import cv2

import mediapipe as mp

import time

mp\_hands = mp.solutions.hands

mp\_draw = mp.solutions.drawing\_utils

# Finger tip landmark indices in MediaPipe

TIP\_IDS = [4, 8, 12, 16, 20] # thumb, index, middle, ring, pinky

def fingers\_up(hand\_landmarks, handedness\_str):

"""

Determine which fingers are up.

Returns a list of 5 integers 0/1 for [thumb, index, middle, ring, pinky].

hand\_landmarks: list of 21 landmarks with x,y,z normalized coords

handedness\_str: 'Left' or 'Right' as reported by MediaPipe

"""

fingers = [0, 0, 0, 0, 0]

# Convert to simple lists for easier access

lm = [(lmk.x, lmk.y, lmk.z) for lmk in hand\_landmarks.landmark]

# Thumb: logic depends on hand side. Compare tip x with IP (landmark 3) or CMC (1)

# For right hand, thumb tip x > thumb ip x indicates thumb is open (to the right on image)

# For left hand, thumb tip x < thumb ip x indicates open (to the left).

# Use landmark 3 (thumb\_ip) vs 4 (thumb\_tip).

if handedness\_str == "Right":

fingers[0] = 1 if lm[TIP\_IDS[0]][0] > lm[3][0] else 0

else:

fingers[0] = 1 if lm[TIP\_IDS[0]][0] < lm[3][0] else 0

# Other fingers: compare tip y to pip y (tip higher on image -> smaller y)

for i, tip\_id in enumerate(TIP\_IDS[1:], start=1):

# pip id = tip\_id - 2 (approx for index/middle/ring/pinky)

pip\_id = tip\_id - 2

fingers[i] = 1 if lm[tip\_id][1] < lm[pip\_id][1] else 0

return fingers

def pattern\_from\_fingers(fingers):

"""Return pattern string (e.g., '01100') and total fingers up."""

pattern = "".join(str(int(b)) for b in fingers)

total = sum(fingers)

return pattern, total

def pretty\_print\_output(gesture\_name, emoji, pattern, total):

"""Print the stylized output exactly like sample."""

print("PS C:\\Users\\ak>\n")

print("GESTURE: " + gesture\_name + "\n")

print("EMOJI:" + emoji + "\n")

print("PATTERN:\n" + pattern + "\n")

print("TOTAL: " + str(total) + " fingers\n")

def run\_detection(use\_webcam=True, simulate\_once=False, image\_path=None):

"""

Run real-time detection (webcam) or single-image detection.

- use\_webcam: True to use webcam (default). If False and image\_path provided, processes that image.

- simulate\_once: if True and no hand found, prints simulated peace output (useful for testing).

"""

hands = mp\_hands.Hands(static\_image\_mode=not use\_webcam,

max\_num\_hands=1,

min\_detection\_confidence=0.6,

min\_tracking\_confidence=0.5)

cap = None

if use\_webcam:

cap = cv2.VideoCapture(0)

if not cap.isOpened():

print("Webcam not found — switching to image mode (or simulation).")

use\_webcam = False

printed\_once = False

while True:

if use\_webcam:

ret, frame = cap.read()

if not ret:

print("Failed camera read — exiting.")

break

img = cv2.flip(frame, 1) # mirror for natural interaction

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

results = hands.process(img\_rgb)

else:

if image\_path:

img = cv2.imread(image\_path)

if img is None:

print("Image not found at", image\_path)

break

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

results = hands.process(img\_rgb)

else:

# no webcam and no image: simulate once if requested

results = None

if results and results.multi\_hand\_landmarks:

# Use first detected hand and handedness

hand\_landmarks = results.multi\_hand\_landmarks[0]

# handedness: results.multi\_handedness exists only when using Hands with detection

handedness\_str = "Right"

try:

handedness\_str = results.multi\_handedness[0].classification[0].label

except Exception:

# default to Right if not available

handedness\_str = "Right"

fingers = fingers\_up(hand\_landmarks, handedness\_str)

pattern, total = pattern\_from\_fingers(fingers)

# Decide gesture name from pattern: Peace = index & middle up (01100)

gesture\_name = "UNKNOWN"

emoji = ""

if pattern == "01100":

gesture\_name = "PEACE"

emoji = "✌️"

elif pattern == "01000":

gesture\_name = "POINT"

emoji = "☝️"

elif pattern == "11111":

gesture\_name = "FIVE"

emoji = "🖐️"

# you can expand mapping here...

# Print to console (match requested format). Only print once to avoid spam.

pretty\_print\_output(gesture\_name, emoji, pattern, total)

printed\_once = True

# Draw landmarks on frame for visualization

mp\_draw.draw\_landmarks(img, hand\_landmarks, mp\_hands.HAND\_CONNECTIONS)

else:

# No hand detected

if simulate\_once and not printed\_once:

# Simulate the peace output exactly as the example so user sees sample output

pretty\_print\_output("PEACE", "✌️", "01100", 2)

printed\_once = True

# Show window if webcam or image mode

if use\_webcam or image\_path:

cv2.imshow("Hand Gesture Detection - press q to quit", img)

k = cv2.waitKey(1 if use\_webcam else 0) & 0xFF

if k == ord('q'):

break

else:

# Not showing anything (simulation). Break after one iteration.

break

# If image mode (single image), exit after one display

if not use\_webcam and image\_path:

break

if cap:

cap.release()

cv2.destroyAllWindows()

hands.close()

if \_\_name\_\_ == "\_\_main\_\_":

# CHANGE these parameters as needed:

# If you want webcam detection set use\_webcam=True (default).

# If you want to test without webcam call with use\_webcam=False and simulate\_once=True

# You can also provide image\_path to process a single photo: image\_path="path/to/photo.jpg"

run\_detection(use\_webcam=True, simulate\_once=False, image\_path=None)