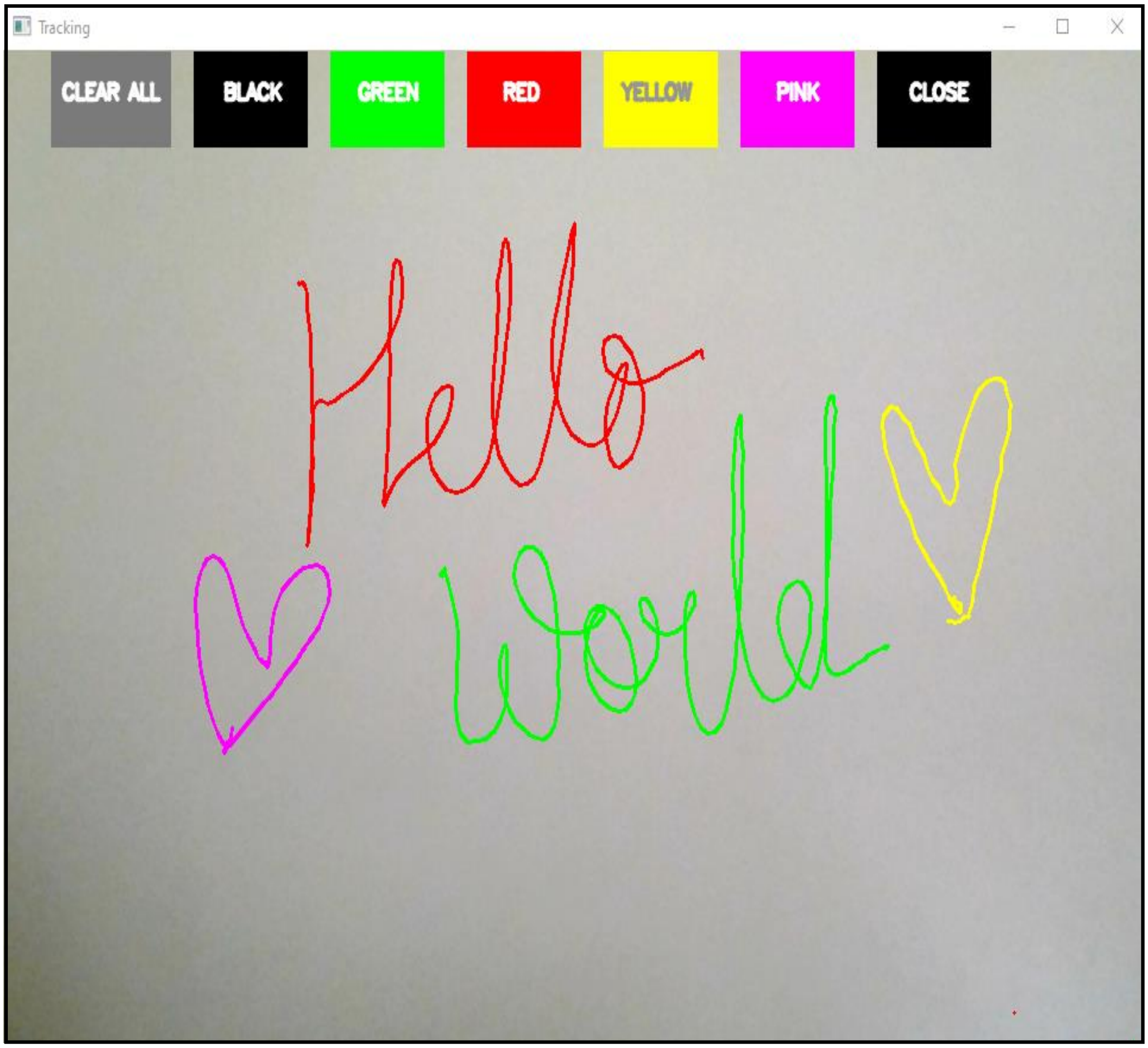


Fig 12 : using different colors

- **Different colors on whiteboard**

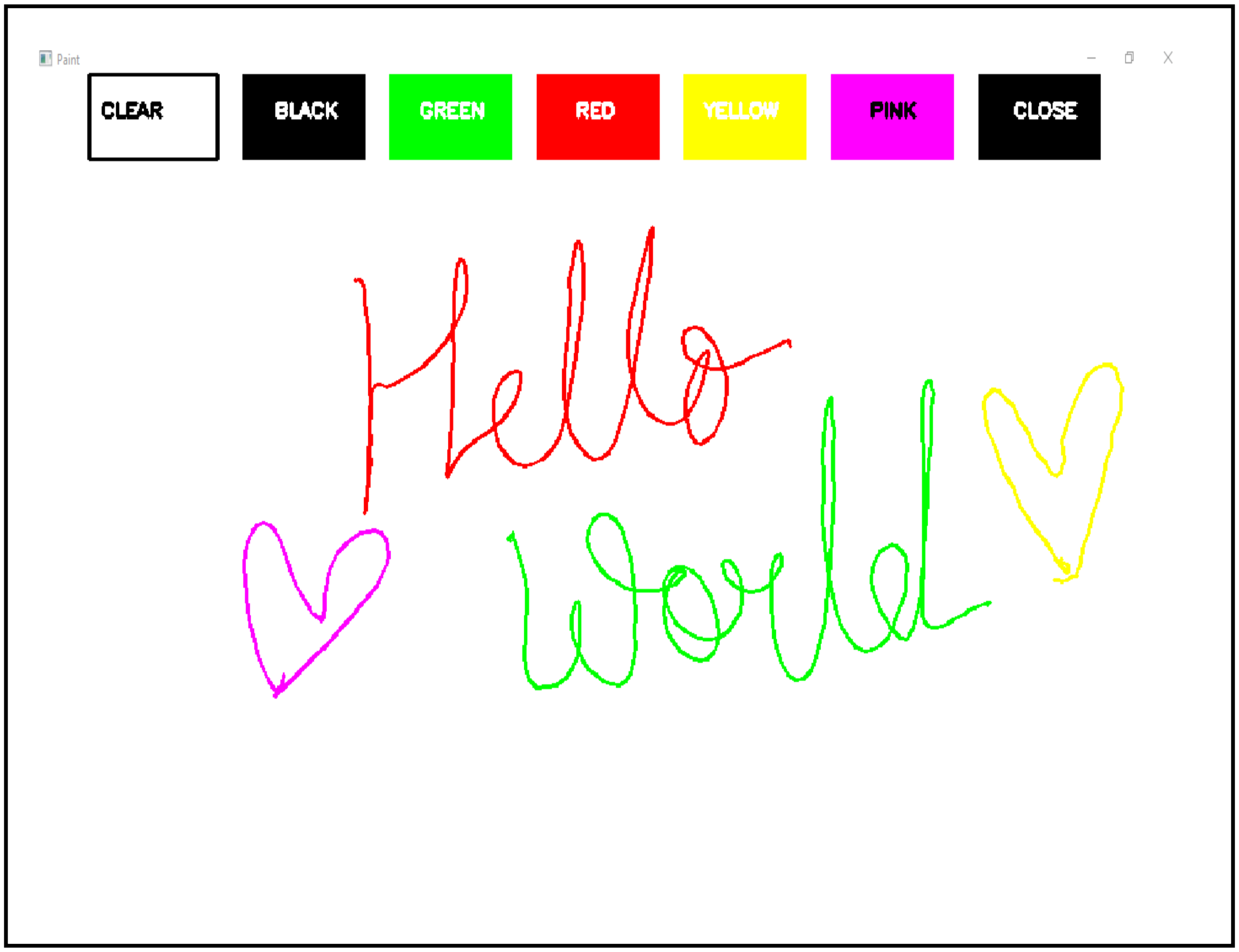


Fig 13 : different colours on whiteboard

- **Choosing the close button to close whole program.**

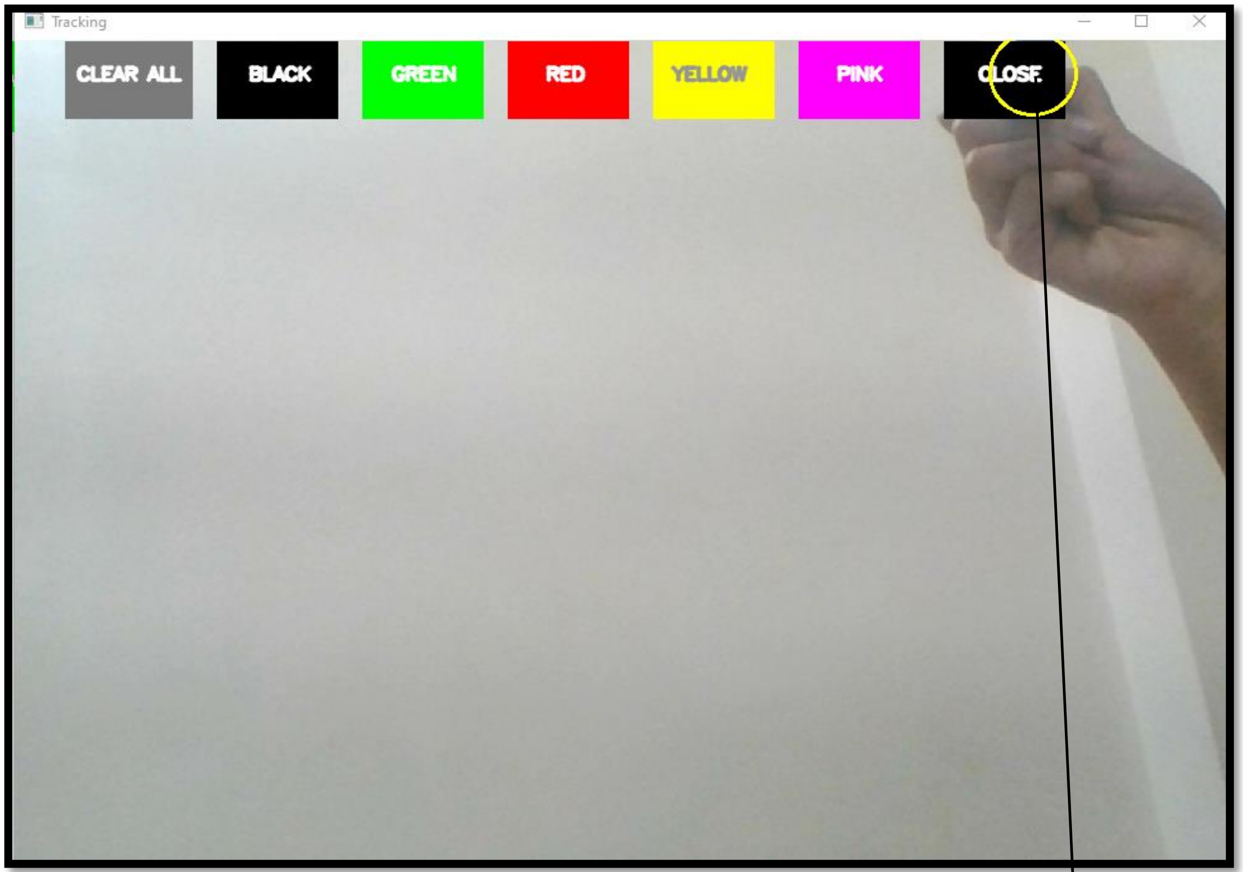


Fig 14: closing the window

By hovering over CLOSE Tab user can close the whole application without pressing any key

Chapter 7
Implementation and Testing

7. IMPLEMENTATION AND TESTING

7.1 Implementation:

Implementation is the stage where the theoretical design is turned into a working project. The most crucial stage in achieving a new successful system and in giving
We have Confidence on our new system for the users that it will work efficiently and effectively. The system will be implemented after thorough testing is done and if it will work according to the specification.
We have done a careful planning, investigation of the current code and its constraints on implementation, design of methods to achieve the changeover and an evaluation of change over methods a part from planning. The major task will be preparing the implementation is testing of the system and debugging it than executing the whole code.

We will proceed with testing of our code with black box testing
We will compare and contrast the expected output and the actual output to find out how is our project responding to the users

The main purpose here is to create the complete source code, including the design assumptions of the project. That’s why at this stage, we worked hard on writing the source code, compiling, etc. The main goal for us at this stage is to give valuable feedback on what’s working (or not) in the system in terms of design. Also, we focused on detecting if the system functions well after introducing changes by performing unit tests.

7.2 Testing:

Software testing is a critical elements quality assurance and testing of our built project.

In our project we have done black box testing as the user need not to know the whole code:

- Black box testing

Input	Expected output	Actual output	result
User holding blue colored pen and waving it in webcam	Drawing on the whiteboard should appear same as the user wave the pen	Drawing on whiteboard appeared as the user waved his hand	✓
User holding a non blue colored pen and waving it in webcam	Drawing on the whiteboard should not appear as it detects only blue color.	Drawing on whiteboard did not appeared as the user waved his hand	✓
User holding two blue colored pen and waving it in webcam	Drawing on the whiteboard should appear same as the user wave both the pens.	Drawing on whiteboard did not appeared as the user waved his hand because it detected 2 blue object and thus output was not precise	✗
User holding blue colored pen having a blue colored background and waving it in webcam	Drawing on the whiteboard should appear same as the user wave the pen	Drawing on whiteboard did not appeared as the user waved his hand because it detected blue object in the background and thus output was not precise	✗

References:

1.GeeksforGeeks: https://www.geeksforgeeks.org/

2.Stack overflow: https://stackoverflow.com/

3.Tutorials point: https://www.tutorialspoint.com/

4.Mygreatlearning: https://www.mygreatlearning.com/blog/opencv-tutorial-in-

5.Github :https://github.com/

6.Google: https://www.google.com/