DETECCIÓN DE PASOS DESDE MODELOS ARTICULADOS

Hito 3 - IPD441

Integrantes
Ignacio Barrera
Francisca Soto
Fecha:
10-07-2024

Contents

1 Problema

Con el fin de entrenar a deportistas, se ha desarrollado una plataforma que busca realizar un análisis cognitivo y motor de estos. El funcionamiento de esta es mediante estímulos presentados a los deportistas, los cuales les indican a qué posición moverse dentro de un esquema de nueve posiciones. Esta plataforma se basa en una solución de análisis de vídeo a través de una cámara frontal frente al circuito de reacción (ver Figura 1).

La solución actual utiliza modelos de segmentación a través de procesamiento digital de imágenes para detectar las posiciones de los pies, la zona a pisar y la intersección de estas. Sin embargo, con el avance de la tecnología y la inestabilidad de los modelos mencionados, surge la propuesta de utilizar modelos de Deep Learning, más específicamente modelos 2D-SPPE [1] (Single Person Pose Estimation). El esquema explicativo de los tipos de modelos de HPE (Human Pose Estimation) se muestra en la figura 2.

Figure 1: Captura de escenario de Análisis Neuromuscular

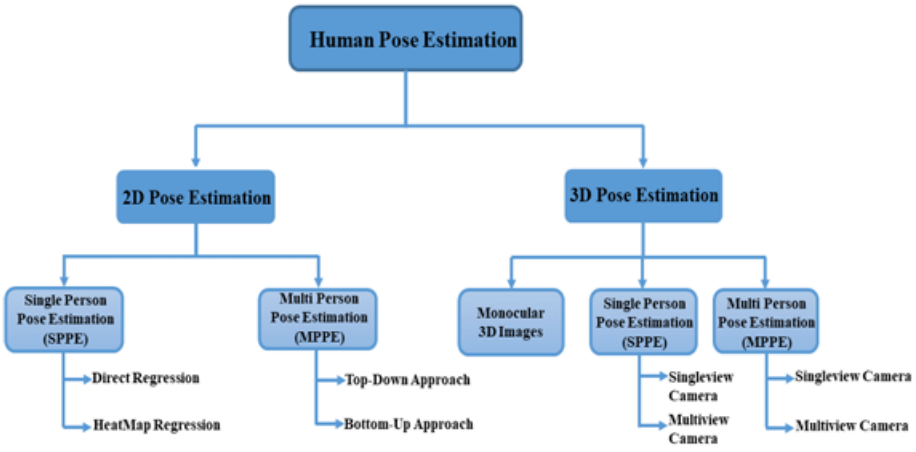

Figure 2: Cuadro resumen de tipos de estimación de pose

2 Solución

La solución que se propone consiste en utilizar dos tipos de estimación de pose humana 2D por medio de modelos kinemáticos (ver a. en figura 3), siendo OpenPose y BlazePose los modelos elegidos. Debido al problema propuesto, nos enfocamos en identificar a una sola persona e identificar si esta pisa el suelo o no, de tal manera que los datos por ambos modelos de estimación de pose se procesarán para detectar las pisadas de los jugadores basándose en un umbral de movimiento entre frames. Para validar los resultados obtenidos, se utilizará un dataset propio de la plataforma de entrenamiento. La propuesta de solución se puede apreciar en la Figura 4.

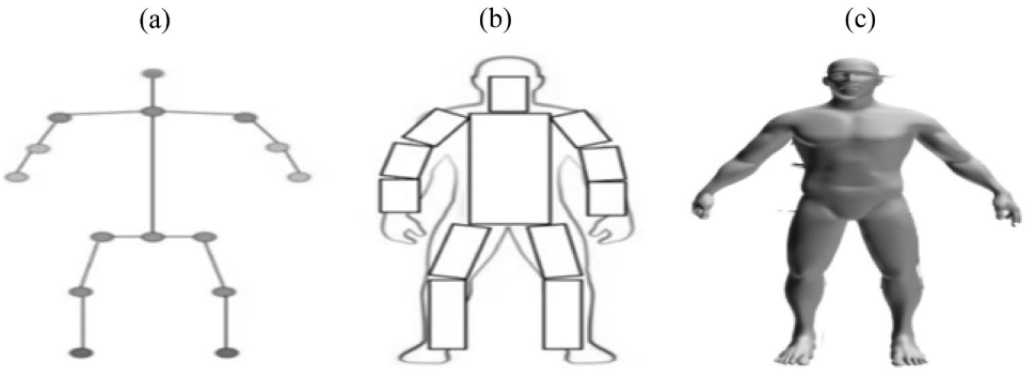

Figure 3: Tipos de esqueletos

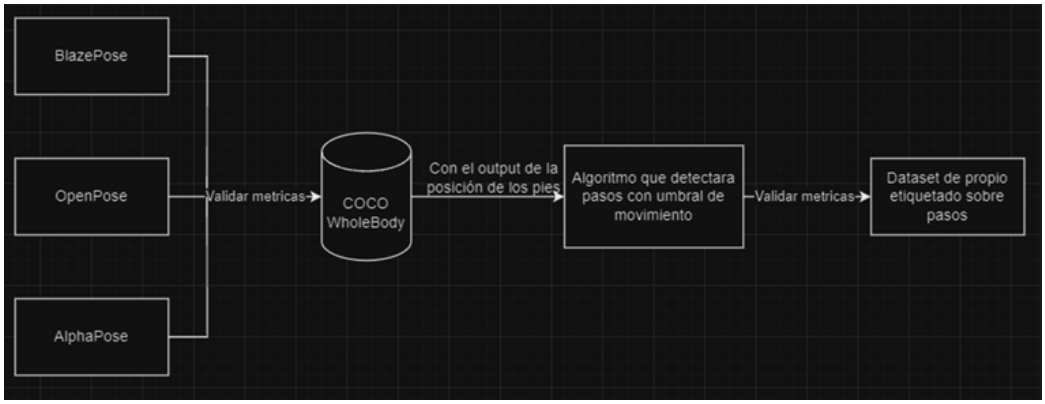

Figure 4: Propuesta de solución

3 Técnicas Utilizadas

En primer lugar, aclaramos que para este hito el algoritmo **AlphaPose** [2] ha sido descartado. Esta decisión ha sido tomada por diversas razones:

- El output de este algoritmo no mantenía una estructura fija, pues modificaba el orden de los keypoints dependiendo de lo que detectaba en cada frame, complicando la

interpretación de los resultados. Esto es particularmente problemático para nuestro objetivo, pues nos interesan exclusivamente los pies.

- Dificultad de implementación de este algoritmo. Los autores proponían configuraciones para el modelo las cuales eran ideales para nuestro plan; sin embargo, estas propuestas contenían problemas de incompatibilidad entre sí, problemas tales como "dimension mismatch" (discrepancias entre los keypoints esperados versus los obtenidos).

- Alto costo computacional. Durante la inferencia realizada con este modelo en el set de evaluación COCO2017, el computador donde se estaban realizando las pruebas, el cual posee componentes tales como la tarjeta gráfica RTX 3060, el procesador i7-12700H y 16 GB de memoria RAM, no logró ejecutar la inferencia de manera correcta. De hecho, para poder realizar esta, cerramos la mayoría de procesos del computador, procesos vitales tales como el explorador de archivos, por lo que llevamos al mínimo los gastos computacionales de otros procesos para poder correr el modelo.

Retomando los modelos que siguen para este hito, utilizamos **OpenPose** [3] ya que permite el tracking de forma nativa para una persona y utiliza tres keypoints en el pie de manera moderada. En esta técnica, la arquitectura propuesta codifica el contexto global, permitiendo un paso de análisis ascendente que mantiene una alta precisión y al mismo tiempo logra un rendimiento en tiempo real, independientemente del número de personas en la imagen. En la Figura 5 se aprecia el esqueleto propuesto por el modelo.

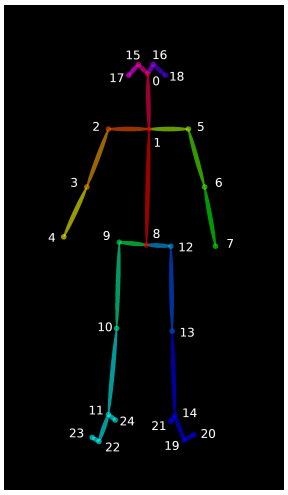

Figure 5: Estructura de OpenPose

La siguiente técnica es utilizando **BlazePose** [4], que realiza tracking de una persona con tres keypoints en el pie, mediante una arquitectura de una red convolucional ligera que produce 33 puntos clave en una persona. Además, permite el uso en tiempo real mediante keypoints como con mapas térmicos. La asociación de los 33 keypoints de BlazePose se puede ver en la Figura 6. En el caso de este modelo, se debe realizar un ajuste a los puntos detectados de tal manera que concuerden con la estimación que realiza COCO. Esto se realiza para poder obtener métricas y poder comparar ambos modelos.

Por último, hemos implementado un algoritmo de detección de pisadas. Este algoritmo se basa en las diferencias de posiciones (x,y) de los keypoints detectados del pie entre cada

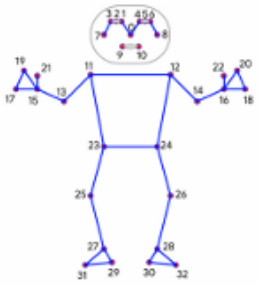

Figure 6: Estructura de BlazePose

frame. Para lograr esto, tenemos una función de suavizado la cual pondera frames vecinos para lograr una estimación sin grandes saltos de posición, simulando un tracking más fluido del pie. Respecto a la parte principal del algoritmo, utilizamos umbrales de máximo movimiento entre frames y cambios de dirección (si el pie cambia de dirección, necesariamente ocurre una pisada) para establecer un criterio que nos diga si ocurrió o no efectivamente un paso. Para más información, revisar en el repositorio del proyecto el archivo "Algoritmo-pisadas.md".

4 Datasets

En este trabajo utilizaremos dos sub-datasets que son extensiones de COCO (Common Objects in Context) [5], que incluye 80 categorías de objetos comunes como personas, vehículos, animales, entre otros. El primer dataset llamado foot-keypoint es de interés, ya que incluye anotaciones de keypoints que indican la ubicación de partes del cuerpo. Específicamente para el caso de estudio se etiquetan seis puntos clave en el pie, como se puede apreciar en la Figura 7.

Además, utilizaremos COCO-WholeBody [6], que es un subdataset de COCO con cinco mil anotaciones de cuerpo entero y contiene 133 puntos de referencia en todo el cuerpo, específicamente 68 en la cara, 42 en las manos, 17 en el cuerpo y 6 en el pie. Este presenta una estructura muy similar a foot-keypoint. Aclaramos que por el momento no hemos logrado tener acceso al dataset foot-keypoint, ya que los servidores de descarga se encuentran caídos, por lo que hasta ahora, solo hemos trabajado sobre WholeBody.

Para este hito, hemos decidido realizar una variante de COCO-WholeBody [6]. En esta variante lo que haremos será filtrar todas las imágenes donde aparezcan múltiples personas, dejando solo las imágenes que tengan una persona o ninguna. Recordemos que dejar imágenes sin personas es importante para tener imágenes de background. El filtro aplicado se encuentra en el repositorio del proyecto bajo el nombre "dataset/filter.ipynb".

5 Resultados y Métricas Obtenidos

Hemos logrado detectar cuando el sujeto del vídeo da pasos a medida que este se mueve. Para identificar el paso dado, se pinta el keypoint elegido como punto de evaluación de color verde; en el caso de OpenPose se utilizó el keypoint asociado al talón, mientras que en BlazePose se utilizó el tobillo. Aclaramos que esto es un parámetro fácilmente manejable.

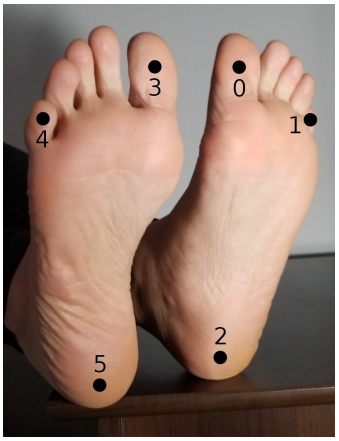

Figure 7: Keypoints pie de COCO

Para simplificar el análisis del vídeo, se guarda cada frame con su estimación de pose, incluyendo la detección de la pisada como una imagen. Además, se genera un archivo JSON que guarda el nombre de cada imagen correspondiente con su número de frame e indica cuál pie está realizando una pisada, siendo "Left" si solo pisa el pie izquierdo, "Right" si solo pisa el pie derecho, "Both" si ambos pies están pisando y "None" si ningún pie está pisando el suelo. Finalmente, el algoritmo desarrollado guarda el vídeo procesado con la detección de poses más información visual respecto a las pisadas.

En particular, con BlazePose, para realizar comparaciones en base de métricas, se realiza una conversión de puntos al formato COCO. En este código se incorporan funcionalidades con respecto al comportamiento de la persona del vídeo, indicando la direccionalidad del sujeto, ya sea movimientos hacia la derecha, izquierda, alejamiento y cercanía a la cámara. Sin embargo, este código considera un umbral entregado mediante el código que se debe adaptar a cada situación en específico.

Para ilustrar los resultados obtenidos, mostraremos los resultados con BlazePose, que se pueden ver en la Figura 8, donde a) indica que la persona está pisando, correspondiente al frame 23, mientras que b) la persona se encuentra en movimiento, frame 24. Además, esto se puede notar en el archivo JSON resultante, lo cual se puede ver como:

```
...
    {
        "frame_index": 23,
        "file_name": "./output/frames\\frame_0023.jpg",
        "stepDetection": true,
        "stepSide": "Both"
    },
    {
        "frame_index": 24,
        "file_name": "./output/frames\\frame_0024.jpg",
        "stepDetection": false,
        "stepSide": "None"
    },
```

```
    {
        "frame_index": 25,
        "file_name": "./output/frames\\frame_0025.jpg",
        "stepDetection": true,
        "stepSide": "Left"
    },
...
```


Este archivo JSON obtenido es similar al que se obtiene con OpenPose, ya que comparten la misma estructura base.

a) b)

Figure 8: Resultados obtenidos en BlazePose

Respecto a las métricas resultantes de este hito, al utilizar nuestra nueva versión filtrada de COCO-WholeBody, hemos notado un aumento de los valores en las métricas de OpenPose. Las métricas con la versión normal las podemos ver en la Figura 9. Respecto a la nueva versión, podemos notar una mejoría de 0.05 en el mAP promediado de todos los umbrales. Las métricas con la versión nueva las podemos ver en la Figura 10.

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets= 20 ] = 0.545
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets= 20 ] = 0.737
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets= 20 ] = 0.582
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets= 20 ] = 0.538
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets= 20 ] = 0.724
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 20 ] = 0.614
 Average Recall     (AR) @[ IoU=0.50      | area=   all | maxDets= 20 ] = 0.770
 Average Recall     (AR) @[ IoU=0.75      | area=   all | maxDets= 20 ] = 0.645
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets= 20 ] = 0.539
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets= 20 ] = 0.728
mAP todos los umbrales: 0.6249645640537065
```

Figure 9: Métricas obtenidas en OpenPose

6 Código

Los códigos desarrollados se pueden encontrar en el siguiente enlace: https://github.com/ignacio-barrera/pose_tracker.

```
DONE (t=0.01s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets= 20 ] = 0.613
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets= 20 ] = 0.781
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets= 20 ] = 0.644
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets= 20 ] = 0.627
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets= 20 ] = 0.712
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 20 ] = 0.687
 Average Recall     (AR) @[ IoU=0.50      | area=   all | maxDets= 20 ] = 0.815
 Average Recall     (AR) @[ IoU=0.75      | area=   all | maxDets= 20 ] = 0.713
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets= 20 ] = 0.628
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets= 20 ] = 0.714
mAP todos los umbrales: 0.6754723555629708
```

Figure 10: Métricas obtenidas en OpenPose

7 Trabajo Futuro

Para trabajo futuro, consideramos que se debe crear un nuevo dataset que nos indique si existe pisada o no por parte del jugador, para lograr evaluar y realizar una comparación más detallada del rendimiento de ambos modelos con el algoritmo de detección de pasos. Si bien fue un objetivo que nos planteamos al comienzo, decidimos concentrarnos en el algoritmo de detección de los pasos de la persona en el vídeo. Para realizar esta comparación, se deben implementar métricas coherentes con la estimación de pose de los modelos. Aunque se obtuvieron métricas, estas aún no son decisivas para elegir cuál modelo es mejor. Además, para poder obtener estas métricas coherentes, se debe incorporar una conversión de los puntos obtenidos en la estimación de pose.

Si bien existe una estimación sobre la dirección de movimiento de la persona utilizando BlazePose, esta estimación se puede complementar con el código ya implementado, de tal manera que la direccionalidad y contador de pasos sean más precisos e incluso puedan detectar cuando la persona pisa los puntos indicados para su movimiento. Este último punto debe ser implementado para ambos modelos de estimación de pose.

References

[1] S. Dubey and M. Dixit, "A comprehensive survey on human pose estimation approaches," *Multimedia Systems*, vol. 29, no. 1, pp. 167–195, 2023.

[2] H.-S. Fang, J. Li, H. Tang, C. Xu, H. Zhu, Y. Xiu, Y.-L. Li, and C. Lu, "Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[3] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7291–7299.

[4] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "Blazepose: On-device real-time body pose tracking," *arXiv preprint arXiv:2006.10204*, 2020.

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," *European conference on computer vision*, pp. 740–755, 2014.

[6] S. Jin, L. Xu, J. Xu, C. Wang, W. Liu, C. Qian, W. Ouyang, and P. Luo, "Whole-body human pose estimation in the wild," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*. Springer, 2020, pp. 196–214.