import cv2

import cvzone

from ultralytics import YOLO

import numpy as np

from directkeys import PressKey, ReleaseKey, up\_arrow\_pressed, down\_arrow\_pressed, left\_arrow\_pressed, right\_arrow\_pressed

import time

import torch

# Khởi tạo webcam và model YOLO

cap = cv2.VideoCapture(1)

model = YOLO('yolov8n-pose.pt').to("cuda")

# Tập hợp phím đang nhấn

current\_key\_pressed = set()

# Tọa độ gốc ban đầu (origin)

origin\_x, origin\_y = 320, 360

neutral\_radius = 60  # Bán kính vùng neutral

# Biến đếm thời gian để cập nhật tọa độ gốc

last\_update\_time = time.time()

update\_interval = 0.5

# Khởi tạo biến đếm khung hình

frame\_counter = 0

# Lưu trữ các keypoints của khung hình trước

previous\_keypoints = None

def update\_keys(new\_keys):

    """Cập nhật trạng thái các phím dựa trên vùng mới."""

    global current\_key\_pressed

    keys\_to\_press = set(new\_keys)

    # Thả các phím không còn được sử dụng

    for key in current\_key\_pressed - keys\_to\_press:

        ReleaseKey(key)

    # Nhấn các phím mới

    for key in keys\_to\_press - current\_key\_pressed:

        PressKey(key)

    # Cập nhật trạng thái

    current\_key\_pressed = keys\_to\_press

while True:

    ret, frame = cap.read()

    if not ret:

        print("Không thể truy cập webcam hoặc luồng video.")

        break

    # Resize khung hình

    frame = cv2.resize(frame, (640, 720))

    height, width = frame.shape[:2]

    # Tạo khung trống để hiển thị overlay

    blank\_image = np.zeros((height, width, 3), dtype=np.uint8)

    # Chuyển đổi khung hình sang grayscale

    gray\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

    # Chạy mô hình YOLO mỗi 3 khung hình

    if frame\_counter % 2 == 0:  # Chạy mô hình YOLO mỗi 2 khung hình

        results = model(frame)

    # Tăng biến đếm khung hình

    frame\_counter += 1

    frame = results[0].plot()  # Hiển thị kết quả từ YOLO

    # Kiểm tra nếu có keypoints

    if results[0].keypoints is not None and len(results[0].keypoints.data) > 0:

        for keypoints in results[0].keypoints.data:

            keypoints = keypoints.cpu().numpy()

            # Đảm bảo có đủ số keypoints

            if keypoints.shape[0] < 17:

                continue

            # Lấy tọa độ điểm keypoint[0] (mũi hoặc trung tâm cơ thể)

            center\_x, center\_y, conf = keypoints[0]

            if conf > 0.5:

                current\_time = time.time()

                if current\_time - last\_update\_time > update\_interval:

                    origin\_x, origin\_y = int(center\_x), int(center\_y)

                    last\_update\_time = current\_time

                # Tính khoảng cách từ keypoint[0] đến gốc tọa độ

                dx = int(center\_x) - origin\_x

                dy = origin\_y - int(center\_y)

                # Danh sách các phím cần nhấn

                keys\_to\_press = []

                # Kiểm tra nếu keypoint[0] nằm trong vùng "Neutral"

                if abs(dx) <= neutral\_radius and abs(dy) <= neutral\_radius:

                    position = "NEUTRAL"

                else:

                    if abs(dy) > abs(dx):  # Chuyển động dọc chiếm ưu thế

                        if dy > 0:

                            position = "UP"

                            keys\_to\_press.append(up\_arrow\_pressed)

                        else:

                            position = "DOWN"

                            keys\_to\_press.append(down\_arrow\_pressed)

                    else:  # Chuyển động ngang chiếm ưu thế

                        if dx > 0:

                            position = "RIGHT"

                            keys\_to\_press.append(left\_arrow\_pressed)

                        else:

                            position = "LEFT"

                            keys\_to\_press.append(right\_arrow\_pressed)

                # Cập nhật các phím được nhấn

                update\_keys(keys\_to\_press)

                if position:

                    cv2.putText(blank\_image, position, (10, 60),

                                cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (255, 255, 255), 2)

                # Vẽ trục cố định tại gốc tọa độ

                cv2.line(blank\_image, (origin\_x, 0), (origin\_x, height), (0, 255, 0), 2)

                cv2.line(blank\_image, (0, origin\_y), (width, origin\_y), (0, 255, 0), 2)

                # Vẽ vùng neutral

                cv2.rectangle(blank\_image,

                              (origin\_x - neutral\_radius, origin\_y - neutral\_radius),

                              (origin\_x + neutral\_radius, origin\_y + neutral\_radius),

                              (255, 255, 0), 2)

            # Kiểm tra Optical Flow (so sánh keypoints giữa khung hình hiện tại và khung hình trước)

            if previous\_keypoints is not None:

                for i, keypoint in enumerate(keypoints):

                    x, y, confidence = keypoint

                    if confidence > 0.7 and i < len(previous\_keypoints):

                        prev\_x, prev\_y, prev\_confidence = previous\_keypoints[i]

                        # Nếu có chuyển động, tính toán sự thay đổi vị trí và vẽ keypoint tương ứng

                        if abs(x - prev\_x) > 2 or abs(y - prev\_y) > 2:

                            cv2.circle(blank\_image, (int(x), int(y)), radius=5, color=(255, 0, 0), thickness=1)

                            cv2.putText(blank\_image, f'{i}', (int(x), int(y) - 10), cv2.FONT\_HERSHEY\_SIMPLEX,

                                        0.4, (255, 255, 255), 1, cv2.LINE\_AA)

            # Lưu trữ keypoints hiện tại để so sánh với khung hình tiếp theo

            previous\_keypoints = keypoints

            # Vẽ các kết nối giữa keypoints

            connections = [

                (3, 1), (1, 0), (0, 2), (2, 4), (1, 2), (4, 6), (3, 5),

                (5, 6), (5, 7), (7, 9), (6, 8), (8, 10),

                (11, 12), (11, 13), (13, 15), (12, 14), (14, 16),

                (5, 11), (6, 12)

            ]

            for part\_a, part\_b in connections:

                if part\_a >= keypoints.shape[0] or part\_b >= keypoints.shape[0]:

                    continue

                x1, y1, conf1 = keypoints[part\_a]

                x2, y2, conf2 = keypoints[part\_b]

                if conf1 > 0.5 and conf2 > 0.5:

                    cv2.line(blank\_image, (int(x1), int(y1)), (int(x2), int(y2)), (255, 0, 255),

                             thickness=2)

    # Hiển thị kết quả

    output = cvzone.stackImages([frame, blank\_image], cols=2, scale=0.80)

    cv2.imshow('Pose Tracking with Dynamic Origin', output)

    # Nhấn 'q' để thoát

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# Thả các phím trước khi thoát

update\_keys([])

cap.release()

cv2.destroyAllWindows()

torch.cuda.empty\_cache()