Air Canvas – Hand Gesture–Based Virtual Drawing System

AirCanvas is a computer vision–based drawing application thatallows usersto draw onavirtualcanvas using hand gestures insteadof traditionalinputdevices. Usinga webcam, the system tracks handmovements inreal time and converts finger gestures into drawing actions.

The project is developed using Python, OpenCV, MediaPipe Hands, and NumPy, enabling accurate hand detection, smooth drawing, color selection, erasing, and canvas clearing through simple gestures. This application demonstrates an intuitive approach to touchless human–computer interaction.

In [ ]:

```
import warnings
warnings.filterwarnings("ignore", category=UserWarning)
import cv2
import mediapipe as mp
import numpy as np
# ------------------------------
# CAMERA
# ------------------------------
cap = cv2.VideoCapture(0)
# ------------------------------
# MEDIAPIPE
# ------------------------------
mp_hands = mp.solutions.hands
mp_draw = mp.solutions.drawing_utils
hands = mp_hands.Hands(max_num_hands=1, min_detection_confidence=0.7)
# ------------------------------
# CANVAS STATE
# ------------------------------




canvas = None
prev_x, prev_y = 0, 0
# ------------------------------
# COLORS & NAMES (BGR)
# ------------------------------
COLORS = [

    (255,  0,  0),  (0,# Blue  #
    255,  0),  (0,  0,Green
    255),  (255,  255,#Red    #
    0),    (255,    0,Cyan    #
    255),   (0,   255,Pink    #
    255),  (255,  255,Yellow  #
    255),             White   #
    (0,  0,  0)       Eraser
]
 COLOR_NAMES = {

    (255,  0,  0): "Blue",  (0,
    255,  0): "Green",  (0,  0,
    255):  "Red",  (255,  255,
    0):   "Cyan",   (255,   0,
    255):  "Pink",   (0,   255,
    255):   "Yellow",   (255,
    255,  255): "White",  (0,
    0,  0): "Eraser"
} current_color = (255, 0, 255)
BRUSH_THICKNESS = 8
#  ____________________________#_ Pink default

_
```

```python
# HELPER FUNCTION
# ------------------------------
def fingers(hand):
    index_up = hand.landmark[8].y < hand.landmark[6].y
    middle_up = hand.landmark[12].y < hand.landmark[10].y
    return index_up, middle_up

# ------------------------------
# MAIN LOOP
# ------------------------------
while True:
    success, frame = cap.read()
    if not success:
        break

    frame = cv2.flip(frame, 1)
    h, w, _ = frame.shape
    if canvas is None:
        canvas = np.zeros((h, w, 3), dtype=np.uint8)

    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    result = hands.process(rgb)

    # ---------- TOP COLOR BAR ----------
    bar_w = (w - 140) // len(COLORS)
    for i, col in enumerate(COLORS):
        cv2.rectangle(frame, (i * bar_w, 0),
                      ((i + 1) * bar_w, 70), col, -1)
        if col == current_color:
            cv2.rectangle(frame, (i * bar_w, 0),
                          ((i + 1) * bar_w, 70), (255, 255, 255), 3)

    # ---------- CLEAR BUTTON ----------
    clear_x1 = w - 140
    clear_x2 = w
    cv2.rectangle(frame, (clear_x1, 0), (clear_x2, 70), (220, 220, 220), -1)
    cv2.putText(frame, "CLEAR",
                (clear_x1 + 30, 45),
                cv2.FONT_HERSHEY_SIMPLEX,
                0.8, (0, 0, 0), 2)

    if result.multi_hand_landmarks:
        hand = result.multi_hand_landmarks[0]
        mp_draw.draw_landmarks(frame, hand, mp_hands.HAND_CONNECTIONS)

        index_up, middle_up = fingers(hand)
        x = int(hand.landmark[8].x * w)
        y = int(hand.landmark[8].y * h)

        # CHANGE COLOR
        if index_up and middle_up and y < 70 and x < clear_x1:
            idx = x // bar_w
            if idx < len(COLORS):
                current_color = COLORS[idx]
            prev_x = 0

        # CLEAR SCREEN
        elif index_up and middle_up and y < 70 and x >= clear_x1:
            canvas = np.zeros((h, w, 3), dtype=np.uint8)
            prev_x = 0

        # DRAW
        elif index_up and not middle_up:
            if prev_x == 0:
                prev_x, prev_y = x, y
            cv2.line(canvas, (prev_x, prev_y), (x, y),
                     current_color, BRUSH_THICKNESS)
            prev_x, prev_y = x, y
            cv2.circle(frame, (x, y), BRUSH_THICKNESS, current_color, -1)
        else:
            prev_x = 0

    # ---------- MERGE CANVAS ----------
    gray = cv2.cvtColor(canvas, cv2.COLOR_BGR2GRAY)
    _, inv = cv2.threshold(gray, 20, 255, cv2.THRESH_BINARY_INV)
    inv = cv2.cvtColor(inv, cv2.COLOR_GRAY2BGR)
    frame = cv2.bitwise_and(frame, inv)
    frame = cv2.bitwise_or(frame, canvas)

    # ---------- SINGLE INSTRUCTION LINE ----------
    cv2.rectangle(frame, (0, h-40), (w, h), (235, 235, 235), -1)
    instruction_text = (

        f"{COLOR_NAMES[current_color]} | "
        "index finger = Draw | "
        "index + middle finger = Change color "
    )
    cv2.putText(frame, instruction_text,

                (10, h-12),
                cv2.FONT_HERSHEY_SIMPLEX,
                0.55, (0, 0, 0), 2)

    cv2.imshow("Air Canvas", frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

cap.release()
cv2.destroyAllWindows()
```

Conclusion

The AirCanvas project successfully implements a gesture-controlled virtual drawing system using real-time hand tracking. It provides a natural and user-friendly alternative to mouse-based interaction while showcasing the practical use of computer vision techniques.

This project serves as a strong foundation for future enhancements such as shape recognition, saving drawings, multi-hand support, and AR/VR integration, making it a valuable addition to a technical portfolio.

```
In [ ]:
                                                    PREM AKKATANGERHAL
```