# Air Canvas Drawing using OpenCV

Author: Deep Rudra, B.Tech in Electronics and Communication Engineering
Duration: Feb 2025 – Mar 2025

## Abstract

This project presents a real-time virtual drawing application using computer vision techniques. The system allows users to draw in the air by moving a colored object, which is tracked through the webcam feed using OpenCV. The drawn path is simultaneously visualized on a digital canvas.

## Objective

The primary objective is to demonstrate human-computer interaction through hand gesture tracking and real-time image processing. It serves as a basic prototype for interactive AR applications and gesture-controlled interfaces.

## Technologies Used

- **Programming Language:** Python
- **Libraries:** OpenCV, NumPy, Collections (Deque)
- **Computer Vision Concepts:** HSV Color Detection, Morphological Filtering, Contour Tracking

## System Architecture

The system captures live webcam frames, processes each frame to isolate the colored marker, identifies its contour, and tracks its movement by appending positions to a deque. A blank canvas is overlaid with lines that follow the marker path, simulating air drawing.

## Implementation

- **Marker Detection:** Converts frames to HSV color space for robust color filtering under varying lighting.
- **Masking and Morphological Ops:** Noise is reduced using erosion and dilation.
- **Drawing Engine:** Maintains separate deques for each color. Draws lines between consecutive points.
- **Interface:** Interactive buttons for switching colors and clearing canvas.

## Key Features

1. Real-time tracking and drawing experience.
2. Multiple color support with dynamic switching.
3. Adjustable HSV thresholds using trackbars.
4. Simple, intuitive GUI layout.

## Conclusion

The Air Canvas project integrates computer vision with intuitive design to create a seamless drawing experience without physical contact. It is highly extendable to gesture recognition, virtual whiteboards, or interactive learning environments.

## Appendix: Full Source Code

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

\#default called trackbar 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)]

\# 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
```

```
\# Canvas setup
paintWindow = np.zeros((471, 636, 3), dtype=np.uint8) + 255
button\_centers = [(90, 33), (207, 33), (322, 33), (437, 33), (552, 33)]
button\_radius = 30
paintWindow = cv2.circle(paintWindow, button\_centers[0], button\_radius, (0, 0, 0), 2)
paintWindow = cv2.circle(paintWindow, button\_centers[1], button\_radius, colors[0], -1)
paintWindow = cv2.circle(paintWindow, button\_centers[2], button\_radius, colors[1], -1)
paintWindow = cv2.circle(paintWindow, button\_centers[3], button\_radius, colors[2], -1)
paintWindow = cv2.circle(paintWindow, button\_centers[4], button\_radius, colors[3], -1)
cv2.putText(paintWindow, "CLEAR", (65, 38), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 0),
cv2.putText(paintWindow, "BLUE", (185, 38), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255,
, 2)
cv2.putText(paintWindow, "GREEN", (300, 38), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255,
), 2)
cv2.putText(paintWindow, "RED", (415, 38), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 2
 2)
cv2.putText(paintWindow, "YELLOW", (505, 38), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (150, 150
0), 2)

\# Webcam and drawing logic continues...
```