```
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()
```