import cv2

from cvlearn import FaceMesh

from cvlearn.Utils import findDistance, find\_rotation

import numpy as np

# Function to draw an arc (partial ellipse)

def draw\_arc(img, x, y, rotation, radius):

center = (x, y)

axes = (radius, radius)

start\_angle = 100

end\_angle = 0

cv2.ellipse(img, center, axes, rotation, start\_angle, end\_angle, (255, 255, 255), int(radius/2))

# Create FaceMeshDetector object

detector = FaceMesh.FaceMeshDetector()

# Initialize cap

cap = None

# Iterate over camera indices to find the correct one

for i in range(10): # You can adjust the range as needed

cap = cv2.VideoCapture(i)

if cap.isOpened():

print(f"Camera found at index {i}")

break

cap.release()

# Check if a camera is found

if cap is None or not cap.isOpened():

print("No camera found.")

exit()

while True:

ret, img = cap.read()

if not ret:

break

img, faces = detector.findFaceMesh(img, draw=False)

if faces:

for face in faces:

# Define facial feature points

face\_points = {

'faceUp': 10,

'faceDown': 152,

'rightEyeUp': 386,

'rightEyeDown': 374,

'leftEyeUp': 159,

'leftEyeDown': 145,

'mouthRight': 78,

'mouthLeft': 308,

'mouthUp': 13,

'mouthDown': 14

}

# Draw ellipse for face

faceLength, face\_pos = findDistance(face[face\_points['faceUp']], face[face\_points['faceDown']], img)

cv2.ellipse(img, (face\_pos[0], face\_pos[1] - 30),

(int(faceLength / 1.5), int(faceLength / 1.5) + 10),

find\_rotation(face[face\_points['faceUp']], face[face\_points['faceDown']]),

0, 360, (52, 225, 255), -1)

# Draw circles for eyes

Right\_eye\_length, R\_eye\_pos = findDistance(face[face\_points['rightEyeUp']], face[face\_points['rightEyeDown']], img)

cv2.circle(img, (R\_eye\_pos[0], R\_eye\_pos[1] - 30), int(Right\_eye\_length / 1.3), (65, 71, 100), -1)

draw\_arc(img, R\_eye\_pos[0]-3, R\_eye\_pos[1]-33, find\_rotation(face[face\_points['rightEyeUp']],

face[face\_points['rightEyeDown']]),

int(Right\_eye\_length/3))

Left\_eye\_length, L\_eye\_pos = findDistance(face[face\_points['leftEyeUp']], face[face\_points['leftEyeDown']], img)

cv2.circle(img, (L\_eye\_pos[0], L\_eye\_pos[1] - 30), int(Left\_eye\_length / 1.3), (65, 71, 100), -1)

draw\_arc(img, L\_eye\_pos[0]-3, L\_eye\_pos[1]-33, find\_rotation(face[face\_points['leftEyeUp']],

face[face\_points['leftEyeDown']]),

int(Left\_eye\_length/3))

# Draw ellipse for mouth

MouthLength1, mouth\_pos = findDistance(face[face\_points['mouthRight']], face[face\_points['mouthLeft']], img)

MouthLength2, \_ = findDistance(face[face\_points['mouthUp']], face[face\_points['mouthDown']], img)

cv2.ellipse(img, (mouth\_pos[0], mouth\_pos[1] - 30),

(int(MouthLength1 / 1.6), int(MouthLength2)),

find\_rotation(face[face\_points['mouthRight']], face[face\_points['mouthLeft']]),

0, 360, (0, 60, 255), -1)

# Draw polygons for eyebrows

right\_eyebrow = [

[face[336][0], face[336][1] - 40],

[face[296][0], face[296][1] - 40],

[face[334][0], face[334][1] - 40],

[face[293][0], face[293][1] - 40],

[face[300][0], face[300][1] - 40],

[face[283][0], face[283][1] - 40],

[face[282][0], face[282][1] - 40],

[face[295][0], face[295][1] - 40],

[face[285][0], face[285][1] - 40]

]

right\_eyebrow\_polygon = np.array([right\_eyebrow], np.int32)

cv2.fillPoly(img, pts=[right\_eyebrow\_polygon], color=(0, 0, 0))

left\_eyebrow = [

[face[70][0], face[70][1] - 40],

[face[63][0], face[63][1] - 40],

[face[105][0], face[105][1] - 40],

[face[66][0], face[66][1] - 40],

[face[107][0], face[107][1] - 40],

[face[55][0], face[55][1] - 40],

[face[52][0], face[52][1] - 40],

[face[65][0], face[65][1] - 40],

[face[53][0], face[53][1] - 40]

]

left\_eyebrow\_polygon = np.array([left\_eyebrow], np.int32)

cv2.fillPoly(img, pts=[left\_eyebrow\_polygon], color=(0, 0, 0))

# Display the image

cv2.imshow('img', img)

if cv2.waitKey(1) & 0xFF == ord('q'): # Exit loop when 'q' is pressed

break

cap.release() # Release the VideoCapture object

cv2.destroyAllWindows() # Close all OpenCV windows