SOURCE CODE

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

# Initialize MediaPipe Hands
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(max_num_hands=1, min_detection_confidence=0.7)
mp_draw = mp.solutions.drawing_utils

# Default callback function for trackbars
def setValues(x):
   pass

# Creating trackbars
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)

# Deques for storing points
bpoints = [deque(maxlen=1024)]
```

```
gpoints = [deque(maxlen=1024)]
rpoints  =  [deque(maxlen=1024)]
ypoints  =  [deque(maxlen=1024)]
# Indexes for colors
blue_index = 0
green_index = 0
red_index = 0
yellow_index = 0
# Kernel for morphology operations
kernel = np.ones((5, 5), np.uint8)
# Colors for drawing
colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (0, 255, 255)]
colorIndex = 0
# Setup canvas for drawing
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), 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)
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,
(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.namedWindow('Paint', cv2.WINDOW_AUTOSIZE)

# Capture from webcam
```

```
cap = cv2.VideoCapture(0)

while True:
    # Read frame
    ret, frame = cap.read()
    frame = cv2.flip(frame, 1)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Get HSV values from trackbars
    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])

    # Add buttons to frame
    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)
    cv2.putText(frame, "CLEAR ALL", (49, 33), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
 (255, 255, 255), 2, cv2.LINE_AA)
    cv2.putText(frame, "BLUE", (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)

    # Detect hand landmarks using MediaPipe
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = hands.process(rgb_frame)

    if results.multi_hand_landmarks:
        for hand_landmarks in results.multi_hand_landmarks:
            # Draw hand landmarks
            mp_draw.draw_landmarks(frame, hand_landmarks, mp_hands.
HAND_CONNECTIONS)
  # Get the index finger tip landmark
            index_tip = hand_landmarks.landmark[mp_hands.HandLandmark.
INDEX_FINGER_TIP]
            h, w, c = frame.shape
            cx, cy = int(index_tip.x * w), int(index_tip.y * h)
            center = (cx, cy)
# Handle button presses based on finger position
            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)]

                    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:
            # Draw on canvas based on selected color
            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)
# Draw lines on frame and canvas
    points = [bpoints, gpoints, rpoints, ypoints]
    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)
    # Display frames
    cv2.imshow("Tracking", frame)
    cv2.imshow("Paint", paintWindow)
  # Exit on 'q' key press
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
  # Clean up
  cap.release()
  cv2.destroyAllWindows()
```