Informe : Detección de Postura con MediaPipe, Hilos y Streamlit

Alumno: David Diaz Curso: Digitales 3

1 de noviembre de 2025

Índice

Resumen

Se presenta el desarrollo completo de una aplicación para detección de posturas humanas (de pie / sentado) mediante MediaPipe, control de concurrencia con hilos y mutex, e interfaz gráfica con Streamlit. El informe documenta el proceso paso a paso desde la concepción, codificación y sincronización hasta la ejecución en contenedor Docker.

1. Objetivos

- Implementar una aplicación Python que procese video en tiempo real con MediaPipe y clasifique la postura.
- Aplicar conceptos de concurrencia: hilos, mutex, semáforos y secciones críticas.
- Mostrar los resultados en una interfaz Streamlit.
- Empaquetar la solución en Docker y documentar la estructura del repositorio.

2. Arquitectura general

El sistema sigue el patrón productor–consumidor:

1. **Hilo de captura**: lee los cuadros de la cámara y los almacena en una variable global.
2. **Hilo de procesamiento**: toma los cuadros, ejecuta MediaPipe y obtiene los landmarks.
3. **Hilo principal**: visualiza los resultados en Streamlit.

El acceso concurrente se controla mediante un **mutex** (`threading.Lock()`) para evitar condiciones de carrera.

3. Código principal

```
import streamlit as st
import cv2
import mediapipe as mp
import threading
import time

st.set_page_config(page_title="Detecci n␣de␣Postura", layout="
   centered")
st.title("              ␣Detecci n␣de␣Postura␣con␣MediaPipe")

st.sidebar.header("Configuraci n␣de␣C mara")
cam_index = st.sidebar.selectbox("Selecciona␣la␣c mara:", [0,1],
   index=0)
```

```
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(min_detection_confidence=0.5,
   min_tracking_confidence=0.5)
mp_drawing = mp.solutions.drawing_utils

frame_lock = threading.Lock()
frame_global = None
results_global = None
running = True

def detectar_postura(landmarks):
cadera_izq = landmarks[mp_pose.PoseLandmark.LEFT_HIP.value]
rodilla_izq = landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value]
cadera_der = landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value]
rodilla_der = landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value]
h_caderas = (cadera_izq.y + cadera_der.y) / 2
h_rodillas = (rodilla_izq.y + rodilla_der.y) / 2
return "      De pie" if (h_rodillas - h_caderas) > 0.15 else "
        Sentado"

def hilo_captura():
global frame_global, running
cap = cv2.VideoCapture(cam_index)
if not cap.isOpened():
st.error("No se puede abrir la c mara.")
running = False
return
while running:
ret, frame = cap.read()
if not ret:
continue
with frame_lock:
frame_global = frame.copy()
time.sleep(0.03)
cap.release()

def hilo_procesamiento():
global frame_global, results_global, running
while running:
with frame_lock:
if frame_global is None:
continue
frame_rgb = cv2.cvtColor(frame_global, cv2.COLOR_BGR2RGB)
results = pose.process(frame_rgb)
with frame_lock:
results_global = results
time.sleep(0.05)

t1 = threading.Thread(target=hilo_captura, daemon=True)
t2 = threading.Thread(target=hilo_procesamiento, daemon=True)
t1.start()
```

```
t2.start()

frame_placeholder = st.empty()
estado_placeholder = st.empty()

while running:
with frame_lock:
if frame_global is None:
continue
frame = frame_global.copy()
results = results_global

'''
if results and results.pose_landmarks:
    mp_drawing.draw_landmarks(frame, results.pose_landmarks,
       mp_pose.POSE_CONNECTIONS)
    estado = detectar_postura(results.pose_landmarks.landmark)
    estado_placeholder.markdown(f"### Estado: **{estado}**")
else:
    estado_placeholder.markdown("### No se detecta persona")

frame_placeholder.image(frame, channels="BGR")
time.sleep(0.05)
'''

running = False
```

4. Empaquetado con Docker

no se llevo a cabo puesto que al Docker aislar esto y empaquetarlo no puede tomar control del hadware fisico de la camara, por lo cual no funcionaria, por lo tanto se recomienda mejor ejecutar directamente con streamlit run

```
source venv/bin/activate
streamlit run main1.py
```

5. Conclusiones

La práctica permitió integrar procesamiento de visión por computador con control de concurrencia. El uso de MediaPipe simplificó la extracción de landmarks, mientras que los hilos y mutex garantizaron sincronización segura. La interfaz Streamlit ofreció una visualización interactiva y el uso de Docker facilitó la portabilidad del proyecto.

6. Referencias

- MediaPipe Pose: `https://ai.google.dev/edge/mediapipe/solutions/examples?hl=es-419`

- Streamlit Docs: `https://docs.streamlit.io/`