

MÁSTER UNIVERSITARIO EN CIENCIA DE DATOS



VNIVERSITAT
DE VALÈNCIA

TRABAJO DE FIN DE MÁSTER

**ANÁLISIS DE LA CARRERA A PARTIR DE DATOS DE
VÍDEO**

AUTOR:
CARLOS MORATA
GUIRADO

TUTOR:
EMILIO SORIA OLIVAS

JULIO, 2022



MÁSTER UNIVERSITARIO EN CIENCIA DE DATOS

TRABAJO DE FIN DE MÁSTER

ANÁLISIS DE LA ARRERA A PARTIR DE DATOS DE VÍDEO

AUTOR:

**CARLOS MORATA
GUIRADO**

TUTOR:

EMILIO SORIA OLIVAS

TRIBUNAL:

PRESIDENTE/A:

VOCAL 1:

VOCAL 2:

FECHA DE DEFENSA:

CALIFICACIÓN:

Resumen

La inteligencia artificial en el deporte está avanzando a una gran velocidad gracias a los avances tecnológicos y los muchos usos que se encuentran hoy en día. En este trabajo, hemos desarrollado un algoritmo de este campo, especializado en calcular las métricas que describen la forma de correr de una persona. Para ello, se ha realizado un estudio de los datos que hemos utilizado como entrada: una colección de 46 vídeos de 24 sujetos distintos grabados en cuatro sesiones distintas, junto a las mediciones de las unidades de medida inercial que tenían. Después, hemos hecho una revisión de los diferentes modelos de estimación de pose y sus tipos, para así poder seleccionar el más acorde a nuestro problema y realizar una estimación en esos vídeos.

A partir de las coordenadas obtenidas en el modelo de estimación de pose, calculamos la evolución temporal de los ángulos de las articulaciones a lo largo del vídeo. Posteriormente, utilizamos la evolución temporal de estos ángulos para calcular diferentes métricas de la carrera, haciendo uso de técnicas de análisis de señales. El algoritmo desarrollado calcula un total de cinco métricas: cadencia de pasos, tiempo de contacto de cada pierna en el suelo y ratio de tiempo en el aire de cada pierna. No obstante, podría modificarse en el futuro para calcular más métricas, ya que el cálculo se basa en las series temporales de las articulaciones de las piernas.

Finalmente, expondremos los resultados en tablas y haremos un análisis de estos. En el análisis, se ha calculado la correlación de los resultados con los valores reales conseguidos por los dispositivos de medida inercial y las métricas de error: raíz del error cuadrático medio y error absoluto medio. En estos resultados hemos observado que el algoritmo obtiene buenos resultados en la predicción de la cadencia de pasos. Sin embargo, no obtiene la misma calidad de predicción en los resultados del tiempo de contacto del pie en el suelo y del ratio de tiempo en el aire.

Índice general

1. Introducción	1
1.1. Unidades de medición inercial	2
1.2. Aplicaciones de la inteligencia artificial en el deporte	4
1.3. Estimación de pose	5
1.3.1. Estimación de pose 2D	5
1.3.2. Estimación de pose 3D	7
1.3.3. Datos	8
2. Datos y modelos	11
2.1. Los datos	11
2.2. Método	13
2.3. Estimación de pose	16
2.3.1. AlphaPose	17
2.3.2. MoveNet	19
2.3.3. BlazePose	21
2.3.4. OpenPose	22
2.4. Procesado de los datos	24
2.4.1. Cálculo de ángulos	25
2.5. Cálculo de métricas	25
2.5.1. Cadencia de pasos	27
2.5.2. Tiempo de contacto del pie en el suelo	29
2.5.3. Ratio de tiempo en el aire de cada pie	31
3. Resultados	33
3.1. Cadencia de pasos	34
3.2. Tiempo de contacto del pie en el suelo	39

3.3. Ratio de tiempo en el aire	48
4. Conclusiones y proyección futura	55
4.1. Conclusiones	55
4.2. Proyección futura	56
A. Código	59

Capítulo 1

Introducción

La biometría es el estudio que busca describir de forma única a las personas basándose en sus rasgos físicos o sus acciones; esto incluye factores como la forma de escribir, la voz, las huellas, la geometría de las manos, el iris o los rasgos faciales. Uno de estos factores que es único y cuyo análisis tiene muchas aplicaciones es la forma de hacer un ejercicio físico. En este trabajo nos centraremos en el análisis de la forma de hacer unos ejercicios en concreto: correr y remo.

El análisis de la marcha consiste en estudiar el movimiento del cuerpo humano, en especial el de las extremidades inferiores [1]. Algunos de los parámetros que se pueden medir son: longitud de la zancada, cadencia, velocidad, tiempo de contacto en el suelo, ratio de tiempo en el aire, etc. El estudio del movimiento humano se ha usado tradicionalmente para el diagnóstico médico, aunque también se puede utilizar para las ciencias del deporte.

Este tipo de análisis es muy utilizado en el ámbito del deporte, pues se emplea tanto para la rehabilitación de una lesión, como para reconocer los fallos en el rendimiento del atleta [2]. Muchas de las lesiones que padecen los atletas proceden de un mal desempeño biomecánico al realizar el ejercicio. De esta forma, al analizar el rendimiento del atleta de esta forma podemos prevenir parte de las lesiones que podría padecer el deportista. En concreto, aquellos deportes en los cuales sería más importante el análisis de marcha serían aquellos que contengan una alta

intensidad de correr o saltar.

El objetivo de este trabajo consiste en realizar un método para analizar la carrera a partir de datos de vídeo. Los datos de entrada para nuestro problema son vídeos, tomados por una sola cámara de diferentes personas corriendo en una cinta. Lo que se quiere conseguir a partir de estos vídeos son métricas que caracterizan la forma de correr de esa persona. Se busca hacer una tarea muy similar a la que realizan algunas unidades de medición inercial pero con la ventaja de necesitar solo una cámara de vídeo.

Este documento empieza, en esta sección, con una revisión de diferentes métodos que se utilizan para estudios biométricos. Entre estos métodos están incluidos algunos dispositivos como las unidades de medición inercial, así como diferentes usos que tiene la inteligencia artificial en el deporte.

En la sección 2, veremos varios algoritmos que se han utilizado para cumplir el objetivo propuesto. Asimismo, explicaremos de forma detallada los datos recogidos para poder elaborar este trabajo. Seguidamente, en la tercera sección, se presentarán los resultados que se han conseguido con el modelo utilizado.

Finalmente, en la sección 4, expondremos las conclusiones a las que se han llegado después de la realización del trabajo. Además, se plantearán posibles vías que se podrían seguir para continuar el trabajo.

1.1. Unidades de medición inercial

Las unidades de medición inercial (IMUs) son dispositivos que miden la velocidad, orientación, fuerza gravitatoria y campo magnético en las tres dimensiones de un sistema de coordenadas local [3]. Este dispositivo electrónico puede utilizar acelerómetros, giroscopios o magnetómetros, pudiendo combinarse entre ellos para conseguir más información.

Es necesario hacer una calibración de las unidades de medición para que el sistema de coordenadas sea una base ortonormal bien alineada

con el sistema de coordenadas exterior. Con este propósito se han ideado diferentes algoritmos que estiman la orientación del sensor respecto de un sistema de coordenadas global fijo.

En nuestro trabajo tomaremos como *ground truth* los resultados de un estudio realizado a unos corredores con unidades de medición inercial (IMUs) de marca RunScribe [4]. Estos IMUs son dispositivos pequeños portables que, pese a su reducido tamaño, son más sensibles y fiables que otros dispositivos anteriores de mayor tamaño. La introducción de este tipo de dispositivos más pequeños ha permitido hacer cambios en la forma de entrenamiento en muchos deportes. En la Figura 1 podemos ver el tamaño de estos dispositivos.¹



Figura 1: Imagen de dos unidades de medición inercial listos para utilizarse.

La información se recoge por estos dispositivos en forma de métricas, las cuales se pueden agrupar por: eficiencia, impacto, moción y derivados, entre otras. Entre ellas ha habido algunas métricas que nos ha parecido más interesantes para nuestro trabajo, las cuales son: Pasos por

¹Imagen de la página web de RunScribe. <https://runscribe.com/runscribe-validation-research-study/>

minuto, tiempo de contacto de cada pie en el suelo y ratio de cada pie en el aire.

1.2. Aplicaciones de la inteligencia artificial en el deporte

El uso de la inteligencia artificial en el deporte ha supuesto grandes avances ya sea en el entrenamiento o en las competiciones profesionales. Uno de los aspectos en los que más está ayudando es en el análisis del movimiento, tanto para la mejora en el entrenamiento como para la prevención de lesiones.

Las técnicas de aprendizaje máquina consisten en la creación de modelos que, a partir de datos, son capaces de generalizar comportamientos. Esto puede ser combinado con la biomecánica para múltiples usos [5]: mejorar la forma de recolección de datos de sensores de deporte, la información que se consigue por diferentes dispositivos, el procesamiento de datos o el uso de estos datos para conseguir mayor conocimiento del deporte o del riesgo de lesión.

Durante los años 90, la computación pasó a tener un papel más importante en la biomecánica y el deporte. Esto se debe a que pasaron de solo participar en la adquisición de datos y su procesamiento a tomar parte de la toma de decisiones. En esta década se iniciaron las primeras investigaciones en el uso de sistemas expertos, la mayoría de las veces desde el campo de la informática médica. Debido a estos primeros usos y a la acelerada evolución del campo de la computación se predijo que se acabaría utilizando con mucha frecuencia la inteligencia artificial en el deporte [6].

Uno de estos primeros sistemas expertos fue GAIT-ER-AID en el año 1992 [7]. Este sistema servía para el diagnóstico de patologías en la forma de andar. A partir de los datos producidos por la instrumentación de un laboratorio, hacía un procesamiento en el cual se reducía la cantidad de datos y se extraía información, con la que se usaba una estrategia de inferencia para el diagnóstico y podaba el árbol de diagnóstico.

El aprendizaje reforzado es una de las aplicaciones de la inteligencia artificial más utilizadas en el deporte. En este tipo de aprendizaje tenemos un agente de software que aprende qué acciones tiene que tomar a partir de la información de respuesta que recibe del entorno. Debido a que en el deporte es relativamente sencillo definir un entorno, un agente y una respuesta, esta metodología es la adecuada para muchos problemas de toma de decisiones en el deporte [8].

Con el auge de las redes neuronales y de la recolección de datos durante la última década, se desarrolló el campo de la visión artificial. Esta consiste en la adquisición, procesamiento y análisis de imágenes o videos mediante la computación. Sus usos más característicos son: clasificación de objetos, detección de objetos, segmentación de imagen y seguimiento de objetos a lo largo de un video.

1.3. Estimación de pose

Dentro del campo de la visión inteligente, la tarea de estimar la pose humana es de gran importancia debido al gran número de usos que tiene en áreas como la robótica, el cine, la salud y el deporte. En el ámbito del deporte, la estimación de pose ayuda en la prevención de lesiones y en el análisis del desempeño del ejercicio al identificar las diferentes articulaciones del cuerpo humano. La gran mayoría de modelos de estimación de pose utilizan redes neuronales y cada vez son más rápidos, ya que incluso pueden hacer la estimación a tiempo real.

1.3.1. Estimación de pose 2D

Los modelos de estimación de pose 2D tienen como objetivo calcular la posición de las diferentes articulaciones del cuerpo humano en una imagen o video. Es 2D porque no busca situarlo en un espacio tridimensional sino que busca estimar la pose en esta proyección bidimensional que es la imagen o el video de entrada al modelo [9].

Los modelos bidimensionales tienen la ventaja de ser menos costosos que los tridimensionales. A nivel de computación son más rápidos

ya que no precisan de cálculos tan pesados como los de tres dimensiones. Y a nivel de conseguir los datos, necesitan menos datos ya que se puede realizar esta estimación de pose con imágenes, o vídeos, de una sola cámara.

Estos modelos se pueden clasificar según si el modelo es de aprendizaje profundo o un modelo hecho a medida para la estimación de pose humana [10]:

- **Modelos de aprendizaje profundo:** son modelos que se utilizan para muchos tipos de problemas, como los basados en redes neuronales. Estos aprenden a realizar la estimación de pose al ver las relaciones entre las partes del cuerpo a partir de grandes bases de datos. Este tipo de modelos tiene más coste computacional, pero mucha mayor precisión.
- **Modelos a medida:** estos modelos son más tradicionales ya que son modelos a medida para aprender las relaciones entre las partes del cuerpo. En comparación con los modelos de aprendizaje profundo son menos costosos computacionalmente y necesitan menor cantidad de datos para ser funcional. Sin embargo, su precisión es mucho más baja que los otros modelos.

Si nos centramos en cómo actúan los modelos en cuanto a los pasos que ejecuta para detectar a las personas y realizar la estimación de pose, podemos separar los modelos en dos tipos [10]:

- **Dos etapas:** son modelos que realizan en dos etapas diferentes la detección de las personas y la estimación de pose. Dentro de este grupo de modelos hay dos subtipos: los modelos de arriba-abajo y de abajo-arriba. Los de arriba-abajo primero detectan a las diferentes personas que hay en imagen y después predicen la pose dentro de la caja contenedora de cada persona. Los modelos de abajo-arriba estiman las coordenadas de las articulaciones primero y a partir de ellas asignan estas coordenadas para componer a las diferentes personas. Los modelos de abajo-arriba sufren cuando hay más personas en la imagen, mientras que los opuestos tienen mayores problemas

al tener partes de personas detrás de otras personas u objetos.

- **Una etapa:** los modelos de una etapa buscan combinar los modelos de detección de persona y de estimación de pose en un solo modelo. Hacen esto para conseguir el objetivo de mejorar algunas eficiencias que tienen los modelos de dos etapas pero no han conseguido tener la misma precisión que estos modelos.

Los modelos de estimación de pose en dos dimensiones también se pueden clasificar según si se utilizan grafos o no [10]:

- **Basados en grafos:** asignan los puntos que hacen referencia a las articulaciones como si fuera un problema de programación dinámica. Realizan esto así para conseguir un menor coste computacional que los modelos que no se basan en grafos.
- **No basados en grafos:** realizan la tarea de asignación de las articulaciones de una forma más intuitiva, ya que lo asignan a cada persona basándose en el resultado de la detección. La precisión de estos modelos depende mucho de la precisión de la detección de las personas.

Para aquellos casos en los que se hace la estimación de pose sobre vídeos, muchas veces se utilizan redes neuronales recurrentes ya que pueden aprender de largas secuencias. Estas se utilizan en combinación con las convolucionales para conseguir hacer la estimación de pose a lo largo de una secuencia de imágenes.

1.3.2. Estimación de pose 3D

Al contrario que la estimación de pose bidimensional, la de tres dimensiones busca reconstruir las articulaciones del cuerpo humano en el espacio de tres dimensiones donde se sitúa la acción de la imagen o el vídeo.

El objetivo es conseguir unas imágenes que, en cada, pixel guarden la distancia de la cámara a dicho punto. Para conseguir la información de esta profundidad se utilizan distintos métodos [11]:

- **Cámaras en estéreo:** es un conjunto de cámaras, de mínimo dos cámaras que se han calibrado en un sistema de referencia tridimensional. Consiguen averiguar esta profundidad gracias la triangulación de las cámaras. Este método puede no ser del todo preciso para puntos en la escena con intensidades o colores parecidos.
- **Tiempo de vuelo:** se ilumina la escena con pulsos de luz y se mide el tiempo en el aire de estos pulsos reflejada por el cuerpo que se mide la profundidad. Este tiempo está correlacionado con la distancia al cuerpo desde la cámara. Puede tener errores relacionados con radiometría, geometría y variaciones de iluminación.
- **Escáneres 3D de luz estructurada:** consiste en proyectar un patrón de luz estructurada infrarroja y observar la deformación geométrica que sufre. Al compararlo con el patrón esperado en caso de que no hubiera ningún objeto puede calcular la profundidad del objeto. Sin embargo, pierde precisión al tener superficies transparentes o con reflejos.

Este tipo de métodos para realizar la estimación de pose en tres dimensiones requiere muchos recursos. No obstante, también se pueden utilizar otros métodos para que solo sea necesaria una cámara y, a partir de los datos bidimensionales de vídeo, conseguir una reconstrucción en tridimensional. Para ello, recientemente, se han creado modelos de aprendizaje profundo que son capaces de conseguir esta reconstrucción con gran precisión.

1.3.3. Datos

Como cualquier método de aprendizaje profundo, los modelos de estimación de pose 2D, necesitan una base de datos con una gran cantidad de imágenes para poder entrenarse y cumplir su función. Los conjuntos de datos más utilizadas para este propósito u otros relacionados con detección de objetos o segmentación de imagen son:

- **LSP (2010):** consiste en 2000 imágenes con la poses anotadas con 14 articulaciones. La mayoría de las fotos son de personas practi-

cando deportes de Flickr [12].

- **FLIC (2013)**: son en torno a 5.000 imágenes de películas. Este conjunto de datos tiene anotadas las posiciones de las articulaciones de casi 20.000 personas en sus imágenes [13].
- **MPII (2014)**: este conjunto de datos contiene alrededor de 25.000 imágenes con más de 40.000 personas con sus poses anotadas. En estas fotografías se puede ver a estas personas realizando 410 actividades diferentes; las imágenes fueron extraídas de YouTube [14].
- **COCO (2014)**: es el que contiene más datos con 330.000 imágenes con 1,5 millones de objetos identificados y 250.000 personas con sus poses anotadas. Este conjunto de datos ha sido construido colaborativamente por mucha partes, entre ellas empresas como Facebook o Microsoft [15]. En la Figura 2 podemos ver una imagen de este conjunto de datos, con las correspondientes poses asignadas.



Figura 2: Imagen perteneciente al conjunto de datos COCO con las poses asignadas a las personas.

Cada conjunto de datos anota de una forma diferente las articulaciones de las personas. Además, hay puntos clave como pueden ser las

orejas, los ojos o la nariz que están incluidos en algunos conjuntos de datos y en otros no.

En el caso de la estimación de pose en 3D hay muchas formas de recolección de los datos. Los conjuntos de datos más utilizados son:

- **Human3.6M (2014):** el conjunto de datos contiene 3,6 millones de poses humanas precisas. Los datos son recogidos por cámaras situadas en 4 puntos de vista diferentes y también 10 sensores de movimiento de alta velocidad. Las acciones realizadas en el conjunto de datos fueron hechas por 5 mujeres y 6 hombres teniendo mucha variabilidad en las formas del cuerpo [16].
- **HumanEva (2010):** consiste en 40.000 capturas de sensores de movimiento y vídeo desde diferentes puntos de vista, que como resultado son más de 250.000 imágenes en total. Las imágenes fueron tomadas a diferentes personas realizando acciones predefinidas [17].

Capítulo 2

Datos y modelos

Nuestro propósito es conseguir un modelo capaz de dar información sobre la forma de correr de diferentes personas a través de vídeos de ellos corriendo. En esta sección describiremos todas las etapas que se han seguido en el trabajo.

2.1. Los datos

El departamento de Fisioterapia de la Universitat de València fue el encargado de conseguir los datos utilizados durante el trabajo. Los datos consisten en unos vídeos de gente corriendo y en unos documentos que contienen los resultados del análisis de unas unidades de medición inercial. Durante el trabajo solo hemos hecho uso de aquellos vídeos que tienen su análisis de las IMUs asociado, puesto que hemos considerado que no podemos saber la precisión de nuestro modelo en aquellos casos que no tenemos la información sobre el análisis de la carrera.

Entre los vídeos que tienen su correspondiente información del análisis de la carrera se pueden distinguir dos grupos: 1) vídeos que se grabó en una sesión el 16 de julio de 2021; 2) vídeos que se grabaron en tres sesiones diferentes el 12, el 16 y el 17 de mayo de 2022.



Figura 3: Ejemplo de captura de un vídeo utilizado de julio de 2021.



Figura 4: Ejemplo de captura de un vídeo utilizado del 12 de mayo de 2022.

- **Julio 2021:** este grupo de vídeos consiste en 17 vídeos de 9 sujetos diferentes (6 varones y 3 mujeres) corriendo en una cinta. En todos los vídeos se situa la cámara enfrente a la izquierda y algo baja respecto del corredor, siendo siempre un ángulo muy parecido en todos los vídeos. Los vídeos tienen una duración aproximada de unos 30 segundos y hay representadas una gran variedad en formas de correr. En la Figura 3 podemos ver una captura de uno de los vídeos.
- **Mayo 2022:** durante los días 12, 16 y 17 de mayo de 2022 se grabaron un total de 29 vídeos más de 15 personas diferentes (4 varones y 11 mujeres). En estos vídeos, la cámara está situada de forma diferente respecto a el otro grupo de vídeos. La cámara está enfrente a la derecha y con un ángulo bajo respecto del corredor. Durante los tres días se grabaron los vídeos desde el mismo ángulo. Al igual

que en los vídeos de julio, tienen una duración de 30 segundos aproximadamente. Podemos observar un ejemplo de estos vídeos en la Figura 4.

Por otra parte, tenemos los resultados del análisis de la carrera realizado por las unidades de medición inercial. Estos datos son recogidos por unos dispositivos portables que se sitúan en los zapatos de los corredores. A partir de las mediciones de aceleración que realizan estos dispositivos el *software* de esta marca calcula diferentes métricas que describen la forma de correr del sujeto.

La cantidad de información en forma de métricas que aporta el *software* es muy grande. Además algunas de las métricas se miden por separado para cada pie. Las principales métricas que calcula las podemos ver en la Tabla 1.

De todas estas métricas las que vamos a considerar inicialmente para conseguir con nuestro modelo serán: pasos/minuto, tiempo de contacto en el suelo de cada pie y ratio de tiempo en el aire de cada pie.

2.2. Método

En la Figura 5 podemos observar los pasos que seguiremos para conseguir los resultados.

Lo primero que haremos será utilizar modelos de estimación de pose sobre los vídeos que tenemos. De esta forma, tendremos una estimación de la posición de las articulaciones y de como evolucionan estas posiciones a lo largo del vídeo. Para ello usaremos modelos de estimación de pose en dos dimensiones. Nos hemos decantado por este tipo de modelos porque tienen mucho menor coste computacional, mayor precisión y no necesitamos estimar la pose de una forma tridimensional. Al estar la cámara situada en un ángulo parcialmente lateral ya podemos ver con exactitud el movimiento del corredor desde nuestro punto de vista.

La estimación de pose nos devuelve las coordenadas en la imagen de la estimación de las diferentes articulaciones o puntos clave del cuerpo

Métrica	Tipo de métrica	Definición	Rango típico
Ritmo	-	Velocidad a la que se mueve el corredor	-
Longitud de zancada	-	Distancia entre dos sucesivos apoyos del mismo pie	1,5 m - 3 m
Ratio de pasos	Eficiencia	Número de pasos que realiza el corredor por minuto	160 pasos/min - 200 pasos/min
Ratio en el aire	Eficiencia	Ratio de tiempo que pasa el pie en el aire en cada paso	0 % - 50 %
Tiempo de contacto	Eficiencia	Tiempo que pasa el pie en contacto con el suelo	190 ms - 400 ms
Impacto	Impacto	Impacto total realizado por paso	6 Gs - 19 Gs
Impacto con el suelo	Impacto	Componente vertical del impacto. Relacionado con el impacto con el suelo	5 Gs - 14 Gs
Frenada	Impacto	Componente horizontal del impacto. Relacionada con las fuerzas de frenado en el paso	4 Gs - 13 Gs
Tipo de zancada	Moción	Zona del pie en la que se recibe el impacto. Desde talón al antepié.	3 (talón) - 12 (antepié)
Excusión de pronación	Moción	Rango total de movimiento angular que realiza el pie entre la mínima y máxima pronación	-2º - -20º
Velocidad máxima de pronación	Moción	La máxima variación angular del pie	200º/s - 900º/s
Simetría	Simetría	Mide la simetría entre ambos pies en eficiencia, impacto y moción	-

Tabla 1: Métricas de la carrera calculadas con el software de las unidades de medición inercial.

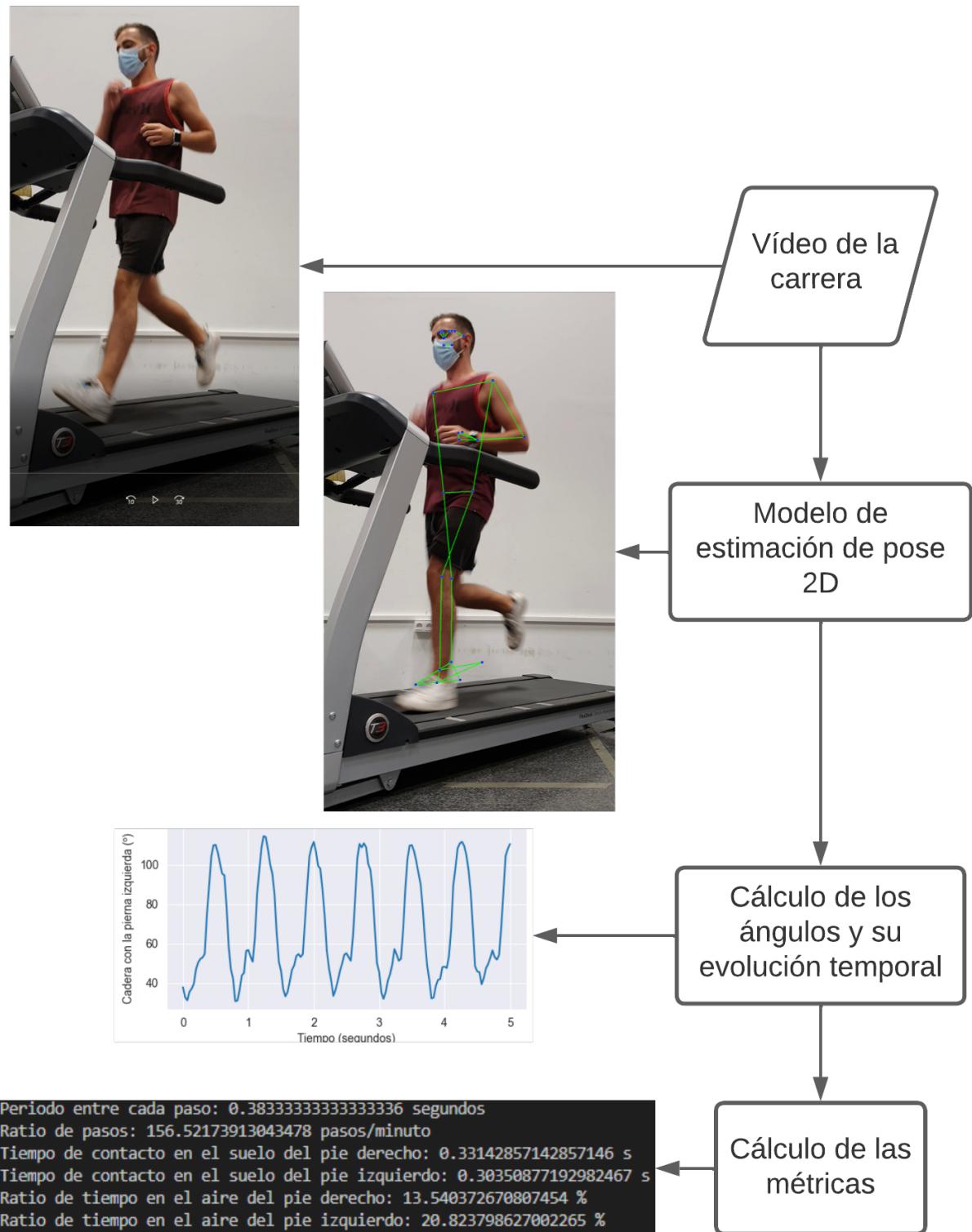


Figura 5: Esquema del protocolo seguido desde la adquisición de los datos hasta el cálculo de los resultados.

realizada por el modelo. A partir de estas coordenadas podemos obtener los vectores que unen estas articulaciones, las cuales dibujan la pose del corredor. Esto es útil ya que, a partir de estos vectores, es posible calcular los ángulos de las articulaciones. Aunque estos ángulos son de la proyección bidimensional en la imagen y no los ángulos reales del corredor, son una buena aproximación por el ángulo de la cámara.

A partir de este punto lo que buscamos es calcular las métricas del análisis de la carrera. Como tenemos los ángulos de las articulaciones en cada una de las imágenes que componen el vídeo, podemos calcular la evolución temporal del ángulo de las articulaciones. Esto lo haremos a través de la evolución de los ángulos de las articulaciones a lo largo de la sucesión de imágenes.

A la hora de calcular las métricas de la carrera, no tenemos una gran cantidad de datos ya que solo tenemos 17 vídeos con las métricas correspondientes. Por ello, realizar un modelo de aprendizaje máquina podría no ser la mejor opción ya que no tendríamos suficientes datos para entrenar el modelo y que este sea preciso. Debido a esto, es una buena opción utilizar herramientas clásicas de análisis de variables y evitar otras de aprendizaje máquina que necesiten datos para entrenar.

2.3. Estimación de pose

Una vez tenemos los datos, el siguiente paso es utilizar algún método de estimación de pose 2D para conseguir una estimación de donde se localizan las diferentes articulaciones o puntos clave del cuerpo en la imagen. Para ello, probaremos algunos modelos de los más potentes y que más se ajusten a nuestro problema.

Las propiedades más valoradas del modelo son las siguientes:

- **Precisión:** como es lógico, en cualquier problema de regresión como es este, la precisión es extremadamente importante. En este caso, hemos decidido tener en cuenta dos aspectos respecto a la precisión: 1) que las métricas de la carrera que calculemos sean lo más precisas

posibles respecto del *ground truth*; 2) que los ángulos calculados por el modelo de estimación de pose sean lo más precisos posibles y que no se vean demasiado afectados por problemas de oclusión (que algún elemento del cuerpo humano no esté visible para la cámara en un determinado momento).

- **Puntos clave:** cada modelo de estimación de pose considera distintos puntos clave. En nuestro problema hay algunos puntos clave que son más importante que otros, en concreto aquellos de las piernas y los pies. Los ángulos que hemos utilizado son los de la rodilla y la cadera pero de cara al futuro sería importante otros ángulos como el tobillo. Alguno de los modelos no señalan el talón del pie, por lo que no se podría calcular con tanta precisión el ángulo del tobillo. Es por esto que para nuestro problema se valora positivamente en los modelos que calculen la posición de la mayor cantidad de puntos clave en las piernas y los pies.
- **Rapidez:** la rapidez con la que se pueden obtener los resultados siempre es un factor a tener en cuenta, pero este puede ser más o menos importante dependiendo del propósito del modelo. En nuestro caso, lo consideramos un factor importante pero menos que los dos citados anteriormente. Creemos que puede ser importante porque un posible futuro uso del modelo podría ser en dispositivos móviles y sería interesante obtener los resultados en un tiempo que no sea excesivo.

A continuación, describiremos los modelos de estimación de pose 2D que hemos utilizado en este trabajo: *MoveNet*, *AlphaPose*, *BlazePose* y *OpenPose*. También haremos un análisis sobre cuáles serían los mejores modelos para nuestro problema.

2.3.1. AlphaPose

El modelo de estimación de pose es un modelo de aprendizaje profundo basado en *transformers*. El proceso que sigue es en dos etapas de arriba-abajo, esto significa que primero utiliza un modelo de detección de humanos y después realiza la estimación de pose. Este modelo tiene

la particularidad de realizar la tarea de estimación de pose de una forma precisa, incluso cuando la detección ha sido imprecisa [18].

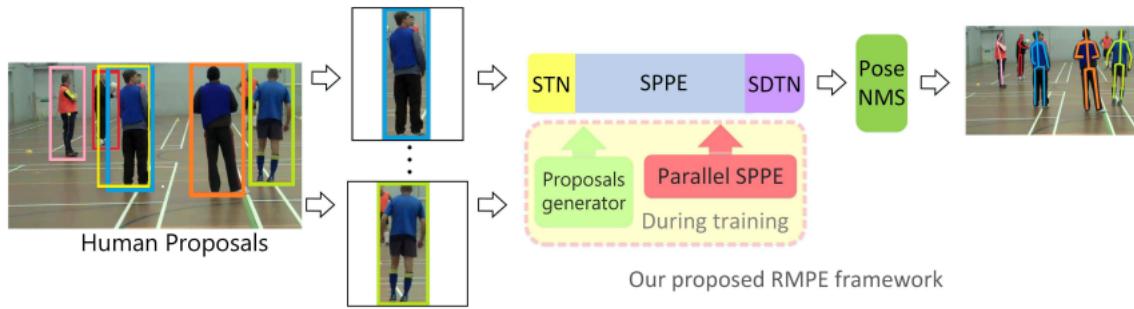


Figura 6: Proceso del modelo AlphaPose.

El proceso que sigue el modelo lo podemos ver en la Figura 6. Las propuestas de detección de humanos se introducen en un sistema llamado “Symmetric STN + SPPE”. El sistema consiste en un *Spatial transformer network* que extrae propuestas de detección de humanos de gran precisión realizando una transformación de coordenadas. En este espacio el SPPE (estimador de pose unipersonal) realiza la estimación de pose. Finalmente, el *Spatial detransformer network* realiza la transformación inversa para volver al espacio de coordenadas original [18].

Únicamente para la fase de entrenamiento del modelo, se utiliza un estimador de pose unipersonal paralelo como regularizador. Este módulo ayuda a que el STN se centre en el área correcta y realice mejor su tarea. Finalmente, se introducen los datos generados por el *Symmetric STN + SPPE* en un módulo llamado *Pose non-maximum suppression* que elimina estimaciones de pose redundantes que se producen inevitablemente en las detecciones de humanos.

Este modelo fue probado en los principales conjuntos de datos de estimación de pose que son MPII y COCO. En MPII obtuvo los mejores resultados de estimación de todas los puntos clave, excepto la cabeza. Cabe señalar que destacó especialmente en la mejoría de estimación de algunas articulaciones difíciles de estimar como las muñecas, los codos, los tobillos y las rodillas. En el conjunto de datos COCO tuvo muy buenos resultados también, pues obtuvo una precisión parecida a los otros

métodos actuales.

AlphaPose es una buena opción sobre todo por la precisión que tiene, ya que es uno de los modelos más precisos que hay. Sin embargo, tiene algunos inconvenientes para la aplicación en nuestro problema: es un modelo mucho más lento que otros, llegando a tener 30 veces el coste computacional que necesitan otros modelos algo menos precisos. También hay que tener en cuenta que no señala el talón, lo cual implica que, de cara al futuro, limita la cantidad de puntos clave de la pierna y el pie que se pueden utilizar para calcular métricas.

2.3.2. MoveNet

MoveNet es un modelo de estimación de aprendizaje profundo que realiza su tarea en dos etapas. Es un modelo de abajo-arriba, lo que significa que primero detecta las partes de las personas en la imagen y después las agrupa formando los diferentes cuerpos. En la Figura 7 podemos ver la arquitectura de MoveNet [19].

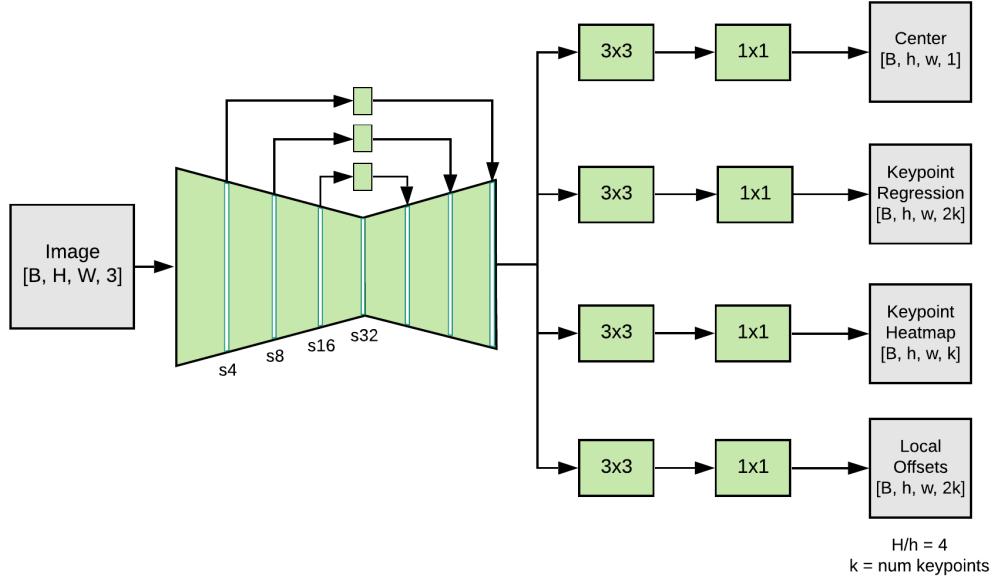


Figura 7: Arquitectura del modelo MoveNet.

El modelo contiene un primer módulo que es el extractor de características, que es la red neuronal convolucional MobileNetV2 con unas modificaciones para permitir un mapa de características de mayor resolución.

El otro módulo del modelo es un conjunto de cuatro cabezas de predicción que realizan la siguientes estimaciones: centro geométrico de las personas; conjunto completo de puntos clave de las personas; localizaciones de todos los puntos clave, independientemente de la persona; y cálculo exacto del punto clave a partir de un mapa de calor calculado por las otras cabezas de predicción.¹

MoveNet se entrenó con los conjuntos de datos COCO y Active (un conjunto de datos interno de Google). Se decidió utilizar Active también porque COCO no es apropiado para la estimación de pose en personas realizando ejercicio ya que no tiene demasiadas imágenes de este estilo. Active, en cambio, es una base de datos de vídeos de personas realizando ejercicio extraída de YouTube.

El modelo tiene dos variantes: *Lightning* y *Thunder*. *Lightning* está especializado para aplicaciones que necesitan mayor velocidad de inferencia a cambio de sacrificar un poco la precisión. No obstante, *Thunder* es más lento, pero tiene una precisión muy alta. Aun así, ambos modelos tienen muy buena precisión y una velocidad por encima de los 30 FPS, por lo que se puede utilizar a tiempo real.

Este modelo no tiene tanta precisión como otros modelos, pero, a cambio, tiene una gran velocidad y, por tanto, podría ser una muy buena opción para aplicaciones deportivas. Sin embargo, el único punto clave que señala en el pie es el tobillo y no muestra otros puntos del pie como algún dedo o el talón y no muestra dificulta su aplicación en nuestro problema.

¹Una cabeza de predicción es una terminación del modelo de aprendizaje profundo cuyo objetivo es devolver una predicción

2.3.3. BlazePose

El modelo de estimación de pose BlazePose tiene una arquitectura formada por redes neuronales convolucionales, que podemos ver en la Figura 8 [20]. Es un modelo en dos etapas, cuyo orden es de arriba-abajo.

El proceso que sigue el modelo empieza por un algoritmo de detección de personas. Este detecta el rostro de las personas y, a partir de este, detecta otras partes de las personas y construye la marco que contiene a la persona en la imagen. Una vez detectada la persona en el vídeo no se vuelve a computar esta parte hasta que la red neuronal indica que no hay ningún humano.

La información obtenida en el detector de personas se introduce en la red neuronal del modelo que realizará la estimación de pose. En la Figura 8 podemos observar la estructura que tiene esta red neuronal. Usa una combinación de mapa de calor, *offsets* y regresión, aunque la relativa a mapa de calor y offsets solo se utiliza durante el entrenamiento de la red.² Esto se hace así para supervisar el *embedding* de bajo coste computacional que tiene. La red neuronal consiste en un pequeño *encoder-decoder* del mapa de calor y en un *encoder* de regresión. Cabe destacar que utilizan conexiones que saltan capas para así tener en cuenta tanto grandes como pequeñas características de la imagen.

²Además de las conexiones usuales, tiene otros dos tipos: las *skip connections*, que saltan capas de la red para mantener un balance entre características de alta y baja escala; y las *stop gradient connections*, cuyo objetivo es que los gradientes del *encoder* de regresión no se retropropaguen hacia las características ya entrenadas del *encoder-decoder* del mapa de calor

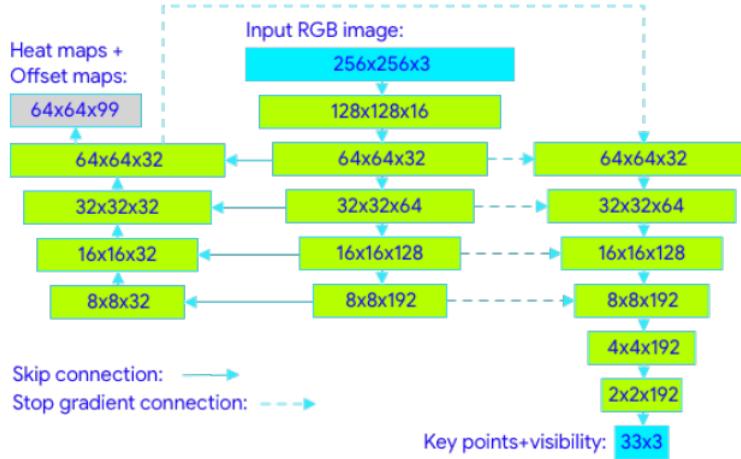


Figura 8: Arquitectura de la red neuronal del modelo BlazePose.

El modelo obtuvo muy buenos resultados sin llegar a ser los mejores. Sin embargo, tiene menor coste computacional que aquellos modelos que obtuvieron mejores resultados, como por ejemplo OpenPose. En relación a los modelos mencionados anteriormente, este modelo si señala varios puntos clave del pie: tobillo, talón y dedo índice del pie. Debido a todo lo anteriormente mencionado, este modelo es una de las mejores opciones a tener en cuenta para nuestro problema.

2.3.4. OpenPose

OpenPose es un modelo de estimación de pose 2D basado en aprendizaje profundo [21]. Desarrolla su tarea en dos etapas siendo estas de abajo-arriba. Tiene la particularidad de utilizar campos de afinidad parcial (PAFs) para medir el nivel de asociación entre partes. Los PAFs son un conjunto de campos vectoriales bidimensionales que codifican la localización y orientación de las extremidades de las personas por toda la imagen.

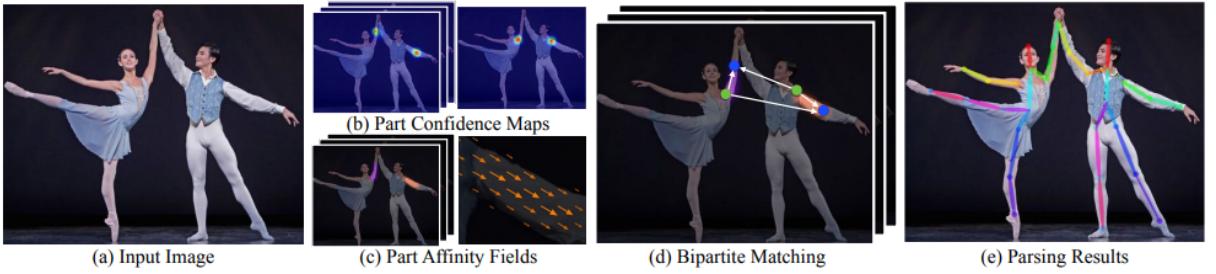


Figura 9: Representaciones del proceso que sigue una imagen durante el proceso de OpenPose: a) Imagen de entrada, b) mapas de calor de los diferentes puntos clave de la pose, c) campos de afinidad parcial (PAFs), d) análisis de las relaciones entre las posibles partes del cuerpo y e) unión de las partes del cuerpo para formar la estimación de pose completa.

En la Figura 9 podemos ver en qué consiste el método utilizado por OpenPose. La imagen inicial se introduce en la red en la que en las primeras capas se hace una extracción de características inicial. Una vez realizado, la red se divide en dos redes paralelas que se centran en conseguir diferentes resultados para después unirlos. La primera rama se encarga de calcular mapas de confianza para cada una de las partes de la pose humana, mientras que la otra predice los PAFs que indican el nivel de asociación entre partes.

Después de estas dos partes separadas, se combinan los resultados de forma que los campos de afinidad parcial unen las articulaciones calculadas en los mapas de confianza. Una vez realizadas estas uniones, podemos componer esto para formar el cuerpo humano completo y, de esta forma, realizar la estimación de la pose completa de las diferentes personas que haya en imagen. El modelo separa algunas tareas de la detección en otros modelos especializados, como en el caso de los pies, las manos y la cara. De esta forma, consigue mayor velocidad y precisión en esas zonas.

El modelo ha sido probado en tres bases de datos diferentes: MPII, COCO y una base de datos propia con imágenes de pies. En las bases de datos de MPII y COCO consigue unos de los mejores resultados en comparación con otros métodos *state-of-the-art*, siendo más rápido que todos ellos, incluso 6 órdenes de magnitud en algunos casos. Al haber

utilizado un modelo específico para los pies, se consigue una precisión muy alta en esta zona sin tener un alto coste computacional.

En general, este modelo y el modelo BlazePose son los que mejor se ajustan a nuestro problema. Este modelo consigue una gran precisión (sobre todo en los pies y en las piernas) sin tener un alto coste computacional, por lo que es rápido. En cambio, BlazePose tiene una precisión algo menor pero es algo más rápido. Otra aspecto a tener en cuenta es que OpenPose se ve menos afectado que BlazePose al tener alguna situación en la que alguna parte del cuerpo humano se encuentra momentáneamente detrás de un objeto o de otra parte. Esto es el caso de personas corriendo, pues la pierna que se encuentra opuesta a la cámara pasa por detrás de la que se encuentra delante de esta. Aun así, ambos modelos son buenos candidatos para usarse en este problema.

2.4. Procesado de los datos

Para realizar el procesado de los datos emplearemos el lenguaje de programación Python. Este lenguaje de programación tiene la ventaja de tener muchas las librerías especializadas cada una en un tipo de proceso. Las principales librerías que he utilizado son: Numpy (uso general); Matplotlib y Seaborn (visualizaciones); Scipy (cálculo); Statsmodels (análisis de señales); Pandas (tratar conjuntos de datos); Opencv (tratamiento de imágenes); y Mediapipe (modelos de aprendizaje máquina).

Como resultado de aplicar los modelos de estimación de pose de los vídeos obtenemos unos archivos de tipo JSON, uno por cada frame del vídeo, tenemos las coordenadas en la imagen de cada uno de los puntos clave registrados por el modelo de estimación. En el caso del modelo BlazePose, como está incluido en la librería MediaPipe, no es necesario este paso intermedio de guardar los resultados en un JSON. De esta forma, con BlazePose podemos utilizar directamente el vídeo original en Python para calcular las métricas.

A partir de estos resultados, ya sea el archivo JSON o los resultados de la librería MediaPipe dentro de Python utilizamos la librería Pandas

para almacenar los resultados en un solo *dataframe*. Una vez tenemos los datos bien organizados, calcularemos los ángulos de las articulaciones.

2.4.1. Cálculo de ángulos

A partir de las coordenadas en las que se encuentran los puntos clave podemos calcular los vectores que los unen, creando así los vectores que definen la orientación entre las articulaciones. Estos vectores se calculan de una forma sencilla:

$$\vec{u} = (q_x - p_x, q_y - p_y) \quad (2.1)$$

Después de haber unido el cuerpo entero calculando estos vectores, ya podemos calcular los ángulos de las articulaciones. Los ángulos los calculamos haciendo uso de la fórmula del ángulo entre dos vectores:

$$\cos(\alpha) = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| \cdot |\vec{v}|} \quad (2.2)$$

Para aquellos ángulos que están entre 0° y 180° se puede aplicar de forma inmediata el *arccos* al resultado y obtener el ángulo, como es el caso del tobillo o la rodilla. Sin embargo, para el caso del ángulo entre la pierna y el torso, si realizamos el mismo proceso obtendríamos ángulos menores de 180° siempre. Si sabemos que el ángulo está entre 90° y 270° , una posible solución es cambiar uno de los vectores por uno perpendicular y sumar 90° al resultado del ángulo. De esta forma, ya tendríamos bien calculados todos los ángulos de las articulaciones. En la Figura 10 podemos contemplar las series temporales de los ángulos de las articulaciones.

2.5. Cálculo de métricas

Ahora que ya tenemos las evoluciones temporales de los ángulos podemos empezar el proceso de cálculo de métricas. La métricas que

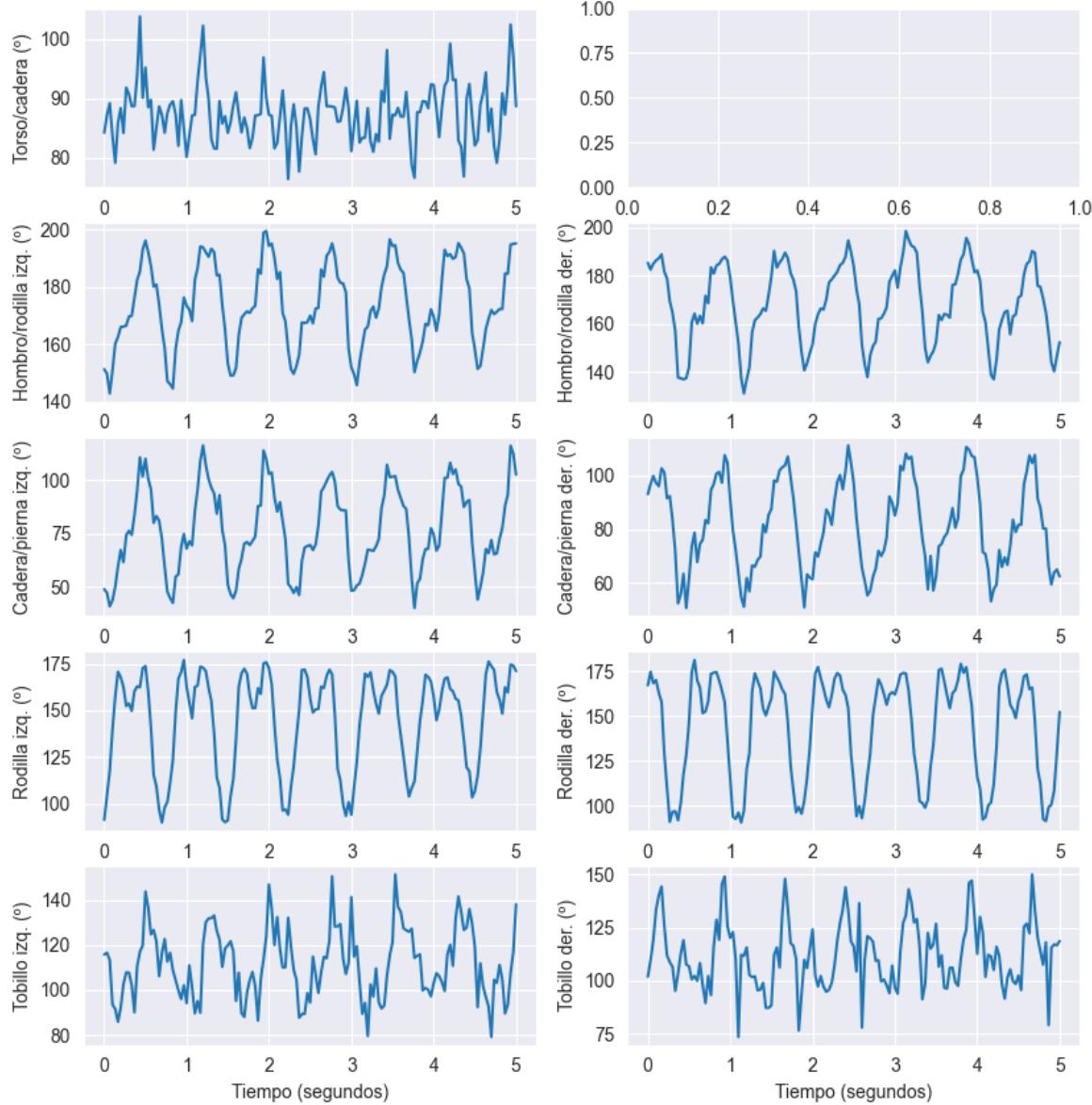


Figura 10: Evolución temporal de los ángulos calculados. (1^a fila) Ángulo entre la cadera y el torso, (2^a fila) ángulos entre los hombros y las rodillas correspondientes, (3^a fila) ángulos entre la cadera y cada una de las rodillas, (4^a fila) ángulos de las rodillas y (5^a fila) ángulos de los tobillos

hemos conseguido calcular a partir de estas series temporales son: pasos/minuto, tiempo de contacto en el suelo de cada pie y ratio de tiempo en el aire de cada pie. Cada una de las métricas las calcularemos de una forma diferente.

El código referente a esta parte se realizó entre la primera sesión y la segunda. Aunque el código esté ideado para que funcionara con, prácticamente, cualquier ángulo de la cámara (exceptuando un ángulo frontal respecto del corredor) afectará el hecho de que se realizara con un solo ángulo. Aun así, esta realizado con la idea de identificar los puntos clave en las redes neuronales sea cual sea el ángulo de la cámara respecto del corredor.

2.5.1. Cadencia de pasos

Las series temporales de los ángulos tienen una componente periódica de gran importancia. Cada vez que un pie realiza un paso es un ciclo que se repite tantas veces como pasos hace. De un ciclo a otro puede haber variaciones, pero durante una misma carrera de una misma persona estos ciclos no suelen variar demasiado.

Debido a este componente cíclico, se ha decidido hacerlo a través de un análisis espectral. A partir de una serie temporal del ángulo de una articulación, en nuestro caso del ángulo de la pierna derecha, aunque podría utilizarse cualquier otro de las piernas, calculamos la función de autocorrelación, la cual podemos ver en la Figura 11. La función de autocorrelación parcial consiste en medir la correlación entre los datos de la serie temporal respecto de los mismos pero k *lags* antes.

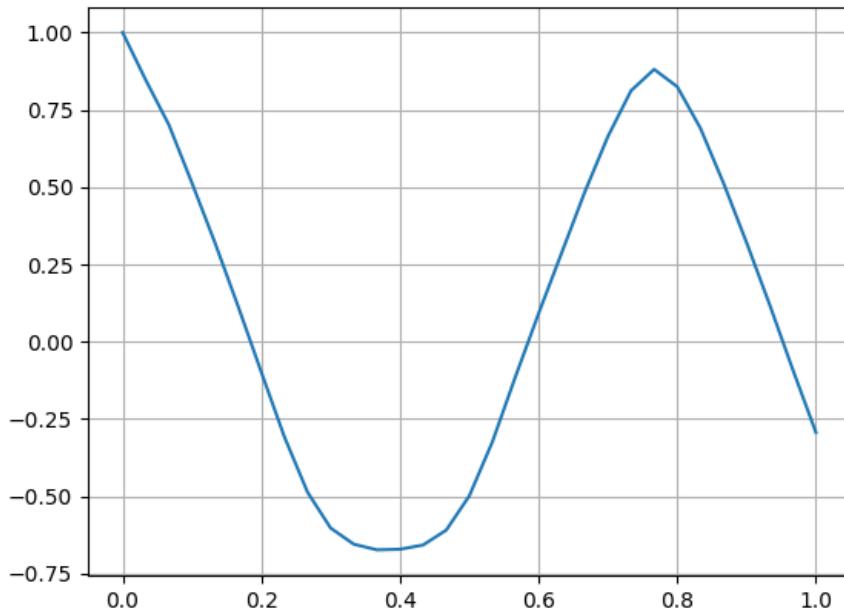


Figura 11: Función de autocorrelación de la serie temporal correspondiente al ángulo de la pierna derecha respecto de la cadera. Eje x: Segundos.

Una vez calculada esta función, seleccionamos el segundo máximo local que encontramos en la función, ya que indica que t segundos más tarde la función original se comporta de una forma similar. Esto lo realizamos porque el objetivo es buscar el periodo medio que hay entre cada ciclo de la serie temporal.

Cuando calculamos la función de autocorrelación tenemos que tener en cuenta que las series temporales son discretas y por lo tanto hay un tiempo mínimo entre cada valor de este *eje x*. En nuestro caso este tiempo entre cada valor viene definido por el número de fotogramas por segundo que en los vídeos utilizados es de 30 fotogramas por segundo. Esto implica un inconveniente que es que el resultado de este máximo de la función de autocorrelación solo puede ser un múltiplo de $33,3\text{ ms}^3$.

³30 fotogramas por segundo implica que hay un fotograma cada $33,3\text{ ms}(1/30s)$

2.5.2. Tiempo de contacto del pie en el suelo

El tiempo de contacto del pie en el suelo es una de las métricas más importantes en el análisis de la carrera, porque es una medida relacionada con la eficiencia de la carrera. Dependiendo de la velocidad a la que vaya el corredor se puede definir ciertos rangos del tiempo de contacto del pie en el suelo para tener mayor eficiencia en cada zancada. Por esto, hemos decidido incluir el cálculo de esta métrica en el trabajo.

El tiempo de contacto del pie en el suelo se define por el tiempo que pasa desde que cualquier parte del pie entra en contacto con el suelo hasta que deja de tener contacto. Normalmente es el talón el primer en tener contacto con el suelo y los dedos los últimos en tenerlo antes de que despegue el pie del suelo. En la Figura 12 tenemos una representación de las etapas que recorre el pie mientras está en contacto con el suelo.⁴

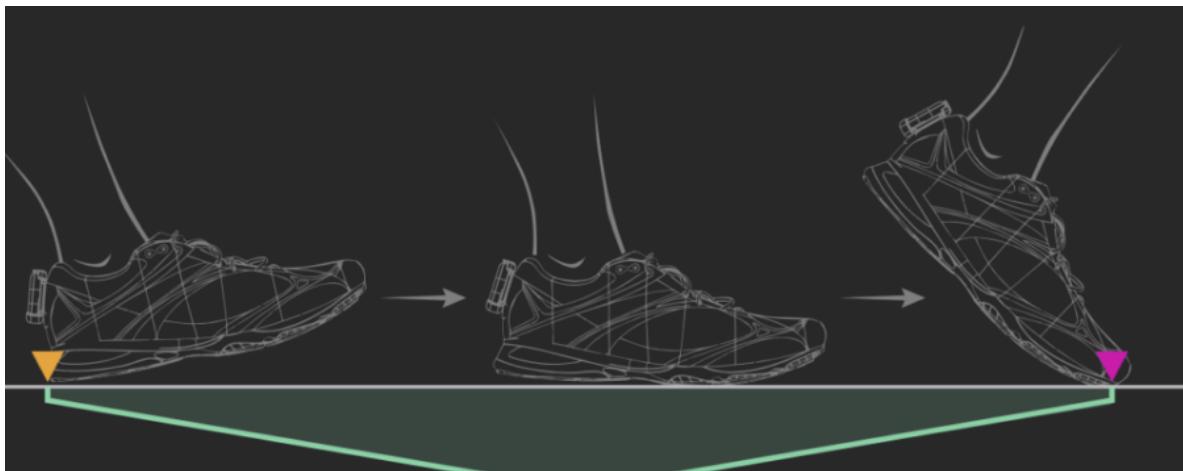


Figura 12: Fases del pie durante el tiempo de contacto con el suelo.

Para nuestro problema, los datos que tenemos de entrada, para poder calcular esta métrica, son las evoluciones temporales de los ángulos de las articulaciones. Por lo tanto, hay que relacionar estos ángulos con el tiempo de contacto en el suelo. Esta fase se podría haber realizado de formas muy distintas, pero en nuestro caso hemos utilizado solo el ángulo de la rodilla.

⁴Imagen tomada de la página web de RunScribe. <https://runscribe.com/metrics/>

Hemos escogido el ángulo de la rodilla ya que tiene los dos puntos de inicio del tiempo de contacto y de final muy marcados [22]. El pie inicia el contacto en un máximo de ángulo de la rodilla, es decir, cuando la rodilla está extendida. Después de esto, la rodilla se flexiona ligeramente mientras el pie está en contacto hasta que llega un nuevo máximo parecido al anterior. Este segundo máximo indica el fin del contacto, porque en este punto la pierna se vuelve a extender. Finalmente, hay un periodo de tiempo que la rodilla se flexiona (mucho más que antes), que es el tiempo que el pie no está en contacto con el suelo. Estas fases están representadas en la Figura 13.

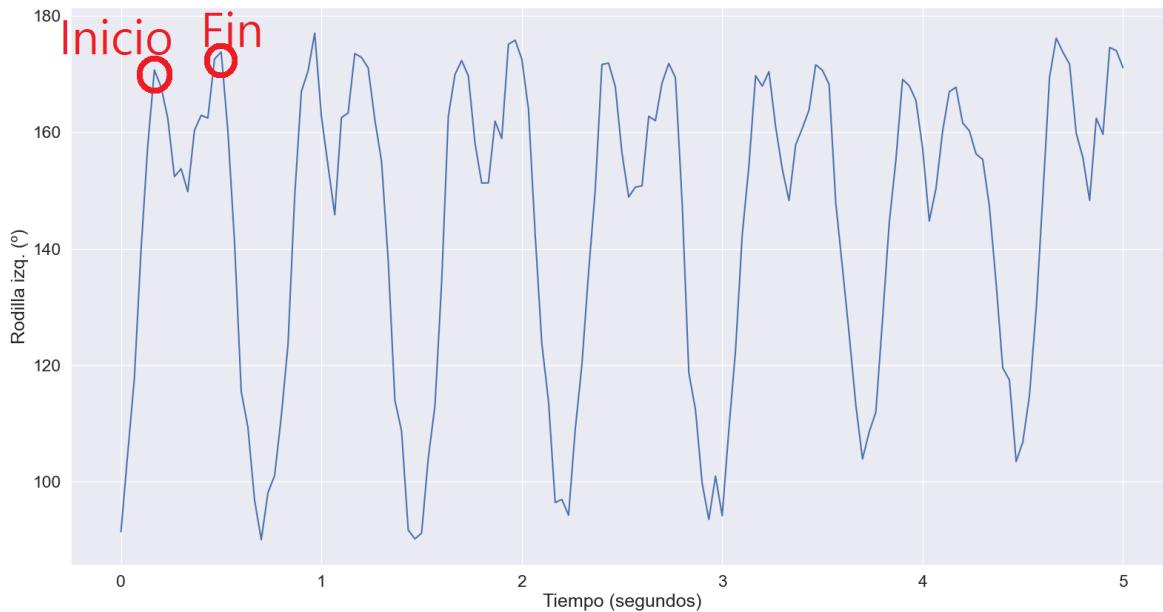


Figura 13: Evolución temporal del ángulo de la rodilla izquierda con los dos puntos característicos del tiempo de contacto en el suelo marcados.

Calcular de esta forma el tiempo de contacto en el suelo tiene un gran inconveniente: es necesario que el patrón de máximos y mínimos esté marcado. El cálculo será erróneo si hay una gran distorsión en este patrón y no se puede definir claramente el mínimo global, el mínimo local y los dos máximos locales.

En el futuro, esta forma de cálculo debería revisarse para poder tener mayor exactitud. Se podrían realizar muchas variaciones ya que

la cantidad de información que tenemos en todas las series temporales calculadas, correspondientes a los ángulos de las articulaciones, es muy grande y podría afrontarse el problema de maneras muy diversas.

2.5.3. Ratio de tiempo en el aire de cada pie

El tiempo en el aire es la cantidad de tiempo que el corredor pasa sin tocar el suelo entre cada zancada. Por lo tanto, es el concepto opuesto al de tiempo de contacto en el suelo. La Figura 14 muestra un gráfico explicativo de esta relación entre métricas. Este tiempo en el aire se calcula para cada pie, para así ver si hay asimetrías en la carrera.

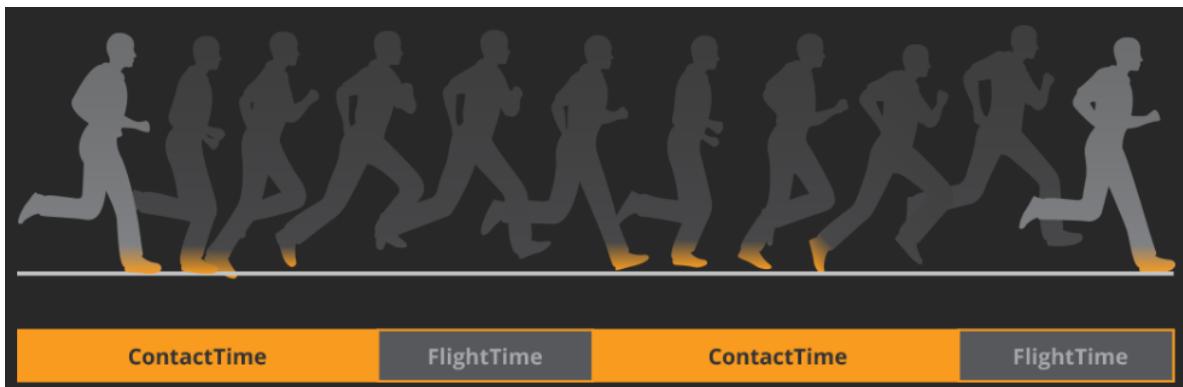


Figura 14: Ilustración explicativa de las fases de la carrera con los tiempos de contacto y el tiempo en el aire.

Debido a esta relación, utilizaremos el resultado obtenido para el tiempo de contacto en el cálculo de esta métrica. De esta forma, calcularemos el ratio de tiempo en el aire de cada pie con la siguiente ecuación:

$$\text{Ratio de tiempo en el aire} = 1 - \frac{\text{Tiempo de contacto en el suelo}}{\text{Periodo entre cada paso}} \quad (2.3)$$

Al calcularse siguiendo el resultado de la métrica anterior, tendremos los mismos inconvenientes en esta métrica que en el cálculo del tiempo de contacto del pie. Sin embargo, es natural calcular de esta forma el ratio de tiempo en el aire y, por ello, no creo que sea un error utilizar este método.

Capítulo 3

Resultados

En esta sección expondremos los resultados que hemos obtenido haciendo uso de los modelos y el método de trabajo seguido. Finalmente, los modelos empleados han sido el BlazePose y el OpenPose. Hemos utilizado solo estos dos porque, de los estudiados, eran los que más se adecuaban a nuestro problema, y por tanto, los que mejores resultados podrían obtener. Otra posible solución podría ser utilizar el resultado de ambos modelos utilizados y realizar la media de sus resultados.

De esta forma, hemos decidido incluir los resultados obtenidos tras aplicar los modelos de estimación de pose BlazePose y OpenPose, y también la media del resultado de estos. Los resultados los analizaremos por separado: por una parte, la cadencia de pasos y, por otra, el tiempo de contacto del pie en el suelo junto al tiempo de vuelo.

Junto a los resultados también haremos un análisis de estos. Uno de los objetivos de este análisis de resultado es ver como de parecidos son estos al *ground truth*, que en este caso son los datos recogidos por las unidades de medición inercial. Para validar nuestros resultados, por tanto, veremos la correlación de estos con los datos tomados por *runscribe*, a parte de calcular algunas métricas de los errores.

Cabe tener en cuenta que el código usado para calcular las métricas se desarrolló después de la primera sesión que se hizo de grabación. Por consiguiente, es esperable que dicho código sea capaz de calcular las

métricas mejor en esta sesión que en las otras, ya que será capaz de localizar mejor los puntos clave de las series temporales en el primer día que en el segundo. Esto se debe, sobre todo, a que la cámara en los otros días se localiza en otra posición respecto del corredor. Aun así, también es interesante ver estas diferencias de precisión entre la primera sesión y el resto.

3.1. Cadencia de pasos

Lo primero que haremos será presentar los resultados obtenidos utilizando los dos modelos y la media junto a los datos de las unidades de medición inercial. También especificamos el sujeto, la velocidad a la que va y la sesión en la que se grabó el vídeo. Estos datos se muestran en la Tabla 2 y 3.

En este caso, los dos modelos y, por tanto, la media, tienen los mismos resultados. Esto ocurre ya que se calcula utilizando la autocorrelación de la serie temporal y el ciclo temporal cambia muy poco de un modelo a otro. También podemos observar como los resultados solo son unos números en concreto, y esta es la razón principal que causa el error. Esto ocurre porque el resultado de la autocorrelación solo puede ser un número entero de fotogramas. Entonces, al calcular cuánto tiempo pasa entre esos fotogramas el resultado solo podrá ser unos valores en concreto (múltiples de 1/30s).

Día	Sujeto	Velocidad (km/h)	Ground truth (pasos/min)	OpenPose (p./min)	BlazePose (p./min)	Media (p./min)
1	1	10	160	156	156	156
1	1	12	168	171	171	171
1	2	10	165	164	164	164
1	2	12	164	171	171	171
1	3	10	177	180	180	180
1	4	10	167	164	164	164
1	4	12	175	171	171	171
1	5	10	168	171	171	171
1	5	12	161	171	171	171
1	6	10	155	156	156	156
1	6	12	162	164	164	164
1	7	10	162	164	164	164
1	7	12	169	171	171	171
1	8	10	156	156	156	156
1	8	12	162	164	164	164
1	9	10	163	171	171	171
1	9	12	185	189	189	189
2	10	10	173	171	171	171
2	10	12	179	180	180	180
2	11	10	160	164	164	164
2	11	12	167	164	164	164
2	12	10	166	164	164	164
2	12	12	176	171	180	176

Tabla 2: Resultados de la cadencia de pasos de las sesiones del día 1 y 2.

Día	Sujeto	Velocidad (km/h)	Ground truth (pasos/min)	OpenPose (p./min)	BlazePose (p./min)	Media (p./min)
3	13	10	177	180	180	180
3	14	10	162	164	164	164
3	14	12	167	164	164	164
3	15	10	160	156	156	156
3	15	12	171	171	171	171
4	16	10	159	156	156	156
4	16	12	169	171	171	171
4	17	10	172	171	171	171
4	17	12	175	171	171	171
4	18	10	189	189	189	189
4	18	12	197	200	200	200
4	19	10	175	171	171	171
4	19	12	182	180	180	180
4	20	10	154	156	156	156
4	20	12	162	164	164	164
4	21	10	158	156	156	156
4	21	12	162	164	164	164
4	22	10	173	171	171	171
4	22	12	181	180	180	180
4	23	10	158	156	156	156
4	23	12	173	171	171	171
4	24	10	164	164	164	164
4	24	12	173	171	171	171

Tabla 3: Resultados de la cadencia de pasos de las sesiones del día 3 y 4.

Para medir la calidad de los resultados, calcularemos el coeficiente de correlación de Pearson entre los datos estimados y el *ground truth*, para ver como se asimilan los resultados. El coeficiente de correlación de Pearson lo acompañaremos del intervalo de 95 % de confianza y el p-valor del test de hipótesis de que la correlación sea estadísticamente significativa. También incluiremos información sobre los errores, como son la raíz del error cuadrático medio (RMSE) y el error medio absoluto (MAE). Estos resultados están expuestos en la Tabla 4.

Coeficiente de correlación de Pearson	Intervalo de 95 % de confianza	p-valor	RMSE	MAE
0,954	(0,918 - 0,975)	<2,2·10 ⁻¹⁶	2,87	2,37

Tabla 4: Resultados del test de correlación y de las métricas de error de la cadencia de pasos.

La correlación es muy alta, lo cual expresa que las estimaciones están muy correlacionadas con los valores reales. Podemos ver esto en la Figura 15, junto a una línea que definiría cuales serían los resultados perfectos. Por otro lado, el RMSE y el MAE también tienen buenos valores ya que son muy bajos en comparación de la escala de los resultados. Los errores, como podemos ver en la Figura 16, tienen una dispersión constante y lineal en función de la estimación. Además, la distribución es similar a la normal, a excepción de algunos valores extremos (Figura 17).

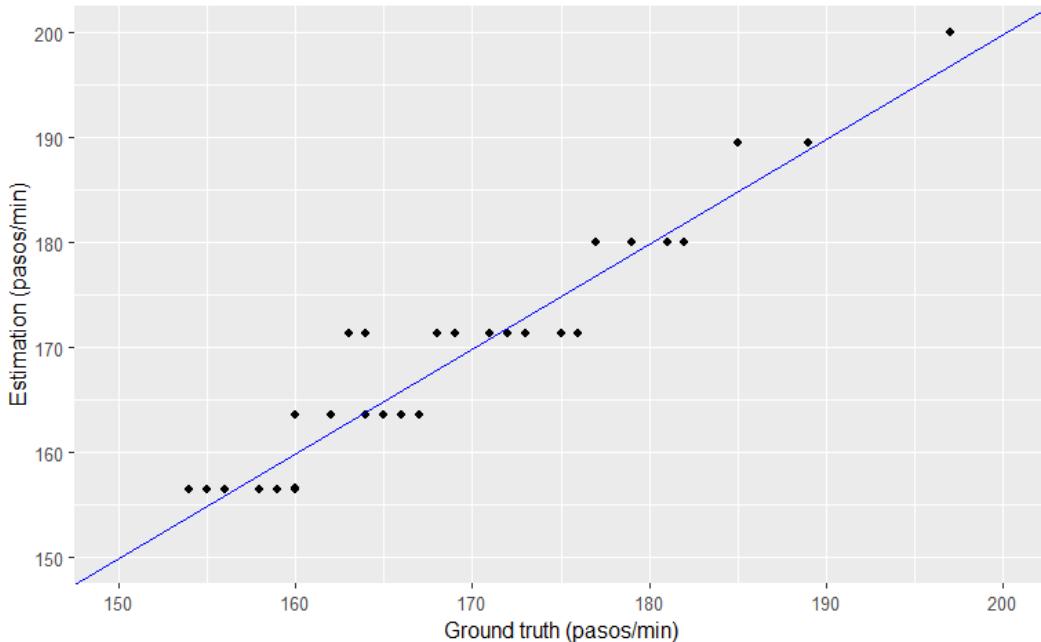


Figura 15: Gráfico de dispersión de la estimación y los valores reales de la cadencia de pasos.

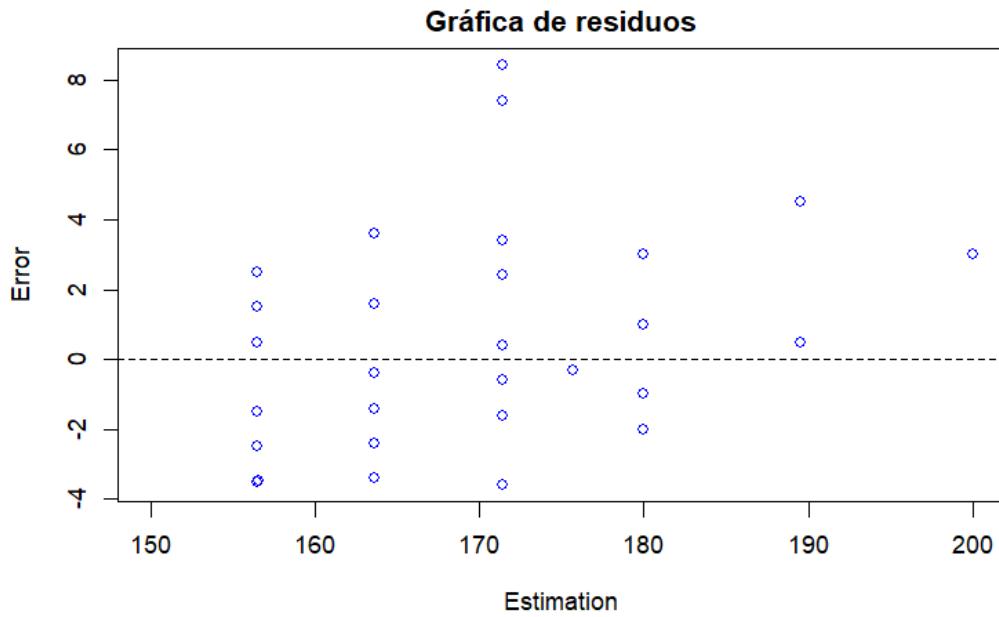


Figura 16: Representación de los errores en función de la estimación de la cadencia de pasos.

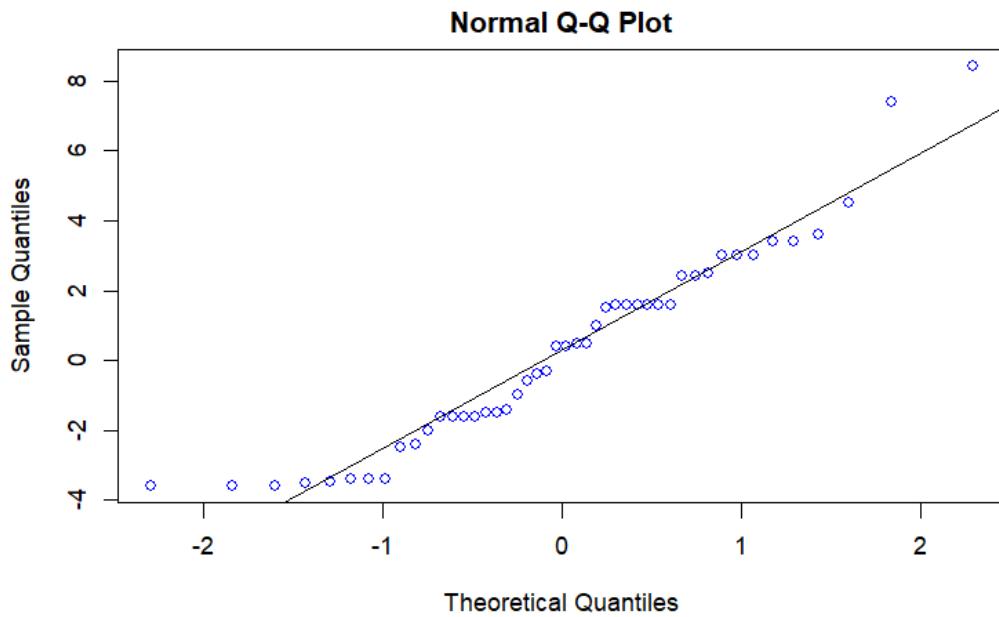


Figura 17: Gráfico que muestra los cuantiles de los datos de la cadencia de pasos en comparación de los cuantiles correspondientes a una distribución normal teórica de la misma cantidad de datos.

3.2. Tiempo de contacto del pie en el suelo

Los resultados de la métrica(tiempo de contacto del pie en el suelo) los presentaremos por separado para cada uno de los pies. Teniendo, por tanto, los resultados del pie izquierdo en las Tablas 5 y 6, y del pie derecho en las Tablas 7 y 8.

Día	Sujeto	Velocidad (km/h)	Ground truth (ms)	OpenPose (ms)	BlazePose (ms)	Media (ms)
1	1	10	310	321	304	312
1	1	12	274	271	271	271
1	2	10	310	309	333	321
1	2	12	276	303	300	301
1	3	10	277	242	279	260
1	4	10	306	306	300	303
1	4	12	278	279	293	286
1	5	10	326	320	326	323
1	5	12	294	297	293	295
1	6	10	307	309	272	290
1	6	12	288	295	300	297
1	7	10	330	310	313	311
1	7	12	300	293	297	295
1	8	10	325	329	319	324
1	8	12	295	285	303	294
1	9	10	399	254	262	258
1	9	12	262	251	243	247
2	10	10	270	264	328	296
2	10	12	257	257	375	316
2	11	10	263	254	348	301
2	11	12	257	231	308	269
2	12	10	309	320	320	320
2	12	12	277	262	320	291

Tabla 5: Resultados del tiempo de contacto del pie izquierdo en el suelo de las sesiones del día 1 y 2.

CAPÍTULO 3. RESULTADOS

Día	Sujeto	Velocidad (km/h)	Ground truth (ms)	OpenPose (ms)	BlazePose (ms)	Media (ms)
3	13	10	306	261	408	334,5
3	14	10	254	248	262	255
3	14	12	235	300	362	331
3	15	10	278	289	435	362
3	15	12	286	267	304	285,5
4	16	10	295	290	352	321
4	16	12	260	272	286	279
4	17	10	340	274	352	313
4	17	12	328	251	306	278,5
4	18	10	269	230	274	252
4	18	12	250	207	237	222
4	19	10	286	267	292	279,5
4	19	12	259	244	285	264,5
4	20	10	311	302	342	322
4	20	12	271	251	322	286,5
4	21	10	311	312	450	381
4	21	12	287	308	440	374
4	22	10	309	248	283	265,5
4	22	12	283	235	333	284
4	23	10	284	275	380	327,5
4	23	12	252	250	295	272,5
4	24	10	352	307	325	316
4	24	12	314	278	284	281

Tabla 6: Resultados del tiempo de contacto del pie izquierdo en el suelo de las sesiones del día 3 y 4.

Día	Sujeto	Velocidad (km/h)	Ground truth (ms)	OpenPose (ms)	BlazePose (ms)	Media (ms)
1	1	10	315	307	331	319
1	1	12	275	255	248	251,5
1	2	10	307	341	325	333
1	2	12	276	296	298	297
1	3	10	277	285	261	273
1	4	10	311	294	286	290
1	4	12	281	285	273	279
1	5	10	340	319	312	315,5
1	5	12	303	295	295	295
1	6	10	315	306	324	315
1	6	12	298	314	310	312
1	7	10	311	333	309	321
1	7	12	294	310	290	300
1	8	10	320	313	319	316
1	8	12	294	294	300	297
1	9	10	378	319	290	304,5
1	9	12	261	237	245	241
2	10	10	269	263	318	290,5
2	10	12	262	264	290	277
2	11	10	291	266	283	274,5
2	11	12	271	248	289	268,5
2	12	10	302	318	336	327
2	12	12	273	279	311	295

Tabla 7: Resultados del tiempo de contacto del pie derecho en el suelo de las sesiones del día 1 y 2.

CAPÍTULO 3. RESULTADOS

Día	Sujeto	Velocidad (km/h)	Ground truth (ms)	OpenPose (ms)	BlazePose (ms)	Media (ms)
3	13	10	297	235	276	255,5
3	14	10	298	265	281	273
3	14	12	352	247	260	253,5
3	15	10	283	274	296	285
3	15	12	286	244	288	266
4	16	10	285	279	309	294
4	16	12	253	243	286	264,5
4	17	10	313	246	272	259
4	17	12	274	237	269	253
4	18	10	265	232	244	238
4	18	12	289	228	257	242,5
4	19	10	292	276	277	276,5
4	19	12	261	238	268	253
4	20	10	310	307	323	315
4	20	12	269	255	284	269,5
4	21	10	322	276	317	296,5
4	21	12	295	259	293	276
4	22	10	287	251	290	270,5
4	22	12	266	227	270	248,5
4	23	10	282	277	333	305
4	23	12	270	254	284	269
4	24	10	351	321	321	321
4	24	12	313	284	304	294

Tabla 8: Resultados del tiempo de contacto del pie derecho en el suelo de las sesiones del día 3 y 4.

A diferencia de la métrica anterior, en el tiempo de contacto del pie en el suelo si observamos mayores diferencias en los resultados dependiendo del modelo de estimación de pose que hayamos utilizado. De forma similar a lo realizado en la métrica anterior, calculamos la correlación, el RMSE y el MAE para los tres modelos en las dos piernas. Podemos ver estos resultados separados por cada pierna en las Tablas 9 y 10.

La correlación en esta métrica es mucho más baja y con mayores p-valores que en la cadencia de pasos. Esto lo podemos ver para los dos modelos y la media en una gráfica de dispersión en las Figura 18 y 19. De entre los modelos, el que mejor funciona es el modelo OpenPose al

Modelo	Coeficiente de correlación de Pearson	Intervalo de 95 % de confianza	p-valor	RMSE	MAE
OpenPose	0,178	(-0,331 - 0,606)	0,49	35,1	22,5
BlazePose	0,145	(-0,351 - 0,593)	0,55	60,9	41,5
Media	0,178	(-0,331 - 0,607)	0,49	39,0	25,9

Tabla 9: Resultados del test de correlación y de las métricas de error del tiempo de contacto del pie izquierdo.

Modelo	Coeficiente de correlación de Pearson	Intervalo de 95 % de confianza	p-valor	RMSE	MAE
OpenPose	0,581	(0,350 - 0,746)	$2,3 \cdot 10^{-5}$	32,1	24,5
BlazePose	0,364	(0,082 - 0,592)	0,013	28,3	20,7
Media	0,525	(0,276 - 0,707)	0,00018	27,1	19,6

Tabla 10: Resultados del test de correlación y de las métricas de error del tiempo de contacto del pie derecho.

tener mayor coeficiente de correlación en ambas piernas. Pero en la pierna derecha el modelo BlazePose tiene menor RMSE y MAE. Sin embargo, si utilizamos la media tenemos resultados estables en ambas piernas, en comparación con los dos modelos. En la pierna izquierda tenemos mejores resultados que en la derecha y esto se debe a que la mayoría de los vídeos tienen un punto de vista en el que la pierna derecha está delante.

En general, los resultados del tiempo del contacto en el suelo son peores que los de la cadencia de pasos, ya que el modelo de estimación de pose no ha podido ser exacto porque la visibilidad de la pierna es reducida. Esto ocurre, por ejemplo, cuando una pierna pasa por delante de la otra. Aunque en este caso, la gran mayoría de las veces no llega a distorsionarse tanto como para que haya grandes errores en el cálculo del tiempo de contacto.

Otro de los inconvenientes de calcularlo así es que existe cierta variabilidad respecto a la persona. Esto ocurre porque también hay cierta variabilidad de precisión del modelo de estimación de pose dependiendo de la persona y de su forma de correr.

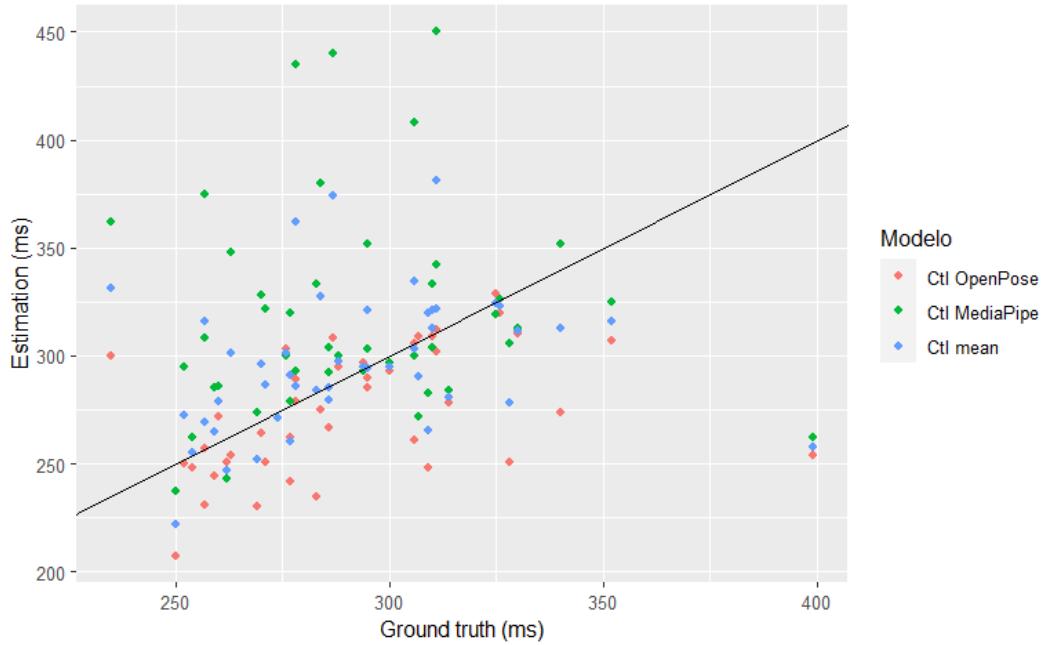


Figura 18: Gráfico de dispersión de la estimación y los valores reales del tiempo de contacto del pie izquierdo en el suelo.

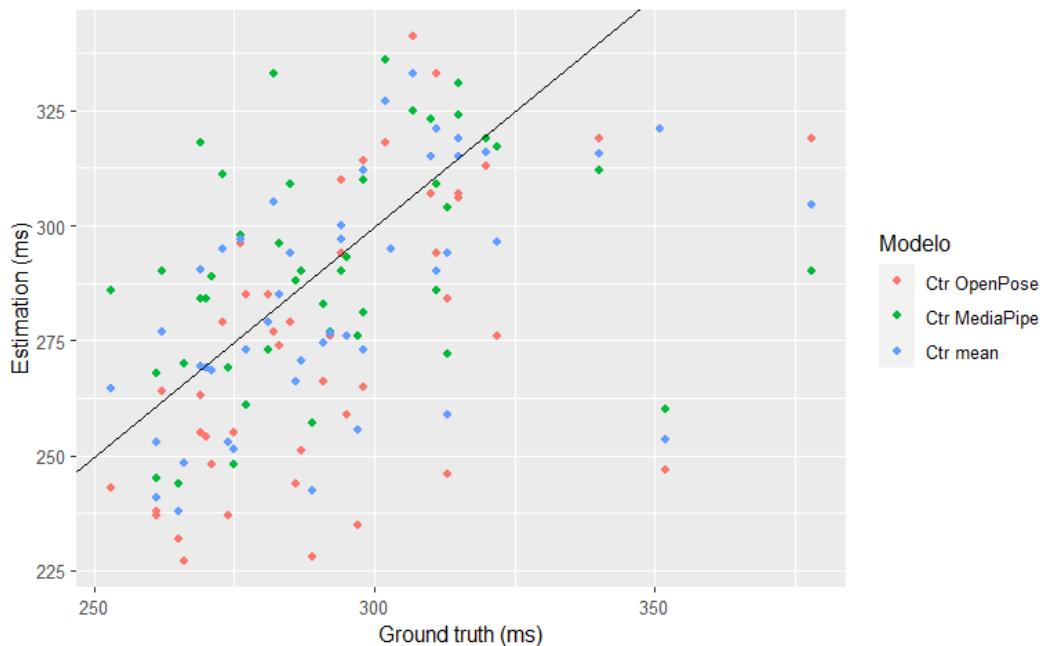


Figura 19: Gráfico de dispersión de la estimación y los valores reales del tiempo de contacto del pie derecho en el suelo.

Ángulo de la cámara	Pierna	Coeficiente de correlación de Pearson	Intervalo de 95 % de confianza	p-valor	RMSE	MAE
Izquierdo	Izq.	0,178	(-0,331 - 0,607)	0,49	36,0	16,5
Derecho	Izq.	0,326	(-0,046 - 0,619)	0,084	40,6	31,4
Izquierdo	Der.	0,624	(0,205 - 0,850)	0,0074	23,0	15,6
Derecho	Der.	0,368	(0,002 - 0,647)	0,049	29,2	22,0

Tabla 11: Resultados del test de correlación y de las métricas de error del tiempo de contacto de los dos pies separados por los ángulos de cámara. Están calculados a partir de la media de las predicciones entre los dos modelos.

El ángulo de la cámara es otro factor que debemos tener en cuenta, ya que la perspectiva que mejor captura las evoluciones temporales de los ángulos de las articulaciones es totalmente lateral respecto de la persona, ya que calcularía el ángulo real de la articulación. Sin embargo, con este ángulo el problema de superposición de objetos tiene mayor importancia. En este trabajo se han realizado grabaciones desde dos ángulos diferentes, ya que se busca disminuir la dependencia con el ángulo de la cámara. Puesto que esta es una razón adicional de la disminución de la precisión.

Para analizar la diferencia entre las sesiones con diferentes ángulos de cámara y cómo afecta, podemos ver en la Tabla 11 tenemos que mejores resultados con el ángulo de la cámara correspondiente a la primera sesión. Esto se debe a que se realizó el programa de cálculo de métricas teniendo solo esos vídeos. También es destacable que obtenemos mejores resultados de la pierna derecha en ambos casos, incluso en el caso de la cámara situada a la izquierda del corredor.

De la misma forma que en la métrica anterior, en las Figuras 20, 21, 22 y 23 se encuentra el análisis de los errores. Esto incluye un gráfico de dispersión para ver la linealidad o no linealidad de los residuos y un *q-q plot* para ver la semejanza con una distribución normal. Se observa que los errores son lineas respecto del valor de estimación y que la dispersión es constante, excepto en algunos casos en los valores más extremos. En los *q-q plots* se contempla cómo la distribución de errores no se parece al de una normal.

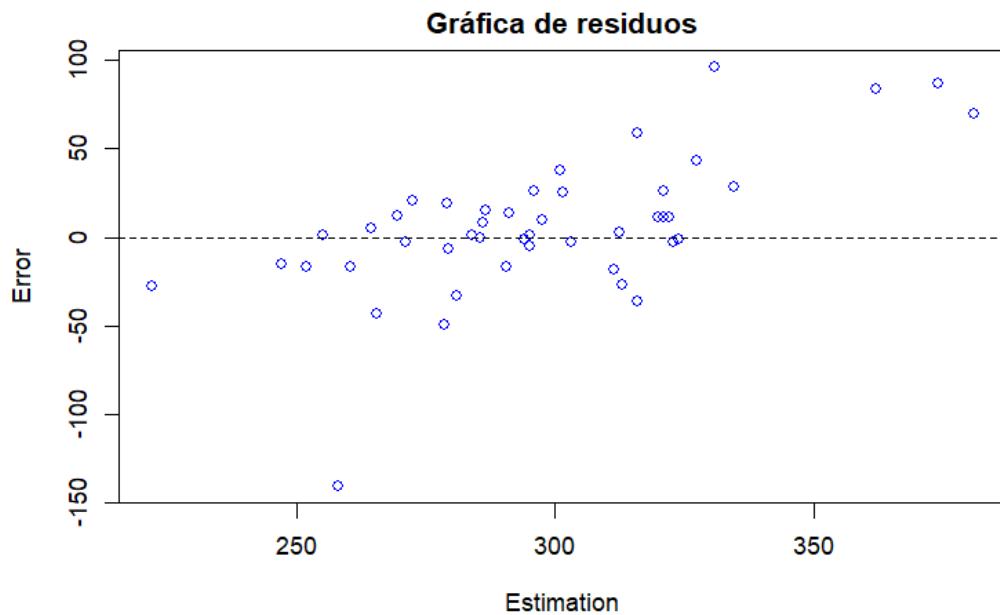


Figura 20: Representación de los errores en función de la estimación del tiempo de contacto del pie izquierdo en el suelo.

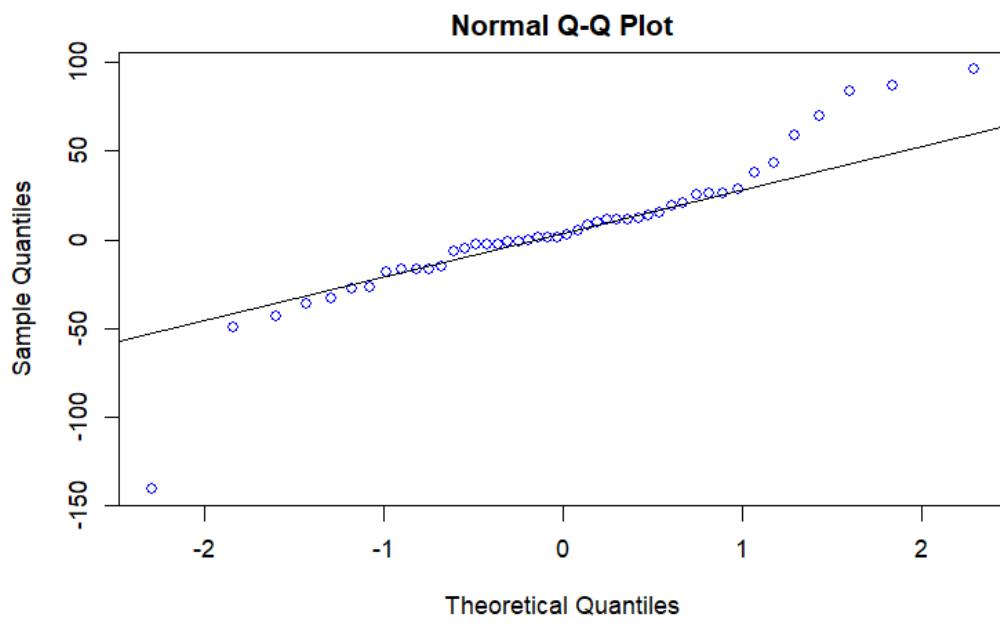


Figura 21: Gráfico que muestra los cuantiles de los datos del tiempo de contacto del pie izquierdo en el suelo en comparación de los cuantiles correspondientes a una distribución normal teórica de la misma cantidad de datos.

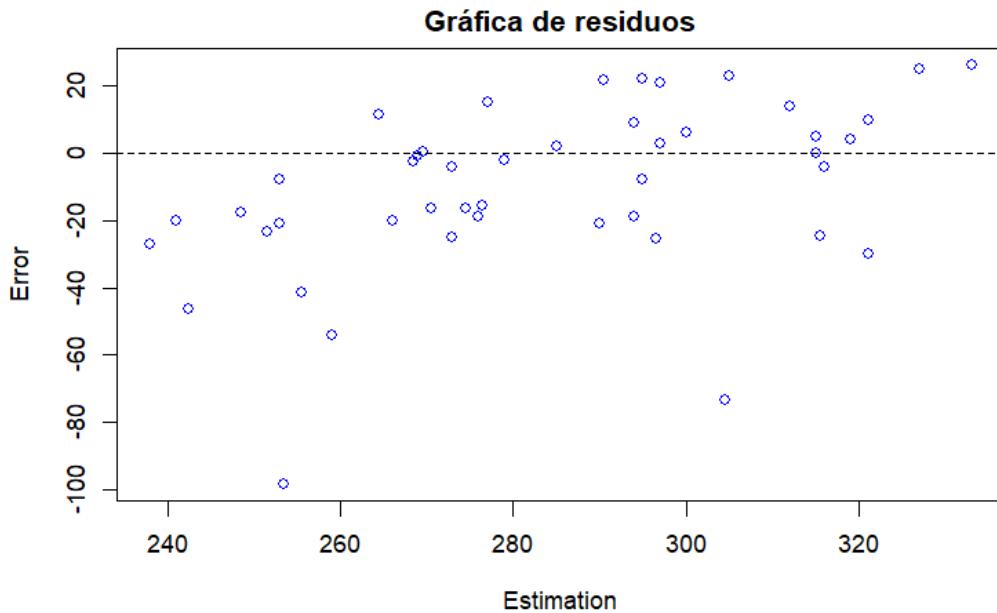


Figura 22: Representación de los errores en función de la estimación del tiempo de contacto del pie derecho en el suelo.

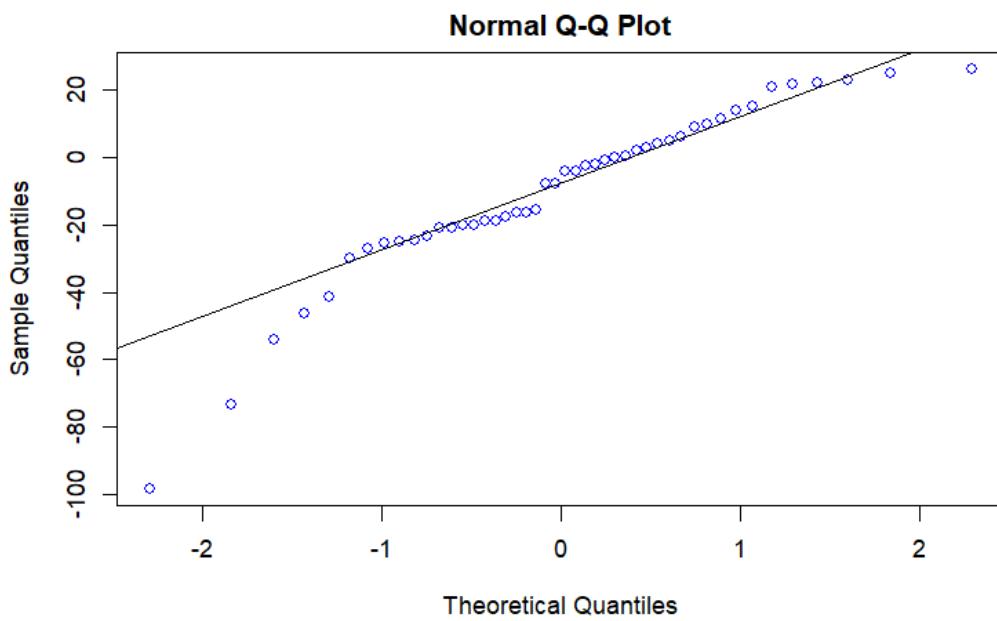


Figura 23: Gráfico que muestra los cuantiles de los datos del tiempo de contacto del pie izquierdo en el suelo en comparación de los cuantiles correspondientes a una distribución normal teórica de la misma cantidad de datos.

3.3. Ratio de tiempo en el aire

La última métrica que hemos medido ha sido el ratio de tiempo en el aire, que al igual que el tiempo de contacto, es diferente para cada pie. Los resultados correspondientes a esta métrica los podemos ver en las Tablas 12, 13, 14 y 15.

Día	Sujeto	Velocidad (km/h)	Ground truth (%)	OpenPose (%)	BlazePose (%)	Media (%)
1	1	10	18	16,1	20,7	18,4
1	1	12	23	22,6	22,4	22,5
1	2	10	15	15,7	9,1	12,4
1	2	12	20	13,4	14,3	13,85
1	3	10	18	27,3	16,4	21,85
1	4	10	15	16,6	18,2	17,4
1	4	12	19	20,4	16,2	18,3
1	5	10	9	8,7	7	7,85
1	5	12	16	15,2	16,3	15,75
1	6	10	20	19,5	29	24,25
1	6	12	22	19,5	18,2	18,85
1	7	10	11	15,6	14,5	15,05
1	7	12	15	16,4	14,9	15,65
1	8	10	15	14,3	16,8	15,55
1	8	12	20	22,4	17,2	19,8
1	9	10	1	27,3	25,1	26,2
1	9	12	19	20,8	23,2	22
2	10	10	22	24,7	6,3	15,5
2	10	12	23	22,8	0	11,4
2	11	10	30	30,8	5	17,9
2	11	12	28	37,1	15,7	26,4
2	12	10	14	12,6	12,8	12,7
2	12	12	19	25,2	4	14,6

Tabla 12: Resultados del ratio del tiempo en el aire del pie izquierdo de las sesiones del día 1 y 2.

3.3. RATIO DE TIEMPO EN EL AIRE

Día	Sujeto	Velocidad (km/h)	Ground truth (%)	OpenPose (%)	BlazePose (%)	Media (%)
3	13	10	10	21,8	0	10,9
3	14	10	32	32,4	28,5	30,45
3	14	12	35	18,2	1,1	9,65
3	15	10	26	24,6	0	12,3
3	15	12	19	23,8	13	18,4
4	16	10	22	24,4	8	16,2
4	16	12	27	22,4	18,4	20,4
4	17	10	7	21,8	0	10,9
4	17	12	9	28,2	12,7	20,45
4	18	10	15	17,2	23	20,1
4	18	12	18	30,9	21	25,95
4	19	10	16	23,6	16,6	20,1
4	19	12	21	26,7	14,5	20,6
4	20	10	20	21,2	10,5	15,85
4	20	12	27	31,6	12,2	21,9
4	21	10	18	18,7	0	9,35
4	21	12	23	29,2	0	14,6
4	22	10	11	29	19	24
4	22	12	15	29,5	0	14,75
4	23	10	25	28,1	1,1	14,6
4	23	12	27	28,4	15	21,7
4	24	10	4	16,1	11,4	13,75
4	24	12	10	20,7	18,9	19,8

Tabla 13: Resultados del ratio del tiempo en el aire del pie izquierdo de las sesiones del día 3 y 4.

CAPÍTULO 3. RESULTADOS

Día	Sujeto	Velocidad (km/h)	Ground truth (%)	OpenPose (%)	BlazePose (%)	Media (%)
1	1	10	16	19,9	13,5	16,7
1	1	12	23	27	29	28
1	2	10	16	6,9	11,3	9,1
1	2	12	20	15,1	14,7	14,9
1	3	10	19	24,5	21,6	23,05
1	4	10	13	19,8	22	20,9
1	4	12	18	18,6	21,2	19,9
1	5	10	5	8,9	11	9,95
1	5	12	13	15,6	15,7	15,65
1	6	10	18	20	15,6	17,8
1	6	12	19	14,5	15,4	14,95
1	7	10	16	9,3	15,6	12,45
1	7	12	17	11,3	16,9	14,1
1	8	10	17	18,2	16,7	17,45
1	8	12	20	19,8	18,2	19
1	9	10	1	8,9	17,2	13,05
1	9	12	19	25	22,6	23,8
2	10	10	23	24,9	9,3	17,1
2	10	12	22	20,7	12,9	16,8
2	11	10	22	27,4	22,7	25,05
2	11	12	24	32,3	21,1	26,7
2	12	10	16	13,3	8,5	10,9
2	12	12	20	20,2	6,8	13,5

Tabla 14: Resultados del ratio del tiempo en el aire del pie derecho de las sesiones del día 1 y 2.

Día	Sujeto	Velocidad (km/h)	Ground truth (%)	OpenPose (%)	BlazePose (%)	Media (%)
3	13	10	13	29,5	17,2	23,35
3	14	10	21	27,6	23,3	25,45
3	14	12	2	32,8	29	30,9
3	15	10	25	28,6	22,6	25,6
3	15	12	19	30,4	17,7	24,05
4	16	10	25	27,2	19,3	23,25
4	16	12	29	30,6	18,3	24,45
4	17	10	14	29,7	22,3	26
4	17	12	21	32,4	23,1	27,75
4	18	10	16	26,5	13,6	20,05
4	18	12	11	24	14	19
4	19	10	15	21,1	20,8	20,95
4	19	12	21	28,6	19,5	24,05
4	20	10	20	19,8	15,7	17,75
4	20	12	27	30,5	22,5	26,5
4	21	10	15	28	17,4	22,7
4	21	12	21	29,1	20	24,55
4	22	10	17	28,2	17,1	22,65
4	22	12	20	31,9	19,1	25,5
4	23	10	26	27,8	13	20,4
4	23	12	23	27,3	18,7	23
4	24	10	4	12,2	12,3	12,25
4	24	12	9	18,7	13,1	15,9

Tabla 15: Resultados del ratio del tiempo en el aire del pie derecho de la sesiones del día 3 y 4.

El ratio de tiempo en el aire está calculado directamente a partir del resultado del tiempo de contacto del pie en el suelo. De esta forma, tenemos las mismas características en los resultados que esta métrica. En la Tabla 16 observamos como el coeficiente de correlación de Pearson no es demasiado alto, por ejemplo, existen casos en los que el p-valor es mayor de 0,05. El modelo que mejores resultados ha dado ha sido el que consiste en la media, si nos fijamos en el RMSE on el MAE. Sin embargo, el que mayor coeficiente de correlación tiene con mayor confianza es el del modelo OpenPose.

Finalmente, como hemos hecho con las métricas anteriores, en las

CAPÍTULO 3. RESULTADOS

Modelo	Pierna	Coeficiente de correlación de Pearson	Intervalo de 95 % de confianza	p-valor	RMSE	MAE
OpenPose	Izq.	0,389	(0,112 - 0,611)	0,0075	8,33	5,67
BlazePose	Izq.	-0,092	(-0,372 - 0,204)	0,54	12,61	9,51
Media	Izq.	0,161	(-0,135 - 0,431)	0,28	8,01	5,63
OpenPose	Der.	0,468	(0,206 - 0,668)	0,0010	8,61	6,62
BlazePose	Der.	0,146	(-0,151 - 0,418)	0,33	7,21	5,15
Media	Der.	0,383	(0,104 - 0,606)	0,0087	6,88	5,17

Tabla 16: Resultados del test de correlación y de las métricas de error del ratio del tiempo en el aire de las dos piernas.

Figuras 24, 25, 26 y 27 vemos la dispersión de los errores, los cuales son lineales y homocedásticos. Asimismo, vemos como, de forma similar a la métrica del tiempo de contacto, la distribución de los errores difiere del de una normal.

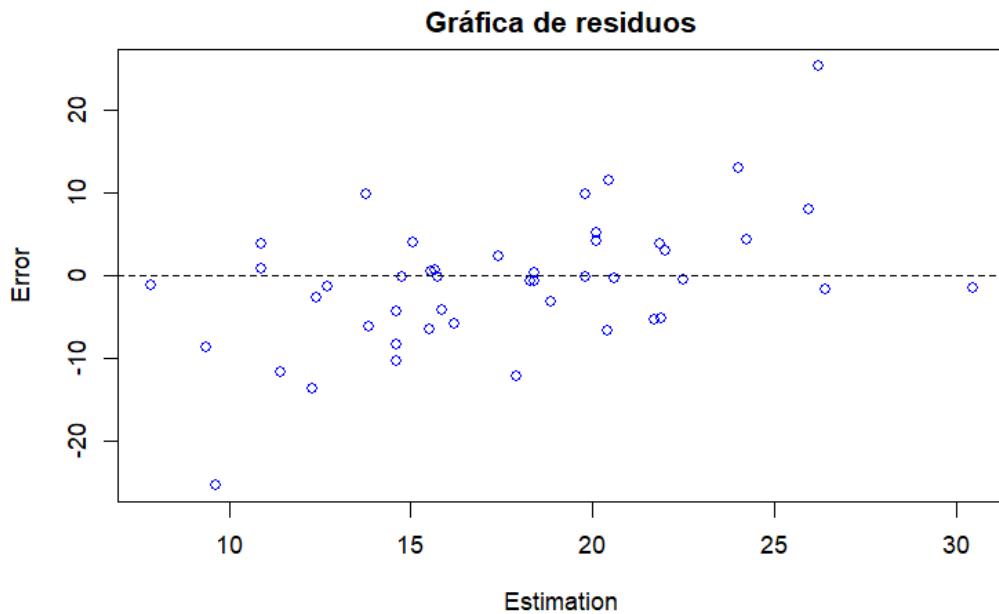


Figura 24: Representación de los errores en función de la estimación del ratio de tiempo en el aire del pie izquierdo.

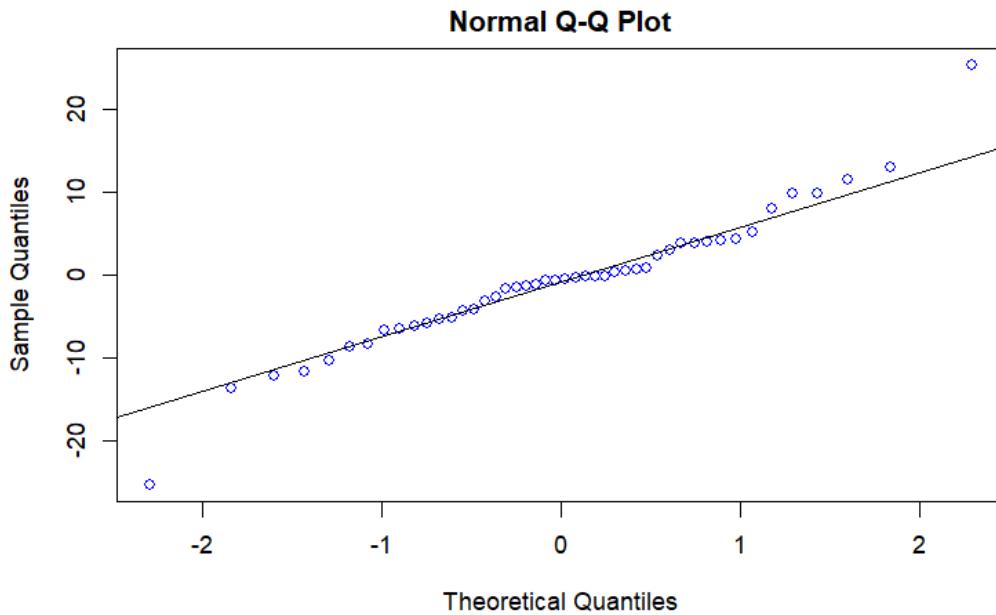


Figura 25: Gráfico que muestra los cuantiles de los datos del ratio de tiempo en el aire del pie izquierdo en comparación de los cuantiles correspondientes a una distribución normal teórica de la misma cantidad de datos.

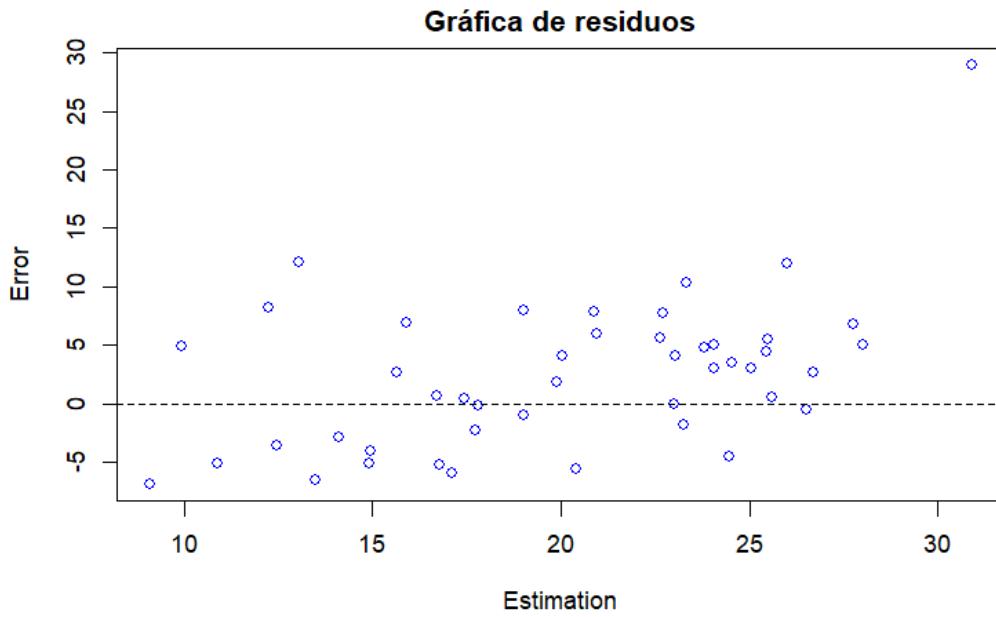


Figura 26: Representación de los errores en función de la estimación del ratio de tiempo en el aire del pie derecho.

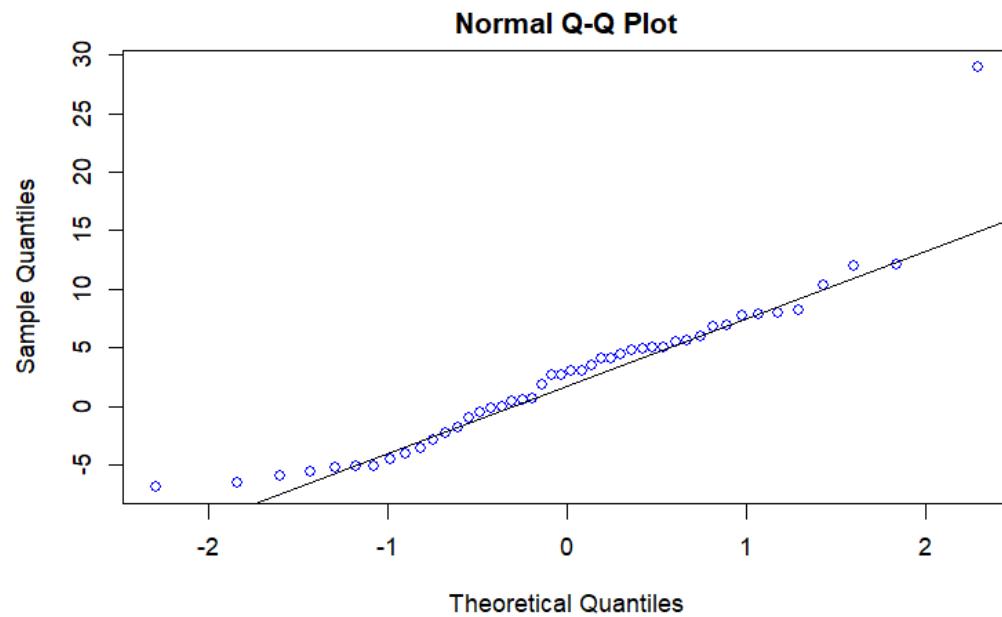


Figura 27: Gráfico que muestra los cuantiles de los datos del ratio de tiempo en el aire del pie izquierdo en comparación de los cuantiles correspondientes a una distribución normal teórica de la misma cantidad de datos.

Capítulo 4

Conclusiones y proyección futura

La inteligencia artificial tiene numerosas aplicaciones en el deporte y en estos últimos años se han dado grandes pasos. Los algoritmos mejoran año a año y cada vez surgen más usos de estos algoritmos de inteligencia artificial, y es por eso que es un campo de la ciencia de datos que está tanto en auge.

4.1. Conclusiones

Los resultados del método utilizado durante este trabajo son bastante dispares. Por una parte, los resultados de la cadencia de paso son muy buenos ya que se asemejan mucho a los resultados obtenidos por las unidades de medición inercial. Teniendo en cuenta como se calcula la cadencia de pasos, la precisión se podría mejorar si utilizamos vídeos que tengan más fotogramas por segundo. De esta forma, los posibles resultados de la cadencia de pasos serían más precisos. Sin embargo, sería necesario el uso de cámaras que graben con más fotogramas por segundo.

Por otra parte, las métricas de tiempo de contacto y ratio de tiempo en el aire tienen peores resultados al tener mayor error cuadrático medio, error absoluto medio y menor correlación con los resultados de las unidades de medición inercial.

Debido a esto, la parte que tiene mayor potencial de mejora es el

cálculo de métricas. Concretamente, el que debería mejorarse más, si se quiere continuar con este proyecto, es el cálculo del tiempo de contacto del pie en el suelo y el ratio de tiempo en el aire.

4.2. Proyección futura

El trabajo puede mejorarse de diversas formas, no solo a nivel de mejorar la predicción, sino también para utilizarlo en diferentes usos. Se podría adaptar el algoritmo para calcular otras métricas o incluso para otro ejercicio diferente.

Para mejorar el cálculo de métricas se podrían realizar cambios en el código referente a cómo calcula cuáles son los máximos que usa para calcular la métrica posteriormente. Asimismo, se puede considerar utilizar varias series temporales para cada pierna y no solo una, para así conseguir estimar mejor el momento exacto en el que el pie entra en contacto con el suelo y en el que despega de él. Por último, otro enfoque posible sería utilizando algún método de aprendizaje máquina que se pueda emplear en varias series temporales cíclicas y cuyo resultado sea de regresión.

En cuanto a los modelos de estimación de pose, también se podría mejorar en este aspecto. Durante el trabajo hemos utilizado modelos de estimación de pose bidimensionales, los cuales hacen la estimación de pose en una proyección 2D sobre la imagen. Con esta estimación obtenemos, por tanto, la proyección de los ángulos en la imagen. Sin embargo, si quisieramos obtener el ángulo real lo que tendríamos que hacer es utilizar algún modelo de estimación de pose tridimensional.

Por último, también mejorarían los resultados si escogemos modelos que usan una parte temporal para estimar la pose, teniendo en cuenta la pose del fotograma anterior. De esta forma, la precisión del modelo sería mayor en aquellos fotogramas en los que una pierna se sitúa por detrás de la otra.

El algoritmo desarrollado durante este trabajo necesita mejorar

diferentes fases para obtener buenos resultados en todos sus cálculos. Sin embargo, durante este trabajo, se ha demostrado que esta aproximación al problema del análisis de la carrera tiene gran potencial y mucha utilidad.

Apéndice A

Código

A continuación presentamos el código para calcular las métricas a partir de las coordenadas de los puntos clave del modelo de estimación de pose.

```
1 from cmath import inf
2 import os, json
3 import numpy as np
4 from scipy.signal import argrelextrema
5 import pandas as pd
6 import seaborn as sns
7 from matplotlib import pyplot as plt
8 import statsmodels.api as sm
9
10
11 # Path to the videos and name of the video
12
13 path = "Videos/Output_Json/"
14 video_name = "Sujeto1_R1_10kmh_20210715_175707"
15
16 print("Video name: ", video_name, "\n")
17
18 # Frame rate of the video
19
20
21 fr = 30
22
23
24 # Number for each joint
```

APÉNDICE A. CÓDIGO

```
25
26 Nose = 0
27 Neck = 1
28 RShoulder = 2
29 RElbow = 3
30 RWrist = 4
31 LShoulder = 5
32 LElbow = 6
33 LWrist = 7
34 MidHip = 8
35 RHip = 9
36 RKnee = 10
37 RAnkle = 11
38 LHip = 12
39 LKnee = 13
40 LAnkle = 14
41 REye = 15
42 LEye = 16
43 REar = 17
44 LEar = 18
45 LBigToe = 19
46 LSmallToe = 20
47 LHeel = 21
48 RBigToe = 22
49 RSmallToe = 23
50 RHeel = 24
51 Background = 25
52
53
54 # equation to calculate angle from 2 vectors
55
56 def vectors_to_angle(vector1, vector2) -> float:
57     x = np.dot(vector1,
58                -vector2)/(np.linalg.norm(vector1)*np.linalg.norm(vector2))
59     theta = np.degrees(np.arccos(x))
60     return theta
61
62 # get a json file and returns a dictionary with the angles of that frame
63 # of the video
64 def json_to_angle(json_name) -> dict:
65     with open(path+video_name+"/"+json_name) as json_file:
66
67         data = json.load(json_file)['people'][0]['pose_keypoints_2d']
```

```

68
69      # coordinates
70  Neck_coor = np.array([data[Neck*3], data[Neck*3+1]])
71  RShoulder_coor = np.array([data[RShoulder*3],
72    ↵  data[RShoulder*3+1]])
72  LShoulder_coor = np.array([data[LShoulder*3],
73    ↵  data[LShoulder*3+1]])
73  MidHip_coor = np.array([data[MidHip*3], data[MidHip*3+1]])
74  LHip_coor = np.array([data[LHip*3], data[LHip*3+1]])
75  RHip_coor = np.array([data[RHip*3], data[RHip*3+1]])
76  LKnee_coor = np.array([data[LKnee*3], data[LKnee*3+1]])
77  RKnee_coor = np.array([data[RKnee*3], data[RKnee*3+1]])
78  LAnkle_coor = np.array([data[LAnkle*3], data[LAnkle*3+1]])
79  RAnkle_coor = np.array([data[RAnkle*3], data[RAnkle*3+1]])
80  LBigToe_coor = np.array([data[LBigToe*3], data[LBigToe*3+1]])
81  RBigToe_coor = np.array([data[RBigToe*3], data[RBigToe*3+1]])

82

83
84      # vectors
85  Torso_vector = MidHip_coor - Neck_coor
86  RTorso_vector = RHip_coor - RShoulder_coor
87  LTorso_vector = LHip_coor - LShoulder_coor
88  Hip_vector = LHip_coor - RHip_coor
89  LFemur_vector = LKnee_coor - LHip_coor
90  RFemur_vector = RKnee_coor - RHip_coor
91  LTibia_vector = LAnkle_coor - LKnee_coor
92  RTibia_vector = RAnkle_coor - RKnee_coor
93  LFoot_vector = LBigToe_coor - LAnkle_coor
94  RFoot_vector = RBigToe_coor - RAnkle_coor

95

96  LFemur_vector_perp = LFemur_vector.copy()
97  LFemur_vector_perp[1] = LFemur_vector[0]
98  LFemur_vector_perp[0] = -LFemur_vector[1]

99

100 RFemur_vector_perp = RFemur_vector.copy()
101 RFemur_vector_perp[1] = RFemur_vector[0]
102 RFemur_vector_perp[0] = -RFemur_vector[1]

103

104      # angles
105  TorsoHip_angle = vectors_to_angle(Torso_vector, Hip_vector)
106  LShoulderFemur_angle = vectors_to_angle(LFemur_vector_perp,
107    ↵  LTorso_vector) + 90
107  RShoulderFemur_angle = vectors_to_angle(RFemur_vector_perp,
108    ↵  RTorso_vector) + 90
108  LHip_angle = vectors_to_angle(LFemur_vector, Hip_vector)

```

```
109     RHip_angle = vectors_to_angle(RFemur_vector, Hip_vector)
110     LKnee_angle = vectors_to_angle(LTibia_vector,
111         ↪  LFemur_vector_perp) + 90
112     RKnee_angle = vectors_to_angle(RTibia_vector,
113         ↪  RFemur_vector_perp) + 90
114     LAnkle_angle = vectors_to_angle(LFoot_vector, LTibia_vector)
115     RAnkle_angle = vectors_to_angle(RFoot_vector, RTibia_vector)
116
117     dict_angles = {"TorsoHip_angle": TorsoHip_angle,
118                     ↪ "LShoulderFemur_angle": LShoulderFemur_angle,
119                     ↪ "RShoulderFemur_angle": RShoulderFemur_angle,
120                         ↪ "LHip_angle": LHip_angle, "RHip_angle":
121                         ↪ RHip_angle, "LKnee_angle": LKnee_angle,
122                         ↪ "RKnee_angle": RKnee_angle,
123                         ↪ "LAnkle_angle": LAnkle_angle, "RAnkle_angle":
124                         ↪ RAnkle_angle}
125
126     return dict_angles
127
128
129 # Plot the angles
130
131
132 def plot_angles(df) -> None:
133
134     with sns.axes_style("darkgrid"):
135         fig, axes = plt.subplots(5, 2, figsize=(9, 30))
136
137         fig.suptitle("Evolución en el tiempo de los ángulos de las
138             ↪ articulaciones")
139
140         sns.lineplot(ax = axes[0, 0], data = df, x = "Time_in_sec", y =
141             ↪ "TorsoHip_angle").set(xlabel = "Tiempo (segundos)", ylabel =
142             ↪ "Torso/cadera (°)")
143         sns.lineplot(ax = axes[1, 0], data = df, x = "Time_in_sec", y =
144             ↪ "LShoulderFemur_angle").set(xlabel = "Tiempo (segundos)",
145
146
147             ↪ "RShoulderFemur_angle").set(xlabel = "Tiempo (segundos)",
```

```

135
136     sns.lineplot(ax = axes[2, 0], data = df, x = "Time_in_sec", y =
137                     "LHip_angle").set(xlabel = "Tiempo (segundos)", ylabel =
138                     "Cadera/pierna izq. (°)")
139     sns.lineplot(ax = axes[2, 1], data = df, x = "Time_in_sec", y =
140                     "RHip_angle").set(xlabel = "Tiempo (segundos)", ylabel =
141                     "Cadera/pierna der. (°)")
142     sns.lineplot(ax = axes[3, 0], data = df, x = "Time_in_sec", y =
143                     "LKnee_angle").set(xlabel = "Tiempo (segundos)", ylabel =
144                     "Rodilla izq. (°)")
145     sns.lineplot(ax = axes[3, 1], data = df, x = "Time_in_sec", y =
146                     "RKnee_angle").set(xlabel = "Tiempo (segundos)", ylabel =
147                     "Rodilla der. (°)")
148     sns.lineplot(ax = axes[4, 0], data = df, x = "Time_in_sec", y =
149                     "LAnkle_angle").set(xlabel = "Tiempo (segundos)", ylabel =
150                     "Tobillo izq. (°)")
151     sns.lineplot(ax = axes[4, 1], data = df, x = "Time_in_sec", y =
152                     "RAnkle_angle").set(xlabel = "Tiempo (segundos)", ylabel =
153                     "Tobillo der. (°)")

154     plt.show()

155     sns.set(font_scale = 1.5)
156     sns.lineplot(data = df, x = "Time_in_sec", y =
157                     "LKnee_angle").set(xlabel = "Tiempo (segundos)", ylabel =
158                     "Rodilla izq. (°)")

159     plt.show()

160     # # Fourier transform
161
162     # def fourier_period(y) -> float:
163     #     # Number of sample points
164     #     N = len(y)
165     #     # Sample spacing
166     #     T = 1.0 / fr
167
168     #     yf = np.fft.fft(y)
169     #     yf_abs = 2.0/N * np.abs(yf[0:N//2])

```

APÉNDICE A. CÓDIGO

```
161 #     xf = np.fft.fftfreq(N, T)[:N//2]
162 #     yf_max = np.copy(yf_abs)
163 #     yf_max[0] = 0.0
164 #     period = xf[np.argmax(yf_max)]
165
166 #     plt.plot(xf, yf_abs)
167 #     plt.xlim(right = 4 * period)
168 #     plt.axvline(period, color='red')
169 #     plt.xlabel('Tiempo (segundos)')
170 #     plt.title("Transformada rápida de Fourier")
171 #     plt.show()
172
173 #     return period
174
175
176 # Autocorrelation
177
178 def autocorrelation(y, t) -> float:
179     s = pd.Series(y)
180
181     x = sm.tsa.acf(s, missing = "drop")
182
183     plt.plot(t[:len(x)], x)
184     plt.grid()
185     plt.show()
186
187     xf = np.copy(x)
188     xf[0] = 0.0
189     xf[1] = 0.0
190     xf[2] = 0.0
191     period = t[np.argmax(xf)]
192
193     return period
194
195
196
197 # The contact time starts with the leg have an local maximum in the
#     angle due to the impact, at the same time the knee has his first
#     angle maximum.
198 # The contact time finishes with the maximum angle of the leg and the
#     second maximum of the knee.
199
200 # Get two largest values in a list
201
202 # def twoLargest(numbers) -> list:
```

```

203 #      x=0
204 #      y=0
205 #      x_loc=0
206 #      y_loc=0
207 #      i = 0
208 #      for loc, n in enumerate(numbers):
209 #          i+=1
210 #          if n > x:
211 #              y = x
212 #              y_loc = x_loc
213 #              x = n
214 #              x_loc = loc
215 #              continue
216 #          if n > y:
217 #              y = n
218 #              y_loc = loc
219
220 #      return [x, y], [x_loc, y_loc]
221
222
223 def maximum(numbers) -> list:
224     z = 0
225     z_loc = 0
226     for loc, n in enumerate(numbers):
227         if n > z:
228             z = n
229             z_loc = loc
230
231     return z, z_loc
232
233 def minimum(numbers) -> list:
234     z = inf
235     z_loc = 0
236     for loc, n in enumerate(numbers):
237         if n < z:
238             z = n
239             z_loc = loc
240
241     return z, z_loc
242
243
244 #old version
245
246 # def contact(y, fr) -> float:
247 #     # a cycle can be determined by a local minimum lower than 120°

```

```
248
249 #     splits = argrelextrema(y.values,
250 #     ↪ np.less)[0][y[argrelextrema(y.values, np.less)[0]]<120]
250
251 #     t_diffs = []
252
253 #     for i, s in enumerate(splits):
254 #         if i>0:
255 #             if s-splits[i-1]<10:
256 #                 continue
257 #             y_s = y[splits[i-1]:s]
258
259 #             _, max_locs =
260 #             ↪ twoLargest(y_s.values[argrelextrema(y_s.values, np.greater)[0]])
261 #             max_ilocs = argrelextrema(y_s.values,
262 #             ↪ np.greater)[0][max_locs]
263
264 #             t_diffs.append((1.0/fr) * abs(max_ilocs[0] -
265 #             ↪ max_ilocs[1]))
266
267
268 #new version
269
270 # def contact(y, y2, fr) -> float:
271 #     # a cycle can be determined by a local minimum lower than 120°
272
273 #     splits = argrelextrema(y.values,
274 #     ↪ np.less)[0][y2[argrelextrema(y.values, np.less)[0]]<120]
274
275 #     t_diffs = []
276
277 #     for i, s in enumerate(splits):
278 #         if i>0:
279 #             if s-splits[i-1]<10:
280 #                 continue
281 #             y_s = y[splits[i-1]:s]
282
283 #             _, max_locs =
284 #             ↪ twoLargest(y_s.values[argrelextrema(y_s.values, np.greater)[0]])
284 #             max_ilocs = argrelextrema(y_s.values,
285 #             ↪ np.greater)[0][max_locs]
285 #             #max_iloc = min(max_ilocs)
```

```

286
287 #           y2_s = y2[splits[i-1]:s]
288
289 #           - , max_iloc = maximum(y2_s)
290 #           - , min_iloc = minimum(y2_s)
291
292
293 #           t = (1.0/fr) * abs(max_iloc - min_iloc)
294
295 #           t_diffs.append(t)
296
297 #           t_diff = np.mean(t_diffs)
298
299
300 #           return t_diff
301
302
303 def contact(y, fr, period, min_dis=9, max_dis=30) -> float:
304     # get the local minimums that are less than 150
305     i_min = argrelextrema(y.values, np.less)[0]
306     i_max = argrelextrema(y.values, np.greater)[0]
307
308     i_s = []
309
310     for i, _ in enumerate(i_min):
311         for j, _ in enumerate(i_max):
312             if i == 0:
313                 continue
314             if (i_min[i] - i_min[i-1]) > (i_max[j] - i_min[i-1]) and
315                 (i_max[j] - i_min[i-1]) > 0:
316                 if (y.iloc[i_max[j]] - y.iloc[i_min[i-1]]) > 40:
317                     i_s.append(i_min[i-1])
318
319     splits = y.iloc[i_s]
320
321     # make sure we do not have two local minimums in less than 0.3
322     # seconds.
323     drop_index = []
324
325     for loc, _ in enumerate(splits):
326         if loc == 0:
327             continue
328
329             if ((splits.index[loc] - splits.index[loc-1]) < min_dis) or
330                 (splits.index[loc] - splits.index[loc-1]) > max_dis:

```

```
328         drop_index.append(splits.index[loc])
329
330     # split the data in each cycle
331
332     splits = splits.drop(drop_index).index
333
334     # print(splits)
335
336     t_diffs = []
337
338     for i, s in enumerate(splits):
339
340         if s == splits[0]:
341             continue
342
343         #y1_s = y1[splits[i-1]:s]
344
345         # get the moment when the angle of the knee with the torso goes
346         # to greater than 180°
347         #try:
348         #    y1_s_180 = min(y1_s.index[y1_s>170].to_list())
349         #except:
350         #    continue
351
352         # this is the moment where the foot do the first contact with a
353         # 0 angle between the knee and the torso
354
355         #first_contact = y1_s_180 - 1
356
357         #launching = y1_s.index[maximum(y1_s)[1]]
358
359         y_s = y[splits[i-1]:s]
360
361         j_max = argrelextrema(y_s.values, np.greater)
362         y_s_maxs = y_s.iloc[j_max]
363         y_s_maxs = y_s_maxs[y_s_maxs.values>150]
364         #print(y2_s_maxs)
365         #if y2_s_maxs.index[0] < 150:
366         #    print(y2_s)
367         #    print(y2_s_maxs)
368         #    print(y2_s_maxs.index)
369
370         first_contact = min(y_s_maxs.index)
371
372         launching = max(y_s_maxs.index)
```

```

371
372     # print("Max dis: ", max_dis)
373     # print("First contact: ", first_contact)
374     # print("Launching: ", launching)
375
376     # get the moment when the knee angle is in its minimum
377     #launching = y2_s.index[minimum(y2_s)[1]]
378
379     t = (1.0/fr) * (launching - first_contact)
380
381     if (t > 0.1) & (t < 2*period/3):
382         t_diffs.append(t)
383
384     # print(t_diffs)
385
386     t_diff = np.mean(t_diffs)
387
388     return t_diff
389
390
391 # Main function
392
393 def main() -> None:
394
395     json_files = [pos_json for pos_json in os.listdir(path+video_name)
396                   if pos_json.endswith('.json')]
397
398     list_of_dicts = []
399     for json_name in json_files[:len(json_files)-1]:
400         try:
401             list_of_dicts.append(json_to_angle(json_name))
402         except:
403             pass
404
405     df_angles = pd.DataFrame(list_of_dicts)
406     df_angles["Time_in_sec"] = [n/fr for n in range(len(df_angles))]
407
408     # print(df_angles)
409
410     df_to_plot = df_angles.loc[(df_angles["Time_in_sec"] >= 0) &
411                                (df_angles["Time_in_sec"] <= 5)]
412
413     plot_angles(df_to_plot)
414

```

APÉNDICE A. CÓDIGO

```
413 period = autocorrelation(df_angles["RHip_angle"] ,  
414     ↵ df_angles["Time_in_sec"])  
415 print("Periodo entre cada paso:", period/2, "segundos")  
416 print("Ratio de pasos:", (2*60.0)/period, "pasos/minuto")  
417  
418 contact_time_r = contact(df_angles["RKnee_angle"], fr, period,  
419     ↵ min_dis=int(period*fr)-5, max_dis=int(period*fr)+5)  
420 contact_time_l = contact(df_angles["LKnee_angle"], fr, period,  
421     ↵ min_dis=int(period*fr)-5, max_dis=int(period*fr)+5)  
422 print("Tiempo de contacto en el suelo del pie izquierdo:",  
423     ↵ contact_time_l, "s")  
424 print("Tiempo de contacto en el suelo del pie derecho:",  
425     ↵ contact_time_r, "s")  
426  
427 flight_ratio_r = 100 * (1 - (2 * contact_time_r / period))  
428 flight_ratio_l = 100 * (1 - (2 * contact_time_l / period))  
429 print("Ratio de tiempo en el aire del pie izquierdo:",  
430     ↵ flight_ratio_l, "%")  
431 print("Ratio de tiempo en el aire del pie derecho:", flight_ratio_r,  
432     ↵ "%")  
433  
434 if __name__ == '__main__':  
435     main()
```

Bibliografía

- [1] T. K. M. Lee, M. Belkhatir and S. Sanei, “A comprehensive review of past and present vision-based techniques for gait recognition”, *Multimedia Tools and Applications*, vol. 72, no. 3, pp. 2833–2869, 2014.
- [2] Y. Wahab and N. A. Bakar, “Gait Analysis Measurement For Sport Application Based On Ultrasonic System,” *IEEE 15th International Symposium on Consumer Electronics*, pp. 20-24, 2011.
- [3] N. Ahmad, R. A. R. Ghazilla, N. M. Khairi, “Reviews on Various Inertial Measurement Unit (IMU) Sensor Applications”, *International Journal of Signal Processing Systems*, vol. 1, no. 2, December 2013
- [4] RunScribe. (2022, January 28). *RunScribe - Wearable IMU - Gait Analysis*. Retrieved May 2022, from <https://runscribe.com/>
- [5] C. Richter, M. O'Reilly and E. Delahunt, “Machine learning in sports science: challenges and opportunities”, *Sports Biomechanics*, pp. 1-7, Routledge, 2021
- [6] A. C. Lapham and R. M. Bartlett, “The use of artificial intelligence in the analysis of sports performance: A review of applications in human gait analysis and future directions for sports biomechanics”, *Journal of Sports Sciences*, vol. 13, no. 3, pp. 229–237, 1995
- [7] G. A. Bekey, J. J. Kim, J. K. Gronley, E. L. Bontrager and J. Perry, “GAIT-ER-AID: An expert system for diagnosis of human gait”, *Artificial Intelligence in Medicine*, 4(4), 293–308, 1992
- [8] G. Liu and O. Schulte, “Deep Reinforcement Learning in Ice Hockey

- for Context-Aware Player Evaluation”, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 3442-3448, 2018
- [9] W. Gong *et al.*, “Human Pose Estimation from Monocular Images: A Comprehensive Survey”, *Sensors*, 16 (12), 1992, 2016
- [10] C. Wang, F. Zhang and S. S. Ge, “A comprehensive survey on 2D multi-person pose estimation methods”, *Engineering Applications of Artificial Intelligence*, vol. 102, pp. 104260, June 2021
- [11] L. L. Presti and M. L. Cascia, “3D skeleton-based human action classification: a survey”, *Pattern Recognition*, vol. 53, pp. 130–147, May 2016
- [12] S. Johnson and M. Everingham, “Clustered pose and nonlinear appearance models for human pose estimation”, *Proceedings of the British Machine Vision Conference*, BMVA Press, pp. 12.1-12.11, September 2010.
- [13] B. Sapp and B. Taskar, “MODEC: multimodal decomposable models for human pose estimation”, *Proceedings of the IEEE Conference of Computer Vision and Pattern Recognition*, pp. 3674-3681, June 2013.
- [14] M. Andriluka, L. Pishchulin, P. Gehler and B. Schiele, “2D human pose estimation: new benchmark and state of the art analysis”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674-3681, June 2014.
- [15] T. Lin *et al.*, “Microsoft COCO: common objects in context”, *Proceedings of the 13th European Conference on Computer Vision*, vol. 8693, pp. 740–755, September 2014.
- [16] C. Ionescu, D. Papava, V. Olaru and C. Sminchisescu, “Human3.6M: Large scale datasets and predictive methods for 3d human sensing in natural environments”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36(7), pp. 1325-1339, July 2014.
- [17] L. Sigal, A. O. Balan and M. J. Black “HumanEva: Synchronized Vi-

- deo and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion”, *International Journal of Computer Vision*, vol. 87(1-2), pp. 4-27, March 2010.
- [18] H. Fang, S. Xie and C. Lu “RMPE: Regional Multi-person Pose Estimation”, *ICCV*, pp. 2334-2343, 2017.
- [19] R. Votel *et al.* “Next-Generation Pose Detection with MoveNet and TensorFlow.js”, *Google Research*, <https://www.tensorflow.org/hub/tutorials/movenet>, May 2021.
- [20] V. Bazarevsky *et al.* “Blazepose: On-device realtime body pose tracking”, *CVPR Workshop*, June 2020.
- [21] Z. Cao, T. Simon, S.E. Wei and Y. Sheikh “Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7291-7299, 2017.
- [22] K. Mattes, S. Wolff and S. Alizadeh “Kinematic Stride Characteristics of Maximal Sprint Running of Elite Sprinters – Verification of the “Swing-Pull Technique””, *Journal of Human Kinetics*, vol.77, pp. 15-24, 2021.

