

▼ Proyecto Final - Algoritmos y Programación III

- Angela Maria Gonzalez Cordoba - A00399435
- Juan Manuel Casanovs Marin - A00400090
- Juliana Filigrana Valencia - A00153988

Primera Entrega

Sistema de anotación de video para análisis de actividades humanas usando modelos de analítica

```
from google.colab import drive  
drive.mount('/content/drive')
```

Mounted at /content/drive

```
!pip install mediapipe opencv-python-headless pandas matplotlib seaborn openpyxl
```

```
Requirement already satisfied: mediapipe in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: openpyxl in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: absl-py in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: attrs>=19.1.0 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: jax in /usr/local/lib/python3.12/dist-packages (from mediapipe>0.10.0)  
Requirement already satisfied: jaxlib in /usr/local/lib/python3.12/dist-packages (from mediapipe>0.10.0)  
Requirement already satisfied: numpy<2 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: opencv-contrib-python in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: protobuf<5,>=4.25.3 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: sounddevice>=0.4.4 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: CFFI>=1.0 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: ml_dtypes>=0.4.0 in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: opt_einsum in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: scipy>=1.11.1 in /usr/local/lib/python3.12/dist-packages
```

Resumen

Este proyecto busca desarrollar una solución para el análisis en tiempo real de actividades específicas de una persona a partir de videos, mediante seguimiento de movimientos articulares y posturales. Se propone usar la metodología CRISP-DM adaptada para guiar la investigación, la recolección y el análisis de datos, así como la evaluación de modelos supervisados para clasificar actividades como caminar, girar y sentarse. En esta primera entrega se presenta la definición del problema, metodología, análisis exploratorio de datos y estrategias para ampliar la base de datos, junto con un análisis ético de la implementación de IA en este contexto.

Introducción

El análisis de actividades humanas en videos tiene aplicaciones importantes en áreas como la salud, el deporte y la seguridad. Este proyecto se enfoca en desarrollar una herramienta que permita detectar y clasificar actividades específicas en video en tiempo real, considerando variaciones en movimientos articulares y posturas. La relevancia de esta solución radica en mejorar la precisión del seguimiento postural y potenciar aplicaciones basadas en la interacción humano-máquina.

Definición del problema y cuestiones de interés

El problema central es desarrollar un sistema capaz de recibir video en tiempo real, procesarlo para identificar movimientos específicos y realizar seguimiento de articulaciones clave, clasificando las actividades en curso. Las preguntas de interés incluyen: ¿Cómo ajustar y entrenar modelos supervisados para distinguir con precisión actividades y posturas? ¿Qué características (ángulos, velocidades) son más relevantes para la clasificación? ¿Cómo responder adecuadamente a distintas perspectivas y velocidades de las personas en el video?

Metodología

Se adopta la metodología CRISP-DM adaptada a este proyecto, abarcando las fases de comprensión del problema, preparación y análisis de los datos, modelado y evaluación. Para la anotación, se combinará etiquetado manual y uso de herramientas como LabelStudio para facilitar la segmentación de actividades clave. Para el seguimiento de articulaciones se emplearán tecnologías como MediaPipe, centradas en landmarks

específicos (cadera, rodillas, hombros, etc.). La extracción de características considera ángulos articulares, velocidades y la inclinación del tronco. Se planifica la implementación de modelos supervisados como SVM o Random Forest para clasificación.

Para la fase de recolección de datos, se grabarán una serie de videos enfocados en capturar diferentes actividades humanas que servirán como base para el entrenamiento y análisis del modelo. Las actividades seleccionadas serán: caminar hacia la cámara, caminar de regreso, girar, sentarse y ponerse de pie.

Todos los videos serán filmados en el mismo lugar, bajo condiciones de iluminación controlada y fondo uniforme, con el fin de reducir el ruido visual y facilitar el seguimiento de las articulaciones por parte del modelo de visión por computadora.

El dispositivo utilizado será un teléfono móvil montado sobre un trípode, lo que garantizará la estabilidad de la cámara y mantendrá la misma posición y altura en todas las grabaciones. Esto permitirá capturar los movimientos sin desplazamiento del punto de vista, asegurando la consistencia espacial necesaria para el análisis posterior.

Variables y objetivos

- Variable dependiente: clase de actividad (caminar hacia, caminar regreso, girar, sentarse, ponerse de pie).
- Variables independientes/observables: coordenadas (x,y,z) de 33 landmarks MediaPipe, ángulos articulares (rodilla, cadera, hombro), velocidad/AC del movimiento, brillo, FPS, resolución.

Muestra y balance

- Tamaño muestral: N participantes ($\geq 10-20$ para diversidad mínima).
- Criterios de inclusión: persona de pie, cuerpo completo visible, sin objetos que tapen articulaciones.
- Criterios de exclusión: clips borrosos, frame drops extremos, ocrusión >30% del cuerpo.

Aleatorización y contrabalanceo

- Orden aleatorio de actividades por participante.
- Contrabalanceo de dirección (entrar/salir por izquierda/derecha) para evitar sesgos de trayectoria.

Preprocesamiento planificado

- Extracción MediaPipe: versión y parámetros
- Normalización: centrar en caderas u hombros; escalar por distancia hombro-hombro; suavizado (filtro Savitzky–Golay o media móvil).

- Features iniciales: ángulos (rodilla, cadera, tronco), velocidades angulares, inclinación lateral, path length de muñeca/tobillo.

EDA planificado (entrega 1)

- A nivel video: #frames, duración, FPS, resolución, brillo promedio, índice de movimiento.
- A nivel landmarks (muestra piloto): histogramas X/Y, trayectorias seleccionadas, boxplots para outliers.

Riesgos y mitigación

- Iluminación variable → luz fija y chequeo de brillo antes de cada sesión.
- Oclusiones → indicar al participante mantener brazos visibles; repetir toma si falla tracking.

Métricas para la evaluación del rendimiento

Para la evaluación del desempeño del sistema se emplearán métricas clásicas de clasificación supervisada: precisión (accuracy), recall y F1-Score. Estas métricas permiten medir qué tan bien el modelo identifica correctamente cada actividad y manejan el balance entre falsos positivos y falsos negativos. También se considerará la capacidad del sistema para generalizar a diferentes personas y condiciones de video.

Recogida y descripción de datos

Los datos consisten en videos capturados con diversas personas realizando actividades específicas: caminar hacia la cámara, caminar de regreso, girar, sentarse y ponerse de pie. Se contemplan variaciones en velocidades, trayectorias y perspectivas para robustecer el modelo. Los videos serán procesados usando MediaPipe para extraer landmarks articulares (cadera, rodillas, tobillos, muñecas, hombros, cabeza). Esta información será exportada a archivos Excel, generando un dataset estructurado con las coordenadas y características derivadas (ángulos, velocidades, inclinaciones), normalizadas y filtradas para eliminar ruido.

Aquí tienes la explicación de la estrategia en formato Markdown, lista para tu informe en Google Colab:

Estrategia de Captura, Etiquetado y Análisis de Movimientos con MediaPipe

1. Captura y Preparación de Videos

Se graban videos de varias personas realizando secuencias de movimientos específicos: **pararse, sentarse, girar, caminar hacia adelante y caminar hacia atrás**. Los videos se organizan y nombran de acuerdo con la persona y la actividad para facilitar su procesamiento posterior.

2. Procesamiento con MediaPipe Pose

Cada video es procesado cuadro a cuadro utilizando MediaPipe Pose, una solución de machine learning que detecta automáticamente **33 puntos de referencia tridimensionales** (landmarks) en el cuerpo humano, incluyendo cabeza, hombros, caderas, rodillas, tobillos y muñecas.

- Para cada frame, MediaPipe genera las coordenadas

$$x, y, z$$

y un valor de visibilidad para cada articulación.

- Estos datos permiten reconstruir la postura y el movimiento de la persona en cada instante del video.

3. Etiquetado Automático de Movimientos

El etiquetado de los movimientos se realiza de forma **automática** y basada en reglas, utilizando la información de los landmarks:

- Se definen criterios biomecánicos para identificar cada movimiento. Por ejemplo:
 - **Sentarse:** Disminución rápida de la altura de la cadera y aumento del ángulo de flexión de rodillas.
 - **Pararse:** Aumento de la altura de la cadera y extensión de las rodillas.
 - **Girar:** Cambios significativos en la orientación de los hombros y caderas.
 - **Caminar:** Desplazamiento alternado de caderas y pies en dirección horizontal.
- Estas reglas se aplican sobre las secuencias de coordenadas para asignar una etiqueta de actividad a cada frame del video.

4. Construcción del DataFrame

Toda la información extraída se organiza en un **DataFrame**:

- Cada fila representa un frame del video.
- Las columnas incluyen las coordenadas de cada articulación, los ángulos relevantes y la etiqueta de movimiento asignada automáticamente.
- Este DataFrame se puede exportar a Excel o CSV para su análisis posterior.

✓ Análisis exploratorio de datos (EDA)

El análisis exploratorio consistirá en examinar la distribución y comportamiento de las variables extraídas con MediaPipe y análisis descriptivo. Se realizarán gráficos de dispersión y líneas temporales para las posiciones articulares, y se calcularán estadísticas descriptivas de ángulos y velocidades. El objetivo es identificar patrones, tendencias o irregularidades que permitan guiar el modelado y optimización. Además, se corroborará la consistencia y calidad de los datos antes del entrenamiento.

Los datos están en el siguiente link: [Link Datos](#)

```
import cv2
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")
```

▼ Análisis

Antes de aplicar cualquier filtrado, recorte o transformación, realizamos un análisis exploratorio de los videos en su forma original para caracterizar su calidad y variabilidad. Esto nos permite entender la fuente de datos real (iluminación, estabilidad, resolución) y establecer una línea base objetiva.

Métrica y variables extraídas por video. Para cada video leemos sus metadatos y calculamos medidas simples pero informativas:

- **fps:** cuadros por segundo (propiedad cv2.CAP_PROP_FPS).
- **frames:** número total de cuadros (cv2.CAP_PROP_FRAME_COUNT).
- **duración_s:** duración en segundos, calculada como frames / fps.
- **resolución:** ancho (cv2.CAP_PROP_FRAME_WIDTH) y alto (cv2.CAP_PROP_FRAME_HEIGHT) en píxeles.
- **brillo_promedio:** promedio del nivel de gris por cuadro, calculado convirtiendo cada cuadro a escala de grises y tomando la media de intensidades. Esta métrica resume la iluminación global del video.
- **movimiento_promedio:** promedio de la magnitud de cambio entre cuadros consecutivos, usando la diferencia absoluta $\text{abs}(\text{frame}_t - \text{frame}_{\{t-1\}})$. Esta métrica aproxima el nivel de movimiento/temblor o cambios de escena.

Procedimiento

1. Listamos los archivos de la carpeta y filtramos por ".mp4".

2. Para cada archivo:

- Abrimos el video con OpenCV (cv2.VideoCapture).
- Leemos fps, frames y resoluciones para derivar duración_s.
- Tomamos el primer cuadro como referencia y luego iteramos sobre el resto:
- Convertimos cada cuadro a escala de grises y acumulamos su media para el brillo_promedio.
- Calculamos la diferencia absoluta con el cuadro anterior y acumulamos su media para el movimiento_promedio.
- Guardamos los resultados en una estructura tipo lista de diccionarios.

3. Al finalizar, consolidamos los resultados en una tabla con las columnas:

- video, duracion_s, frames, fps, ancho, alto, brillo_promedio, movimiento_promedio.

```
carpeta_videos = '/content/drive/MyDrive/PROYECTO APO/VideosAPO' # carpeta donde están los videos
datos = []

for archivo in os.listdir(carpeta_videos):
    if archivo.endswith('.mp4'):
        video = cv2.VideoCapture(os.path.join(carpeta_videos, archivo))
        fps = video.get(cv2.CAP_PROP_FPS)
        frames = int(video.get(cv2.CAP_PROP_FRAME_COUNT))
        duracion = frames / fps

        # Resolución
        ancho = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
        alto = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))

        # Brillo y movimiento promedio
        frame_count = 0
        brillo_total = 0
        movimiento_total = 0

        ret, frame1 = video.read()
        if not ret:
            continue

        while True:
            ret, frame2 = video.read()
            if not ret:
                break
            # Brillo promedio
            gris = cv2.cvtColor(frame2, cv2.COLOR_BGR2GRAY)
            brillo_total += np.mean(gris)
            # Movimiento
            diff = cv2.absdiff(frame1, frame2)
            movimiento_total += np.mean(diff)

            frame_count += 1
            frame1 = frame2

        datos.append({
            'video': archivo,
            'duracion_s': duracion,
            'frames': frames,
            'fps': fps,
            'ancho': ancho,
            'alto': alto,
            'brillo_promedio': brillo_total / frame_count,
            'movimiento_promedio': movimiento_total / frame_count
        })

video.release()
```

```

frame1 = frame2
frame_count += 1

brillo_promedio = brillo_total / frame_count
movimiento_promedio = movimiento_total / frame_count

datos.append({
    'video': archivo,
    'duracion_s': duracion,
    'frames': frames,
    'fps': fps,
    'ancho': ancho,
    'alto': alto,
    'brillo_promedio': brillo_promedio,
    'movimiento_promedio': movimiento_promedio
})

video.release()

```

Crear DataFrame y revisar los datos

```

df = pd.DataFrame(datos)
print(df.describe())

```

	duracion_s	frames	fps	ancho	alto	\
count	18.000000	18.000000	18.000000	18.000000	18.000000	
mean	15.055000	576.777778	38.268311	741.333333	577.777778	
std	2.643403	227.604031	13.830183	176.981222	162.232787	
min	11.466667	343.000000	29.896907	464.000000	480.000000	
25%	13.190000	397.000000	29.927433	560.000000	480.000000	
50%	14.981667	473.500000	29.936767	848.000000	480.000000	
75%	15.908333	775.000000	52.449124	848.000000	744.000000	
max	22.863333	1029.000000	59.941748	848.000000	832.000000	
	brillo_promedio	movimiento_promedio				
count	18.000000	18.000000				
mean	143.892220	1.168171				
std	3.030601	0.375145				
min	138.502537	0.483080				
25%	142.224678	0.892146				
50%	144.048499	1.333862				
75%	146.367528	1.470228				
max	148.156281	1.543853				

Muestra: 18 videos, cálculos consistentes (frames/fps ≈ duración).

Duración: media ~15.1 s; hay 1 outlier largo (22.86 s) → recortar o analizar por ventanas.

FPS: mediana ~30; mezcla de ~30 y ~60 fps → estandarizar a 30 fps o normalizar “por segundo”.

Resolución: familias 848×480 , 560×480 y algunas alturas $744-832$; un caso $464 \times 480 \rightarrow$ unificar a 848×480 y corregir orientación/padding.

Brillo: estable ($\approx 144 \pm 3/255$) \rightarrow no requiere normalización fuerte.

Movimiento: media 1.17, mediana 1.33; algunos muy bajos (~0.48) \rightarrow marcarlos para revisión/estrategia aparte.

Graficar las métricas

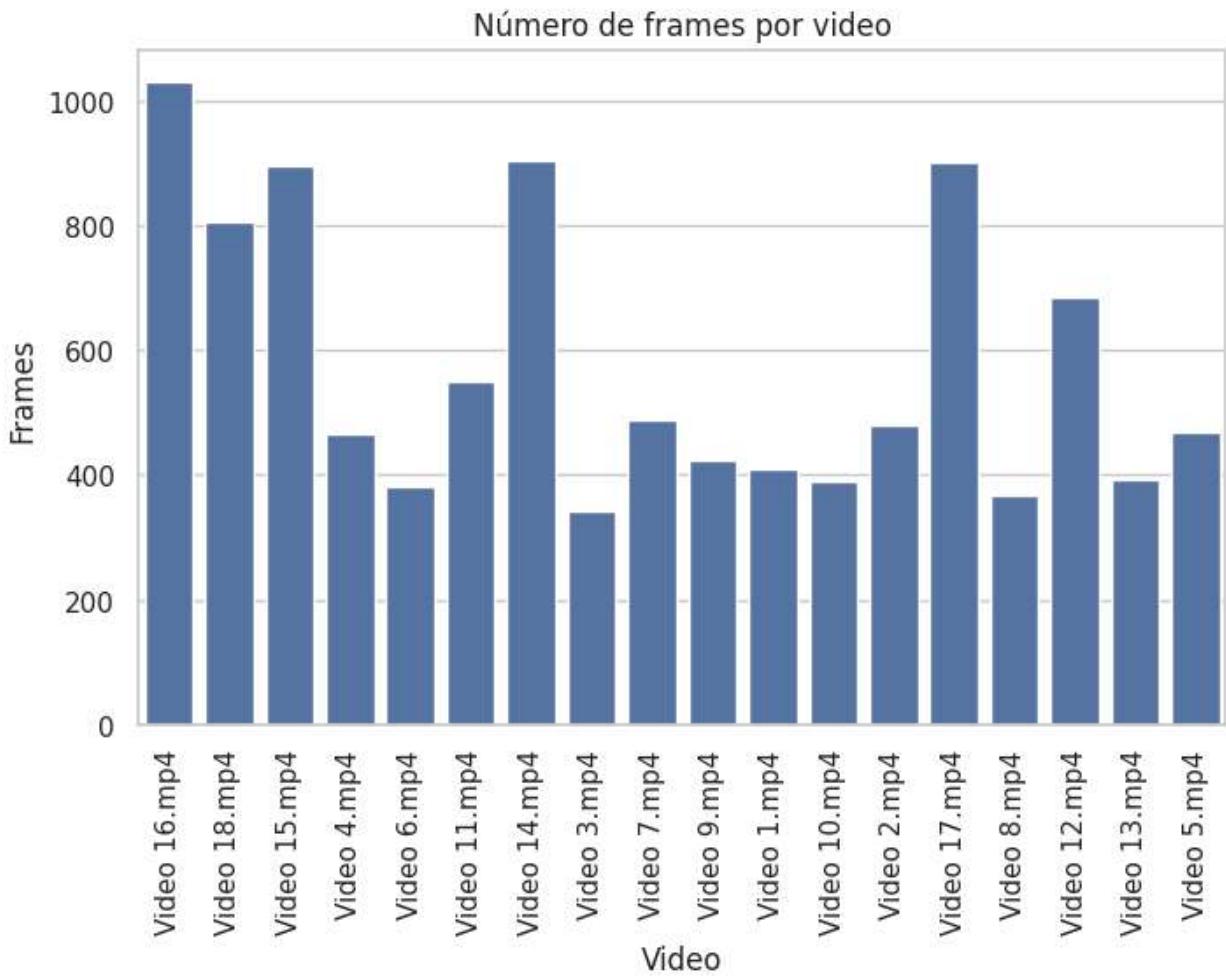
```
plt.figure(figsize=(8,5))
sns.histplot(df['duracion_s'], bins=10, kde=True)
plt.title("Distribución de la duración de los videos (segundos)")
plt.xlabel("Duración (s)")
plt.ylabel("Cantidad de videos")
plt.show()
```



El histograma muestra que la mayoría de los videos tienen una duración entre 13 y 16 segundos, lo que indica que el conjunto de grabaciones es relativamente homogéneo en su longitud. Esto sugiere que los videos fueron grabados bajo condiciones o protocolos similares, probablemente con una duración planificada.

Frames por video

```
plt.figure(figsize=(8,5))
sns.barplot(x='video', y='frames', data=df)
plt.xticks(rotation=90)
plt.title("Número de frames por video")
plt.ylabel("Frames")
plt.xlabel("Video")
plt.show()
```



El gráfico muestra la cantidad total de frames por video, lo cual refleja tanto la duración de cada grabación como la tasa de cuadros por segundo (FPS) con la que fueron capturados.

En términos simples: más frames = video más largo o con mayor FPS.

Se evidencia una variabilidad notable en el número de frames entre los videos. Algunos, como Video 16.mp4 y Video 17.mp4, superan los 900–1000 frames, mientras que otros como Video 3.mp4 o Video 6.mp4 tienen menos de 400 frames.

El conjunto de datos presenta videos con entre 350 y 1000 frames, con una media aproximada de 600 frames.

FPS de los videos

```
plt.figure(figsize=(8,5))
sns.boxplot(y=df['fps'])
plt.title("Distribución de FPS en los videos")
plt.ylabel("FPS")
plt.show()
```



El boxplot representa la variabilidad de la tasa de cuadros por segundo (FPS) entre los videos del conjunto de datos. Los valores de FPS reflejan la fluidez del movimiento en cada grabación: a mayor FPS, más suave y detallada es la representación del movimiento.

El rango total de FPS va aproximadamente de 30 a 60 FPS, lo que indica que los videos fueron capturados con dos configuraciones principales: algunos con 30 FPS (estándar) y otros con 60 FPS (alta tasa de refresco).

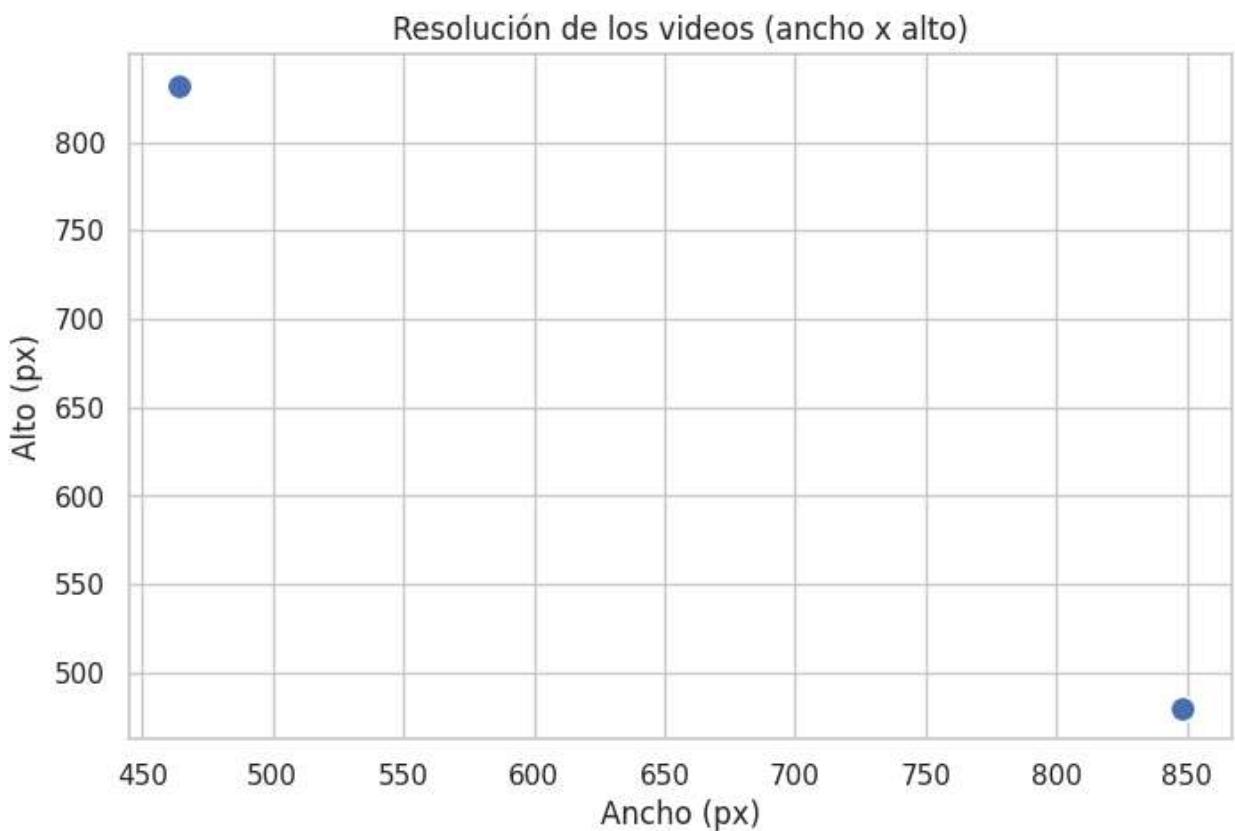
La mediana se encuentra alrededor de 50 FPS, lo cual muestra una ligera tendencia hacia grabaciones más fluidas.

El rango intercuartílico (IQR) abarca valores entre 30 y 53 FPS, lo que refleja una moderada variabilidad en la velocidad de grabación.

No se observan valores atípicos extremos, lo que indica que todos los videos se mantuvieron dentro de un rango razonable de tasas de captura.

Resolución de los videos

```
plt.figure(figsize=(8,5))
sns.scatterplot(x='ancho', y='alto', data=df, s=100)
plt.title("Resolución de los videos (ancho x alto)")
plt.xlabel("Ancho (px)")
plt.ylabel("Alto (px)")
plt.show()
```



Se observan solo dos puntos, lo que indica que los videos se encuentran grabados en dos resoluciones distintas:

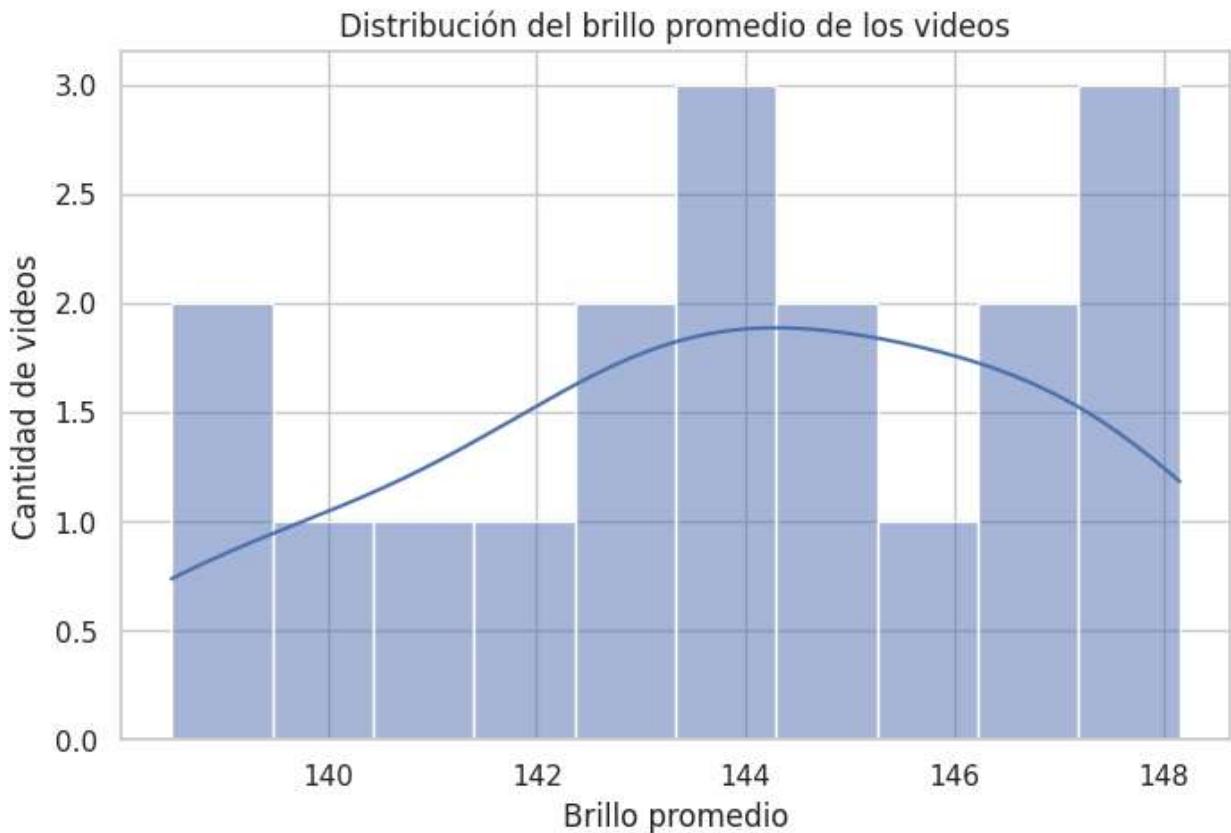
Una alrededor de 450×850 px, correspondiente probablemente a videos en formato vertical (retrato).

Otra cerca de 850×480 px, que sugiere videos en formato horizontal (paisaje).

Esta diferencia sugiere que el dataset incluye videos capturados desde diferentes dispositivos o en distintas orientaciones de cámara.

Brillo promedio

```
plt.figure(figsize=(8,5))
sns.histplot(df['brillo_promedio'], bins=10, kde=True)
plt.title("Distribución del brillo promedio de los videos")
plt.xlabel("Brillo promedio")
plt.ylabel("Cantidad de videos")
plt.show()
```



Los valores de brillo se concentran entre 138 y 148, lo que demuestra una homogeneidad en las condiciones de iluminación durante la grabación.

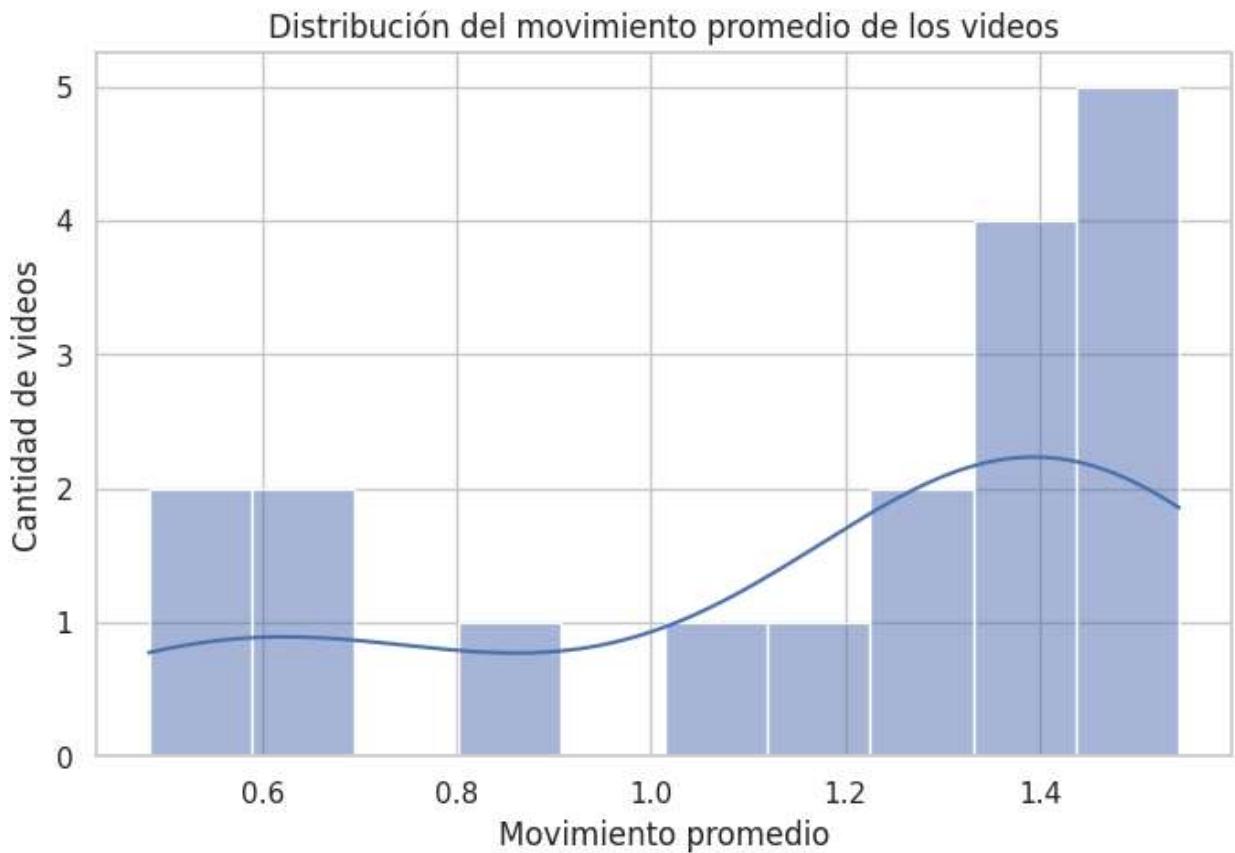
La mayor concentración de videos se encuentra alrededor de 143–145, indicando que la mayoría fueron grabados en entornos moderadamente iluminados, probablemente interiores o zonas con luz natural indirecta.

No se observan valores extremos (muy oscuros o sobreexpuestos), lo que implica una buena consistencia visual en el conjunto de datos.

La curva suavizada (línea azul) sugiere una distribución levemente bimodal, es decir, algunos videos ligeramente más claros y otros más oscuros, pero sin diferencias significativas.

Movimiento promedio

```
plt.figure(figsize=(8,5))
sns.histplot(df['movimiento_promedio'], bins=10, kde=True)
plt.title("Distribución del movimiento promedio de los videos")
plt.xlabel("Movimiento promedio")
plt.ylabel("Cantidad de videos")
plt.show()
```



Se observa una tendencia creciente hacia la derecha, lo que significa que varios videos presentan niveles medios a altos de movimiento (entre 1.2 y 1.5 unidades promedio).

Solo unos pocos videos muestran valores inferiores a 0.8, lo que indica que la mayoría de las grabaciones contienen actividad corporal significativa, posiblemente caminatas, giros o cambios de postura.

La curva suavizada (línea azul) refuerza esta tendencia, mostrando un pico en el rango alto del eje X, lo que confirma que el conjunto de datos no está dominado por escenas estáticas.

No se observan outliers pronunciados, lo que sugiere una distribución continua y controlada del movimiento entre los videos.



Análisis Exploratorio de Datos (EDA) – Resultados de MediaPipe

En esta sección realizamos un análisis exploratorio del archivo `all_coords_full.csv`, que contiene las coordenadas extraídas por MediaPipe a partir de los videos procesados. El propósito es conocer la estructura de los datos, verificar su calidad (valores faltantes, dimensiones, rangos) y explorar el comportamiento de las coordenadas de los puntos corporales detectados (landmarks).

```
BASE = "/content/drive/MyDrive/PROYECTO APO"

ALL_CSV      = "/content/drive/MyDrive/PROYECTO APO/mediapipe/all_coords_full.csv"
SUM_CSV      = "/content/drive/MyDrive/PROYECTO APO/mediapipe/metrics_summary.csv"
IDX_MAP      = "/content/drive/MyDrive/PROYECTO APO/mediapipe/landmark_index_map.c

import os
assert os.path.exists(ALL_CSV), f"No encuentro {ALL_CSV}. Asegúrate de correr el
print("Rutas OK:\n", ALL_CSV, "\n", SUM_CSV, "\n", IDX_MAP)
```

Rutas OK:

```
/content/drive/MyDrive/PROYECTO APO/mediapipe/all_coords_full.csv
/content/drive/MyDrive/PROYECTO APO/mediapipe/metrics_summary.csv
/content/drive/MyDrive/PROYECTO APO/mediapipe/landmark_index_map.csv
```

Imports, flags y helpers

```
import os, re, glob, math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Que matplotlib pinte inline en Colab
%matplotlib inline

# ===== FLAGS =====
INCLUDE_POSE_ANALYSIS  = True
INCLUDE_HANDS_ANALYSIS = True
INCLUDE_FACE_ANALYSIS  = False    # True = MUY pesado (468*3 columnas)
COMPUTE_VELOCITIES     = True     # Deriva velocidades (más lento)

def load_header_columns(csv_path):
    with open(csv_path, "r", encoding="utf-8") as f:
        header = f.readline().strip().split(",")
    meta = [c for c in header if c in ("video", "frame", "time_s", "has_pose", "has_")]
    lm_x = [c for c in header if c.startswith("landmark_") and c.endswith("_x")]
    idxs = sorted({int(re.findall(r"landmark_(\d+)_x", c)[0]) for c in lm_x})
    return meta, idxs
```

```

def choose_landmark_columns(indices, pose=True, hands=True, face=False):
    ranges = []
    if pose: ranges.append(range(0, 33))
    if hands: ranges += [range(33, 54), range(54, 75)]
    if face: ranges.append(range(75, 75+468))
    idx_set = set(indices)
    final_idxs = []
    for r in ranges:
        final_idxs.extend([i for i in r if i in idx_set])
    cols = []
    for i in final_idxs:
        cols += [f"landmark_{i}_x", f"landmark_{i}_y", f"landmark_{i}_z"]
    return sorted(set(cols)), final_idxs

def dist2d(df, i, j):
    ax, ay = f"landmark_{i}_x", f"landmark_{i}_y"
    bx, by = f"landmark_{j}_x", f"landmark_{j}_y"
    if ax not in df or ay not in df or bx not in df or by not in df:
        return pd.Series([np.nan]*len(df))
    return np.sqrt((df[ax]-df[bx])**2 + (df[ay]-df[by])**2)

def add_pose_proxies(df):
    # 11(L_shoulder), 12(R_shoulder), 23(L_hip), 24(R_hip)
    need = [11,12,23,24]
    need_cols = sum(([f"landmark_{i}_x", f"landmark_{i}_y"] for i in need), [])
    if not all(c in df.columns for c in need_cols):
        return df
    df["shoulder_width"] = dist2d(df, 11, 12)
    df["hip_width"] = dist2d(df, 23, 24)
    sx = df[["landmark_11_x", "landmark_12_x"]].mean(axis=1)
    sy = df[["landmark_11_y", "landmark_12_y"]].mean(axis=1)
    hx = df[["landmark_23_x", "landmark_24_x"]].mean(axis=1)
    hy = df[["landmark_23_y", "landmark_24_y"]].mean(axis=1)
    df["torso_height"] = np.sqrt((sx-hx)**2 + (sy-hy)**2)
    return df

def compute_velocities(df, idx_list):
    if not {"video", "frame", "time_s"}.issubset(df.columns):
        return pd.DataFrame()
    df = df.sort_values(["video", "frame"])
    parts = []
    for i in idx_list:
        cx, cy = f"landmark_{i}_x", f"landmark_{i}_y"
        if cx not in df or cy not in df:
            continue
        dx = df.groupby("video")[cx].diff()
        dy = df.groupby("video")[cy].diff()
        dt = df.groupby("video")["time_s"].diff().replace(0, np.nan)
        v = np.sqrt(dx**2 + dy**2) / dt
        parts.append(pd.DataFrame({"video": df["video"], "frame": df["frame"], f"v_{i}_x": dx, f"v_{i}_y": dy, "dt": dt, "v": v}))
    if not parts:
        return pd.DataFrame()
    vel = parts[0]
    for p in parts[1:]:
        vel = pd.concat([vel, p], axis=1)
    return vel

```

```
    vel = vel.merge(p, on=["video", "frame"], how="outer")
    return vel
```

Descubrir columnas y cargar dataset (con filtros)

```
meta_cols, all_indices = load_header_columns(ALL_CSV)
lm_cols, idx_list = choose_landmark_columns(
    all_indices,
    pose=INCLUDE_POSE_ANALYSIS,
    hands=INCLUDE_HANDS_ANALYSIS,
    face=INCLUDE_FACE_ANALYSIS
)
usecols = meta_cols + lm_cols
print(f"Meta: {meta_cols}")
print(f"Landmarks: {len(idx_list)} índices → {len(lm_cols)} columnas")

df = pd.read_csv(ALL_CSV, usecols=usecols, low_memory=False)
print("Dimensiones:", df.shape)
df.head()
```

Meta: ['video', 'frame', 'time_s', 'has_pose', 'has_lh', 'has_rh', 'has_face']
 Landmarks: 75 índices → 225 columnas
 Dimensiones: (20728, 232)

	video	frame	time_s	has_pose	has_lh	has_rh	has_face	landmark_0_x	landma
0	video 1	1	0.000000	1	1	1	1	0.474635	0
1	video 1	2	0.033415	1	1	1	1	0.474662	0
2	video 1	3	0.066830	1	1	1	1	0.474667	0
3	video 1	4	0.100244	1	1	1	1	0.474663	0
4	video 1	5	0.133659	1	1	1	1	0.474663	0

5 rows × 232 columns

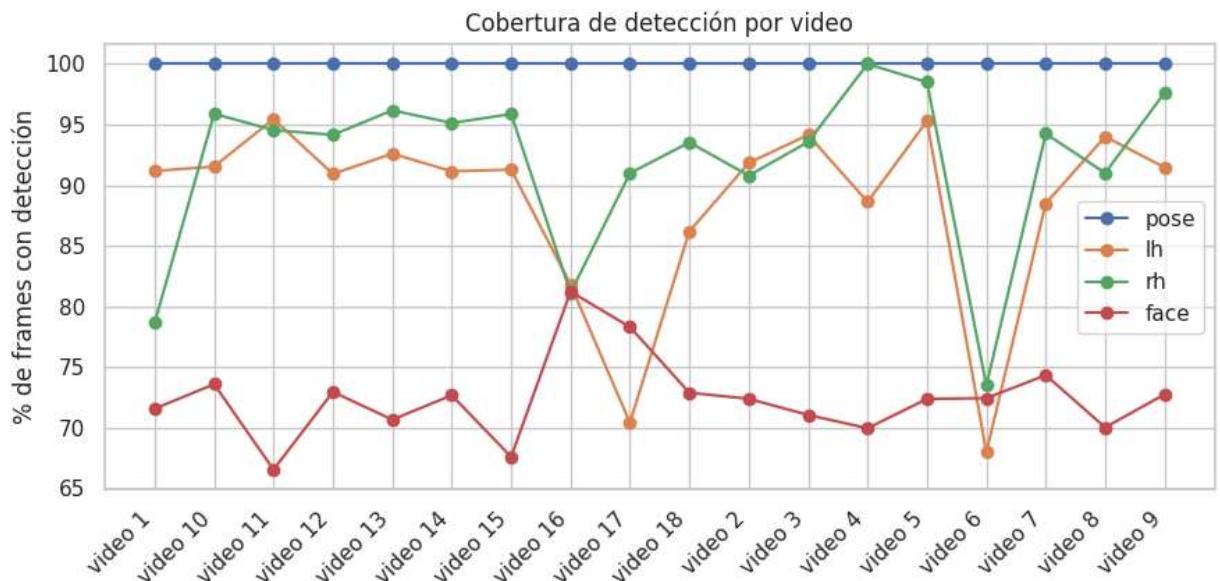
Cobertura de detección por video y gráfico

```
cov = df.groupby("video")[["has_pose", "has_lh", "has_rh", "has_face"]].mean().rese...
cov[["has_pose", "has_lh", "has_rh", "has_face"]] *= 100.0
cov.columns = ["video", "pose_%", "lh_%", "rh_%", "face_%"]
display(cov)
OUTPUT_DIR = "/content/drive/MyDrive/PROYECTO APO/mediapipe"

plt.figure(figsize=(9,4.5))
```

```
for col in ["pose_%", "lh_%", "rh_%", "face_%"]:
    if col in cov.columns:
        plt.plot(cov["video"], cov[col], marker="o", label=col.replace("_%", ""))
plt.xticks(rotation=45, ha="right")
plt.ylabel("% de frames con detección")
plt.title("Cobertura de detección por video")
plt.legend()
png = os.path.join(OUTPUT_DIR, "eda_cobertura_por_video.png")
plt.tight_layout(); plt.savefig(png, dpi=140); plt.show()
print("Guardado:", png)
```

	video	pose_%	lh_%	rh_%	face_%
0	video 1	100.0	91.176471	78.676471	71.568627
1	video 10	100.0	91.538462	95.897436	73.589744
2	video 11	100.0	95.454545	94.545455	66.545455
3	video 12	100.0	90.935673	94.152047	72.953216
4	video 13	100.0	92.602041	96.173469	70.663265
5	video 14	100.0	91.150442	95.132743	72.676991
6	video 15	100.0	91.284916	95.865922	67.597765
7	video 16	100.0	81.809339	81.128405	81.225681
8	video 17	100.0	70.411568	90.989989	78.309232
9	video 18	100.0	86.194030	93.532338	72.885572
10	video 2	100.0	91.841004	90.794979	72.384937
11	video 3	100.0	94.152047	93.567251	71.052632
12	video 4	100.0	88.626609	100.000000	69.957082
13	video 5	100.0	95.289079	98.501071	72.376874
14	video 6	100.0	67.979003	73.490814	72.440945
15	video 7	100.0	88.501027	94.250513	74.332649
16	video 8	100.0	94.005450	91.008174	70.027248
17	video 9	100.0	91.469194	97.630332	72.748815



Guardado: /content/drive/MyDrive/PROYECTO APO/mediapipe/eda_cobertura_por_video.

La gráfica representa el porcentaje de frames en los que MediaPipe logró detectar landmarks de cada tipo:

- Pose (azul): puntos del cuerpo (tronco, brazos, piernas, etc.)
- LH (naranja): mano izquierda
- RH (verde): mano derecha
- Face (rojo): puntos faciales

Cada línea muestra la consistencia de detección por video.

La detección del cuerpo (pose) se mantiene en 100% para todos los videos, lo que significa que el modelo logró rastrear la postura corporal completa en cada cuadro, sin interrupciones.

→ Esto demuestra buena iluminación, encuadre completo del cuerpo y estabilidad en las grabaciones.

Las detecciones de manos (LH y RH) presentan una variabilidad moderada:

- En la mayoría de los videos, superan el 90% de cobertura, lo que indica que las manos estuvieron visibles la mayor parte del tiempo.
- Sin embargo, en algunos videos (como el video 6 y video 16) la detección cae por debajo del 80%, probablemente porque las manos salieron del encuadre o ocurrieron occlusiones (por ejemplo, al girar o sentarse).

La detección facial (rojo) muestra los porcentajes más bajos y variables, entre 65% y 75% en la mayoría de los videos.

Faltantes por landmark (NaN %) y gráfico TOP-20

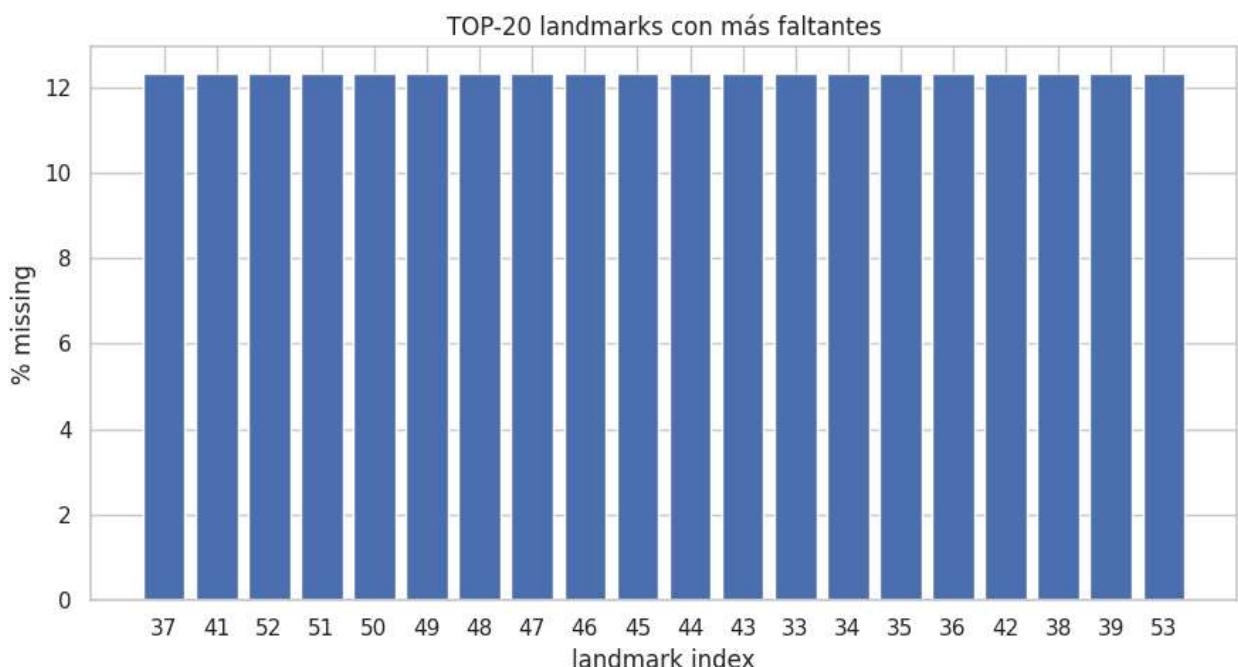
```

na_rows = []
for i in idx_list:
    colx = f"landmark_{i}_x"
    present = df[colx].notna().mean() * 100.0
    na_rows.append({"landmark_index": i, "present_%": present, "missing_%": 100.0 - present})
na_df = pd.DataFrame(na_rows).sort_values("missing_%", ascending=False)
display(na_df.head(10))

top20 = na_df.head(20)
plt.figure(figsize=(9,5))
plt.bar([str(i) for i in top20["landmark_index"]], top20["missing_%"])
plt.xlabel("landmark index"); plt.ylabel("% missing")
plt.title("TOP-20 landmarks con más faltantes")
png = os.path.join(OUTPUT_DIR, "eda_missing_top20.png")
plt.tight_layout(); plt.savefig(png, dpi=140); plt.show()
print("Guardado:", png)

```

landmark_index	present_%	missing_%
37	37	87.678503
41	41	87.678503
52	52	87.678503
51	51	87.678503
50	50	87.678503
49	49	87.678503
48	48	87.678503
47	47	87.678503
46	46	87.678503
45	45	87.678503



Guardado: /content/drive/MyDrive/PROYECTO APO/mediapipe/eda_missing_top20.png

El gráfico indica los índices de landmarks (puntos anatómicos) que más frecuentemente no fueron detectados o resultaron en valores nulos (NaN) a lo largo del conjunto de datos. El eje Y representa el porcentaje de frames donde esos puntos no fueron capturados correctamente.

Los landmarks listados (33–53 y algunos cercanos) tienen un porcentaje de ausencia cercano al 12.3%, lo que significa que aproximadamente 1 de cada 8 cuadros no contiene

coordenadas válidas para esos puntos.

- La mayoría de estos índices corresponden a las manos (izquierda y derecha), según el mapeo de MediaPipe Holistic: 37–53 → suelen representar dedos y articulaciones distales (falanges) de las manos.

La pérdida de detección en estos landmarks se debe principalmente a:

- Oclusiones parciales (cuando la mano se tapa con el cuerpo o sale del encuadre).
- Movimiento rápido, que provoca desenfoque o pérdida temporal de rastreo.

Condiciones de iluminación o ángulo de cámara que reducen el contraste entre la piel y el fondo.

Los landmarks del rostro y el cuerpo no aparecen en este top, lo que sugiere que las manos son las regiones más propensas a errores de detección.

Proxies de tamaño (pose): hombros, caderas, torso

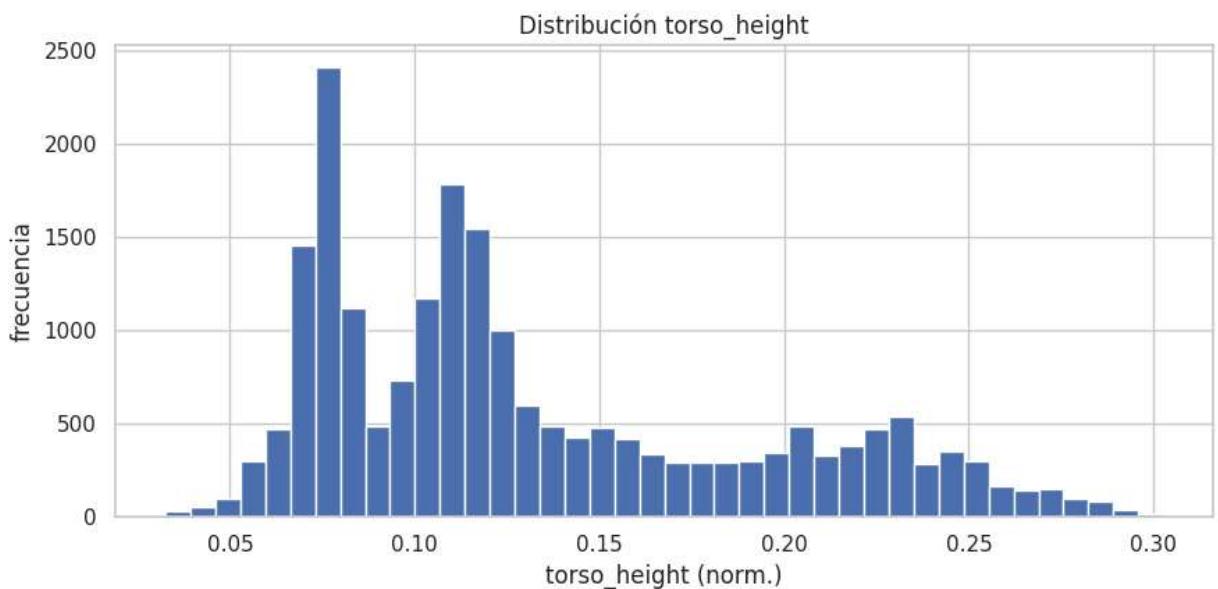
```
if INCLUDE_POSE_ANALYSIS:
    df_pose = add_pose_proxies(df.copy())
    have_cols = {"shoulder_width", "hip_width", "torso_height"}.issubset(df_pose.columns)
    if have_cols:
        display(df_pose[["video", "shoulder_width", "hip_width", "torso_height"]].head())

        # Histograma torso_height
        plt.figure(figsize=(9,4.5))
        df_pose["torso_height"].hist(bins=40)
        plt.xlabel("torso_height (norm.)"); plt.ylabel("frecuencia")
        plt.title("Distribución torso_height")
        png = os.path.join(OUTPUT_DIR, "eda_torso_height_hist.png")
        plt.tight_layout(); plt.savefig(png, dpi=140); plt.show()
        print("Guardado:", png)

        # Promedios por video
        pose_sizes = (df_pose.groupby("video")[["shoulder_width", "hip_width", "torso_height"]]
                      .mean().reset_index())
        display(pose_sizes)

        plt.figure(figsize=(9,4.5))
        for c in ["shoulder_width", "hip_width", "torso_height"]:
            plt.plot(pose_sizes["video"], pose_sizes[c], marker="o", label=c)
        plt.xticks(rotation=45, ha="right"); plt.legend(); plt.ylabel("valor proxy")
        plt.title("Proxies de tamaño por video")
        png = os.path.join(OUTPUT_DIR, "eda_pose_sizes_by_video.png")
        plt.tight_layout(); plt.savefig(png, dpi=140); plt.show()
        print("Guardado:", png)
    else:
        print("No puedo calcular proxies de pose (faltan columnas necesarias).")
else:
    print("INCLUDE_POSE_ANALYSIS=False → sección omitida.")
```

	video	shoulder_width	hip_width	torso_height
0	video 1	0.102699	0.066087	0.084337
1	video 1	0.102546	0.065912	0.083661
2	video 1	0.102372	0.065827	0.083012
3	video 1	0.102241	0.065704	0.082639
4	video 1	0.102121	0.065457	0.082296



Guardado: /content/drive/MyDrive/PROYECTO APO/mediapipe/eda_torso_height_hist.pkl

	video	shoulder_width	hip_width	torso_height
0	video 1	0.146234	0.085712	0.131954
1	video 10	0.154297	0.086716	0.139812
2	video 11	0.170341	0.094768	0.152579
3	video 12	0.156380	0.089574	0.138386
4	video 13	0.149763	0.089394	0.130997
5	video 14	0.151697	0.087836	0.129607
6	video 15	0.150188	0.084873	0.125905
7	video 16	0.144939	0.078042	0.117953
8	video 17	0.138882	0.075959	0.114828
9	video 18	0.146173	0.083892	0.138407
10	video 2	0.158026	0.090679	0.142058
11	video 3	0.147500	0.085395	0.124272
12	video 4	0.156394	0.088592	0.141554
13	video 5	0.159646	0.090695	0.138174
14	video 6	0.140057	0.080591	0.129055

Distribución del torso_height normalizado

15	video 7	0.166083	0.090941	0.149479
----	---------	----------	----------	----------

La mayoría de los valores se concentran entre 0.05 y 0.12, con dos picos principales, uno en torno a 0.07 y otro cerca de 0.11. Esto sugiere que en gran parte de los frames el torso mantiene una proporción corporal estable y realista, lo que indica detecciones consistentes.

La cola defecha (valores entre 0.20 y 0.30) representa casos menos frecuentes donde la altura del torso parece mayor. Estos valores pueden corresponder a:

Cambios de posición corporal (por ejemplo, cuando la persona se inclina o se acerca a la cámara).

No se observan valores extremos negativos ni picos fuera del rango anatómicamente esperable, lo cual confirma que el modelo mantiene coherencia geométrica en la mayoría de los cuadros.

Proxies de tamaño corporal promedio por video

La anchura de hombros (shoulder_width) mantiene valores estables entre 0.14 y 0.17 video, siendo la medida más grande de las tres. Esto indica que el modelo de MediaPipe detectó de forma consistente la parte superior del cuerpo, sin grandes variaciones entre videos. Los picos, como en el video 12 y el video 7, podrían corresponder a posturas más abiertas o leves acercamientos a la cámara.

La anchura de caderas (hip_width) presenta valores más bajos ($\approx 0.08-0.09$) y bastante uniformes, lo que sugiere que la cámara mantuvo una distancia estable y que las caderas permanecieron dentro del campo de visión durante la mayoría de los movimientos.

La altura del torso (torso_height) muestra una variación intermedia ($\approx 0.11-0.15$). Los descensos, como en los videos 16 y 17, indican momentos en los que la persona se sienta, gira o se encorva, reduciendo temporalmente la distancia vertical entre hombros y caderas. En cambio, valores altos, como en el video 11 o el 7, sugieren una postura erguida o de pie.

Haz doble clic (o pulsa Intro) para editar

Haz doble clic (o pulsa Intro) para editar

```
import pandas as pd
import matplotlib.pyplot as plt

# Ruta del dataset generado por MediaPipe
ruta = "/content/drive/MyDrive/PROYECTO APO/mediapipe/all_coords_full.csv"
df = pd.read_csv(ruta)

# Información general del dataset
print(" Estructura del dataset:")
print(f"Filas: {df.shape[0]}, Columnas: {df.shape[1]}")
```

```
print("\nColumnas disponibles:", list(df.columns))
print("\nVista previa de los datos:")
display(df.head())
```

Estructura del dataset:
Filas: 20728, Columnas: 1636

Columnas disponibles:

```
['video', 'frame', 'time_s', 'has_pose', 'has_lh', 'has_rh', 'has_face', 'landmark_0_x', 'landma...
```

Vista previa de los datos:

	video	frame	time_s	has_pose	has_lh	has_rh	has_face	landmark_0_x	landma...
0	video 1	1	0.000000	1	1	1	1	0.474635	0
1	video 1	2	0.033415	1	1	1	1	0.474662	0
2	video 1	3	0.066830	1	1	1	1	0.474667	0
3	video 1	4	0.100244	1	1	1	1	0.474663	0
4	video 1	5	0.133659	1	1	1	1	0.474663	0

5 rows × 1636 columns

Cada fila representa un frame procesado, con las coordenadas de los landmarks (puntos del cuerpo) detectados por MediaPipe. El dataset puede incluir columnas como frame, video, landmark, x, y, z, o nombres específicos por punto corporal (left_shoulder_x, right_knee_y, etc.).

▼ Revisión de valores faltantes y estadísticos generales

```
# Valores nulos por columna
print("Valores nulos por columna:")
print(df.isnull().sum().sort_values(ascending=False).head(10))

# Resumen estadístico general
print("\n Resumen estadístico de variables numéricas:")
display(df.describe())
```

Valores nulos por columna:

```
landmark_270_y      5602
landmark_360_x      5602
landmark_363_y      5602
landmark_363_x      5602
landmark_362_z      5602
landmark_362_y      5602
landmark_362_x      5602
landmark_361_z      5602
landmark_361_y      5602
landmark_361_x      5602
dtype: int64
```

Resumen estadístico de variables numéricas:

	frame	time_s	has_pose	has_lh	has_rh	has_fa
count	20728.000000	20728.000000	20728.0	20728.000000	20728.000000	20728.000000
mean	330.875434	7.691770	1.0	0.876785	0.918854	0.72973
std	234.948894	4.687776	0.0	0.328692	0.273066	0.44410
min	1.000000	0.000000	1.0	0.000000	0.000000	0.00000
25%	144.000000	3.743329	1.0	1.000000	1.000000	0.00000
50%	288.000000	7.512611	1.0	1.000000	1.000000	1.00000
75%	461.250000	11.267580	1.0	1.000000	1.000000	1.00000
max	1028.000000	22.796579	1.0	1.000000	1.000000	1.00000

8 rows × 1635 columns

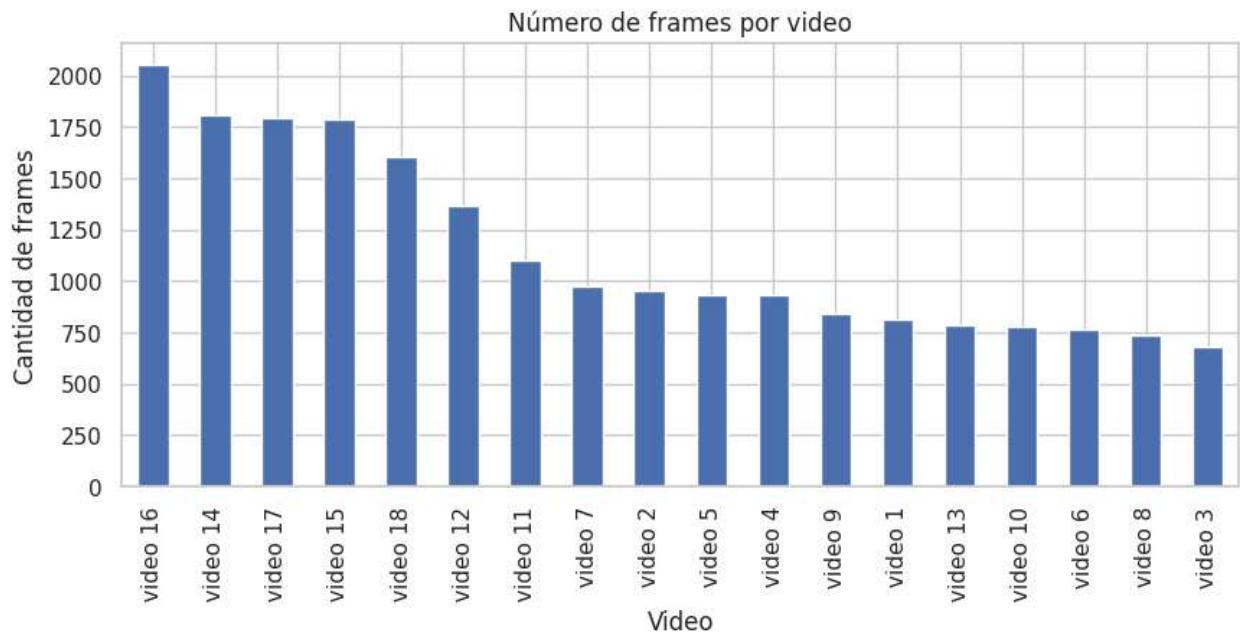
En esta etapa del análisis exploratorio se revisó la calidad y estructura del dataset generado tras el procesamiento con MediaPipe. El objetivo fue identificar la presencia de valores faltantes y obtener un resumen descriptivo de las variables numéricas principales.

- Valores nulos:

Los resultados muestran que varias columnas correspondientes a landmarks (por ejemplo, landmark_270_y, landmark_360_x, landmark_363_y, etc.) presentan 5602 valores nulos cada una. Esto indica que en ciertos frames MediaPipe no logró detectar correctamente algunos puntos corporales, posiblemente debido a occlusiones, movimientos rápidos o pérdida temporal del esqueleto durante la captura. Estos datos faltantes deberán considerarse en las etapas de limpieza o imputación, ya que pueden afectar el entrenamiento del modelo.

▼ Distribución de frames por video

```
if 'video' in df.columns:  
    conteo = df['video'].value_counts()  
    conteo.plot(kind='bar', figsize=(10,4), title='Número de frames por video')  
    plt.xlabel('Video')  
    plt.ylabel('Cantidad de frames')  
    plt.show()
```



- El gráfico muestra la cantidad de frames (cuadros) que fueron extraídos de cada video analizado con MediaPipe. Se observa una variación notable en la duración o número de frames entre los diferentes videos.
- El video 16 es el que más frames posee (alrededor de 2000), lo que sugiere que es el video más largo o el que se procesó con mayor número de cuadros.
- En contraste, el video 3 presenta la menor cantidad de frames (cerca de 700), indicando que puede ser más corto o que tuvo menos fotogramas válidos detectados.
- En general, los videos entre el 14 y el 17 concentran una mayor cantidad de información visual, mientras que los videos 1, 3, 6, 8 y 10 tienen una cantidad más reducida.

Esta distribución desigual de frames por video podría influir en los resultados posteriores del análisis de coordenadas o métricas de movimiento, ya que los videos con más frames

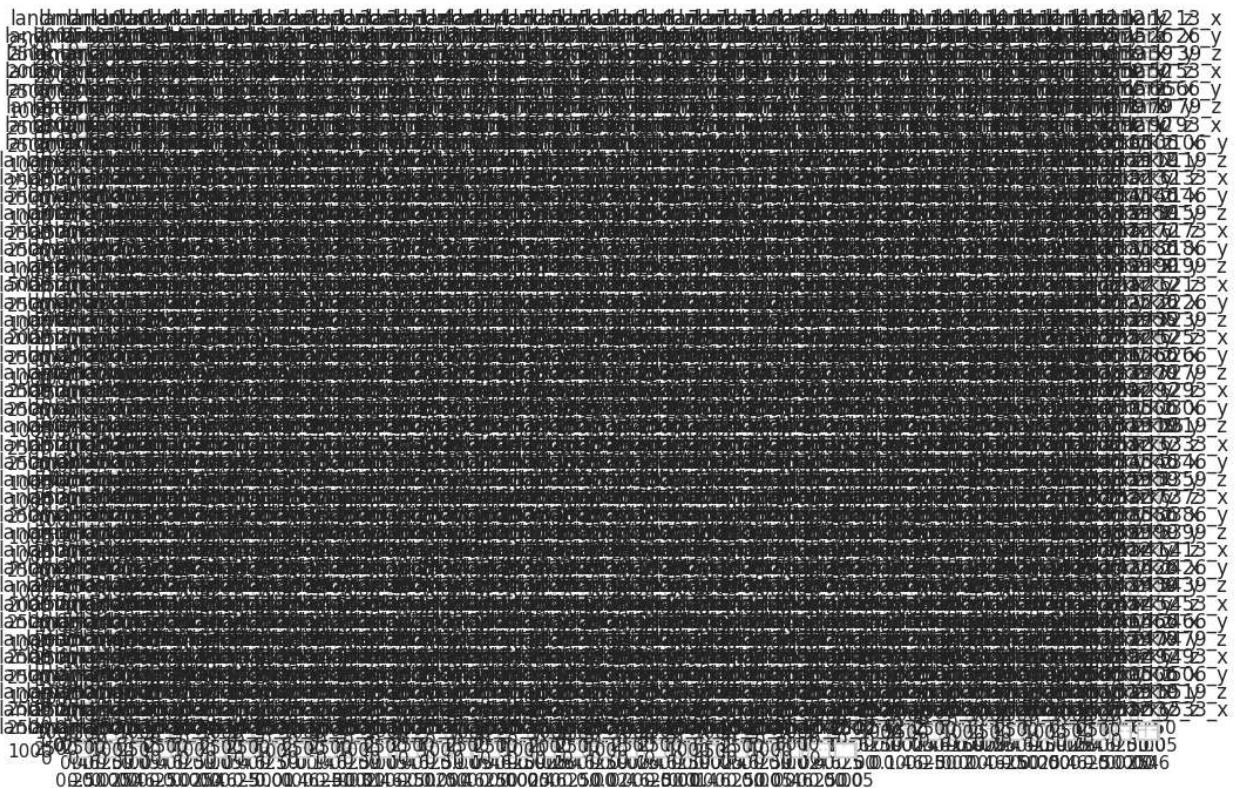
aportan más datos y podrían sesgar las estadísticas globales. Será importante normalizar o equilibrar el número de frames por video antes de realizar comparaciones directas.

▼ Distribución de coordenadas (x, y, z)

```
coord_cols = [c for c in df.columns if any(k in c for k in ['x', 'y', 'z'])]

df[coord_cols].hist(figsize=(12,8), bins=30)
plt.suptitle('Distribución de coordenadas detectadas por MediaPipe', fontsize=14
plt.show()
```

Distribución de coordenadas detectadas por MediaPipe



- Debido a la gran cantidad de puntos (más de 400 landmarks con tres coordenadas cada uno), el gráfico se ve muy denso y poco legible en conjunto.
- Aun así, permite notar que la mayoría de las coordenadas se concentran en rangos medios, lo que indica una captura consistente de posiciones corporales dentro del encuadre del video.
- No se observan valores extremos visibles, lo que sugiere ausencia de outliers marcados en las detecciones.

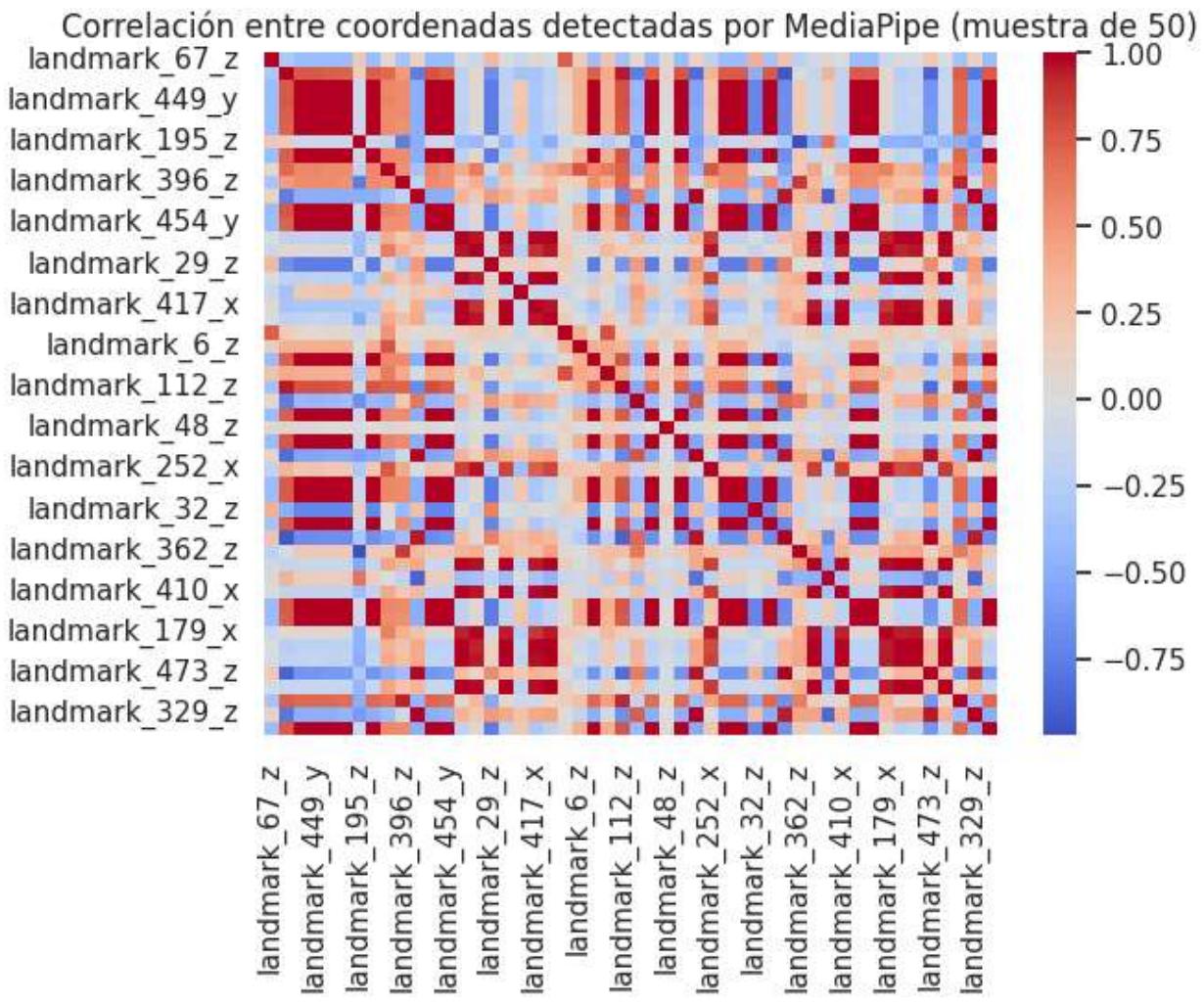
Aunque el gráfico general resulta difícil de leer por la cantidad de variables, evidencia una distribución estable de las coordenadas capturadas por MediaPipe, reflejando que el modelo ha identificado los puntos corporales de manera uniforme a lo largo de los videos.

▼ Correlación entre coordenadas

```
import seaborn as sns
import numpy as np

# Calcular matriz de correlación de las coordenadas
corr = df[coord_cols].corr()

# Graficar solo una muestra de columnas si son muchas
sample_cols = np.random.choice(coord_cols, size=50, replace=False) # 50 columna
sns.heatmap(df[sample_cols].corr(), cmap='coolwarm', center=0)
plt.title('Correlación entre coordenadas detectadas por MediaPipe (muestra de 50')
plt.show()
```



- Se observan varios bloques rojos que indican coordinación entre landmarks cercanos, como los de una misma extremidad o zona facial.
- Las áreas azules o neutras muestran movimientos independientes entre partes del cuerpo, lo que es esperable en datos de pose humana.
- No se aprecian correlaciones extremas en todo el conjunto, lo que sugiere una buena distribución de variabilidad entre las coordenadas, sin redundancia excesiva.

El análisis de correlación confirma que los datos capturados por MediaPipe mantienen coherencia estructural, representando adecuadamente las relaciones espaciales naturales del cuerpo humano. Esto indica que los datos están listos para usarse en análisis posteriores o modelos de clasificación/postura.

▼ Análisis de Métricas y Acciones Detectadas

Explorar las estadísticas agregadas obtenidas de los clips etiquetados (caminar, girar, sentarse, ponerse de pie, etc.) para evaluar cómo varían las métricas físicas (brillo, movimiento, duración, fps, resolución) entre distintas acciones.

```

import pandas as pd
import matplotlib.pyplot as plt

# Ruta del archivo de resumen
ruta_sum = "/content/drive/MyDrive/PROYECTO APO/mediapipe/metrics_summary.csv"

df_sum = pd.read_csv(ruta_sum)
print("Estructura del resumen:")
print(df_sum.info())
display(df_sum.head())

```

Estructura del resumen:

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   video            36 non-null    object  
 1   frames           36 non-null    int64   
 2   fps              36 non-null    float64 
 3   duracion_s       36 non-null    float64 
 4   ancho            36 non-null    int64   
 5   alto              36 non-null    int64   
 6   brillo_promedio  36 non-null    float64 
 7   movimiento_promedio  36 non-null    float64 
 8   frames_pose_detectados  36 non-null    int64   
 9   frames_mano_izq    36 non-null    int64   
 10  frames_mano_der   36 non-null    int64   
 11  frames_rostro     36 non-null    int64  
dtypes: float64(4), int64(7), object(1)
memory usage: 3.5+ KB
None

```

	video	frames	fps	duracion_s	ancho	alto	brillo_promedio	movimiento_promedio	
0	video 1	408	29.93	13.633	480	848	146.223	1	
1	video 1	408	29.93	13.633	480	848	146.223	1	
2	video 10	390	29.92	13.033	480	848	142.611	1	

▼ Distribución de duración, brillo y movimiento

```

fig, axes = plt.subplots(1, 3, figsize=(15,5))

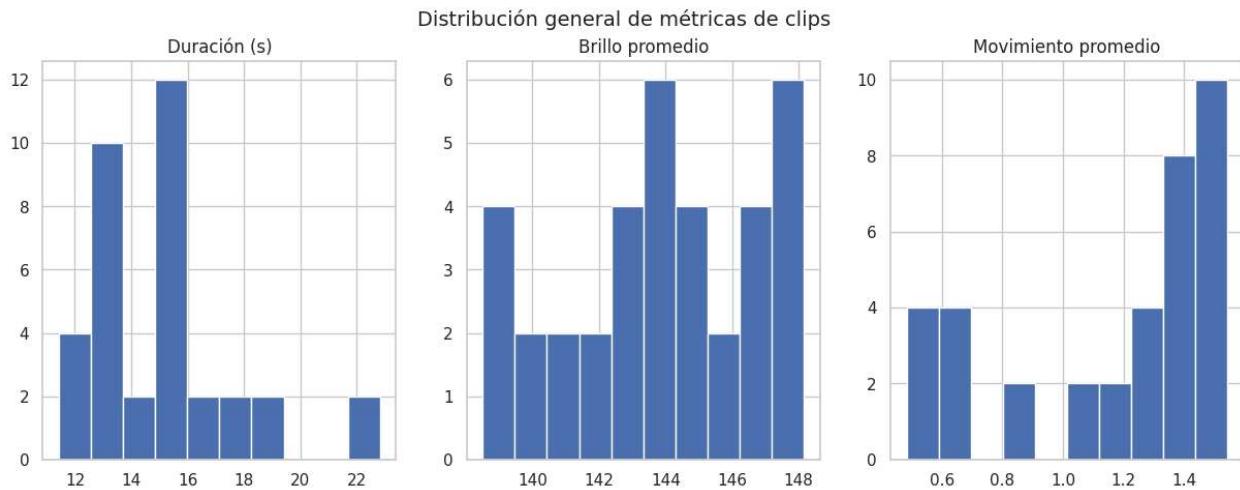
df_sum['duracion_s'].hist(ax=axes[0], bins=10)
axes[0].set_title("Duración (s)")

df_sum['brillo_promedio'].hist(ax=axes[1], bins=10)
axes[1].set_title("Brillo promedio")

```

```
df_sum['movimiento_promedio'].hist(ax=axes[2], bins=10)
axes[2].set_title("Movimiento promedio")

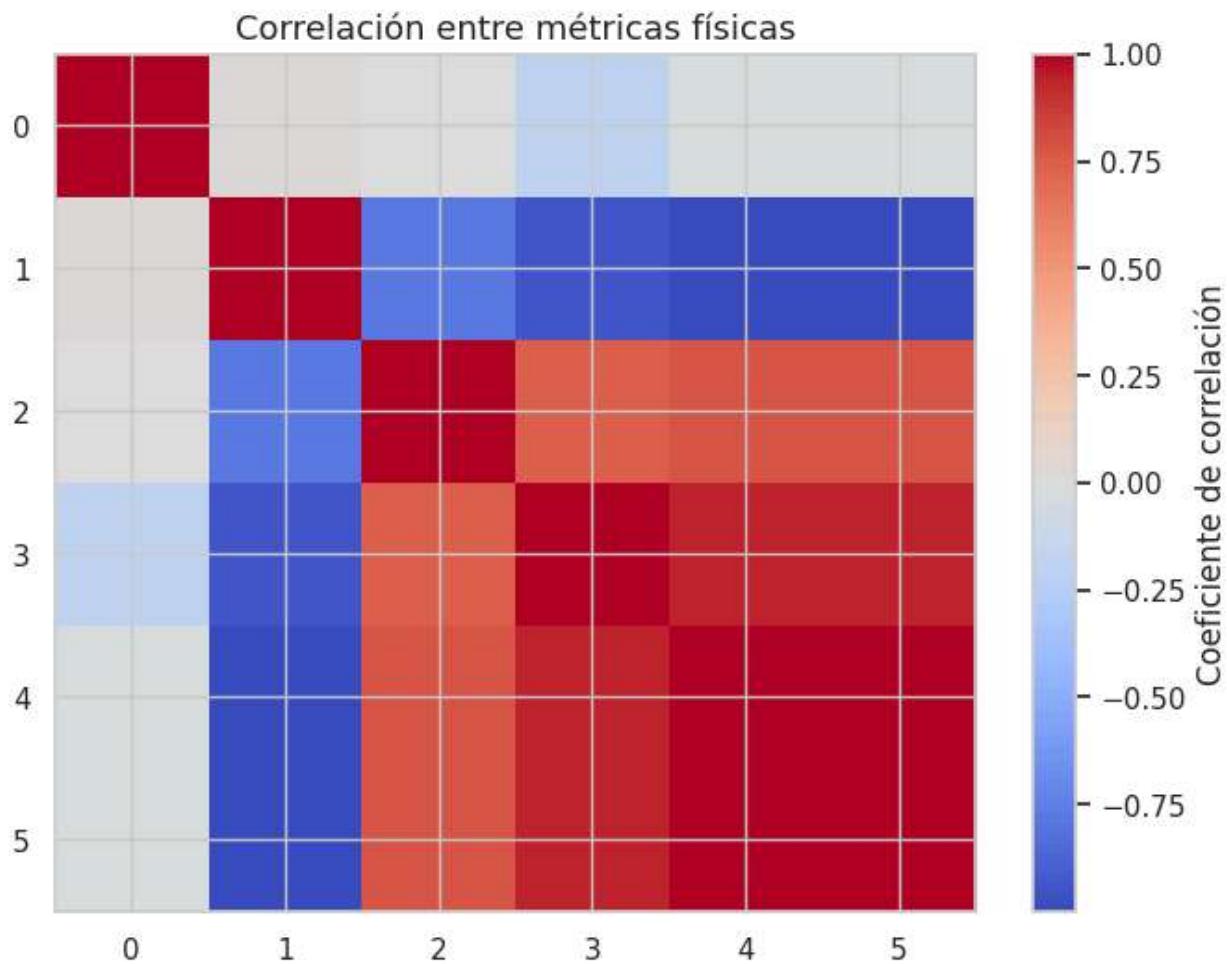
plt.suptitle("Distribución general de métricas de clips", fontsize=14)
plt.show()
```



Los datos presentan una buena homogeneidad y calidad. Las métricas muestran un comportamiento coherente con el tipo de actividades analizadas, lo que sugiere que los clips fueron grabados y procesados de manera uniforme.

Correlaciones entre métricas físicas

```
corr = df_sum[['duracion_s','fps','brillo_promedio','movimiento_promedio','ancho']]
plt.figure(figsize=(8,6))
plt.imshow(corr, cmap='coolwarm', aspect='auto')
plt.colorbar(label="Coeficiente de correlación")
plt.title("Correlación entre métricas físicas", fontsize=13)
plt.show()
```

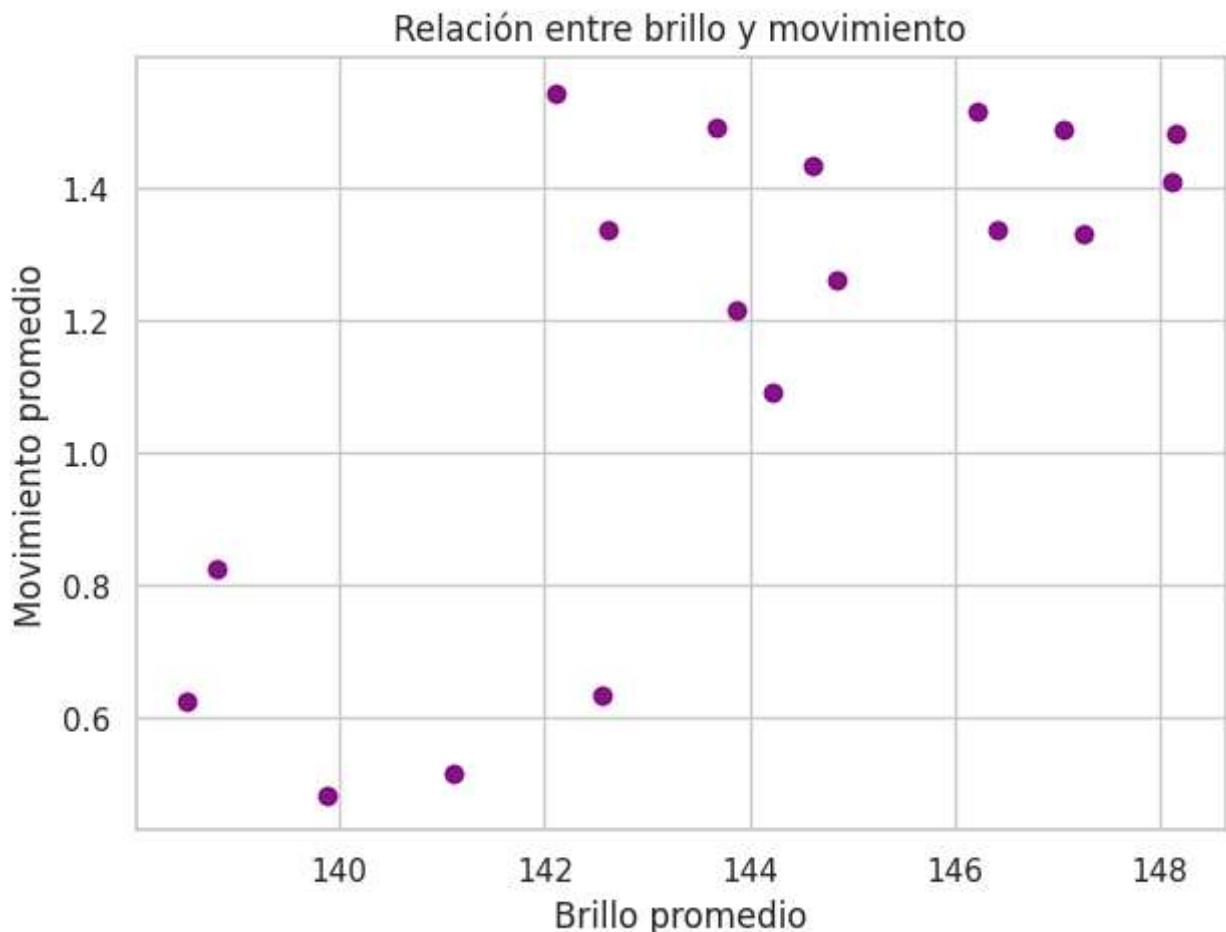


Las correlaciones evidencian que:

- Las métricas de estructura (duración, frames, resolución) están fuertemente relacionadas entre sí.
- Las métricas visuales (brillo y movimiento) son más independientes, lo cual es útil para el análisis posterior con MediaPipe, ya que pueden aportar información complementaria sobre la dinámica del cuerpo y la escena.

▼ Dispersión entre movimiento y brillo

```
plt.figure(figsize=(7,5))
plt.scatter(df_sum['brillo_promedio'], df_sum['movimiento_promedio'], c='purple')
plt.xlabel("Brillo promedio")
plt.ylabel("Movimiento promedio")
plt.title("Relación entre brillo y movimiento")
plt.show()
```



La relación sugiere que los clips con mejor iluminación podrían haber permitido una detección más precisa de los cambios de posición, lo que se traduce en mayores valores de movimiento promedio. Sin embargo, no todos los videos más brillantes muestran mucho movimiento, lo que indica que la iluminación no es el único factor que influye en esta métrica.

▼ Visualización de Landmarks Corporales (MediaPipe)

Explorar visualmente los puntos anatómicos (landmarks) detectados por MediaPipe para:

- Verificar la calidad del seguimiento corporal.
- Detectar errores o pérdidas de tracking.
- Comparar posiciones entre diferentes acciones o frames

```
import pandas as pd
import matplotlib.pyplot as plt

ruta_coords = "/content/drive/MyDrive/PROYECTO APO/mediapipe/all_coords_full.csv"
df_coords = pd.read_csv(ruta_coords)

print("Datos cargados:", df_coords.shape)
```

```
display(df_coords.head())
```

Datos cargados: (20728, 1636)

	video	frame	time_s	has_pose	has_lh	has_rh	has_face	landmark_0_x	landma
0	video 1	1	0.000000	1	1	1	1	0.474635	0
1	video 1	2	0.033415	1	1	1	1	0.474662	0
2	video 1	3	0.066830	1	1	1	1	0.474667	0
3	video 1	4	0.100244	1	1	1	1	0.474663	0
4	video 1	5	0.133659	1	1	1	1	0.474663	0

5 rows × 1636 columns

▼ Seleccionar un frame o acción para visualizar

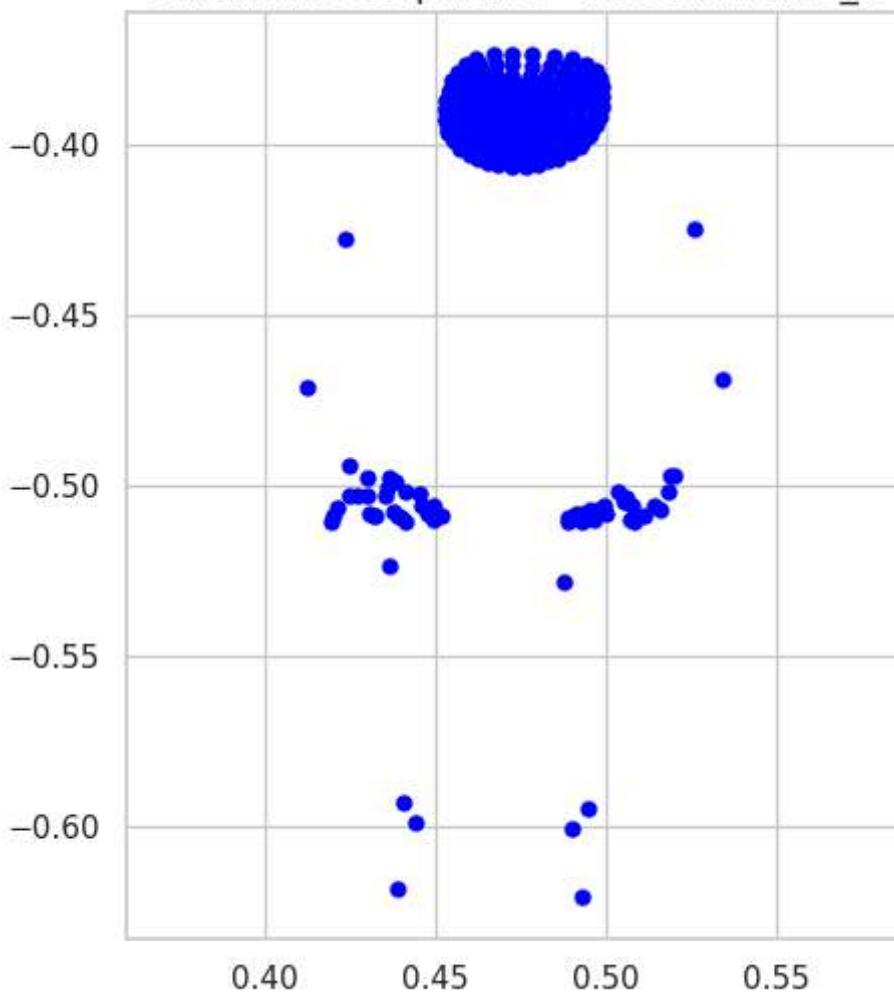
```
# Filtrar un video o acción específica (si existe la columna 'label')
if 'label' in df_coords.columns:
    accion = df_coords['label'].unique()[0] # toma el primer nombre de acción
    subset = df_coords[df_coords['label'] == accion].head(1)
else:
    accion = "frame_1" # nombre genérico si no hay columna 'label'
    subset = df_coords.head(1)

# Obtener coordenadas X e Y de todos los landmarks
landmarks_x = [col for col in df_coords.columns if col.endswith('_x')]
landmarks_y = [col for col in df_coords.columns if col.endswith('_y')]

x = subset[landmarks_x].values.flatten()
y = subset[landmarks_y].values.flatten()

plt.figure(figsize=(5,6))
plt.scatter(x, -y, c='blue', s=25)
plt.title(f"Landmarks corporales - Acción: {accion}", fontsize=13)
plt.axis('equal')
plt.show()
```

Landmarks corporales - Acción: frame_1



En este frame, la persona parece estar en posición erguida o semi-estática, sin movimientos amplios visibles. Los landmarks se detectaron correctamente, lo que confirma que el modelo de MediaPipe logró mapear las articulaciones principales. Sin embargo, se recomienda invertir o escalar el eje Y para representar la postura con orientación más natural (cabeza arriba, pies abajo)

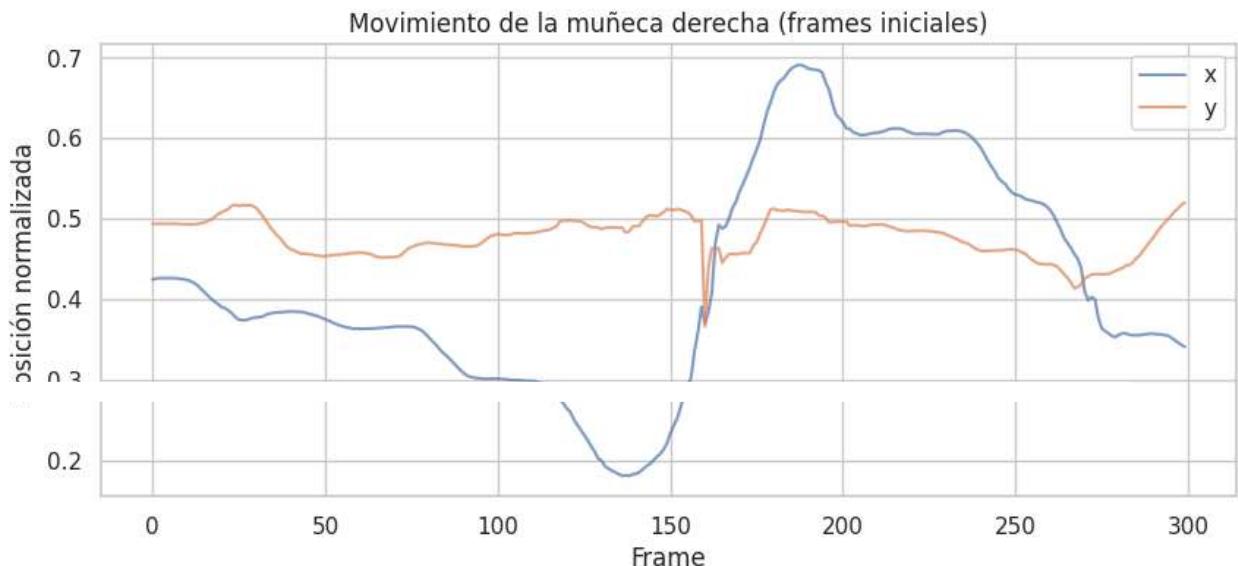
Visualizar movimiento promedio de un punto (por ejemplo, muñeca)

```
import numpy as np

# Seleccionamos las coordenadas de la muñeca derecha (landmark 16 según MediaPipe)
cols_wrist = [c for c in df_coords.columns if '16_' in c]

plt.figure(figsize=(10,4))
plt.plot(df_coords[cols_wrist[0]][:300], label='x', alpha=0.7)
plt.plot(df_coords[cols_wrist[1]][:300], label='y', alpha=0.7)
plt.title("Movimiento de la muñeca derecha (frames iniciales)")
plt.xlabel("Frame")
plt.ylabel("Posición normalizada")
```

```
plt.legend()
plt.show()
```



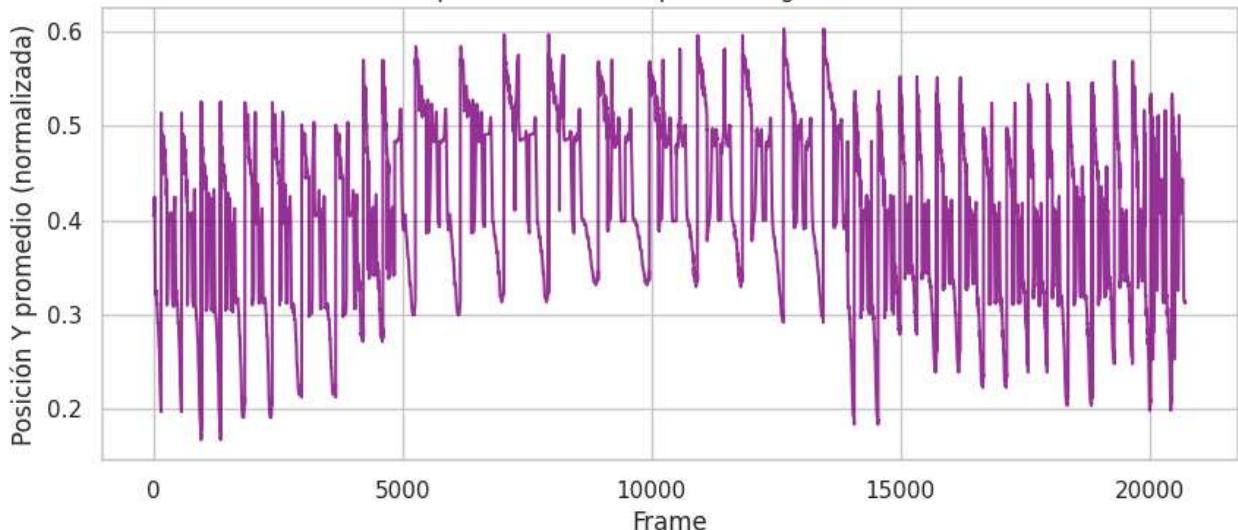
La gráfica muestra que el proceso automatizado con MediaPipe logra capturar de forma precisa los cambios en la posición y movimiento de la muñeca derecha durante los primeros instantes del video. Se observa que la posición x tiene un descenso sostenido, seguido de un cambio abrupto y posterior estabilización, mientras que la posición y permanece relativamente estable, con pequeñas variaciones en los momentos clave. Esta dinámica revela que la muñeca realiza un movimiento lateral amplio y rápido en un tramo específico, lo que puede corresponder a gestos como estiramiento, giro o preparación para una actividad principal.

▼ Comparar landmarks promedio entre frames

```
# Promedio de las coordenadas Y por frame (altura promedio del cuerpo)
coords_y = [c for c in df_coords.columns if '_y' in c]
mean_y = df_coords[coords_y].mean(axis=1)

plt.figure(figsize=(10,4))
plt.plot(mean_y, color='purple', alpha=0.8)
plt.title("Altura promedio del cuerpo a lo largo de los frames")
plt.xlabel("Frame")
plt.ylabel("Posición Y promedio (normalizada)")
plt.show()
```

Altura promedio del cuerpo a lo largo de los frames



A lo largo de la secuencia, los ciclos de subida y bajada son frecuentes y consistentes, lo