
**UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA**

INSTITUTO DE ELECTRÓNICA Y MECATRÓNICA

INGENIERÍA EN MECATRÓNICA

# SISTEMA DE CAPTURA Y RÉPLICA DE MOVIMIENTO PARA MIEMBRO INFERIOR

TESIS PARA OBTENER EL TÍTULO DE:

INGENIERO EN MECATRÓNICA

PRESENTA:

ERICK AQUILEO ARANGO GÓMEZ

DIRECTORA DE TESIS:

DRA. ESTHER LUGO GONZÁLEZ

CO-DIRECTOR:

DR. ANTONIO ORANTES MOLINA

HUAJUAPAN DE LEÓN, OAXACA, MÉXICO, SEPTIEMBRE 2025

## *Dedicatoria*

*Dedicado a mis padres, por darme la oportunidad de superarme, por el amor, apoyo y paciencia que me han brindado durante toda mi vida.*

*A mi hermano Luis, por su guía, consejos y por su apoyo incondicional.*

# Agradecimientos

Agradezco a mis padres Luis Arango y Flor Gómez, por su inmenso sacrificio, por los valores y principios que me han enseñado, por estar dispuestos a dar más por nosotros y por siempre incentivarme a superarme. Por permitirme alcanzar este sueño y creer en mí, eternamente gracias.

A mi hermano Luis Alberto, quien en mi vida se ha convertido en un ejemplo a seguir como estudiante y como persona, por su apoyo en los momentos difíciles, por sus ánimos, consejos y paciencia. Afortunado de ser tu hermano menor, gracias por todo.

A mi familia, por todo el apoyo y consejos que me brindaron. A mis abuelos, tíos y primos, por su amor y por los buenos momentos juntos.

A los amigos que me acompañaron durante cada etapa de mi vida académica, por haber compartido tantas experiencias juntos, por su apoyo, las risas y la alegría. Sin ellos, el camino no hubiese sido tan divertido.

Agradezco a cada profesor a que me incentivo y apoyo durante todos mis años como estudiante, sin su enseñanza, paciencia y vocación, esto no hubiese sido posible, muchas gracias.

# Resumen

Este trabajo trata sobre un sistema de captura y réplica de movimiento centrado en el plano sagital del miembro inferior del cuerpo humano, que no hace uso de marcadores o sensores para la adquisición de información sobre el movimiento analizado. Se desarrollan dos sistemas de captura, diferenciados por el uso de distintas herramientas de detección de articulaciones, *MediaPipe* y *MMPose*, ambos sistemas analizan frames de videos que muestran a personas realizando ejercicios de miembro inferior desde la perspectiva lateral izquierda, calculando el ángulo de movimiento de tres articulaciones, cadera, rodilla y tobillo. A partir de los ángulos calculados, la reproducción del movimiento capturado se lleva a cabo mediante un modelo de péndulo doble de tres eslabones, de forma virtual en el entorno de simulación CoppeliaSim y de manera física con el uso combinado de una tarjeta Arduino UNO y de actuadores físicos GM25-370. Con la realización de pruebas de estabilidad y coherencia biomecánica se determina el sistema de captura de movimiento con mejor desempeño, además, para ambos sistemas de réplica se evalúa la precisión en la reproducción del movimiento analizado.

# Índice

# Índice de Figuras

# Índice de Tablas

# Capítulo 1

# Introducción

A los sistemas de captura de movimiento se les han dado diversas aplicaciones, una de las más interesantes es en la medicina como herramienta para el seguimiento en el mejoramiento de la salud de pacientes. Hoy en día, con el avance de la electrónica y la robótica se han desarrollado sistemas de captura aún más complejos que posibilitan la réplica del movimiento capturado, gracias a esto se ha extendido su uso en sistemas de rehabilitación para pacientes con poca o nula movilidad en alguna de sus extremidades a través de la captura de ejercicios y la réplica de estos, facilitando la mejoría en la movilidad del paciente, así como en su calidad de vida.

Para sistemas de captura y réplica de movimiento en el sector de rehabilitación motriz se requiere de una adquisición de datos fiables y precisos. Este tipo de necesidad puede ser cubierta a través del uso de dispositivos especializados como cámaras o sensores, y aunque algunas empresas como Vicon™ ya se especializan en brindar sistemas de captura fiables y de calidad, estos están lejos de ser accesibles para la mayoría de personas interesadas. Por lo anteriormente descrito, muchos proyectos optan por el uso de sistemas personalizados que emplean recursos de libre acceso, como Kinect SDK, en conjunto con hardware de menor calidad, a costa de un margen de error más alto en los resultados [1]; además, dependiendo de la forma en la que se desarrollen estos sistemas se puede llegar a requerir de un conocimiento avanzado sobre el tema. Este proyecto se centra en el desarrollo de sistemas de captura de movimiento que

permitan replicar el movimiento humano tanto en un entorno simulado como en un sistema físico de tres eslabones, a modo de representación de un miembro inferior.

## 1.1. Estructura de tesis

En este proyecto se presenta el desarrollo de un sistema de captura y réplica de movimiento de miembro inferior. En el capítulo dos se aborda el marco teórico, el cual está compuesto por conceptos necesarios para el desarrollo del proyecto, como lo son, los tipos de sistemas de captura de movimiento y sus principales características, algunos de los recursos que pueden ser utilizados para el desarrollo o implementación de los sistemas mencionados anteriormente, así como conceptos importantes sobre la marcha humana y las características de los diversos movimientos que se generan en las articulaciones de los miembros inferiores del cuerpo humano. Para el capítulo tres se detalla el desarrollo del proyecto, se propone un diseño para los sistemas de captura y para los sistemas de réplica de movimiento, además se especifican las herramientas que serán utilizadas en el proceso de desarrollo. En el capítulo cuatro se implementa el diseño y se verifica la funcionalidad de los sistemas de captura y réplica a través de la realización de pruebas. En el capítulo cinco se presenta el análisis y comparación de resultados de los sistemas de captura y la precisión de los sistemas de réplica. Para el capítulo seis se muestran las conclusiones del desarrollo del proyecto y se plantean los trabajos futuros.

## 1.2. Estado del arte

La visión artificial es una rama de la inteligencia artificial (IA), que se define como la capacidad de sistemas informáticos para adquirir información significativa a partir de imágenes digitales, videos u otras entradas visuales [2]. Este recurso posibilita que los sistemas informáticos puedan tomar decisiones o emitan recomendaciones basadas en la interpretación de la información. En el desarrollo de la investigación llevada a cabo por los neurofisiólogos H. Hubel y N. Wiesel en el año de 1959 [3], se describía la respuesta de las neuronas corticales de un gato doméstico a estímulos visuales, concluyendo por los resultados obtenidos que el procesamiento de imágenes comienza por las formas más simples, líneas, segmentos o bordes. Con los diversos avances en informática, el uso de la visión artificial se ha extendido a diversas áreas, por mencionar algunas: en la astronomía para la observación e identificación de cuerpos celestiales, para aplicaciones de medicina como el diagnóstico de patologías, en sistemas industriales automatizados de inspección, para la verificación del cumplimiento de normas de calidad en diversos productos mecánicos, orgánicos, etc [4]. Dentro de las diversas aplicaciones de esta tecnología, una de las más interesantes es su uso para el seguimiento, detección y captura de movimiento.

Los sistemas de captura de movimiento o por sus siglas en inglés *Mocap* ( *Motion capture*) consisten en la captura de datos sobre el movimiento mecánico del cuerpo humano, utilizando visión artificial principalmente, y la representación del mismo en sistemas digitales [5]. Este tipo de sistemas son utilizados ampliamente en animación 3D con fines de entretenimiento, en películas o series, para el control remoto de robots o como una interfaz con las computadoras. Los sistemas *Mocap* permiten además la captura de coordenadas angulares, así como la adquisición de velocidades y aceleraciones, haciendo uso de diversos recursos tecnológicos como cuartos especializados o acondicionados, software dedicado y cámaras de alta resolución de espectros específicos, además del uso de sensores y marcadores o trajes para la obtención de mejores resultados. Con la posibilidad de tener información precisa acerca del movimiento

humano, se da pie a la creación de sistemas complejos que hacen uso de los sistemas de captura de movimiento y la robótica para toma de decisiones, actividades de alto riesgo o réplica de movimiento en algún modelo.

## 1.3. Aplicación de sistemas basados en visión artificial para la captura, análisis o réplica de movimiento

El uso de sistemas de análisis y/o captura de movimiento utilizados en sectores de investigación facilitan todo tipo de operaciones, como la realización y comparación de pruebas para la validación de información visual compleja de analizar a simple vista. Sistemas de captura de movimiento, como el aplicado en el proyecto "*Sistema de seguimiento visual para la rehabilitación*" desarrollado por Gildardo Azcárate [6] sugieren que, el uso en conjunto de un sistema de visión artificial con técnicas médicas, son clave para el mejoramiento de la salud de un paciente, principalmente, al contribuir o confirmar diagnósticos médicos visuales, como radiografías, tomografías o resonancias, las cuales suelen depender de la experiencia y percepción del profesional de la salud, lo que en consecuencia puede llevar a interpretaciones variables.

Las aplicaciones de la visión artificial no se ven limitadas únicamente al análisis y comparación de información visual, pues gracias a la conjunción de sistemas de visión artificial con la robótica, se conciben sistemas mucho más complejos. Sistemas de control robótico a través de movimientos espaciales, son hoy en día un hecho, tal y como se demuestra en el proyecto de Andrea Condo sobre el desarrollo de un sistema de visión artificial para el control de un brazo robótico Mitsubishi®[7]. Este proyecto combina la visión artificial con el uso un brazo robótico de la marca Mitsubishi®, para crear un sistema que puede identificar posición y orientación del miembro superior de un operario, y utilizar la información obtenida para controlar un sistema electro mecánico, el brazo robótico, con el fin de permitir la realización de tareas específicas, las cuales pueden llegar a ser denominadas de alto riesgo para un ser humano.

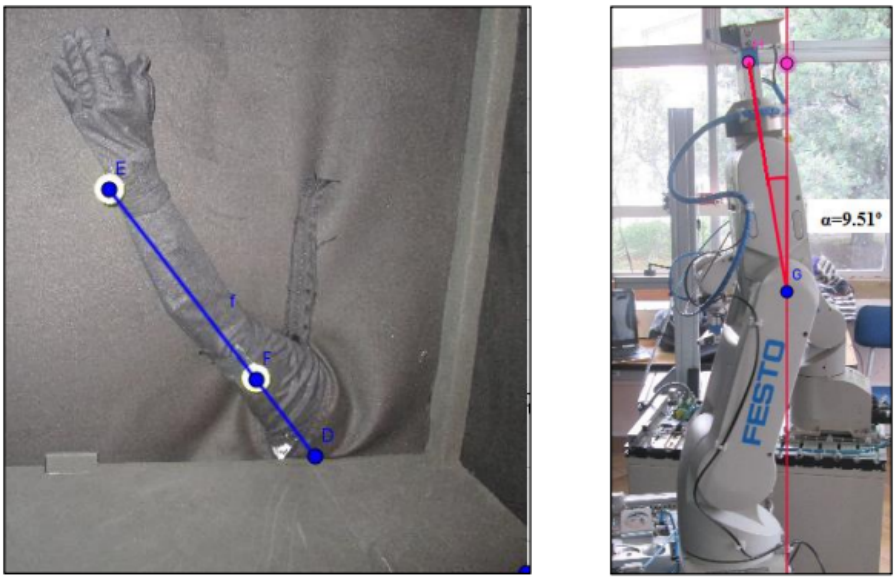



Figura 1.1: Control de brazo robótico Mitsubishi® mediante visión artificial [7].

En el proyecto "*Control de exoesqueleto de rehabilitación de mano con leap motion controller*" realizado por David Sierra y colaboradores en el año 2022 [1], se desarrolla un nuevo sistema de control para su uso en conjunto con un sistema *Leap Motion Controller* o por sus siglas LMC. El objetivo principal fue la obtención de un sistema de control que permita la captura de movimientos de la mano de una persona sin el uso de elementos de contacto, como guantes, sensores, electrodos, etc., la aplicación del sistema completo del proyecto, el cual incluye el uso de un exoesqueleto ROBHAND para la mano, es el ejemplo perfecto sobre como la utilización en conjunto de sistemas de captura y réplica de movimiento son una gran herramienta en el campo de la rehabilitación. En el proyecto se realizó la captura de movimiento de las articulaciones de la mano sana del paciente a través del sistema LMC, replicándolo con el exoesqueleto ROBHAND en la extremidad afectada, todo esto con la finalidad de que un paciente pueda ser capaz de realizar ejercicios de terapia para la recuperación de la movilidad en el miembro afectado, y si bien aunque el proyecto sólo ha sido probado en pacientes sanos, este promete ser un gran avance para la mejoría de personas con problemas de movilidad en las manos.

## 1.4. Planteamiento del problema

La implementación de sistemas de *Mocap* en proyectos orientados a la réplica de movimiento, y en particular en prototipos diseñados para rehabilitación, presenta diversos problemas que comprometen su efectividad y/o aplicabilidad. Uno de los principales desafíos radica en la accesibilidad de los sistemas de captura de movimiento. La dificultad de adquisición de sistemas comerciales, ha sido tradicionalmente una barrera significativa para el uso de esta tecnología [1] [8].

Sistemas de captura de movimiento de bajo costo, denominados personalizados, que son desarrollados por el usuario, suelen comprometer la precisión en los datos recopilados, esto debido al uso de hardware económico de mala calidad, lo que limita su utilidad para aplicaciones que requieren de datos precisos y confiables. Se debe destacar que la facilidad de uso y la implementación son importantes para la utilización de sistemas de este tipo, pues la complejidad en la configuración y calibración de los sistemas puede desalentar su uso, especialmente en usuarios que no tienen experiencia o que carecen de los conocimientos técnicos. Para dar una solución a todo lo descrito anteriormente, el desafió de este proyecto consiste en el desarrollo y evaluación de sistemas de captura de movimiento, que garanticen que los rangos de movimiento obtenidos estén alineados con los parámetros fisiológicos descritos en la literatura, deben ser accesibles o sin costo alguno, fáciles de usar e implementar, desarrollados para integrarse con un sistema de réplica de movimiento virtual y físico, de péndulo doble con tres eslabones, para su uso en la investigación o en el desarrollo futuro de sistemas de rehabilitación. Debido a la complejidad del proyecto y a la necesidad de realizar ajustes continuos en función de los resultados obtenidos, el problema se aborda con un enfoque metodológico de carácter iterativo, que permite validar y mejorar los sistemas en las distintas fases de su desarrollo.

## 1.5. Justificación

De acuerdo a diversos proyectos que basan su investigación o análisis en el uso de sistemas de captura, mayoritariamente con fines de rehabilitación al facilitar el proceso de recuperación de los pacientes [9] [10] [6], se coincide con varios puntos expuestos anteriormente. En primer lugar, el costo de adquisición, debido a que si se planteara la opción de adquirir un sistema de captura de movimiento ofrecido por alguna empresa especializada en el campo, como Vicon™, en el desarrollo del proyecto o investigación, los costos asociados a la licencia del software, y el probable requerimiento de hardware adicional (cámaras, sensores, trajes especiales, etc.), vuelven inviable la adquisición de los productos. Esta principal limitación, obliga a los usuarios o instituciones a recurrir a sistemas de uso libre o personalizados, lo que conlleva a un nuevo problema, la dificultad de implementación, debido a que generalmente, esta alternativa exige un nivel considerable de conocimiento sobre programación, para el desarrollo de un sistema que no solo cumpla con los objetivos específicos del proyecto, sino que también garantice resultados consistentes y confiables.

Para el área de rehabilitación o prótesis, se sabe que la utilización de sistemas de captura ayuda a determinar el mejoramiento de un paciente, y con la integración de sistemas de réplica de movimiento, se garantiza la realización precisa y constante de ejercicios terapéuticos, para una recuperación más rápida y eficiente [6] [1]. Actualmente, investigadores que trabajan en el desarrollo de prototipos o sistemas de rehabilitación que hacen uso de la captura y/o réplica de movimiento, optan por la utilización de sistemas *Mocap* simples y de uso libre, el empleo extendido del dispositivo Kinect es un ejemplo de esto, y si bien al no requerir de harware adicional para su implementación se reducen costos, los resultados que brindan estos dispositivos de captura no siempre alcanzan un nivel de precisión óptimo, enfrentándose a fluctuaciones y variaciones en los datos capturados, sin mencionar el tiempo necesario para la capacitación de personal para el uso de los sistemas. Al desarrollar un sistema de captura y

réplica de movimiento, que no requiera de extensos conocimientos del lenguaje sobre el que está desarrollado, simple y fácil de utilizar, se busca facilitar su uso para comparación de datos sobre el avance de un paciente al realizar ejercicios de rehabilitación, y al mismo tiempo brindar una herramienta para el desarrollo de nuevos prototipos para la recuperación.

## 1.6. Hipótesis

Es posible calcular el movimiento angular de las articulaciones del miembro inferior del cuerpo humano, sin la necesidad de marcadores o sensores para la adquisición de información, manteniendo la coherencia biomecánica de los rangos de movimiento registrados.

## 1.7. Objetivos

### 1.7.1. Objetivo general

Realizar un sistema de captura de movimiento centrado en la pierna humana, para obtener los ángulos en el plano sagital y reproducirlos tanto en un modelo virtual como en uno físico de péndulo doble con tres eslabones.

### 1.7.2. Objetivos específicos

Para conseguir el objetivo general se plantean los siguientes objetivos específicos a cumplir durante el desarrollo del proyecto:

- ✓ Determinar criterios de diseño del sistema de captura de movimiento mediante el análisis de sistemas comerciales existentes.
- ✓ Desarrollar al menos dos sistemas de captura de movimiento, que sean capaces de almacenar los grados de movimiento de las articulaciones de interés.
- ✓ Analizar y comparar los datos obtenidos a partir de la realización de pruebas específicas, como el análisis de movimientos de miembro inferior realizados por diferentes personas.
- ✓ Analizar los resultados y determinar cuál de los sistemas desarrollados es la mejor opción, de acuerdo a criterios como la estabilidad de los datos capturados y facilidad de uso o modificación.
- ✓ Integrar al sistema de captura de movimiento un sistema de réplica físico de tres eslabones, y su equivalente en un entorno simulado.

## 1.8. Metodología de desarrollo

Dado que este proyecto se enfoca principalmente en el desarrollo de software o programas, se opta por utilizar la metodología iterativa. Esta metodología permite dividir el proyecto en tareas más pequeñas y manejables, con retroalimentación constante en cada etapa o iteración, a su vez, estas tareas pueden subdividirse en iteraciones más específicas, lo que facilita el desarrollo continuo, reduce errores y permite ajustes tempranos [11].

Como punto de partida se identifica el objetivo o producto final, un sistema de captura y réplica de movimiento para miembro inferior, este objetivo se divide en dos tareas clave, el desarrollo de sistemas de captura de movimiento y el desarrollo de sistemas de réplica de movimiento. Las tareas resultantes se toman como iteraciones y a su vez son divididas en iteraciones más pequeñas, el desarrollo de los sistemas de captura esta dividido nuevamente, debido que se realizarán al menos dos sistemas de captura, cada uno con diferencias clave, de manera similar el desarrollo de los sistemas de réplica se divide en dos iteraciones, una para el desarrollo del sistema de réplica virtual y otra para la réplica de movimiento físico.

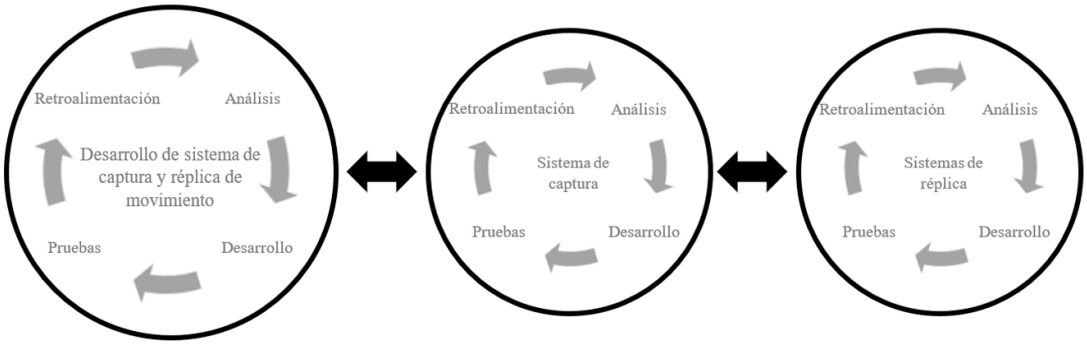



Figura 1.2: Metodología iterativa para el proyecto.

Cada iteración se llevará a cabo mediante un proceso circular, como se muestra en la Figura 1.2, estos ciclos comenzarán con una etapa de análisis, seguido del desarrollo, la realización de pruebas donde se detectan posibles errores y se evalúa el desempeño general, por último la retroalimentación de todo el proceso para la detección de mejoras y la corrección de errores.

Las actividades necesarias para completar cada iteración, se distribuyen tal como se muestra a continuación:

1. **Análisis.**
   - ✓ Planteamiento de necesidad.
   - ✓ Objetivos.
   - ✓ Requerimientos.
2. **Desarrollo.**
   - ✓ Diseño preliminar.
   - ✓ Diseño conceptual.
   - ✓ Diseño detallado.
   - ✓ Implementación.
3. **Pruebas.**
   - ✓ Evaluación de desempeño.
   - ✓ Detección de errores.
4. **Retroalimentación.**
   - ✓ Corrección de errores.
   - ✓ Implementación de mejoras.

# Capítulo 2

# Marco teórico

Para la elaboración del proyecto hay que tener en cuenta los conceptos básicos sobre captura de movimiento, ya que es imprescindible contar con un conocimiento claro de los recursos disponibles para el desarrollo de sistemas orientados al tema. En este capítulo se revisan conceptos sobre la marcha humana con énfasis en las principales articulaciones del miembro inferior, incluyendo los movimientos articulares más relevantes, como la flexión y extensión de cadera, rodilla y tobillo, para adquirir una noción más clara sobre los rangos de movimiento de estas articulaciones. Además, se revisan algunos de los modelos mecánicos más recurrentes para la representación de un miembro inferior, tema importante para la elaboración de un sistema de réplica de movimiento centrado en esta región del cuerpo humano.

## 2.1. Tipos de sistemas de captura y análisis de movimiento

De acuerdo a la investigación y comparación de sistemas de captura llevada a cabo por Gómez Echeverry y sus colegas [12], se puede destacar la siguiente información sobre la clasificación y características de los distintos sistemas de captura existentes. Para los sistemas de captura de movimiento que usualmente utilizan cámaras, de cualquier espectro, se les conoce como sistemas de visión u ópticos, utilizan datos recopilados por una o más cámaras para inferir

el movimiento que realiza algún cuerpo, generalmente con la utilización de marcas adheridas al cuerpo del sujeto, con este método se pueden obtener resultados muy fiables aunque en algunos sistemas ópticos más avanzados no se requiere del uso de marcas, por esta razón se considera que los sistemas ópticos se dividen en dos grupos, sistemas ópticos con marcadores y sin marcadores.

La desventaja más evidente en el caso del uso de marcadores es la posible obstrucción de la visión o identificación de estos ya sea por el mismo movimiento del sujeto o por causas externas como el ambiente de experimentación, luces o reflejos generalmente, obstáculos que los sistemas sin marcadores no presentan. Un problema que comparten ambos sistemas es su costo, dificultad de implementación y de operación, pues se requiere de cámaras de mediana a alta gama para la obtención de buenos resultados en tiempo real, sin mencionar la necesidad de un ambiente de experimentación adecuado.

Si bien los sistemas ópticos son los más empleados para proyectos que involucran captura de movimiento, estos no son los únicos que existen pues se puede recurrir al uso de alternativas similares e inclusive mejores dependiendo del propósito, algunos de estos sistemas, de hecho, no hacen uso de cámaras, pero mantienen el uso de sensores para la obtención de información relevante sobre el movimiento, ejemplo de esto son algunos de los siguientes sistemas.

- ✓ **Sistemas de captura inerciales.**

  Este tipo de sistemas son categorizados como una alternativa para la medición y captura del movimiento o cinemática del cuerpo humano. La captura inercial permite el cálculo angular de articulaciones o extremidades haciendo uso de sensores como giroscopios y magnetómetros. Este tipo de sistemas se destacan principalmente por su alta exactitud en los resultados obtenidos, además, debido a que no requieren obligatoriamente de un ambiente de experimentación, facilita la portabilidad y manipulación.

- ✓ **Sistemas de captura magnéticos.**

  Este tipo de sistemas *Mocap* son utilizados principalmente en estudios de campo,

generalmente se recurre a ellos para análisis deportivos. A diferencia de los sistemas inerciales, carecen de alta precisión en el análisis de determinadas áreas del cuerpo, principalmente en la cadera, aún con esto y gracias a su gran facilidad de uso en diversidad de entornos, como lo es dentro del agua, pueden brindar información cuantitativa fiable comparable a sistemas mucho más costosos y voluminosos, además de no requerir de un montaje gracias a su portabilidad.

Los sistemas descritos anteriormente, son las opciones más conocidas y utilizadas para la captura de movimiento frente al uso de los sistemas ópticos, pero debido a la extensa necesidad de sistemas de captura para entornos o experimentos con características específicas se debe hacer mención de los sistemas especializados, como la captura por ultrasonido o la fluoroscopia, esta última más enfocada al análisis de los miembros inferiores, rodilla y tobillo.

## 2.2. Recursos para el desarrollo de sistemas Mocap

El desarrollo de sistemas de captura de movimiento básicos y complejos puede llevarse a cabo a través de la programación en diversos tipos de lenguajes, generalmente cada lenguaje maneja recursos que facilitan el desarrollo de sistemas de captura de movimiento, además, muchas de estas herramientas son de uso libre o código abierto, aunque por supuesto también se pueden encontrar recursos de software y hardware ofrecidos por empresas, que aunque gracias a sus entornos o interfaces hacen del desarrollo de sistemas de captura una tarea mucho más sencilla, siempre están sujetos a un precio para su adquisición.

- **MediaPipe.**

  Uno de los recursos más sencillos de utilizar es *MediaPipe*, el cual puede ser programado de acuerdo a la necesidad del usuario a través de múltiples lenguajes como, *Python*, C++, JavaScript, entre otros. *MediaPipe* es una herramienta de código abierto desarrollada por la empresa Google, la misma define a este recurso como un conjunto de herramientas y

bibliotecas para aplicar rápidamente técnicas de inteligencia artificial y de aprendizaje automático, para su uso en aplicaciones propias, personalizando y modificando estos recursos de acuerdo a las necesidades del usuario [13].

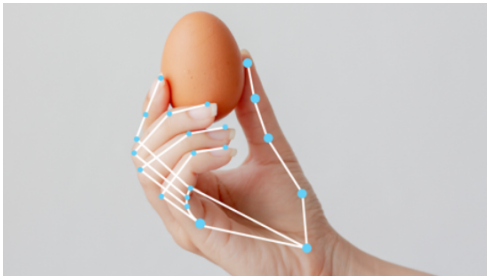


Figura 2.1: Detección y reconocimiento de una mano con MediaPipe [13].

Uno de los aspectos que hace destacar a *MediaPipe* del resto de herramientas es que requiere de recursos mínimos, pudiendo ser ejecutado en hardware bastante limitado, admitiendo gráficos multimodales y dependiendo únicamente de herramientas básicas como *Opencv* para entrada o salida de vídeo, $Tensorflow$ para la gestión del aprendizaje automático o $FFMPEG$ para salida y entrada de sonido. Dentro de la extensa gama de usos que se le ha dado a este software, destacan.

- ✓ Detección y reconocimiento de rostros.
- ✓ Detección, reconocimiento y seguimiento del cuerpo humano.
- ✓ Detección de movimientos específicos en manos o rostro.

Múltiples proyectos han sido desarrollados utilizando este recurso, como el trabajo presentado por Xoan Teira y colegas, en el que se utiliza *MediaPipe* para la detección del uso de mascarillas a través del reconocimiento facial [14], de igual forma se han desarrollado proyectos que buscan facilitar la vida de personas con capacidades diferentes, utilizando como principal recurso a *MediaPipe*. Sistemas de reconocimiento de lenguaje de señas y su interpretación [15], o el desarrollo de exoesqueletos para el tratamiento de defectos congénitos [16], son solo algunos ejemplos de lo que *MediaPipe* puede llegar a hacer.

- **OpenPose.**

  OpenPose es una biblioteca de código abierto, disponible para su utilización en distintos lenguajes de programación como Java, *Python* o C++. Se diseñó este recurso para su uso en la detección y/o seguimiento de puntos específicos del cuerpo humano, en las que se incluyen, entre otras partes, a la mano, las piernas, el rostro y los pies, permitiendo a su vez el análisis de más de una persona de forma simultánea [17].

  El funcionamiento de OpenPose consta de un proceso de cinco pasos [18].

  - ✓ Adquisición de una imagen o fotogramas secuenciales de un video.
  - ✓ Utilizando redes neuronales convolucionales o CNN ("*Convolutional neural network*"), se identifican y localizan puntos claves específicos en el cuerpo, manos y cara de las personas en la imagen.
  - ✓ Asociación de puntos clave. Utilizando *Parts Affinity Fields* o por sus siglas PAFs, se asocian correctamente las partes del cuerpo en las imágenes o fotogramas.
  - ✓ Emparejamiento bipartito. En este punto se realiza el proceso de emparejamiento de candidatos a partes del cuerpo, para la formación de esqueletos completos y coherentes.
  - ✓ Montaje de cuerpo completo. Se obtiene un modelo de cuerpo completo de las personas en las imágenes o videos.

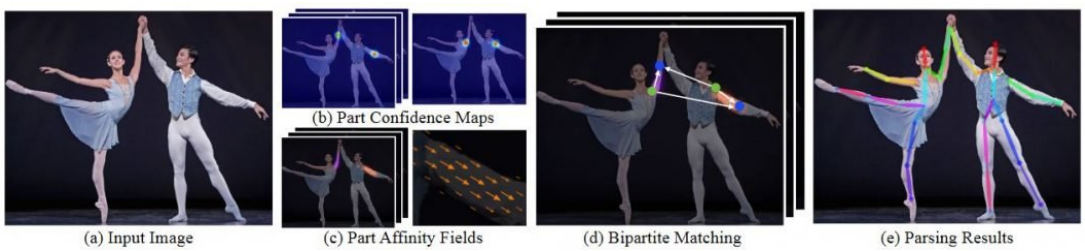



Figura 2.2: Proceso de análisis con OpenPose [18].

- **Vicon.**

  Si bien al utilizar Vicon™ no se desarrollan de manera personal los sistemas de captura, los sistemas ópticos con marcadores Vicon™ son considerados los líderes, por su alta precisión debido al ecosistema que forman sus productos, ya que la empresa ofrece sistemas completos con elementos especializados. Vicon™ ofrece cámaras de muy alta gamma para la captura de movimiento, como lo son los modelos *Valkyrie* y *Vero*.

  *Valkyrie* es descrita por la misma marca como la cámara especializa más poderosa en el mercado, contando en el modelo VK26 con 26.2 Mega Píxeles (MP) y una velocidad nativa de 150 fotogramas por segundo (fps), gracias a su diseño puede ser utilizada en casi cualquier entorno, incluidos ambientes húmedos o inclusive exteriores [19], además, se ofrecen modelos de cámaras enfocados a diferentes propósitos como la velocidad, revolución o estroboscópicos dependiendo de las necesidades del comprador.

Figura 2.3: Cámara valkyrie VK26 [19].

Los sistemas *Mocap* de Vicon™ son controlados por software desarrollado por la misma marca, y al igual que con sus cámaras estos pueden ser especializados para distintas áreas, Nexus es el software todo en uno ofrecido por la marca, puede ser utilizado tanto en ámbitos científicos como en sectores como el deporte o la animación, ya que cuenta con plantillas de modelos humanos de forma nativa, aunque no se ve limitada al trabajo exclusivo con personas.

- **Kinect SDK.**

  Kinect SDK es un conjunto de herramientas y bibliotecas ofrecido por Microsoft® enfocadas al desarrollo y creación de aplicaciones con captura, reconocimiento de movimiento y sonido, haciendo uso de un dispositivo Kinect [20].


Figura 2.4: Dispositivo Kinect [21].

En el sector de captura de movimiento sin marcadores, los dispositivos Kinect ofrecidos por la empresa Microsoft® son considerados de las mejores opciones en el mercado debido a su relación, precio beneficio. Kinect estima geometría en 3D de una escena utilizando una cámara integrada a 30 fps, posee un sensor de profundidad con resolución espacial de 640 x 480 píxeles, y micrófono para reconocimiento de voz [21], este producto en principio fue desarrollado por Microsoft® para su uso en algunos videojuegos junto con la consola Xbox 360.

El desarrollo de Kinect no iba más allá de convertirse en un control para detectar e interpretar el movimiento de un jugador en ciertos juegos compatibles, pero gracias a su fácil adquisición, manipulación y a la liberación de Kinect SDK por parte de Microsoft® hoy en día es utilizado como herramienta para la adquisición de datos y seguimiento de trayectoria en proyectos que involucran captura o análisis de movimiento, en áreas científicas e inclusive deportivas. Un ejemplo de aplicación del sistema Kinect se ve en el proyecto realizado por Muños Cardona y colegas, en donde hacen uso del dispositivo para el análisis de la biomecánica del movimiento humano para el desarrollo de un sistema de rehabilitación [10].

## 2.3. La marcha humana

Tomando información del artículo "*Fases de la marcha humana*" realizada por Martín Nogueras y sus colegas en el año de 1999 [22]. Se define como marcha al proceso de locomoción en el que el cuerpo humano se mueve hacia adelante, mientras ambos miembros inferiores soportan el peso del mismo de manera alternativa. Durante el proceso de movimiento de un punto A a un punto B, el cuerpo humano pasa a través de cuatro fases o momentos, Figura 2.5.

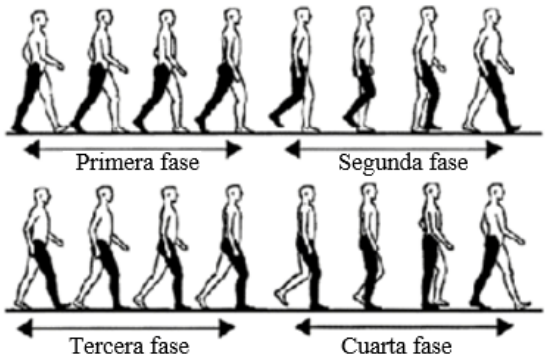



Figura 2.5: Las cuatro fases de la marcha humana [22].

Durante el primer momento, el miembro inferior que se encuentra en una posición atrasada se inclina hacia adelante como resultado de la extensión de la cadera, preparándose para despegar el pie del suelo. En el segundo momento, el mismo miembro inferior atrasado concreta el despegue del pie, iniciando un movimiento de flexión en la rodilla, el tobillo y la cadera, desplazándose hacia el frente mientras se prepara para volver a tener contacto con la superficie, pasando a ser el miembro adelantado. Durante parte de la segunda fase, el peso total del cuerpo es sostenido únicamente por el miembro que se encontraba en una posición adelantada.

En el tercer momento, el pie establece el contacto con el suelo a través del talón, para inmediatamente, entrar en contacto con la superficie, mientras la rodilla y la cadera continúan flexionándose para absorber la nueva distribución del peso corporal entre ambos miembros inferiores. En la fase final, el miembro adelantado se prepara para sostener el peso del cuerpo, manteniendo el equilibrio y permitiendo que el otro miembro se desplace hacia adelante.

### 2.3.1. Movimientos de la cadera

Utilizando información extraída del libro, "*Atlas de la anatomía humana de Netter*" [23], se destacan las siguientes características para los movimientos de flexión y extensión de la cadera en el plano sagital.

- ✓ **Flexión.**

  Es el movimiento hacia adelante del muslo al abdomen, los grados durante este movimiento son aproximadamente de 120 a 140 grados.

- ✓ **Extensión.**

  En la cadera este movimiento consiste en mover el muslo hacia atrás desde una posición neutral, aumentando el ángulo entre el muslo y el torso como se puede ver en la Figura 2.6, dependiendo de la anatomía de la persona el movimiento oscila en rangos de 10 a 20 grados.

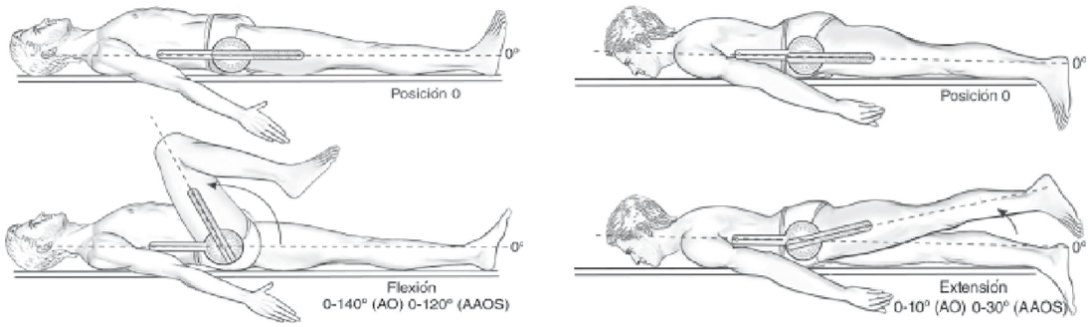



Figura 2.6: Proceso de flexión de la cadera [24].

### 2.3.2. Movimientos de la rodilla

De acuerdo a información proporcionada por Panesso y sus colegas en la investigación de "*Biomecánica de la rodilla*" del año 2008 [25], se destacan las siguientes características para cada movimiento que puede realizar la articulación.

- ✓ **Flexión.**

  La flexión de la rodilla consiste en doblar o elevar el miembro inferior, este movimiento oscila entre valores máximos de 130 a 140 grados, dependiendo de la anatomía.

- ✓ **Extensión.**

  En el proceso de extensión de la rodilla, los ligamentos que forman parte de la estructura interna se tensan dando lugar al enderezamiento del miembro inferior, este proceso tiene ángulos máximos que varían de 0 a 10 grados.

- ✓ **Rotaciones.**

  Esta articulación cuenta con dos tipos de rotaciones, ambas, medial y lateral, tienen un máximo de 30 a 40 grados de rotación.

Algunas otras fuentes de información manejan variaciones en los valores de los grados de flexión y extensión, siendo para el primero rangos entre 120 a 150 grados y para extensión rangos entre 5 a 10 grados [26][27].

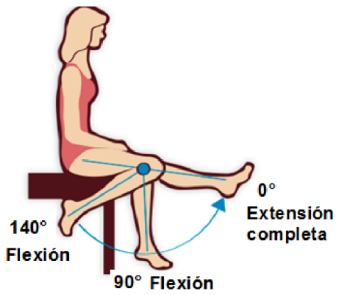



Figura 2.7: Flexión y extensión de la rodilla [28].

### 2.3.3. Movimientos del tobillo

Con la información proporcionada del libro "*Biomecánica de la marcha humana*" [27], se sabe que los principales movimientos de la articulación del tobillo son:

- ✓ **Dorsiflexión.**

  El movimiento del pie hacia arriba, con los dedos dirigidos hacia la espinilla, el rango normal de la dorsiflexión del tobillo es aproximadamente de 10 a 30 grados.

- ✓ **Plantar-flexión.**

  Lo opuesto a la dorsiflexión, es el movimiento del pie hacia abajo, con los dedos alejándose de la espinilla, el rango de movimiento para este movimiento va de 40 a 50 grados.

Cada uno de los rangos de movimiento depende de la anatomía propia de cada persona, de la edad y de la flexibilidad que estas posean.

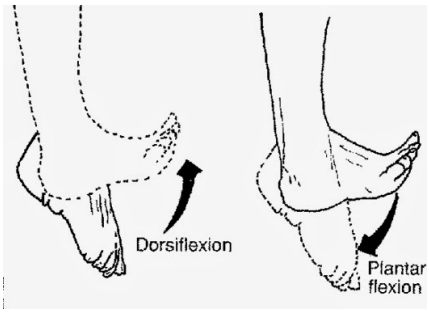



Figura 2.8: Movimientos del tobillo [29].

## 2.4. Modelos de representación de miembro inferior

Actualmente, se han desarrollado varios tipos de sistemas mecánicos que buscan igualar en condiciones y funcionamiento a las articulaciones involucradas en los miembros inferiores del cuerpo humano, los sistemas mecánicos que involucran barras o eslabones en su construcción son los más utilizados, para propósitos de investigación o de forma comercial en prótesis avanzadas.

La representación más sencilla de la extremidad inferior, la proporcionan los modelos de dos eslabones, este tipo de modelos representan al miembro como dos segmentos rígidos conectados por una articulación. Un segmento representa el muslo y el otro representa la pierna, actuando como puntos de rotación la cadera y la rodilla.

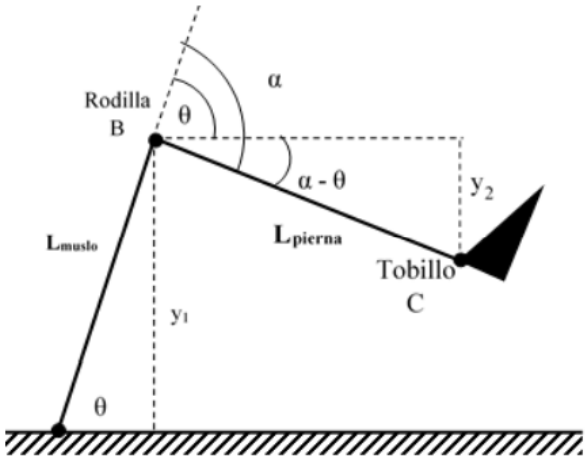



Figura 2.9: Modelo de representación de dos eslabones [30].

Los mecanismos de tres barras, a diferencia del modelo anterior toman en cuenta un pie conectado al miembro, el cual a su vez esta conectado a un cuerpo con puntos de rotación en el tobillo, rodilla y cadera. Este tipo de modelos son útiles para analizar movimientos complejos como la marcha, y aunque pueden llegar a estar alejados en ciertos aspectos del movimiento real que tienen las articulaciones, funcionan bastante bien para propósitos de investigación básicos y en ocasiones son la base de sistemas de miembro inferior más complejos.

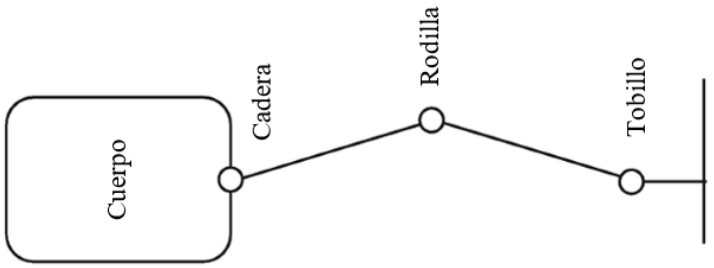



Figura 2.10: Modelo de representación de tres eslabones [31].

# Capítulo 3

# Desarrollo de los sistemas de captura y réplica de movimiento

El desarrollo de este proyecto inició con la elaboración de un diseño conceptual, el cual permite establecer los principios fundamentales para el software de captura de movimiento y los modelos de réplica de movimiento. El área en la cual se enfoca la captura de movimiento, el planteamiento de los requisitos y el establecimiento de características de los sistemas de captura y réplica, son descritos en el diseño conceptual.

## 3.1. Diseño conceptual

El objetivo de este proyecto es el desarrollo de al menos dos sistemas de captura de movimiento de miembro inferior en el plano sagital, capaces de replicarlo a través de un modelo virtual y físico. Los sistemas de captura y réplica requieren de una mínima complejidad, utilizando recursos de software de uso libre, la captura de movimiento se centrará en el miembro inferior izquierdo a través del plano sagital del cuerpo humano, por ello las articulaciones de mayor interés son la cadera, la rodilla y el tobillo, convirtiendo al movimiento angular de estas articulaciones en el principal parámetro de referencia para los sistemas de réplica.

El diseño del software de captura de movimiento, toma como base las distintas fases de funcionamiento que utilizan la mayoría de sistemas personalizados [32], al igual que en algunos sistemas comerciales como los de Vicon, las cuales permiten registrar y visualizar el movimiento, aunque para las necesidades del proyecto se requiere de la adición de dos etapas extra para que el sistema almacene los ángulos de movimiento de las articulaciones, una para el cálculo de ángulos y otra para el registro de resultados:

1. **Captura.** Es el inicio del proceso, consta del registro de movimiento que generalmente es a través de imágenes captadas por una cámara.

2. **Detección y reconocimiento de articulaciones.** Se resaltan o detectan los puntos clave del cuerpo del sujeto en movimiento, con el uso de sensores y marcadores o con alguna herramienta de software para análisis y detección.

3. **Cálculo de vectores.** Una vez identificados los puntos clave del cuerpo se crean vectores a modo de segmentos, conectando cada una de las articulaciones o puntos, para crear una aproximación de la estructura del cuerpo analizado.

4. **Cálculo de ángulos.** En esta fase se utiliza la información de etapas anteriores para el cálculo del ángulo de movimiento de las articulaciones de interés.

5. **Aplicación de filtros.** Para obtener resultados más fieles al movimiento que se esta capturando, diversos sistemas utilizan filtros para suavizar los datos resultantes, evitando y reduciendo errores o fluctuaciones.

6. **Visualización.** Con los vectores y datos resultantes, a través del uso de software dedicado a la animación y renderizado como, *Blender*, *Nexus*, *Unity*, etc., se genera la visualización del movimiento capturado, con cuerpos simples generados a partir de los vectores calculados.

7. **Registro de resultados.** Se almacenan los ángulos de movimiento calculados, para análisis posteriores y para facilitar el proceso de réplica de movimiento.

La Figura 3.1, muestra el primer diagrama de flujo diseñado para los programas de captura de ángulos de movimiento, en este se observa que además del uso de las etapas descritas anteriormente, se añade el cálculo de ángulos y el registro de datos. Todas las etapas presentes en este primer diagrama de flujo, cubren las necesidades del proyecto en cuestión de software de captura, un programa que registra el movimiento de una persona en el plano sagital y a partir de las imágenes capturadas, obtiene el ángulo de movimiento de las articulaciones de interés a través del cálculo del ángulo entre dos segmentos o vectores, almacenando los resultados para su posterior análisis.

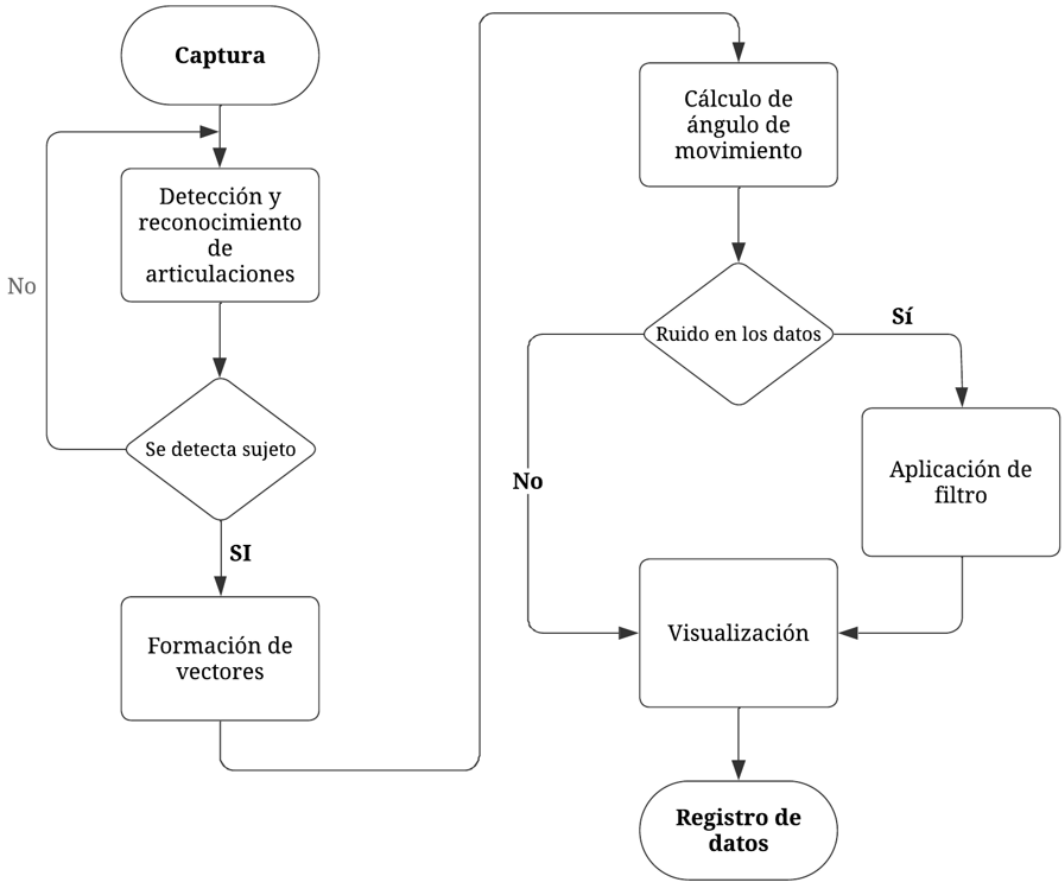



Figura 3.1: Diagrama de flujo del software de captura.

El proceso de la réplica del movimiento, que ha sido registrado por el software de captura, debe realizarse a través de un modelo virtual y su equivalente físico. El sistema de réplica virtual deberá emplear la información resultante del programa de captura, es decir, el modelo virtual utilizará los grados de movimiento calculados para accionar los respectivos actuadores de la cadera, rodilla y tobillo.

Lo anterior es posible con una conexión entre el software de captura y el programa donde se desarrolla el modelo virtual, por lo que se requiere que este último cuente con las características necesarias para recibir datos y utilizarlos como señales de control para el accionamiento de actuadores virtuales. Tanto el programa de conexión como el de captura deberán integrarse en una sola aplicación, para evitar la necesidad de alternar entre distintos programas, por lo que justo después de la etapa de almacenamiento de datos al final del diagrama de la Figura 3.1 se ubicaría el envío de ángulos, para dar inicio al proceso de réplica de movimiento.

Para conseguir que la réplica de movimiento sea más fiel al real, se requiere que los motores conformen una estructura muy similar a la parte del cuerpo que se analiza, de la variedad de modelos de representación de miembro inferior, se considera que un sistema de péndulo doble de tres eslabones es ideal para este proyecto, debido a que contiene al mismo tiempo las tres articulaciones de interés así como las partes del cuerpo que las conectan, Figura 2.10.

En la Figura 3.2, se muestra un diseño conceptual del modelo estructural del miembro inferior, la estructura podría estar colocada de forma fija en una superficie por el extremo superior, dejando en libre movimiento las articulaciones de A) La cadera, B) La rodilla y C) El tobillo. Ambos sistemas de réplica deberán utilizar el mismo modelo de representación, por lo que teniendo esto en cuenta, el modelo físico será impreso en 3D utilizando PLA como materia prima para la impresión del modelo, ya que es un material ligero pero resistente, características suficientes considerando que el modelo físico solo deberá soportar su propio peso y su única función es representar visualmente el movimiento capturado. Los ángulos de rotación de cada una de las articulaciones del modelo de réplica físico, serán realizados por motores con encoders

puesto que se requiere la constante retroalimentación sobre la posición del motor, para verificar que el movimiento replicado coincida con el capturado.

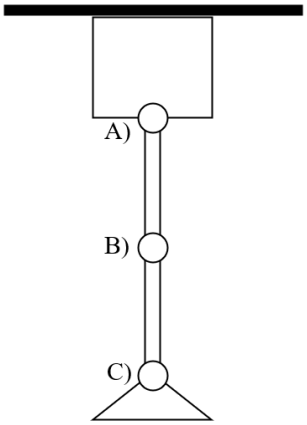



Figura 3.2: Diseño conceptual del modelo de réplica de movimiento.

En el sistema virtual, el control de los actuadores dependerá de los ángulos generados por el programa de captura de movimiento, estos valores determinarán en todo momento la posición de los motores. El software que contendrá el modelo de réplica virtual deberá recibir los datos de los ángulos capturados y enviarlos a los actuadores de cada articulación, permitiendo que el modelo reproduzca el movimiento de la manera más precisa posible. En cuanto al sistema de réplica físico, los actuadores que representan las articulaciones del miembro inferior serían controlados mediante una tarjeta programable, esta tarjeta recibirá los ángulos de movimiento capturados y los enviará a los motores de cada una de las articulaciones, tal como sucede con el modelo de réplica virtual.

La comunicación entre la tarjeta, los datos del sistema de captura y los actuadores será gestionada por un programa dedicado, que asegurará la correcta transmisión y aplicación de los ángulos a los motores. Dentro de la extensa variedad de programas dedicados a la simulación de modelos robóticos, existen algunos con la capacidad de controlar actuadores físicos empleando los mismos parámetros de un entorno simulado, esta característica es importante para tomar en cuenta en el proceso de selección del software de modelado.

## 3.2. Selección de herramientas para el desarrollo del proyecto

Para asegurar el funcionamiento planteado en la sección anterior, los sistemas que conforman este proyecto, requieren de la integración de diversas herramientas de software. En esta sección se describen detalladamente los recursos empleados en el desarrollo de los programas de captura y réplica de movimiento, incluyendo el lenguaje de programación, las bibliotecas especializadas y los distintos complementos utilizados.

### 3.2.1. Recursos análisis de movimiento

*Python* ha sido el lenguaje de programación seleccionado para la elaboración de los programas del proyecto, esto debido a la facilidad de uso y al amplio soporte para la resolución de problemas o dudas, gracias a la activa comunidad de usuarios del lenguaje. También, se considera a este lenguaje de programación como la mejor opción por ser compatible con diversas bibliotecas enfocadas a la creación de proyectos basados en visión artificial, lo que aumenta en gran medida la disponibilidad de recursos para el desarrollo del proyecto.

Para optimizar la programación, la administración de paquetes y el análisis de datos, para cada programa se empleo *JupyterNotebook* como entorno de desarrollo, por su capacidad de ejecutar programas completos mediante celdas de manera modular y organizada, a su vez este entorno se utiliza dentro del gestor Anaconda, el cual facilita la instalación y gestión de bibliotecas necesarias para el proyecto.

Tomando como referencia el diagrama de la Figura 3.1, la definición de los recursos necesarios para el análisis de movimiento comienza con la selección de las herramientas para la etapa de detección y reconocimiento de articulaciones, esto con el fin de evitar errores de compatibilidad entre las herramientas más esenciales y el resto de recursos. Al finalizar, se retoma el flujo del diagrama, continuando con la etapa de captura.

**Etapa de detección y reconocimiento de articulaciones.**

Para cumplir con las necesidades de esta etapa existen diversas opciones, algunas comparten características y enfoques, pero muchas mantienen diferencias significativas. Debido a esto y a los objetivos a cumplir de este proyecto, se realiza un análisis sobre las herramientas disponibles, buscando en mayor medida el cumplimiento de los siguientes criterios:

1. Precisión en la detección de articulaciones.
2. Soporte para el análisis en plano sagital o 2D.
3. Amplia compatibilidad con recursos complementarios.
4. Facilidad de implementación y uso.
5. Flexibilidad y personalización.

En este análisis, se evalúan herramientas para la detección de articulaciones de acuerdo al cumplimiento de los requerimientos y criterios de los programas a desarrollar, se considera el estado actual y las características de las herramientas según la información oficial de cada una [13] [17] [33] [34]. Se utiliza una escala del 0 al 10, en la que 0 representa un incumplimiento total y 10 un alto cumplimiento de los criterios y necesidades del proyecto, esta representación permite identificar qué recursos destacan sobre el resto de opciones de forma sencilla 3.1.

Tabla 3.1: Comparación cuantitativa de herramientas de análisis de movimiento.

| **Criterio** | **MediaPipe** | **OpenPose** | **MMPose** | **DeepLabCut** | **AlphaPose** |
|---|---|---|---|---|---|
| **Precisión en detección** | 8 | 9 | 9 | 10 | 7 |
| **Funcionamiento 2D** | 8 | 9 | 9 | 8 | 10 |
| **Compatibilidad** | 9 | 7 | 10 | 8 | 9 |
| **Facilidad de uso** | 10 | 6 | 7 | 5 | 5 |
| **Flexibilidad** | 8 | 7 | 9 | 10 | 9 |

De acuerdo a la evaluación anterior, se opta por hacer uso de la biblioteca *MediaPipe* y el framework *MMPose*, ambas herramientas son ampliamente utilizadas para la detección de articulaciones, pero presentan diferencias clave en precisión y eficiencia computacional, diferencias que hacen especialmente relevante su análisis comparativo dentro del contexto específico de este proyecto.

*MediaPipe*, diseñado para ser eficiente y accesible, permite la estimación de poses en tiempo real con alto rendimiento en dispositivos de recursos limitados, utiliza modelos de inferencia pre-entrenados integrados en la biblioteca sin necesidad de configuración o descarga previa, y solo requiere un procesador o *CPU* para el análisis de imágenes.

Por otra parte, *MMPose* es una solución más robusta que necesita de la carga de un modelo de aprendizaje profundo como *HRNet*, *MobileNet* o *ResNet*, proporcionando mayor precisión y personalización, aunque a costa de un mayor consumo de recursos, incluyendo la necesidad de una tarjeta gráfica o *GPU* para un procesamiento de imágenes rápido y eficiente.

Aunque algunas de las otras alternativas pueden llegar a ofrecer un rendimiento superior al de las herramientas seleccionadas, también presentan características que dificultan o impiden su uso en este proyecto, tal como se indica a continuación:

- **Openpose:** Caracterizada por una alta precisión, desde el año 2022, esta herramienta dejó de recibir soporte técnico. Actualmente, su instalación se ha vuelto complicada debido a la falta de algunas dependencias necesarias y a la incompatibilidad con versiones recientes de software.

- **DeepLabCut:** No posee modelos pre-entrenados, requiriendo de un proceso de entrenamiento para un modelo personalizado, que implica el etiquetado manual de articulaciones en múltiples imágenes del cuerpo humano, hay que destacar también que su arquitectura original no está diseñada para ofrecer una respuesta rápida, lo que limita su uso en análisis en tiempo real.

- **AlphaPose:** Para obtener resultados precisos es necesario entrenar adecuadamente un modelo de red neuronal, un proceso complejo y que requiere de mucho tiempo. Además, si el modelo no está correctamente optimizado, el sistema de detección puede volverse propenso a errores y a consumir más recursos computacionales.

**Etapa de captura.**

Retomando el orden del diagrama de la Figura 3.1, la siguiente etapa a cubrir es la captura o adquisición de imágenes. Esta tarea puede llevarse a cabo de dos formas distintas:

1. **En tiempo real:** Capturando imágenes a través de una cámara de vídeo y procesándolas de manera simultanea.

2. **Utilizando videos pregrabados:** El sistema analizaría imágenes en videos previamente grabados, permitiendo un mayor control en los movimientos que se pretende analizar.

El sistema de detección y captura puede contar con ambas alternativas, pero la opción de utilizar videos pre-grabados se considera como el método ideal para la posterior comparación de resultados entre sistemas de captura, ya que de esta forma se garantiza que los movimientos analizados son los mismos para ambos programas.

Para cubrir esta etapa de adquisición de imágenes, se escoge a *Opencv* (Open Source Computer Vision Library), que es una biblioteca que permite obtener imágenes en tiempo real por medio de su módulo "*cv2*", haciendo uso de una cámara o en su defecto procesando videos pre-grabados en diversos formatos. Su flexibilidad y eficiencia la convierten en una herramienta ideal para aplicaciones de captura y procesamiento de imágenes en tiempo real, además es compatible con diversas bibliotecas y frameworks de inteligencia y visión artificial. Se descartan opciones como $Scikit-image$ y $SimpleCV$ por su baja velocidad y versatilidad, *Opencv* ofrece un mayor rendimiento, mejor soporte multiplataforma y una amplia variedad de funciones que facilitan el procesamiento de imágenes.

**Etapa de formación de vectores y cálculo de ángulos.**

Dado que los sistemas de captura operan en un plano 2D y no consideran rotaciones articulares, como la medial y lateral de la rodilla, es necesario seleccionar un método matemático eficiente para el cálculo de ángulos en este plano. Debido a esta necesidad, se elige a la ley de cosenos como el método a emplear para el cálculo de ángulos en los programas de análisis de movimiento.

$$c^2 = a^2 + b^2 - 2ab\cos\theta \tag{3.1}$$

El cálculo de ángulos mediante la ley de cosenos se basa en el uso de las distancias entre tres articulaciones para formar un triángulo. A partir de la longitud de sus lados, se obtiene el ángulo deseado despejando $\theta$ en la fórmula, sin necesidad de definir vectores [35].

$$cos\theta = \frac{a^2 + b^2 - c^2}{2ab} \tag{3.2}$$

La ley de cosenos, solo requiere de cálculos básicos, sumas, productos y potencias, reduciendo la carga de procesamiento en los programas de captura. Además, opera de forma natural en el plano 2D, ofreciendo resultados precisos sin necesidad de extrapolaciones complejas.

**Etapa de detección de ruido.**

Se asume que los datos capturados por los programas de análisis de movimiento siempre contendrán ruido en menor o mayor medida, por lo que a los datos resultantes se les aplicará siempre un filtro para mejorar la precisión y estabilidad al disminuir las fluctuaciones.

Por lo que se decidió implementar un filtro en las primeras etapas de los programas, concretamente, justo después de la detección de articulaciones, esto debido a que *MediaPipe* o *MMPose*, en ocasiones pueden dar como resultado la detección de coordenadas con errores o fluctuaciones, por variaciones en la iluminación, calidad del video o pequeños errores en la estimación del modelo que se está usando. La aplicación de un filtro en esa posición, ayuda a suavizar los datos, reducir el ruido y mejorar la estabilidad del seguimiento de las articulaciones.

Para la selección de un filtro que suavice y disminuya el ruido en las coordenadas detectadas, se realiza un proceso de selección en el que se evalúan los siguientes aspectos:

1. **Robustez:** Debe ser capaz de manejar valores atípicos o fluctuaciones en las posiciones capturadas de las articulaciones, asegurando que los datos sean consistentes.

2. **Eficiencia:** Tiene que poder procesar los datos en tiempo real, con una mínima latencia. El filtro no debe afectar de forma significativa la velocidad de ejecución del programa de captura.

3. **Predicción:** Al tener en cuenta la probabilidad de fallo por parte de *MediaPipe* o *MMPose*, perdiendo posiciones o entregando coordenadas con fluctuaciones, es necesario que el filtro tenga la capacidad de corregir este tipo de errores, y una solución es la predicción.

Conforme a los criterios planteados anteriormente, se considera como una buena opción la implementación de un filtro de Kalman, ya que cumple con todos los requisitos necesarios, suaviza datos ruidosos de manera eficiente, procesa información en tiempo real sin generar retardos significativos y reduce valores atípicos. Además, su capacidad predictiva lo hace destacar en la estimación de datos faltantes, como se ha demostrado en diversos estudios [36] [37], también cabe mencionar que este filtro ya ha sido utilizado en proyectos de captura de movimiento con resultados bastante aceptables [38].

Si bien algunas alternativas como el filtro de partículas están por encima de las capacidades del filtro seleccionado, generalmente su implementación resulta ser una tarea muy compleja ya que parámetros como, la cantidad de partículas, la distribución inicial y la función de medición deben estar correctamente definidos, si un ajuste falla el filtro puede volverse inexacto, además, por involucrar múltiples cálculos estadísticos su uso conlleva un alto costo computacional, debido al manejo de un gran número de partículas (posibles soluciones) en cada ciclo.

Existen distintas variantes del filtro de Kalman, sin embargo, en este proyecto se utiliza la versión lineal discreta, cuyo funcionamiento consta de dos fases recursivas, predicción y corrección, Figura 3.3.

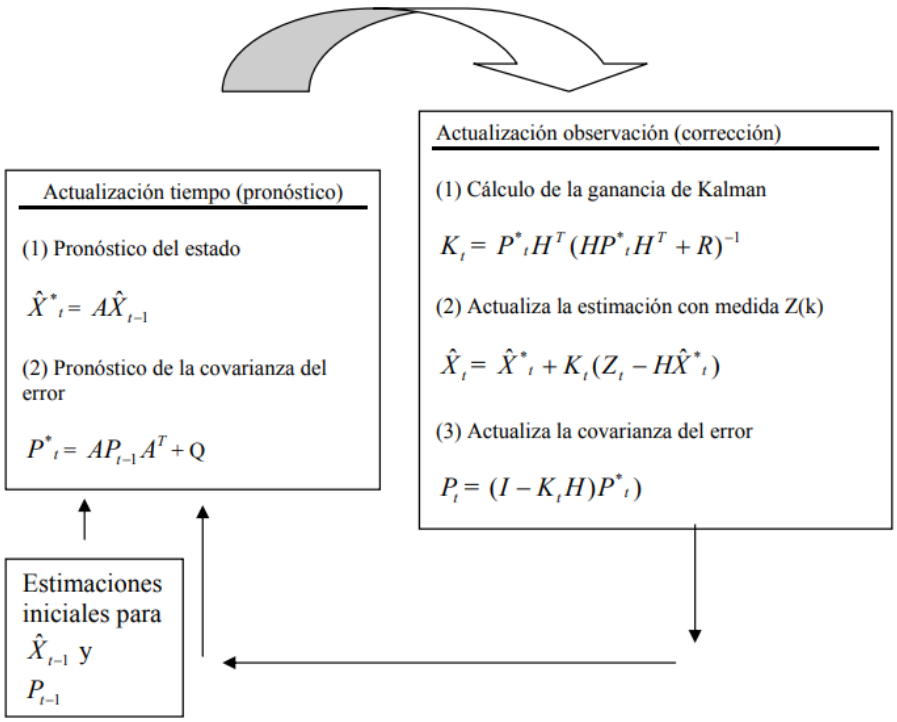



Figura 3.3: Fases del Filtro Kalman [39].

1. **Fase de predicción.**

   Se estima el valor actual de la coordenada articular, utilizando la ecuación de predicción de Kalman, la cual toma en cuenta la posición detectada en el frame anterior.

$$\hat{X}^* = A\hat{X}_{t-1} \tag{3.3}$$

   Luego, se actualiza la incertidumbre de la predicción considerando tanto el error acumulado de la coordenada anterior como el ruido del modelo.

$$P^*_t = AP_{t-1}A^T + Q \tag{3.4}$$

2. **Fase de corrección.**

   Cuando se recibe una nueva medición $Z_t$, el filtro la compara con la predicción anterior y la utiliza para mejorar la estimación de la coordenada articular. Primero, se calcula la confianza en la medición respecto a la predicción, según el nivel de incertidumbre asociado a cada una.

$$K_t = P_t^* H^T (H P_t^* H^T + R)^{-1} \tag{3.5}$$

   Después, se ajusta la coordenada estimada combinando la predicción con la medición, en una proporción determinada por la ganancia de Kalman $K_t$.

$$\hat{X}_t = \hat{X}_t^* + K_t (Z_t - H\hat{X}_t) \tag{3.6}$$

   Finalmente, se actualiza la incertidumbre de la estimación para reflejar la mejora lograda tras incorporar la nueva medición.

$$P_t = (I - K_t H) P_t^* \tag{3.7}$$

Para el manejo de los ángulos de movimiento calculados por los sistemas de captura, se opta por utilizar un filtro Gaussiano. Aún con la aplicación del filtro Kalman para estabilizar las coordenadas articulares detectadas y con ello los ángulos estimados, los resultados finales pueden seguir presentando pequeñas variaciones o picos residuales debido a fluctuaciones muy breves o movimientos atípicos no completamente corregidos por el primer filtro. La combinación de dos filtros permite aprovechar las ventajas de Kalman en la estimación dinámica y del Gaussiano en el refinamiento final de los datos, para obtener resultados más robustos y representativos del movimiento que se esta analizando.

El proceso del filtro Gaussiano se compone de dos fases principales, las cuales trabajan de forma complementaria para suavizar datos y reducir el impacto del ruido en señales o mediciones:

- **Suavizado.**

  Con una secuencia de ángulos de movimiento o ventana de una sola dimensión, se aplica una media ponderada sobre cada ángulo, en la que dependiendo de la cercanía al valor central los valores vecinos tienen diferente peso. Esta ponderación de valores sigue la forma de una distribución Gaussiana.

$$G(x) = \frac{1}{\sqrt{2\sigma}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.8}$$

  Con $x$ como la distancia al valor central y $\sigma$ siendo la desviación estándar.

- **Aplicación.**

  Para cada ángulo $\theta$, el filtro calcula una nueva versión suavizada $\hat{\theta}$.

$$\hat{\theta}[n] = \sum_{i=-k}^{k} G(i) \cdot \theta[n+i] \tag{3.9}$$

  La variable $k$ es el radio de la ventana de suavizado, $G(i)$ son los coeficientes normalizados que asignan más peso a los valores cercanos al valor central

Se selecciona al filtro Gaussiano sobre otros métodos más simples, como el filtro de media móvil, debido a su capacidad para suavizar los datos de forma más eficiente y sin generar desplazamientos o distorsiones, así como por su bajo requerimiento computacional debido a las simples operaciones matemáticas que se realizan durante su uso.

**Etapa de visualización.**

La fase de visualización del programa de captura es simple, ya que al implementar *Opencv* para la etapa adquisición de imágenes, se puede hacer uso del módulo "*cv2*" y más concretamente de su función integrada "*cv2.imshow*", para mostrar en pantalla los resultados del procesamiento de la captura de movimiento en tiempo real. Los puntos clave del miembro inferior, las líneas que representan las conexiones entre articulaciones e incluso los grados de movimiento calculados son sobrepuestos en cada uno de los frames del video de entrada en una nueva ventana gráfica generada por el programa, obteniendo un resultado similar a lo visto en la Figura 2.1. De esta manera, se visualiza directamente cómo se están detectando y analizando los movimientos, facilitando la identificación de posibles errores.

**Etapa de registro de resultados.**

Por último para la etapa de registro de ángulos de movimiento, se opta por realizar el almacenamiento de datos utilizando un archivo xlsx, a través de la función "*to_excel*" del módulo *Pandas*, debido a la gran facilidad de manejo de grandes cantidades de datos en archivos de ese tipo. Además, ya que se analizan y comparan los datos resultantes de ambos programas de captura, el registro de datos mediante formato xlsx permite el manejo de la información en otros programas de *Python*, lo cual facilita la posterior aplicación de diversos métodos de análisis de datos.

### 3.2.2. Recursos de software y hardware para réplica de movimiento

**Réplica de movimiento mediante un modelo virtual.**

En el sistema de réplica de movimiento virtual se utiliza el programa CoppeliaSim, pues permite la construcción de modelos mecánicos o la importación de estos desde otros programas como SolidWorks, cuenta con múltiples protocolos de comunicación en tiempo real como $TCP/IP$, $UDP$ y $ROS$, los cuales constituyen características ideales para la transmisión de información entre los programas de captura y el modelo de representación virtual.

Además, es un programa de uso libre en su versión para la educación, la cual posee todas las herramientas necesarias para el proyecto. CoppeliaSim cuenta con motores de simulación física avanzados, como *ODE* y *Bullet*, los cuales permiten simular las dinámicas del movimiento del miembro inferior con gran precisión. Cabe destacar que el programa también posee una excelente visualización 3D en tiempo real, lo que facilita la observación del movimiento de las articulaciones y del modelo completo desde diferentes ángulos.

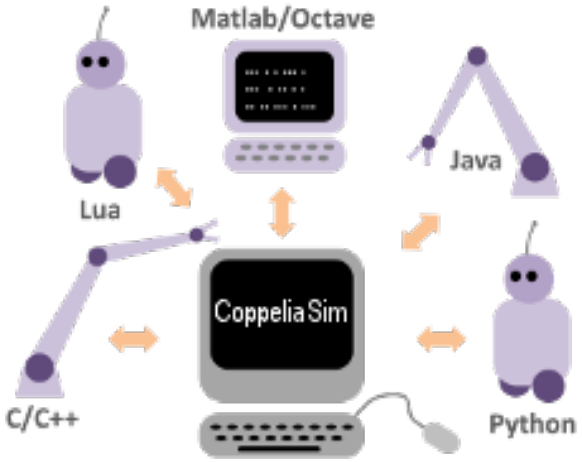



Figura 3.4: Control en CoppeliaSim por API remota [40].

Las ventajas que aporta CoppeliaSim no solo benefician al sistema de réplica virtual, con los protocolos de comunicación *API* también es posible usar el modelo virtual en el envío de información para la realización de movimientos en el modelo físico.

**Réplica de movimiento mediante un modelo físico.**

Para la reproducción del movimiento angular de las articulaciones mediante el uso de un sistema electrónico, es necesario utilizar motores que cumplan con las siguientes características específicas:

- **Control de posición.** Los motores deben permitir un control del ángulo de giro, ya que su función principal es replicar los movimientos capturados.

- **Retroalimentación de posición.** Es necesario que los motores cuenten con sistemas de retroalimentación, como encoders, para confirmar su posición real en cada momento, y para verificar la fidelidad del movimiento enviado con el realizado.

- **Compatibilidad.** El motor debe ser compatible con una gran parte de las tarjetas programables disponibles, para facilitar la implementación y reemplazo de componentes, ampliando las posibilidades de actualización y adaptación.

Por sus características se selecciona el motor reductor modelo GM25-370 (Figura 3.5), en su versión de 330 *RPM* con caja reductora de relación 1:34, para su integración en el sistema de réplica físico. El motor está equipado con un encoder que permite obtener retroalimentación precisa sobre su posición angular y velocidad de rotación en tiempo real, esencial para sincronizar su movimiento con los datos capturados.

Figura 3.5: Motor GM25-370 con encoder [41].

Para el control de velocidad y dirección del motor, se hace uso de una tarjeta programable Arduino UNO junto con el controlador Tb6612fng [42], lo que concreta un sistema en el que se pueden recibir los ángulos de movimiento a reproducir utilizando comunicación serial USB a la tarjeta, la cual a su vez convertirá la información recibida en pulsos PWM para controlar el movimiento de los motores, recibiendo la retroalimentación del encoder que proporciona información sobre la posición y la velocidad del motor para ajustar el movimiento y garantizar que se alcancen los ángulos de movimiento recibidos.

Durante la etapa de diseño conceptual se desarrolló un diagrama de flujo que representaba el funcionamiento previsto del programa de captura, Figura 3.1. Sin embargo, a partir de la selección definitiva de las herramientas, se realizaron algunos ajustes en el proceso general, como la incorporación del filtro de Kalman y la eliminación de algunas etapas. Debido a esto, se presenta en la Figura 3.6, una versión actualizada del diagrama de flujo que refleja con mayor precisión el funcionamiento final del programa.

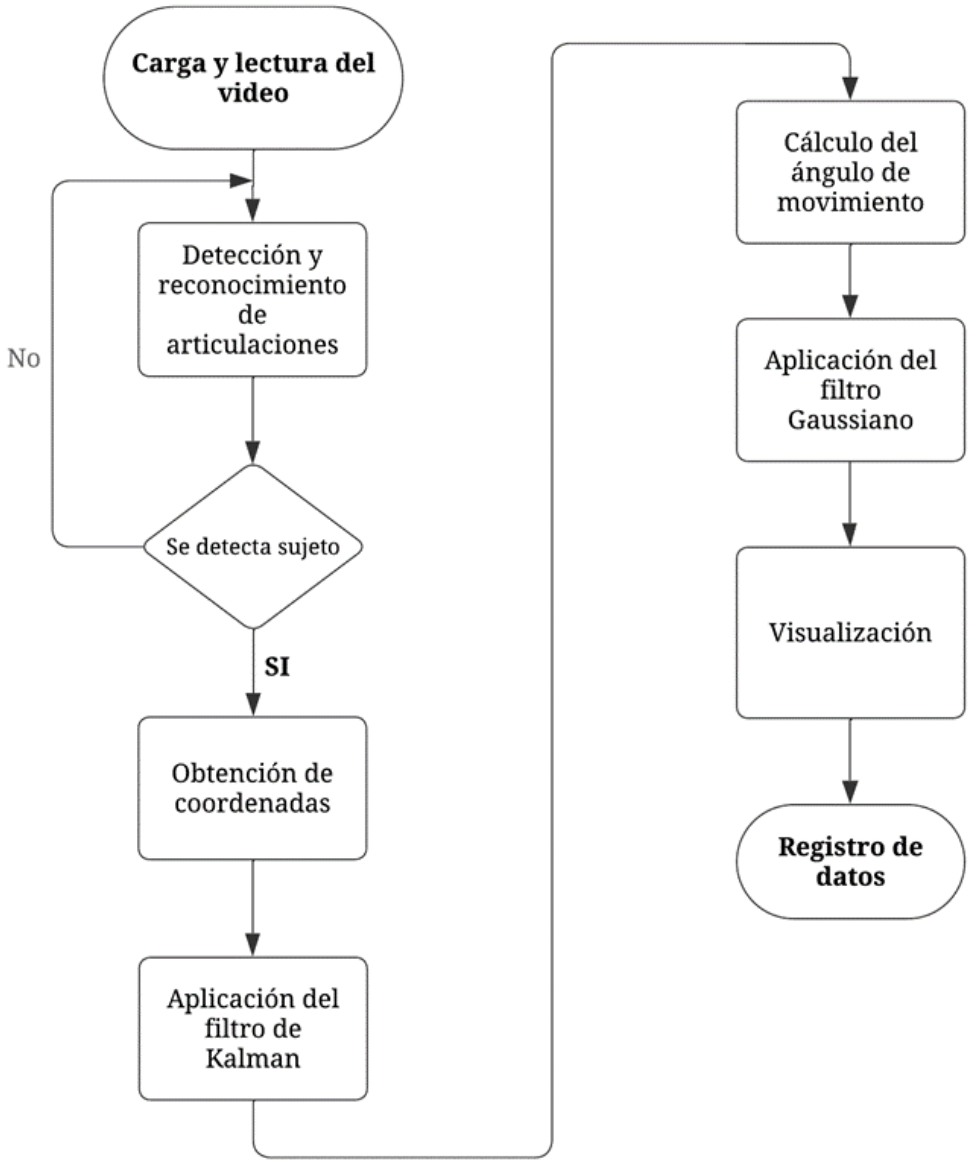



Figura 3.6: Diagrama de flujo de los sistemas de captura de movimiento.

# Capítulo 4

# Implementación

En esta sección, se describen los códigos de cada uno de los programas de captura utilizando las herramientas seleccionadas y descritas en el **Capítulo 3**. El tema de implementación abarca desde la configuración de cada una de las etapas del proceso de captura, hasta la integración de los datos o ángulos resultantes en los sistemas de réplica de movimiento.

Se crearon entornos de programación distintos para cada sistema de captura, con el fin de evitar conflictos entre versiones de los complementos, ya que de esta forma cada biblioteca de detección de articulaciones cuenta con sus respectivas herramientas en versiones totalmente compatibles.

## 4.1. Programa de captura de movimiento basado en MediaPipe

Para la implementación de este sistema se instaló un IDE para el lenguaje de programación *Python* en versiones compatibles, en la información oficial de la biblioteca [13], se recomienda utilizar versiones del lenguaje comprendidas entre la 3.9 y la 3.12, para este proyecto en especifico se instaló la versión 3.11.5.

*MediaPipe* se encuentra disponible en el repositorio oficial de paquetes de *Python* (PyPi), por lo que el proceso de instalación de la biblioteca se lleva a cabo con el comando, *pip install mediapipe*. De la misma forma se instalan las herramientas complementarias del programa, *Opencv* para el manejo de imágenes, *FilterPy* para el uso del filtro Kalman, *NumPy* para la realización de operaciones matemáticas y *Pandas* para la exportación de los datos resultantes, todas en versiones compatibles entre si.

Tabla 4.1: Versiones de las herramientas implementadas en el entorno MediaPipe.

| **Recurso** | **Versión** |
|---|---|
| **MediaPipe** | 0.10.8 |
| **Opencv** | 4.11.0.86 |
| **filterPy** | 1.4.5 |
| **Numpy** | 1.24.3 |
| **pandas** | 2.0.3 |

Con todos los recursos listos en el entorno, se verifica el funcionamiento de *MediaPipe* a través del análisis de una imagen de prueba, con el uso de un programa de ejemplo del repositorio oficial. Los resultados de esta prueba son mostrados en la Figura 4.1.

Figura 4.1: Prueba de funcionamiento biblioteca MediaPipe.

En la imagen resultante de la prueba, se puede apreciar que *MediaPipe* si es capaz de identificar y colocar puntos de referencia o *landmarks* sobre las articulaciones del cuerpo human, en este caso identificando los 28 puntos de referencia más importantes, lo que confirma que el proceso de instalación se realizó correctamente, por lo que se pudo avanzar con seguridad a la escritura del programa de captura.

Este programa basado en *Mediapipe* se estructura en secciones, lo siguiente es una descripción de cada sección junto con los procesos más relevantes que se llevan a cabo en cada una.

**Sección 1: Bibliotecas.**

El código inicia con la declaración de los módulos o bibliotecas que contienen las herramientas que se utilizan a lo largo del programa, todas estas bibliotecas se declaran a través del comando *import*.

```
import cv2  # Biblioteca para procesamiento de imágenes y videos
import mediapipe as mp  # Framework de visión artificial para detección de poses
import numpy as np  # Operaciones matemáticas y manejo de arrays
import pandas as pd  # Manejo y exportación de datos a Excel
import scipy.ndimage  # Aplicación de filtro Gaussiano
from filterpy.kalman import KalmanFilter  # Implementación del filtro de Kalman
```

Figura 4.2: Declaración de bibliotecas MediaPipe.

**Sección 2: Inicialización y configuración.**

En esta sección se inicializa *MediaPipe*, Figura 4.3, utilizando su módulo *pose* para una detección de articulaciones de cuerpo completo. En la misma sección, se incluyen también las líneas de código correspondiente para cargar el video que se analiza.

```
# Inicializar MediaPipe Pose
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(static_image_mode=False,
                    model_complexity=2,
                    enable_segmentation=False,
                    min_detection_confidence= 0.9,
                    min_tracking_confidence=0.5)
mp_drawing = mp.solutions.drawing_utils


# Ruta y carga del video
video_path = "C:\\Users\\erick\\Downloads\\Ejercicio_1.mp4"
cap = cv2.VideoCapture(video_path)
```

Figura 4.3: Etapa de inicialización y configuración MediaPipe.

Dentro del módulo *pose*, se establecen algunos parámetros de funcionamiento para la biblioteca. La configuración *static-image-mode*, con valor *False* mejora el análisis de frames con

relación, necesario debido al uso de videos o imágenes consecutivas relacionadas, a través de *model-complexity* se controla la precisión y rapidez del modelo integrado, actualmente con valor 2 se establece el uso de la versión más robusta y precisa del modelo, *min-detection-confidence* define un umbral mínimo de confianza para considerar que se ha detectado una articulación, se establece un valor de 90% para prevenir falsos positivos, con un margen del 10% para evitar que el programa nunca detecte articulaciones. Por último *min-tracking-confidence*, establece el umbral mínimo de confianza pero ahora para rastrear la pose entre fotogramas, si la confianza en el seguimiento cae por debajo del valor establecido el modelo volverá a detectar la pose desde cero, con la realización de pruebas se determino que el valor actual de 0.5 o 50% mantiene un buen equilibrio, con rastreo estable sin re-calcular excesivamente.

Para la carga de video, únicamente se define su ruta de acceso y se utiliza la función *cv2.VideoCapture()* de la biblioteca *Opencv*, para abrir el archivo para su lectura. Con el objeto *cap* se procesa cada uno de los fotogramas, obteniendo información adicional como resolución, FPS, duración, etc.

**Sección 3: Definición de funciones.**

Dentro de este apartado, se define la función para hacer uso del filtro Kalman, se configura el mismo y se establece el propósito de su uso, Figura 4.4.

```
def crear_filtro_kalman():
    kf = KalmanFilter(dim_x=4, dim_z=2)
    kf.F = np.array([[1, 0, 1, 0], [0, 1, 0, 1], [0, 0, 1, 0], [0, 0, 0, 1]])
    kf.H = np.array([[1, 0, 0, 0], [0, 1, 0, 0]])
    kf.P *= 500
    kf.Q = np.eye(4) * 0.5
    kf.R *= 1
    return kf

def actualizar_Kalman(kf, landmark):
    z = np.array([landmark.x, landmark.y])
    kf.predict()
    kf.update(z)
    return kf.x[:2].flatten()
```

Figura 4.4: Definición y configuración del filtro Kalman en MediaPipe.

Como se menciono en el capítulo anterior, el funcionamiento del filtro Kalman se resume en estimar y corregir valores, pero para entender el funcionamiento del filtro en el programa, se presenta el siguiente proceso ejemplificado, complementando la información de la selección de herramientas.

**Funcionamiento del filtro de Kalman en la estimación de coordenadas**

Primero, dentro de la función *crear_filtro_kalman* se definen las siguientes matrices que son utilizadas en las operaciones del filtro:

- **Matriz de transición.**

  El modelo de predicción del filtro Kalman asume que la nueva posición depende de la posición anterior más la velocidad multiplicada por el tiempo transcurrido, como el análisis se realiza por cada frame se establece que el tiempo o $\Delta t$ es 1. Por lo que la matriz de transición se define como.

$$kf.F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

- **Matriz de observación.**

  En el proceso de filtraje, la información de interés sobre la posición de articulaciones estará junto a datos adicionales como velocidad, esta matriz de observación se encarga de extraer únicamente la información más relevante, mantiene un tamaño de 2x4 debido a que se observan solo dos variables, la posición de las articulaciones en el eje "$x$" y en el eje "$y$".

$$kf.H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.2}$$

- **Incertidumbre en la estimación inicial.**

  La matriz de varianza de error $kf.P$, representa la confianza que tiene el sistema en su estado actual. En el código se utiliza un valor de incertidumbre de 50, lo que significa que se confía en que la estimación inicial es bastante precisa.

$$kf.P = \begin{bmatrix} 50 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix} \tag{4.3}$$

- **Matriz de ruido del proceso.**

  La matriz definida $kf.Q$, modela la incertidumbre en la predicción, si es muy grande significa que el filtro confía más en las mediciones de *MediaPipe*, si es pequeño, confía más en el modelo del filtro. En el programa se le asigna un valor pequeño de 1 a esta matriz.

$$kf.Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

- **Matriz de medición.**

  Esta matriz representa el ruido en los resultados de *MediaPipe*. En este programa se asume que todos los resultados obtenidos de la biblioteca son lo bastante confiables, por lo que se asigna un valor de 5.

$$kf.R = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \tag{4.5}$$

En cada frame del video analizado, *MediaPipe* obtiene una serie coordenadas en los ejes "$x, y$" las cuales representan la posición actual de las articulaciones del cuerpo. En la definición de la función *actualizar_filtro_kalman*, se almacenan estas coordenadas para su uso en el filtro, a través de la variable $z = np.array([landmark.x, landmark.y])$.

Tabla 4.2: Valores de posición hipotéticos.

| z | landmark.x | ladmark.y |
|---|---|---|
| **Frame 1** | 121 | 235 |
| **Frame 2** | 126 | 248 |

1. **Predicción.**

   El filtro predice la nueva posición de las articulaciones basándose en el estado anterior y la matriz de transición. Tomando las dos mediciones de la tabla 4.2, el proceso de predicción se desarrolla tal como se describe a continuación.

   Para aplicar la fórmula de predicción se debe definir un estado inicial $kf.x$, el cual esta conformado por la posición junto a sus velocidades "$vx$" y "$vy$", realizando una sustracción de coordenadas entre frames consecutivos para la obtención de estas últimas. Para los valores del **Frame 1**, las velocidades tienen valor cero.

$$kf.x = [121, 235, 0, 0] \tag{4.6}$$

   Con el estado inicial definido, se aplica la fórmula 3.3 de Kalman, para obtener la predicción de la próxima posición.

$$kf.x^* = kf.F \cdot kf.x = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 121 \\ 235 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 121 \\ 235 \\ 0 \\ 0 \end{bmatrix} \tag{4.7}$$

Y se actualiza la incertidumbre.

$$kf.P^* = (kf.F)\cdot(kf.P)\cdot(kf.F)^T + kf.Q = \begin{bmatrix} 101 & 0 & 50 & 0 \\ 0 & 101 & 0 & 50 \\ 50 & 0 & 51 & 0 \\ 0 & 50 & 0 & 51 \end{bmatrix} \tag{4.8}$$

2. **Corrección.**

   En este punto, el filtro Kalman compara su predicción con una nueva medición real de *MediaPipe* y ajusta su estado.

   Primero se calcula la ganancia de Kalman, siguiendo la fórmula 3.5.

$$K = (kf.P^*)(kf.H)^T \cdot ((kf.H)(kf.P^*)(kf.H)^T + R)^{-1}$$

$$K = \begin{bmatrix} 101 & 0 \\ 0 & 101 \\ 50 & 0 \\ 0 & 50 \end{bmatrix} \begin{bmatrix} 1/106 & 0 \\ 0 & 1/106 \end{bmatrix} = \begin{bmatrix} 0{,}95284 & 0 \\ 0 & 0{,}95284 \\ 0{,}471 & 0 \\ 0 & 0{,}471 \end{bmatrix} \tag{4.9}$$

   Posteriormente, se realiza la corrección utilizando la fórmula 3.6 pero cambiando los términos en $kf.y = kf.x^* + K(Z_t - (kf.H)(kf.x^*))$.

$$kf.y = \begin{bmatrix} 121 \\ 235 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0{,}95284 & 0 \\ 0 & 0{,}95284 \\ 0{,}471 & 0 \\ 0 & 0{,}471 \end{bmatrix} \begin{bmatrix} 5 \\ 13 \end{bmatrix} = \begin{bmatrix} 125{,}7642 \\ 247{,}38692 \\ 2{,}35 \\ 6{,}123 \end{bmatrix} \tag{4.10}$$

Al final del proceso, se obtiene la próxima posición actualizada [125.764, 247.386] y la estimación de las velocidades correspondientes [2.35, 6.123], con estos resultados, la operación recursiva puede continuar estimando nuevas posiciones, comenzando de nuevo con la actualización de la incertidumbre y después obteniendo una nueva predicción para el siguiente estado. En el código, todos los cálculos anteriores se realizan internamente en el método *kf.predict*, dentro de la definición de la función *crear_filtro_kalman* únicamente se establecen los parámetros fundamentales del filtro Kalman.

Dentro de la misma sección del programa, también se establece el procedimiento para el cálculo de ángulos de movimiento, Figura 4.5.

```
def calcular_angulo(p1, p2, p3):

    A = np.array(p1)
    B = np.array(p2)  # Punto central
    C = np.array(p3)

    # Calcular distancias entre los puntos
    AB = np.linalg.norm(A - B)
    BC = np.linalg.norm(B - C)
    AC = np.linalg.norm(A - C)

    # Evitar divisiones por cero
    if AB == 0 or BC == 0:
        return 0.0
```

Figura 4.5: Definición de función para el cálculo de ángulos.

La función *calcular_angulo*, realiza las operaciones para determinar los ángulos de movimiento utilizando la ley de cosenos. Para la aplicación del método, la función recibe como parámetros tres coordenadas o puntos $P1, P2, P3$, los cuales se almacenan en $A, B, C$ como variables de tipo array, para facilitar la realización de operaciones matemáticas posteriores. Y se lleva a cabo el cálculo de la norma euclidiana entre los tres puntos de referencia, para determinar las distancias $AB, BC, AC$.

Con las nuevas variables se aplica la ley de cosenos, Figura 4.6. Primero determinando el coseno del ángulo en el punto central $B$, usando $np.clip()$ para evitar errores numéricos, lo que asegura que el resultado *cos_theta* esté dentro del rango $[-1, 1]$, para finalmente, utilizar la función $np.arccos(cos\_theta)$ para el cálculo del ángulo, convirtiendo el resultado de radianes a grados con $np.degrees()$ y almacenando el resultado en *angulo*.

```
# Aplicar ley de cosenos
cos_theta = (AB**2 + BC**2 - AC**2) / (2 * AB * BC)
cos_theta = np.clip(cos_theta, -1.0, 1.0)  # Evitar errores numéricos
angulo = np.degrees(np.arccos(cos_theta))

return angulo
```

Figura 4.6: Código para la aplicación de la ley de cosenos.

**Sección 4: Inicialización de variables.**

En esta etapa, principalmente se inicializan las variables necesarias para el procesamiento del video y el cálculo de ángulos, Figura 4.7.

```
# Filtros de Kalman para cada articulación
kf_cadera = crear_filtro_kalman()
kf_rodilla = crear_filtro_kalman()
kf_tobillo = crear_filtro_kalman()
kf_pie = crear_filtro_kalman()
kf_talon = crear_filtro_kalman()
kf_hombro = crear_filtro_kalman()
```

Figura 4.7: Creación de filtros Kalman en MediaPipe.

Primero, se definen los filtros Kalman para cada una de las articulaciones, cada variable $kf$ crea un filtro independiente para cada articulación. En la figura anterior, se puede notar que además de las articulaciones de interés se crean filtros para el hombro, el pie y el talón, esto debido a que estas articulaciones son utilizadas como referencia para el cálculo del ángulo de dos articulaciones de interés, y para mejorar la visualización de la estructura del miembro inferior en una ventana de salida.

Justo después como se muestra en la Figura 4.8, se crean listas que almacenarán los ángulos de movimiento más recientes de cada articulación de interés. Estas listas son necesarias para la aplicación del filtro Gaussiano en los resultados.

```
# Listas que almacenarán los últimos valores de los ángulos detectados
historial_cadera = []  # Almacena los últimos valores de la cadera
historial_rodilla = []  # Almacena los últimos valores de la rodilla
historial_tobillo = []  # Almacena los últimos valores del tobillo

# Definir el tamaño de la ventana para el filtro Gaussiano
ventana_gaussiana = 10

# Lista para guardar los angulos de movimiento
datos_angulos = []
```

Figura 4.8: Creación de historial Gaussiano para cada articulación.

La cantidad de valores que se almacenan temporalmente en las listas, dependerá de la ventana gaussiana, el filtro mantiene un equilibrio entre suavidad y tiempo de respuesta al

utilizar una ventana de 10. Al final de esta sección se crea la lista vacía *datos_angulos*, en donde se almacenarán los ángulos de movimiento calculados de cada frame, para ser exportados en un archivo xlsx.

**Sección 5: Captura y procesamiento.**

Esta es la sección más importante del programa, en ella se procesa el video frame por frame, se extraen las posiciones de las articulaciones con *MediaPipe* y se envía la información a las diferentes funciones previamente definidas.

La Figura 4.9 muestra el código para la extracción de frames del video, mediante el bucle *while cap.isOpened()*. La función *cap.read()*, proporciona los frames del video uno a uno, almacenando temporalmente cada uno de ellos en *frame*, la variable *ret* devuelve un valor "*False*" cuando el video ha terminado o no queda ningún frame, fungiendo como un indicador o bandera, que detiene el bucle.

```
#Lectura del video frame por frame
while cap.isOpened():
    ret, frame = cap.read()
    if ret == False:
        break
```

Figura 4.9: Extracción de frames con Opencv.

El uso de *Opencv* para la lectura del video da como resultado imágenes en formato BGR, por lo que se requiere de una conversión a formato RGB para ser utilizadas con *MediaPipe*. La Figura 4.10, muestra la función *cv2.cvtColor*, la cual realiza la conversión de cada uno de los frames del video de BGR a RGB utilizando el código *cv2.COLOR_BGR2RGB*, y almacenando el resultado en *frame_rgb*.

```
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

Figura 4.10: Conversión de imagen de BGR a RGB.

La detección de articulaciones comienza después de la conversión de imagen. En la Figura 4.11 se puede observar a la función *pose.process()*, con la cual *MediaPipe* analiza la imagen almacenada en *frame_rgb*, y devuelve el objeto *results* que contiene las coordenadas de las articulaciones e información adicional como la confianza en la detección. Se verifica si se ha detectado una pose en el frame actual, de no ser así el código dentro del bloque "if" no se ejecutará, provocando que el programa espere hasta el siguiente frame. Con *results.pose_landmarks.landmark*, se extrae toda la información sobre las coordenadas de cada articulación detectada, devolviendo una lista de puntos clave que se almacena en *landmarks*.

```
results = pose.process(frame_rgb)
if results.pose_landmarks:
    landmarks = results.pose_landmarks.landmark
```

Figura 4.11: Análisis de frames con MediaPipe.

La fracción de código en la Figura 4.12, muestra el envío de puntos clave o coordenadas al filtro Kalman.

```
# Filtrar coordenadas con Kalman, cada articulacion posee su filtro
cadera = actualizar_Kalman(kf_cadera, landmarks[mp_pose.PoseLandmark.LEFT_HIP])
rodilla = actualizar_Kalman(kf_rodilla, landmarks[mp_pose.PoseLandmark.LEFT_KNEE])
tobillo = actualizar_Kalman(kf_tobillo, landmarks[mp_pose.PoseLandmark.LEFT_ANKLE])
pie = actualizar_Kalman(kf_pie, landmarks[mp_pose.PoseLandmark.LEFT_FOOT_INDEX])
talon = actualizar_Kalman(kf_talon, landmarks[mp_pose.PoseLandmark.LEFT_HEEL])
hombro = actualizar_Kalman(kf_hombro, landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER])
```

Figura 4.12: Selección manual de puntos de interés.

Tomando como ejemplo la primer línea del código que corresponde a la articulación de la cadera, el programa realiza dos tareas, primero accede a las coordenadas correspondientes de la cadera izquierda que se encuentran dentro de *landmarks* mediante el código *mp_pose.PoseLandmark.LEFT.HIP*, posteriormente estas coordenadas son empleadas en la función *actualizar_kalman*, junto con la instancia del filtro de Kalman que maneja el estado actual de la posición de la cadera *kf_cadera*, los datos de la articulación son usados por el filtro Kalman y las coordenadas filtradas son almacenadas en *cadera*. Este proceso se realiza

para cada una de las articulaciones de interés y de referencia, modificando únicamente la instancia y la dirección para acceder a las coordenadas correspondientes de cada articulación.

Para visualizar el funcionamiento del programa, se muestra la detección de articulaciones y puntos de referencia en una ventana de salida, esto permite identificar errores de detección de forma rápida y simple. *Opencv* puede dibujar las articulaciones de interés y los puntos de referencia en una ventana, sobre los frames del video que se esta analizando en tiempo real, pero para ello se requiere convertir las coordenadas normalizadas obtenidas por *MediaPipe* a coordenadas en píxeles, debido a que *Opencv* solo trabaja con coordenadas enteras en píxeles.

La conversión de coordenadas a pixeles se realiza en la fracción del código de la Figura 4.13, tomando como ejemplo la conversión para los datos de la cadera, las coordenadas obtenidas por *MediaPipe*, tienen valores entre 0 y 1 en relación al tamaño del video, por lo que para mostrar los puntos en una ventana de salida se multiplican por el ancho y el alto del video que se está analizando. El código $int(cadera[0] * ancho\_video)$ multiplica la coordenada en "$x$" con el ancho del video y $int(cadera[1] * alto\_video)$ realiza la multiplicación de la coordenada en "$y$" con el alto del video, guardando el resultado de esta operación en la variable $cadera\_px$.

```
# Convertir coordenadas a píxeles
cadera_px = (int(cadera[0] * ancho_video), int(cadera[1] * alto_video))
rodilla_px = (int(rodilla[0] * ancho_video), int(rodilla[1] * alto_video))
tobillo_px = (int(tobillo[0] * ancho_video), int(tobillo[1] * alto_video))
pie_px = (int(pie[0] * ancho_video), int(pie[1] * alto_video))
talon_px = (int(talon[0] * ancho_video), int(talon[1] * alto_video))
hombro_px = (int(hombro[0] * ancho_video), int(hombro[1] * alto_video))
```

Figura 4.13: Conversión de coordenadas normalizadas a pixeles.

Después de la conversión de coordenadas a pixeles, se realiza el envío de parámetros para el cálculo de ángulos de movimiento. Cada línea del fragmento de código de la Figura 4.14, utiliza la función $calcular\_angulo$, que toma tres puntos de referencia $P1, P2, P3$ , y devuelve el ángulo formado en el punto central o $P2$, almacenando el resultado en la variable $angulo\_$ de cada articulación. En esta parte del programa se definen los puntos de referencia que son considerados para calcular el ángulo de movimiento de cada una de las articulaciones de interés.

```
#Calcular ángulos
angulo_cadera = calcular_angulo(hombro, cadera, rodilla)
angulo_rodilla = calcular_angulo(cadera, rodilla, tobillo)
angulo_tobillo = calcular_angulo(rodilla, tobillo , pie)
```

Figura 4.14: Cálculo de ángulos con MediaPipe.

En la Figura 4.15 se muestran tres diferentes siluetas en las que se han trazado las conexiones entre las articulaciones de referencia y de interés, estas conexiones son las que se toman en cuenta para calcular el ángulo de movimiento de las tres articulaciones del miembro inferior.

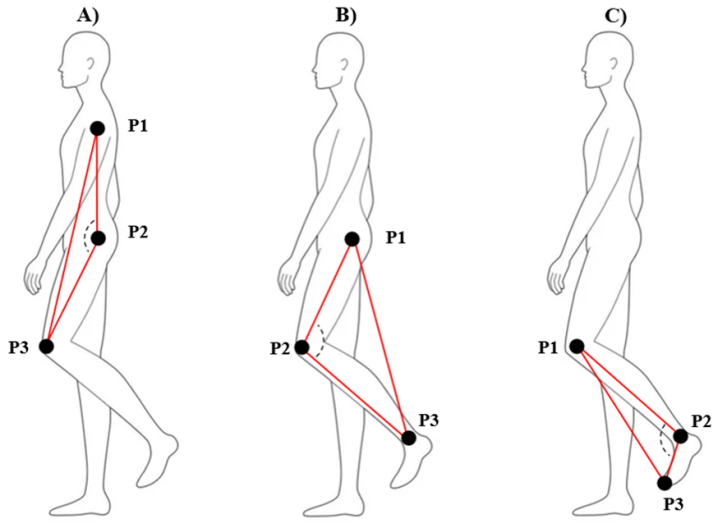



Figura 4.15: Sistema de referencias para el cálculo de ángulo de movimiento.

La silueta A) muestra la conexión entre el hombro, la rodilla y la cadera para determinar el ángulo de movimiento de esta última, la silueta B) muestra la conexión entre la cadera y el tobillo para determinar el ángulo de movimiento de la rodilla, y la silueta C) muestra la conexión del pie con la rodilla, para determinar el ángulo del tobillo. Debido a estas relaciones los videos analizados deben mostrar siempre a la persona de cuerpo completo, con las articulaciones mencionadas completamente visibles.

Aún dentro de la sección cinco y con el ángulo de movimiento ya calculado y almacenado en la variable correspondiente, comienza la etapa de suavizado de resultados. Para hacer uso del filtro Gaussiano en los resultados obtenidos, primero se almacenan estos valores dentro de las listas creadas anteriormente. En la Figura 4.16, se puede observar la función *.append()*, con la cual se almacenan los resultados de movimiento de cada articulación en su historial correspondiente, actuando como una memoria temporal.

```
# Aplicar filtro Gaussiano
historial_cadera.append(angulo_cadera)
historial_rodilla.append(angulo_rodilla)
historial_tobillo.append(angulo_tobillo)

if len(historial_cadera) > ventana_gaussiana:
    historial_cadera.pop(0)
    historial_rodilla.pop(0)
    historial_tobillo.pop(0)
```

Figura 4.16: Registro de resultados en la ventana Gaussiana.

El filtro solamente toma en cuenta los resultados más recientes, por lo que se verifica que los historiales se actualicen constantemente, si algún historial llega a contener un número superior de datos al del valor establecido en la ventana Gaussiana, la función *.pop(0)*, elimina el valor más antiguo dentro de la lista, dejando espacio para ingresar los valores recientes. En la Figura 4.17, la función *scipy.ndimage.gaussian_filter*, aplica un filtro Gaussiano al historial de cada articulación, suavizando los datos almacenados. *sigma* determina la desviación estándar de la distribución Gaussiana, controlando el grado de suavizado de los datos, con el valor actual de 2 se suavizan los datos sin perder demasiada precisión.

```
angulo_cadera = scipy.ndimage.gaussian_filter(historial_cadera, sigma=2)[-1]
angulo_rodilla = scipy.ndimage.gaussian_filter(historial_rodilla, sigma=2)[-1]
angulo_tobillo = scipy.ndimage.gaussian_filter(historial_tobillo, sigma=2)[-1]
```

Figura 4.17: Aplicación de filtro Gaussiano.

La función utilizada para aplicar el filtro Gaussiano suaviza todos los valores dentro del historial o ventana, devolviendo una nueva lista con todos los resultados filtrados,

pero se utiliza el parámetro $[-1]$, para almacenar unicamente el último elemento de la lista en la variable correspondiente, por ejemplo, $[angulo\_cadera]$, el proceso se realiza de esa forma ya que el valor filtrado utilizado es más cercano al valor actual que se esta capturando.

Una vez que los ángulos de movimiento calculados ya han pasado por todos los procesos necesarios, se almacenan para después ser exportados. En la Figura 4.18, se usa *datos_angulos* como una lista temporal que mantiene los ángulos de movimiento de la cadera, rodilla y tobillo, para ser exportados al finalizar el programa, dejando los valores más antiguos al principio.

```
# Almacenar datos de los ángulos
datos_angulos.append([angulo_cadera, angulo_rodilla, angulo_tobillo])
```

Figura 4.18: Almacenamiento de ángulos de movimiento.

Dentro de la Figura 4.19, se realiza el dibujado de puntos de interés y las líneas que los conectan en una ventana de salida. Utilizando las variables que contienen las coordenadas de las articulaciones en pixeles y la función *cv2.circle()* de *Opencv*, se sobreponen círculos en donde se ubican las articulaciones, y con la función *cv2.line()* se dibuja una línea entre dos articulaciones que se definan.

```
# Dibujar líneas entre las articulaciones
cv2.line(frame, cadera_px, rodilla_px, (255, 255, 255), 2)
cv2.line(frame, rodilla_px, tobillo_px, (255, 255, 255), 2)
cv2.line(frame, tobillo_px, pie_px, (255, 255, 255), 2)
cv2.line(frame, tobillo_px, talon_px, (255, 255, 255), 2)
cv2.line(frame, pie_px, talon_px, (255, 255, 255), 2)

# Dibujar puntos de articulaciones
cv2.circle(frame, cadera_px, 5, (0, 0, 255), -1)
cv2.circle(frame, rodilla_px, 5, (0, 0, 255), -1)
cv2.circle(frame, tobillo_px, 5, (0, 0, 255), -1)
cv2.circle(frame, pie_px, 5, (0, 0, 255), -1)
cv2.circle(frame, talon_px, 5, (0, 0, 255), -1)
```

Figura 4.19: Dibujado de líneas y puntos en ventana de salida.

La primer línea del código, dibuja la conexión entre la cadera y rodilla utilizando sus respectivas coordenadas filtradas ($cadera\_px, rodilla\_px$), los valores $(255,255,255)$ definen el color de la línea, blanco en formato BGR. Cada círculo y línea es dibujado en la variable $frame$, que almacena temporalmente uno a uno los frames del video.

Un buen agregado para complementar el dibujado de círculos y líneas, es la colocación de los grados de movimiento calculados, con sus respectivas variaciones en tiempo real. En la Figura 4.20 se muestra el fragmento de código que añade los ángulos de movimiento a la ventana de salida, junto con las líneas y puntos previamente definidos. Se utiliza la función $cv2.putText()$ de $Opencv$, donde se indica la posición del texto y el contenido a mostrar dentro de la variable $frame$. Por ejemplo, la línea $f"cadera : int(angulo\_cadera)"$ especifica que se muestre el ángulo de la cadera.

```
# Mostrar los ángulos en pantalla
cv2.putText(frame, f"Cadera: {int(angulo_cadera)}", (cadera_px[0] - 50, cadera_px[1] - 20),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2, cv2.LINE_AA)
cv2.putText(frame, f"Rodilla: {int(angulo_rodilla)}", (rodilla_px[0] - 50, rodilla_px[1] - 20),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2, cv2.LINE_AA)
cv2.putText(frame, f"Tobillo: {int(angulo_tobillo)}", (tobillo_px[0] - 50, tobillo_px[1] - 20),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2, cv2.LINE_AA)
```

Figura 4.20: Implementación de ángulos calculados en tiempo real.

Justo después, se indica la posición donde se mostrará el texto, el fragmento $cadera[0]-50, cadera[1]-20$ ubica el texto en las coordenadas de la articulación de la cadera, pero con un ligero desplazamiento. Por último, tambien se definen las propiedades del texto, como el tipo de fuente, tamaño y color.

Al final de la sección cinco se encuentra la función $cv2.imshow$, Figura 4.21, que permite visualizar el proceso de análisis de movimiento en una ventana de salida, recibiendo dos parámetros, el título de la ventana y el objeto a mostrar, en este caso la variable $frame$. Cada vez que $frame$ se actualice, la ventana reflejará los cambios en la posición de líneas y puntos, junto con los nuevos ángulos de movimiento conforme se detectan nuevas poses.

```
    cv2.imshow("Captura de Movimiento - Pierna Izquierda", frame)
if cv2.waitKey(frame_delay) & 0xFF == ord('q'):
    break
```

Figura 4.21: Código para mostrar ventana de salida.

**Sección 6: Exportación de datos.**

Dentro de la última sección, se realiza el proceso de exportación de datos, Figura 4.22.

```
df = pd.DataFrame(datos_angulos, columns=["Cadera", "Rodilla", "Tobillo"])
df.to_excel("angulos_movimiento.xlsx", index=False)
```

Figura 4.22: Exportación de datos en archivo xlsx.

En este fragmento del código, $df$ toma los datos que se almacenaron previamente en la lista $datos\_angulos$, y con el comando $columns = [''Cadera'','' Rodilla'','' Tobillo'']$, se nombran las tres columnas de variables para que los ángulos almacenados correspondan con su articulación. Por último se define que $df$ sea exportado como un archivo de tipo xlsx.

### 4.1.1. Pruebas de funcionamiento programa de captura basado en MediaPipe

Con el programa desarrollado se realizaron pruebas de funcionamiento para comprobar que su desempeño es el que se esperaba, para ello se analizó una serie de videos del repositorio de ejercicios de muestra de la revista **Women´s Health** [43], en donde algunas personas realizan ejercicios breves que incluyen movimientos de miembro inferior, todos con una resolución de imagen de **1080x1080** pixeles y una tasa de **30** frames por segundo.

- **Prueba 1:**

  Insertando la ruta de la ubicación del video y ejecutando el programa se obtuvieron los resultados mostrados en las Figuras 4.23 y 4.24.


Figura 4.23: Prueba 1: Frame 01 MediaPipe.

Este primer análisis demuestra que el sistema de detección funciona, se puede visualizar la identificación de las articulaciones de interés con puntos rojos, y la unión de estos puntos por líneas blancas, además tal como se estableció en el código del programa, los resultados del cálculo del ángulo de movimiento también son mostrados, cada uno cerca de su articulación correspondiente.

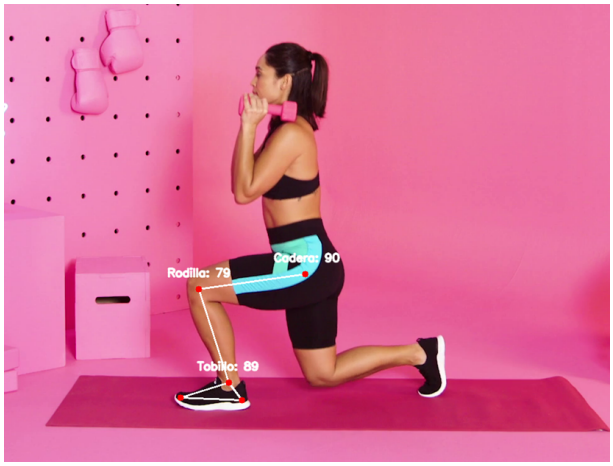



Figura 4.24: Prueba 1: Frame 30 MediaPipe.

El almacenamiento de los resultados también funciona correctamente, ya que si se almacenan los ángulos de movimiento calculados en un archivo ”xlsx”, separando la información en tres columnas, una por cada articulación.

| | A | B | C |
|---|---|---|---|
| 1 | **Cadera** | **Rodilla** | **Tobillo** |
| 2 | 178.6456 | 174.3154 | 102.6223 |
| 3 | 178.7162 | 174.5158 | 102.9064 |
| 4 | 179.1793 | 174.4759 | 103.2611 |
| 5 | 178.2504 | 173.9908 | 103.7185 |
| 6 | 175.8507 | 173.2382 | 104.1968 |
| 7 | 172.4727 | 172.306 | 104.1117 |
| 8 | 168.9151 | 171.3302 | 103.7771 |
| 9 | 165.7174 | 170.375 | 103.4382 |
| 10 | 163.3951 | 169.5936 | 103.2172 |

Figura 4.25: Resultados de MediaPipe almacenados.

Con estos primeros resultados se podía afirmar que el sistema funciona como se esperaba, pero debido a que la ley de cosenos calcula el ángulo de las articulaciones utilizando tres puntos de referencia $A, B, C$, si estos puntos de referencia están alineados, $B$ exactamente entre $A$ y $C$, se obtendrá como resultado 180 grados para el ángulo de movimiento, esto sucede específicamente con la articulación de la rodilla y cadera, cuando el cuerpo se encuentra en posición de pie. Dichos valores, no representan el mecanicismo real del cuerpo humano, pues cuando una persona está de pie, se considera una posición neutral

con valores de 0 a 10 grados para la cadera y rodilla. Para resolver este inconveniente se optó por restar los valores angulares de las tres articulaciones en la posición neutral o de pie.

```
#Calcular ángulos
angulo_cadera = 180 - calcular_angulo(hombro, cadera, rodilla)
angulo_rodilla = 180 - calcular_angulo(cadera, rodilla, tobillo)
angulo_tobillo = 90 - calcular_angulo(rodilla, tobillo , pie)
```

Figura 4.26: Modificación de valores angulares en posición neutral.

Con esta modificación en el código, el programa interpreta los valores de la posición neutra del cuerpo humano como 0 grados para las tres articulaciones de interés, en el caso del tobillo el valor que se sustrae difiere al del resto de articulaciones, ya que los puntos o referencias que se tomaron para el cálculo del ángulo de esta articulación al estar de pie, lo que se forma es un ángulo de 90 grados entre la rodilla, el tobillo y el pie.

- **Prueba 2:**

  Con las modificaciones implementadas, nuevamente se ejecutó el programa para analizar el mismo video de la primer prueba, obteniendo ahora los resultados de las Figuras 4.27 y 4.28.


Figura 4.27: Prueba 2: Frame 01 MediaPipe.


Figura 4.28: Prueba 2: Frame 120 MediaPipe.

En este análisis, se observa que los valores angulares de las articulaciones en la posición neutral ya tienen más sentido bio-mecánicamente. Además, ahora se puede identificar la dirección del movimiento, recibiendo valores positivos y negativos, diferenciando flexión y extensión, como lo visto en el tobillo de la Figura 4.28.

- **Prueba 3:**

  En esta prueba una mujer realiza ejercicios boca arriba, el fondo del video cambia de color, pero al ser solido se sigue destacando a la persona. Ejecutando el programa con este nuevo video, se obtiene el resultado de las Figuras 4.29 y 4.30.

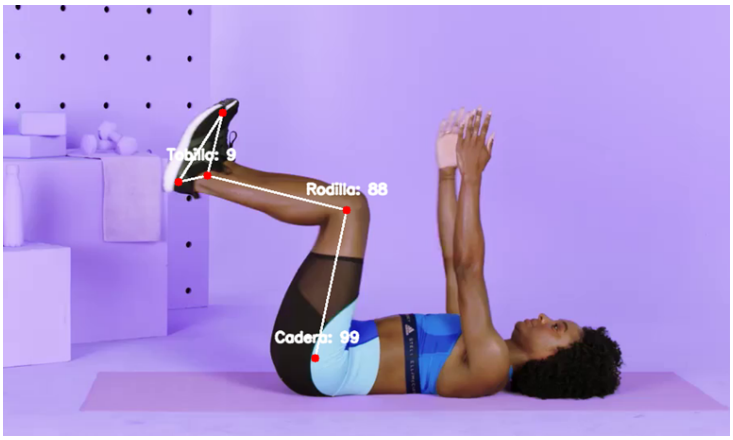



Figura 4.29: Prueba 3: Frame 01 MediaPipe.


Figura 4.30: Prueba 3: Frame 20 MediaPipe.

Aún con el cambio de posición de la persona analizada, la detección es la correcta, y al igual que con el anterior programa se mantienen valores coherentes al momento de estar en una posición neutral, con cero grados para las tres articulaciones.

## 4.2. Programa de captura de movimiento basado en MMPose

A diferencia de *MediaPipe*, *MMPose* al estar basado en redes neuronales profundas, requiere herramientas avanzadas como *PyTorch* y *TorchVision*, además, para un mejor procesamiento del análisis de imágenes la documentación oficial [33], recomienda hacer uso de una tarjeta gráfica o *GPU* junto con el sistema *CUDA*, el cual permite utilizar las tarjetas gráficas de la marca *NVIDIA* para realizar cálculos de alto rendimiento.

El proceso de preparación del entorno se realiza de la siguiente manera. Se instaló un interprete de lenguaje de *Python* altamente compatible con el framework, siendo la versión 3.8.20 la implementada. Dentro del entorno se implementó el sistema *CUDA* mediante la pagina oficial de *NVIDIA*, utilizando la versión 11.8 y se instaló la versión compatible con *CUDA* del complemento *PyTorch*, con el comando *pip*.

Mediante el uso del gestor de paquetes de *OpenMMLab*, *MIM*, se incorpora al entorno el complemento *MMCV*, recurso que proporciona herramientas para la manipulación de datos y modelos, de *MMEngine*, un marco de trabajo que facilita la integración de modelos en *MMPose*. Paquetes adicionales como *Opencv*, *NumPy*, *Pandas*, *FilterPy* y *Scipy*, se instalaron utilizando nuevamente el comando *pip*, en las mismas versiones que en el anterior entorno de *MediaPipe*.

Para la instalación de *MMPose*, se describen dos formas de instalar el framework en la documentación oficial, una que permite la modificación del código fuente de la herramienta y otra que maneja a *MMPose* como otro paquete de *Python*, debido a la naturaleza de este proyecto se seleccionó la segunda opción, por lo que únicamente se ejecutó el comando *mim install "mmpose = 1,3,2"*, la versión 1.3.2 es una versión compatible con todos los demás complementos, Tabla 4.3.

Tabla 4.3: Versiones de herramientas implementadas en entono MMPose.

| Recurso | Versión |
|---|---|
| **MMpose** | 1.3.2 |
| **CUDA** | 11.8 |
| **PyTorch** | 2.4.1 |
| **MMCV** | 2.1.0 |
| **MMEngine** | 0.8.0 |
| **scipy** | 1.10.1 |

El desarrollo del sistema de captura basado en *MMPose* sigue la misma estructura y lógica del programa basado en *MediaPipe*, por lo que a continuación, se detallarán únicamente los cambios realizados en cada sección del código en comparación con el primer programa de captura.

**Sección 1: Bibliotecas.**

Con el uso de un nuevo sistema de detección de articulaciones, en esta sección se definieron nuevas bibliotecas, Figura 4.31. El programa cuenta con las mismas funciones del sistema de detección anterior, por lo que se mantienen las bibliotecas que permiten el uso del filtro Kalman y Gaussiano, la implementación de una ventana de salida y el almacenamiento de los resultados.

```
import cv2                      # Biblioteca para procesamiento de imágenes y videos
import numpy as np              # Operaciones matemáticas y manejo de arrays
import pandas as pd             # Manejo y exportación de datos a Excel
import scipy.ndimage            # Aplicación de filtro Gaussiano
import mmcv                                 # Herramientas adicionales para MMPose
import torch                                # Manejo de operaciones en GPU
from filterpy.kalman import KalmanFilter  # Implementación del filtro de Kalman
from mmengine import Config                 # Configuración del modelo MMPose
from mmpose.apis import init_model, inference_topdown  # Inferencia de poses con MMPose
```

Figura 4.31: Declaración de bibliotecas MMPose.

Adicionalmente se integran las bibliotecas correspondientes para hacer uso *MMpose* como *PyTorch* y *MMCV*.

**Sección 2: Inicialización y configuración.**

Cualquier modelo de aprendizaje requiere dos archivos esenciales para ser utilizado con *MMPose*, un archivo de configuración que define aspectos técnicos como la arquitectura del modelo, el tamaño de entrada y los parámetros de entrenamiento y evaluación, así como un archivo de pesos o *checkpoints*, que contiene los valores numéricos aprendidos por la red neuronal tras ser entrenada con grandes volúmenes de datos. Estos pesos permiten que el modelo realice inferencias sobre nuevos datos de manera efectiva.

Para seleccionar el modelo de aprendizaje, se hace uso de la información de la investigación *"RTMPose: Real-Time Multi-Person Pose Estiomation based on MMPose"* [44], en la que se realizó una comparación entre algunos de los modelos de aprendizaje de *MMPose* más precisos y eficientes, como *HRNet*, *MobileNet*, *FastPose* y *RTMPose*. Los resultados de esta investigación indicaron que *RTMPose* ofrece un buen equilibrio entre rendimiento y complejidad, permitiendo ajustar la precisión según la variante utilizada del modelo, la Figura 4.32 muestra una gráfica de los resultados de la investigación en la que se puede observar el sobresaliente rendimiento del modelo *RTMPose* frente a otras opciones.

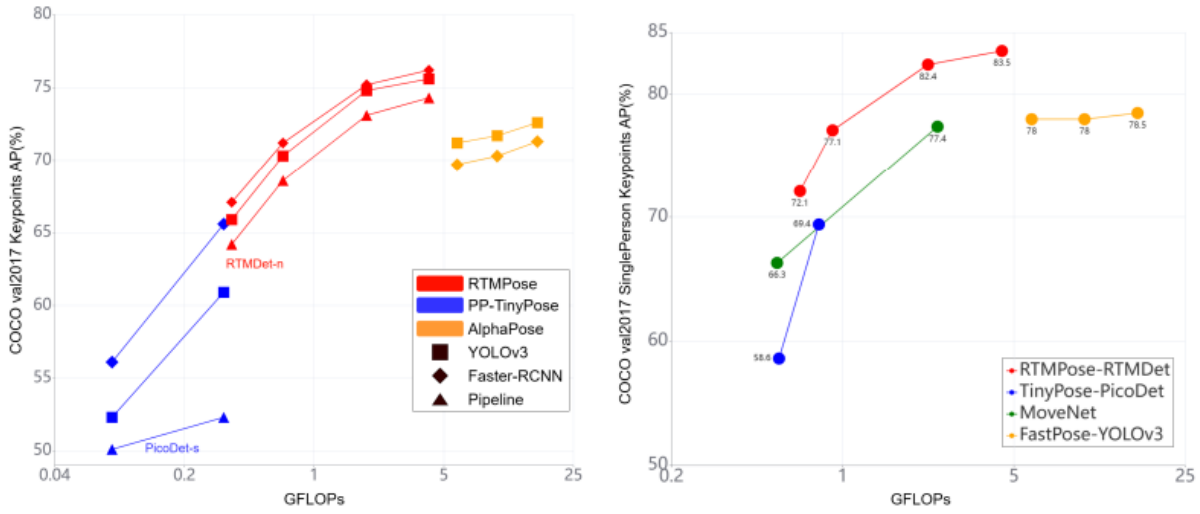



Figura 4.32: Comparación de porcentajes de precisión entre bibliotecas de estimación de código abierto [44].

*RTMPose* cuenta con diferentes versiones, para este entorno se implementa $RTMPose-X$, ya que alcanza un nivel de precisión AP superior al 86 %, lo que garantiza una alta exactitud en la detección de puntos clave, aunque esta variante es la que requiere de más recursos computacionales.

Para el uso del modelo $RTMPose-X$ en el programa, se descargan los archivos de configuración correspondientes a través del repositorio oficial de *MMPose* [45]. La Figura 4.33, muestra la inicialización del modelo de aprendizaje en el programa, en donde primero, se define la ruta de acceso al archivo de configuración del modelo mediante $CONFIG\_FILE$, y la ruta de acceso al archivo de pesos utilizando, $CHECKPOINT\_FILE$.

```
# Inicialización del modelo MMPose

CONFIG_FILE = "C:/Users/erick/mmpose/configsrtmpose-x_8xb256.py"
CHECKPOINT_FILE = "C:/Users/erick/.cache/torch/rtmpose-x_simc.pth"

# Selección del dispositivo (GPU/CPU)
modelo_pose = init_model(CONFIG_FILE,
                         CHECKPOINT_FILE,
                         device="cuda",
                         cfg_options={'weights_only': True})
```

Figura 4.33: Inicialización de MMPose.

En la variable $modelo\_pose$, se inicializa el modelo de estimación, se agregan las rutas para los archivos de configuración, el tipo de hardware a utilizar, $device = cuda$ para hacer uso de la tarjeta gráfica disponible, y el parámetro $cfg\_options$ que en su configuración $weights\_only : True$ evita que se carguen componentes no esenciales del archivo de *checkpoints* (como optimizadores o metadatos).

**Sección 3: Definición de funciones.**

Para este programa, se implementa una función de normalización de coordenadas, Figura 4.34, debido a que las detecciones de *MMPose* están expresadas en pixeles y el filtro Kalman ya establecido está adaptado para su uso con parámetros normalizados. El proceso de normalización se lleva a cabo con la función $obtener\_coordenada$, la cual recibe como

parámetro un índice de identificación "*idx*" correspondiente a una articulación específica, accede a sus coordenadas y las divide por las dimensiones del video.

```
# Normalizacion de coordenadas
def obtener_coordenada(idx):
    x, y = keypoints[idx][:2]
    return x / ancho_video, y / alto_video
```

Figura 4.34: Normalización de coordenadas en MMPose.

**Sección 5: Captura y procesamiento de video.**

En esta sección, tal como se muestra en la Figura 4.35, la lectura del video frame a frame se mantiene, pero se extrae su información de forma distinta. Primero, la función *inference_topdown*, utiliza el modelo definido anteriormente, para detectar y localizar articulaciones en los frames del video, devolviendo una lista con información sobre las personas detectadas y sus articulaciones con coordenadas en pixeles, en la variable *resultados*.

```
# Lectura del video frame por frame
while cap.isOpened():
    ret, frame = cap.read()
    if ret == False:
        break

    # Inferencia de pose con MMPose (Reemplaza MediaPipe)
    resultados = inference_topdown(modelo_pose, frame)
    keypoints = resultados[0].pred_instances.keypoints[0]
```

Figura 4.35: Adquisición de información de frames con MMPose.

En *resultados*, se contiene el atributo *pred_instances*, dentro de este atributo la información se organiza como se muestra en la Figura 4.36. Para hacer uso de la información almacenada, el programa primero accede a *resultados*[0], que contiene todos los resultados de la primera y única persona detectada por el modelo, después utiliza *.pred_instances.keypoints*[0] para extraer solamente las coordenadas de las articulaciones detectadas para dicha persona, descartando información adicional como cajas delimitadoras.

```
resultados = [
    Instancia1 (Persona 1):
        - pred_instances:
            - keypoints: [ [[x1,y1,s1], [x2,y2,s2], ..., [xn,yn,sn]] ]
            - keypoint_scores
            - bboxes
            - otros atributos
    Instancia2 (Persona 2):
        - pred_instances:
            - keypoints: [ [[x1,y1,s1], [x2,y2,s2], ..., [xn,yn,sn]] ]
            - keypoint_scores
            - bboxes
            - otros atributos
]
```

Figura 4.36: Organización de instancias predichas por MMPose.

Todo el proceso anterior deja a la variable *keypoints*, únicamente con la información sobre las coordenadas de las articulaciones y la confianza del modelo sobre la detección, [*x*, *y*, *score*]. Dentro de *keypoints*, las coordenadas están organizadas por índices específicos del 0 al 26, Figura 4.37, en donde cada identificador corresponde a una articulación particular del modelo de detección.

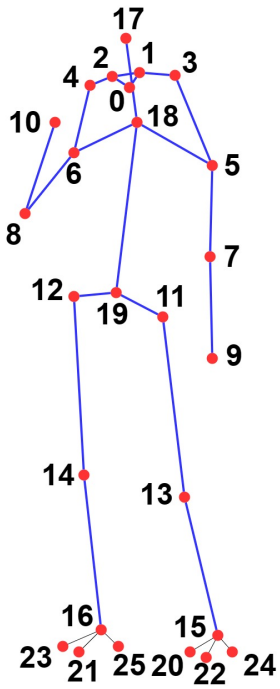



Figura 4.37: Identificadores de Keypoints para el modelo RTMPose-X [45].

Los índices son utilizados en el fragmento de código de la Figura 4.38, donde se realiza el envío de parámetros a la función del filtro Kalman, justo después de la obtención de coordenadas. En esta parte del programa al igual que en el anterior, se eligen los puntos de referencia para el cálculo del ángulo de las articulaciones de interés, este sistema utiliza las mismas referencias mostradas en la Figura 4.15.

```
# Filtro Kalman con coordenadas de MMPose
cadera = actualizar_Kalman(kf_cadera, obtener_coordenada(11))
rodilla = actualizar_Kalman(kf_rodilla, obtener_coordenada(13))
tobillo = actualizar_Kalman(kf_tobillo, obtener_coordenada(15))
pie = actualizar_Kalman(kf_pie, obtener_coordenada(20))
hombro = actualizar_Kalman(kf_hombro, obtener_coordenada(5))
```

Figura 4.38: Envió de coordenadas al filtro Kalman en MMPose.

Tomando como ejemplo la primer línea del código que realiza el proceso de filtración de la posición de la cadera, la función *obtener_coordenada*(11) extrae del arreglo keypoints las coordenadas en píxeles correspondientes a la articulación identificada con el índice 11 del modelo (cadera izquierda) y las normaliza. La coordenada normalizada se utiliza como entrada en la función *actualizar_Kalman*, junto con la instancia *kf_cadera*, almacenando el resultado del proceso de filtraje con Kalman en la variable *cadera*.

Para este punto el programa de captura conserva la misma estructura general, reutilizando tanto las funciones como las variables empleadas previamente en el programa basado en *MediaPipe*, se calcula el ángulo de movimiento con la ley de cosenos, el resultado pasa por un filtro Gaussiano, los ángulos resultantes son almacenados y se muestra el funcionamiento del programa en una ventana de salida, dibujando las líneas, puntos y ángulos justo como en el anterior programa.

### 4.2.1. Pruebas de funcionamiento programa de captura basado en MMPose

El nuevo sistema de captura basado en *MMPose*, fue probado utilizando los mismos videos empleados en las pruebas realizadas con el programa basado en *MediaPipe*.

- **Prueba 1.**

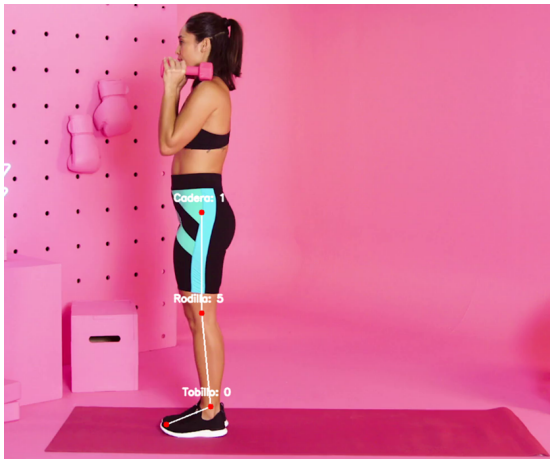



Figura 4.39: Prueba 1: Frame 01 MMPose.


Figura 4.40: Prueba 1: Frame 50 MMPose.

En este primer análisis, el programa realiza correctamente la identificación de articulaciones de interés y de referencia, se muestran los ángulos en su correspondiente articulación y también se almacenan de forma correcta los resultados.

| | A | B | C |
|---|---|---|---|
| 1 | **Cadera** | **Rodilla** | **Tobillo** |
| 2 | 1.888188 | 5.630683 | -0.74945 |
| 3 | 0.993886 | 5.573953 | -1.18356 |
| 4 | 1.856218 | 6.052254 | -1.22239 |
| 5 | 3.9904 | 6.725892 | -1.61466 |
| 6 | 7.143794 | 7.256884 | -1.21915 |
| 7 | 10.46552 | 7.945607 | 0.625942 |
| 8 | 13.97051 | 8.516427 | 2.049271 |
| 9 | 17.56881 | 10.1292 | 2.969589 |
| 10 | 20.17426 | 12.35127 | 3.53757 |

Figura 4.41: Resultados de MMPose almacenados.

- **Prueba 2.**

  En las figuras 4.42 y 4.43, se aprecian algunos de los frames capturados al realizar un segundo análisis.


Figura 4.42: Prueba 2: Frame 01 MMPose.

En esta segunda prueba de funcionamiento se observa un resultado similar a lo visto con *MediaPipe*, hay una correcta identificación de las articulaciones de interés, manteniendo lo ángulos de movimiento capturados dentro del rango de movimiento angular de cada una de las articulaciones.


Figura 4.43: Prueba 2: Frame 220 MMPose.

Al igual que con el sistema de captura basado en *MediaPipe*, la posición de la persona analizada no interfiere con el proceso de captura, pues no se detectó ningún tipo de error durante la ejecución de este análisis.

- **Prueba 3.**

  Insertando un video con obstáculos sobre el miembro analizado, como en las pruebas de *MediaPipe*, se obtuvo el resultado mostrado en las Figuras 4.44, 4.45 y 4.46.

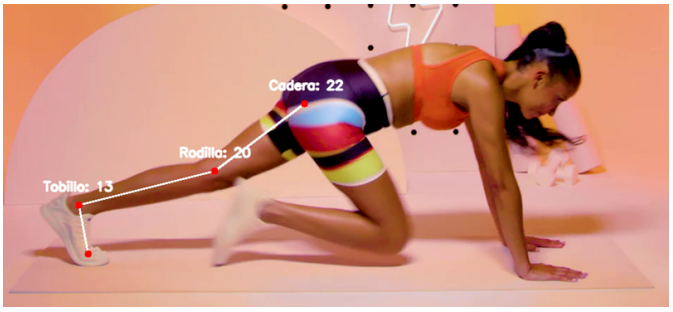



Figura 4.44: Prueba 3: Frame 50 MMPose.

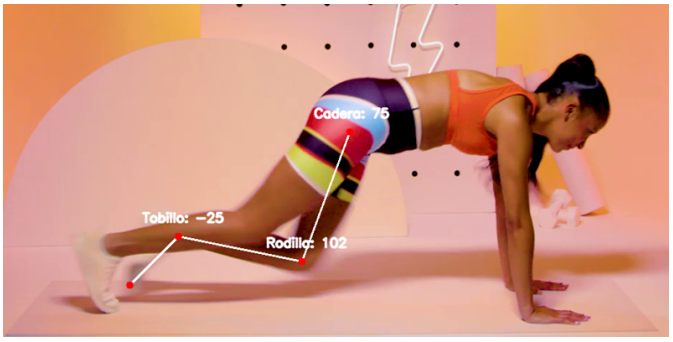



Figura 4.45: Prueba 3: Frame 160 MMPose.

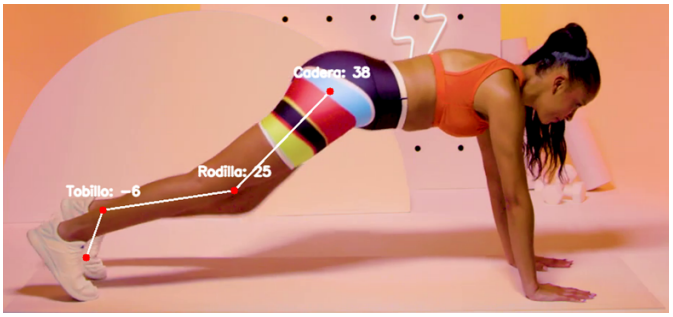



Figura 4.46: Prueba 3: Frame 190 MMPose.

Se puede notar que, incluso con un obstáculo cubriendo el miembro inferior de interés, el sistema de detección mantiene un buen desempeño, además durante la ejecución del análisis no se observaron errores significativos en la localización de las articulaciones. Por lo que con esta última prueba, se consideró este segundo sistema de captura terminado y listo para ser comparado con el sistema basado en *MediaPipe*.

## 4.3. Sistemas de réplica de movimiento

Para la implementación de los sistemas de réplica de movimiento se desarrolló un modelo de miembro inferior, y debido a que el diseño es utilizado en el sistema de réplica virtual así como en el físico, el modelo contempla su impresión en 3D, considerando las medidas de los motores GM25-370.

Aunque CoppeliaSim se emplea como el entorno de simulación del sistema de réplica virtual, su funcionalidad como programa de modelado es limitada ya que carece de funcionalidades comunes, lo que dificulta la realización de operaciones como extracciones, cortes o el ensamblaje, pero gracias a su compatibilidad con modelos externos se utiliza el programa SolidWorks como el entorno para el desarrollo del diseño de miembro inferior.

El prototipo cuenta con un diseño simplificado de 40$cm$ de longitud, que se conforma únicamente por una base fija, que también cumple la función de cadera, y tres eslabones móviles que representan al muslo, la pierna y el pie, adicionalmente perpendicular a cada articulación se agrega el modelo del motor GM25-370, Figura 4.47.

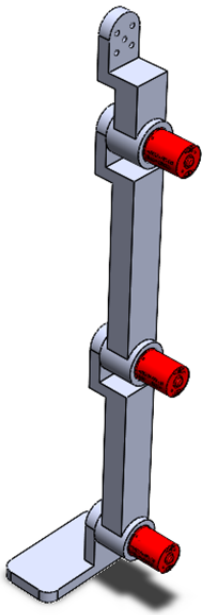


Figura 4.47: Modelo virtual del sistema de réplica de movimiento en SolidWorks.

La pieza de la base, Figura 4.48a, cuenta con múltiples orificios para su colocación en una superficie solida, pero sigue el mismo diseño que los eslabones del muslo y pierna Figura 4.48b, además, cada pieza del modelo posee un espacio para la colocación de cada uno de los actuadores.

(a) Base fija. (b) Eslabón central.

Figura 4.48: Pieza de cadera (a) y muslo (b) del modelo virtual en SolidWorks.

El eslabón representativo del pie, está conformado por una base rectangular con bordes redondeados y por una pestaña de sujeción, compartiendo parte del diseño de las anteriores piezas.

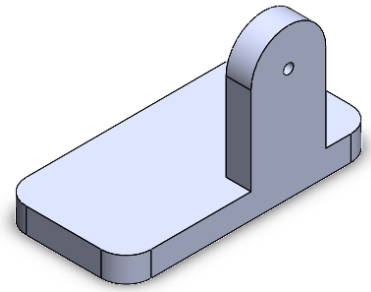


Figura 4.49: Diseño en SolidWorks del modelo virtual del eslabón para el pie .

Cada pieza del modelo se conecta con la anterior mediante el eje de cada motor, tal como se muestra en la Figura 4.50. De esta manera en la impresión 3D del modelo, se reduce el número de elementos necesarios para el ensamblaje y se mejora la calidad del movimiento articular. A excepción de la base fija, ninguna otra pieza requiere de elementos de sujeción.

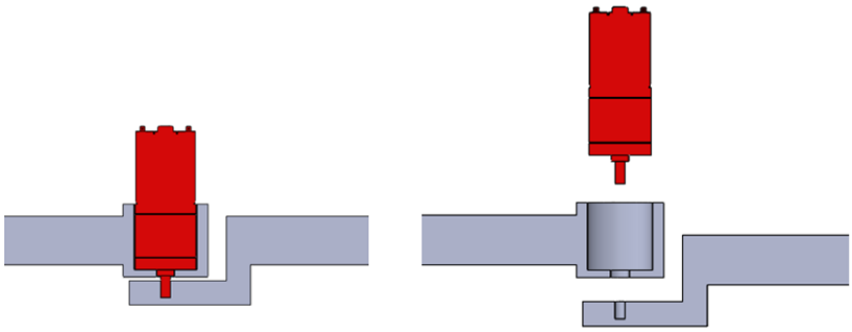


Figura 4.50: Sección explosionada de la unión de dos eslabones en SolidWorks.

Para hacer uso del modelo en el programa de simulación, se exporta el diseño de SolidWorks como un archivo **URDF**, totalmente compatible con CoppeliaSim. La Figura 4.51 muestra el modelo virtual una vez importado en el entorno de simulación de CoppeliaSim.


Figura 4.51: Modelo de réplica virtual en CoppeliaSim.

Es necesario que a cada una de las piezas se le asigne su clase correspondiente, para que el modelo responda ante las órdenes de movimiento. Todos lo eslabones se establecen como *links* y se agregan objetos tipo *joint* en cada unión de las piezas, estos últimos funcionan como actuadores dentro de CoppeliaSim, y son los utilizados para accionar el modelo, por lo que se configuran sus propiedades virtuales de acuerdo a las características de los motores GM25-370, se asigna un rango de movimiento de ±180°, un control de posición activado y una velocidad máxima de 330 RPM o 1980°/s, también se ajusta el torque máximo a un valor de 1.5 N·m, cercano a las características reales de los motores, Figura 4.52.

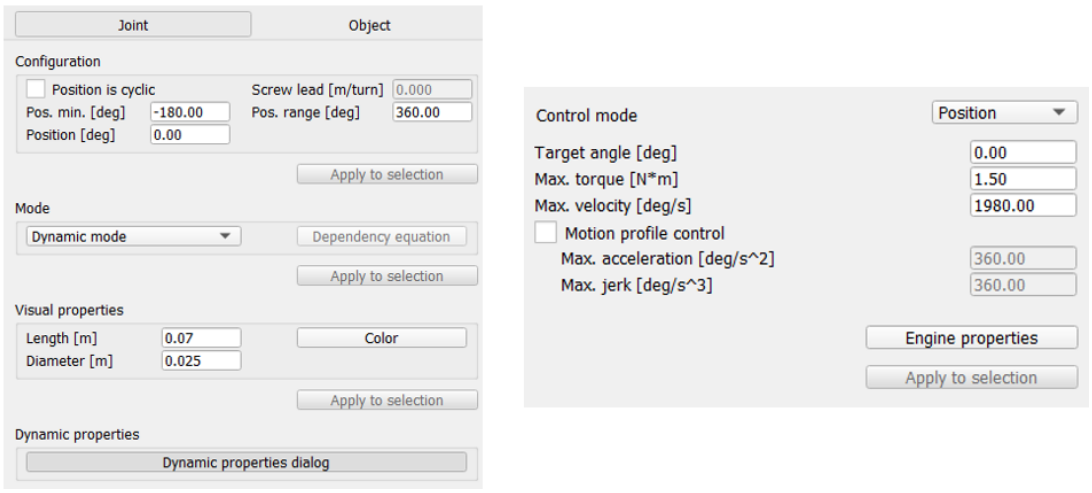



Figura 4.52: Propiedades de los actuadores virtuales.

### 4.3.1. Comunicación CoppeliaSim-Python

El control remoto del movimiento de los actuadores virtuales mediante un programa personalizado requiere que se establezca previamente una conexión entre *Python* y el modelo virtual. Esta comunicación se realiza mediante la Remote API, el mecanismo de control remoto que proporciona CoppeliaSim, que permite enviar instrucciones desde un programa en *Python* hacia el entorno de simulación en tiempo real. La conexión entre ambos programas se configura a través del siguiente procedimiento.

- **Manipulación de archivos para Remote API.**

  Junto con la instalación de CoppeliaSim se incluyen archivos necesarios para la comunicación con *Python*, específicamente *remoteApi.dll*, *sim.py* y *simConst.py*, ubicados en sus rutas respectivas "*programming/remoteApiBindings*" y "*remoteApiBindings/python/python*" dentro del directorio de instalación. Estos archivos deben ser copiados al directorio del proyecto en *Python* para habilitar la interacción remota.

- **Habilitar puerto de conexión.**

  Dentro de la simulación del modelo, se debe crear un script no hilado, Figura 4.53, en el que se habilite la comunicación mediante un puerto, en este caso el "19999", utilizando el

comando en lenguaje LUA *simRemoteApi.start*(19999) en una función de inicialización. Con el comando de comunicación, cada vez que se ejecute la simulación del modelo, se inicia un intento de comunicación remota a través del puerto definido.

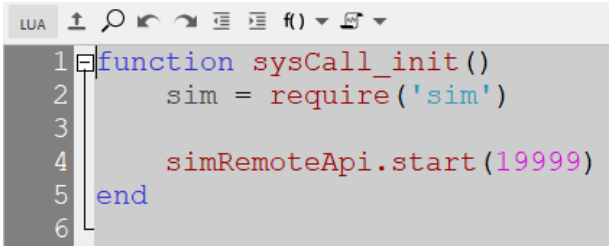



Figura 4.53: Configuración de puerto de comunicación en CoppeliaSim.

### 4.3.2. Programa de control de actuadores virtuales

El envío de los ángulos de movimiento al modelo virtual puede realizarse de dos formas distintas, un primer método consiste en una transmisión simultánea, es decir, integrando el envío de los ángulos calculados directamente en los programas de captura, de modo que el movimiento sea replicado justo después de haber sido detectado, la segunda opción consta de realizar el envío de ángulos posteriormente, una vez que se ha analizado por completo el video, y los ángulos capturados han sido almacenados en el archivo xlsx, separando el proceso de réplica y el de captura.

Dado que la mayor parte del código requerido para ambas opciones es muy similar, se optó por implementar tanto el envío de ángulos de forma simultánea como el envío posterior al análisis del video. Esta decisión es tomada con el objetivo de identificar la opción más adecuada para el sistema de réplica de movimiento, y la mejor manera de obtener este resultado es con la comparación de ambas alternativas.

El control de actuadores desde *Python*, al igual que los programas de captura, sigue una estructura que garantiza el envío y recepción de la información para accionar elementos en CoppeliaSim. Las etapas generales que debe seguir este proceso en cualquiera de las dos alternativas planteadas anteriormente, son las siguientes.

**Establecer conexión con CoppeliaSim.**

El programa de envío establece una vía de comunicación con el modelo virtual utilizando *sim.simxStart*. Esta función crea una conexión entre *Python* y CoppeliaSim definiendo la dirección IP local, el puerto TCP/IP (”19999”), el tiempo de espera y la configuración del hilo de comunicación.

```
clientID=sim.simxStart('127.0.0.1',19999,True,True,2000,5)
if clientID == 0: print("conectado a", port)
else: print("no se pudo conectar")
return clientID
```

Figura 4.54: Función de comunicación con CoppeliaSim.

**Obtención de identificadores o handlers de los actuadores.**

Para que el programa de control pueda accionar individualmente cada uno de los actuadores o *joints* virtuales, primero se extrae su identificador único. En la Figura 4.55, se muestra una fracción de código en donde la función *sim.simxGetObjectHandle* obtiene y almacena los identificadores de las piezas con el nombre *joint*1, *joint*2 y *joint*3, que corresponden a los motores de las articulaciones del modelo virtual.

```
ret,joint1=sim.simxGetObjectHandle(clientID,'joint1',sim.simx_opmode_blocking)
ret,joint2=sim.simxGetObjectHandle(clientID,'joint2',sim.simx_opmode_blocking)
ret,joint3=sim.simxGetObjectHandle(clientID,'joint3',sim.simx_opmode_blocking)
print(joint1, joint2,joint3)
```

Figura 4.55: Función de obtención de identificadores.

**Envío de ángulos de movimiento.**

Finalmente mediante el vínculo establecido entre el programa de control y los actuadores, y con el uso de los ángulos calculados se generan instrucciones para el movimiento de las articulaciones virtuales. El programa utiliza la función *sim.simxSetJointTargetPosition*, en la cual se envía desde *Python* una orden a CoppeliaSim para mover los actuadores *joint*1, *joint*2 y *joint*3, a una posición angular definida a partir del valor *angulo_cadera*, *angulo_rodilla* y *angulo_tobillo* respectivamente, convirtiendo los valores a radianes antes de ser enviados con

la función *np.deg2rad*. La orden se ejecuta en modo *blocking*, lo que implica que se envía el valor angular y el programa espera a que el motor alcance dicha posición.

```
sim.simxSetJointTargetPosition(clientID, joint1, np.deg2rad(angulo_cadera), sim.simx_opmode_oneshot)
sim.simxSetJointTargetPosition(clientID, joint2, np.deg2rad(angulo_rodilla), sim.simx_opmode_oneshot)
sim.simxSetJointTargetPosition(clientID, joint3, np.deg2rad(angulo_tobillo), sim.simx_opmode_oneshot)
```

Figura 4.56: Envió de valores angulares para la réplica de movimiento.

Para la integración con los programas de captura, el proceso de envío de ángulos de movimiento al modelo virtual es separado de acuerdo a las secciones principales de los programas. El establecimiento de la conexión con CoppeliaSim y la obtención de los identificadores de los actuadores, toman lugar en la sección tres de definición de funciones, por último, el envío de ángulos de movimiento se coloca justo después del almacenamiento de resultados en el archivo xlsx, en la sección cinco, captura y procesamiento de video.

El envío de ángulos de movimiento mediante el uso del archivo de resultados, implica la implementación de funciones para la lectura y manipulación de la información contenida en el archivo. Para el envío de posición mediante el uso del archivo de resultados, se implementa la función *pd.read_excel*, para la lectura de los datos, y para el envío en secuencia de los valores angulares se utiliza un bucle *for* para recorrer fila por fila los datos contenidos en el archivo, Figura 4.57. Extrayendo los valores en grados desde las columnas y asignándolos a variables individuales para ser procesados y enviados a los actuadores virtuales.

```
for index, fila in datos.iterrows():
    # Lectura de los valores en grados desde el archivo Excel
    angulo_cadera = fila["Cadera"]
    angulo_rodilla = fila["Rodilla"]
    angulo_tobillo = fila["Tobillo"]

    # Envío de los valores angulares a los actuadores virtuales
    sim.simxSetJointTargetPosition(clientID, joint1, np.deg2rad(angulo_cadera), sim.simx_opmode_oneshot)
    sim.simxSetJointTargetPosition(clientID, joint2, np.deg2rad(angulo_rodilla), sim.simx_opmode_oneshot)
    sim.simxSetJointTargetPosition(clientID, joint3, np.deg2rad(angulo_tobillo), sim.simx_opmode_oneshot)
```

Figura 4.57: Bucle for para el envío de ángulos a CoppeliaSim.

### 4.3.3. Sistema de réplica de movimiento físico

El sistema de réplica físico se divide en dos partes importantes, un script en Python encargado de la lectura y el envío de los ángulos de movimiento a los motores GM25-370 mientras al mismo tiempo recibe retroalimentación sobre su posición, y un segundo programa en lenguaje **C++** implementado en una tarjeta Arduino UNO, la cual es responsable del control directo de los actuadores y de la lectura de la información de su encoder integrado. Este procedimiento deja fuera el uso del programa CoppeliaSim para el control de los motores, lo que facilita el envió de posiciones al no tener que elaborar un programa especial en lenguaje LUA en el entorno de simulación.

**Programa de lectura y envío de datos en Pyhton.**

El programa de envío de datos requiere una vía de comunicación directa con el Arduino UNO, junto con la especificación del archivo a utilizar para el envío de datos. En la Figura 4.58 se muestra el fragmento de código donde se definen parámetros de la conexión serial, así como el tiempo entre cada envío de ángulos.

```
Puerto_serial = 'COM3'
Baudios = 9600
fps = 24
TIEMPO_ENTRE_ANGULOS = 1.0 / fps

Archivo = 'angulos_movimiento.xlsx'
Columnas = ['Cadera', 'Rodilla', 'Tobillo']
Archivo_salida = 'replay_respuesta.xlsx'

df = pd.read_excel(Archivo)
```

Figura 4.58: Configuración inicial para comunicación con Arduino UNO.

Nuevamente se utiliza la función *pd.read_excel* para la lectura de los datos de un archivo definido previamente, estableciendo también las columnas donde se encuentra la información de cada una de las articulaciones, para posteriormente almacenar estos datos en listas separadas.

En la Figura 4.59, se muestra el fragmento de código encargado de establecer la conexión entre el programa de Python y el Arduino UNO, mediante la función *serial.Serial*. En el código se define el puerto serial $'COM3'$, y se establece la velocidad de transmisión de datos, $Baudios = 9600$.

```
arduino = serial.Serial(Puerto_serial, Baudios, timeout=3)
time.sleep(2)
```

Figura 4.59: Función de conexión por puerto serial con Arduino UNO.

Después de establecer la conexión serial para comenzar con el envío de ángulos a los motores, primero se define una iteración utilizando la longitud de una de las listas que almacenan los ángulos de movimiento, en este caso los de la cadera *ang_cadera*, Figura 4.60. Además, se toma el tiempo de inicio del ciclo para controlar la frecuencia del envió.

```
for i in range(len(ang_cadera)):
    tiempo_inicio = time.time()

    mensaje = f"{ang_cadera[i]},{ang_rodilla[i]},{ang_tobillo[i]}\n"
    arduino.write(mensaje.encode('utf-8'))
    print(f"[{i+1}] Enviado: {mensaje.strip()}")
```

Figura 4.60: Envió de ángulos de movimiento al sistema de réplica físico.

La transmisión de ángulos hacia la tarjeta se realiza mediante la función *arduino.write*, en donde se utiliza una cadena de texto tipo *string* que contiene tres ángulos (Cadera, Rodilla, Tobillo), los cuales cambian con cada iteración por el índice $i$. Para que el Arduino pueda utilizar los datos enviados, $utf-8$ convierte el texto *string* en una secuencia de bytes, los cuales pueden ser interpretados por la tarjeta.

Se calcula el tiempo transcurrido desde el inicio del ciclo, y se utiliza para determinar si el programa debe realizar el siguiente envío de ángulos o seguir esperando. En el código mostrado en la Figura 4.61, la sentencia $if\ restante > 0$, se encarga de detener temporalmente el programa para que no se envíen ángulos antes de tiempo.

```
tiempo_transcurrido = time.time() - tiempo_inicio
restante = TIEMPO_ENTRE_ANGULOS - tiempo_transcurrido
if restante > 0:
    time.sleep(restante)
```

Figura 4.61: Sentencia de tiempo de espera entre envío de ángulos al sistema de réplica físico.

**Programa de control de actuares en Arduino UNO.**

Dentro del programa de control de actuadores, como en el resto de programas primero se realiza una serie de definiciones, en este caso se establecen primero las conexiones correspondientes con cada uno de los motores y sus encoders, Figura 4.62.

```
// Motor 1: Cadera
const int PWM1 = 5;
const int AIN1 = 7;
const int AIN2 = 8;
const int ENCA1 = 2;
const int ENCB1 = 3;

// Motor 2: Rodilla
const int PWM2 = 6;
const int BIN1 = 9;
const int BIN2 = 10;
const int ENCA2 = A4;
const int ENCB2 = A5;

// Motor 3: Tobillo
const int PWM3 = 11;
const int CIN1 = 12;
const int CIN2 = 13;
const int ENCA3 = A0;
const int ENCB3 = A1;
```

Figura 4.62: Definición de conexiones en la tarjeta Arduino UNO.

Se definen variables que almacenan la posición actual de los motores en ticks y el último valor codificado para interpretar el sentido de giro del encoder, Figura 4.63.

```
volatile long pos1 = 0, pos2 = 0, pos3 = 0;
volatile int last1 = 0, last2 = 0, last3 = 0;

double Kp = 2.0, Ki = 0.001, Kd = 0.5;

long target1 = 0, target2 = 0, target3 = 0;
double err1 = 0, err2 = 0, err3 = 0;
double lastErr1 = 0, lastErr2 = 0, lastErr3 = 0;
double int1 = 0, int2 = 0, int3 = 0;
```

Figura 4.63: Parámetros y variables de error para control PID.

Debido a que se utilizan motores DC en este sistema, los cuales tienden a no llegar o sobrepasar la posición deseada, se aplica un control PID para mantener la precisión y estabilidad en el proceso de réplica. En el fragmento de código de la figura anterior se definen los parámetros de este controlador, al igual que las variables de objetivo en ticks, el error actual, error previo y error de acumulación.

Justo después de definir los parámetros del control PID comienza la inicialización, donde lo más importante es la configuración de todos los pines de motores y de encoders como entrada o salida a través de *pinmode*, y la asignación de interrupciones a los pines de los encoders para la actualización de las pociones de los motores mediante *attachInterrupt*, Figura 4.64.

```
Serial.begin(9600);

pinMode(PWM1, OUTPUT); pinMode(AIN1, OUTPUT); pinMode(AIN2, OUTPUT);
pinMode(PWM2, OUTPUT); pinMode(BIN1, OUTPUT); pinMode(BIN2, OUTPUT);
pinMode(PWM3, OUTPUT); pinMode(CIN1, OUTPUT); pinMode(CIN2, OUTPUT);

pinMode(ENCA1, INPUT_PULLUP); pinMode(ENCB1, INPUT_PULLUP);
pinMode(ENCA2, INPUT_PULLUP); pinMode(ENCB2, INPUT_PULLUP);
pinMode(ENCA3, INPUT_PULLUP); pinMode(ENCB3, INPUT_PULLUP);

attachInterrupt(digitalPinToInterrupt(ENCA1), leerEncoder1, CHANGE);
attachInterrupt(digitalPinToInterrupt(ENCB1), leerEncoder1, CHANGE);
attachInterrupt(digitalPinToInterrupt(ENCA2), leerEncoder2, CHANGE);
attachInterrupt(digitalPinToInterrupt(ENCB2), leerEncoder2, CHANGE);
attachInterrupt(digitalPinToInterrupt(ENCA3), leerEncoder3, CHANGE);
attachInterrupt(digitalPinToInterrupt(ENCB3), leerEncoder3, CHANGE);
```

Figura 4.64: Configuración de pines.

En el bucle principal de este programa, se realiza la recepción de la linea de datos que se ha enviado desde el programa de Python, Figura 4.65, para después proceder con la separación de posiciones a través de la identificación de comas en la linea de datos.

```
if (Serial.available()) {
  String entrada = Serial.readStringUntil('\n');
  entrada.trim();

  int idx1 = entrada.indexOf(',');
  int idx2 = entrada.lastIndexOf(',');
```

Figura 4.65: Sentencia if para la recepción de ángulos de movimiento.

En seguida, mediante *entrada.substring* se extraen los ángulos de las articulaciones desde la linea de datos recibida, convirtiendo al mismo tiempo los datos en ticks para su uso en cada motor, Figura 4.66.

```
float ang1 = entrada.substring(0, idx1).toFloat();
float ang2 = entrada.substring(idx1 + 1, idx2).toFloat();
float ang3 = entrada.substring(idx2 + 1).toFloat();

target1 = long((ang1 / 360.0) * TICKS_POR_REV);
target2 = long((ang2 / 360.0) * TICKS_POR_REV);
target3 = long((ang3 / 360.0) * TICKS_POR_REV);
```

Figura 4.66: Extracción de posiciones.

Posteriormente tal como se muestra en la Figura 4.67, se llama a la función *moverMotor*, para la realización de los cálculos PID y el accionamiento del motor de acuerdo a los parámetros correspondientes.

```
moverMotor(target1, pos1, err1, lastErr1, int1, PWM1, AIN1, AIN2);
moverMotor(target2, pos2, err2, lastErr2, int2, PWM2, BIN1, BIN2);
moverMotor(target3, pos3, err3, lastErr3, int3, PWM3, CIN1, CIN2);
```

Figura 4.67: Función de accionamiento de actuadores.

Y por último mediante una sentencia $if$, Figura 4.68, se evalúa si los motores han alcanzado la posición deseada dentro de un margen de tolerancia y de ser así con la función *detenerMotores*, el movimiento de los motores es interrumpido, enviando al mismo tiempo la posición actual de cada uno de los motores.

```
if (todoListo()) {

  detenerMotores();
  Serial.print("OK:");
  Serial.print((pos1 * 360.0) / TICKS_POR_REV, 2); Serial.print(",");
  Serial.print((pos2 * 360.0) / TICKS_POR_REV, 2); Serial.print(",");
  Serial.println((pos3 * 360.0) / TICKS_POR_REV, 2);
  return;

}
```

Figura 4.68: Sentencia if de evaluación para la detención de actuadores.

Este ciclo se repite nuevamente una y otra vez hasta que la comunicación serial sea detenida en el programa de python, o lo que es igual a que se terminen los ángulos de movimiento de las articulaciones en el archivo de resultados.

# Capítulo 5

# Análisis y comparación de resultados

En esta sección se exponen los resultados de las pruebas comparativas realizadas a los dos sistemas de captura desarrollados, con el propósito de determinar la superioridad de uno de los sistemas sobre otro. Se realizaron pruebas de evaluación de niveles de ruido en las mediciones, ante movimientos lentos, además, se verificó la coherencia biomecánica de las mediciones obtenidas tomando como referencia los rangos de movimiento promedio de los miembros inferiores del cuerpo humano vistos en el **Capítulo 2**.

En este apartado también se presenta el análisis de resultados de las pruebas realizadas en los sistemas de réplica de movimiento. Las evaluaciones realizadas, se centraron en determinar el método más eficiente para la transmisión de ángulos de movimiento al sistema virtual, en la verificación de precisión de la réplica en los actuadores de ambos sistemas, y el tiempo de ejecución de las rutinas de movimiento analizadas.

Todos los videos utilizados durante la realización de pruebas han sido seleccionados debido a que utilizan como sujetos de prueba a personas del sexo femenino, que en su mayoría cuentan con ropa deportiva más ajustada lo que permite una mejor detección por parte de los sistemas de captura, además comparten similitudes en su entorno, pues el sujeto de prueba siempre resalta

en todas las tomas. Debido al requerimiento de videos de cuerpo completo que mostraran la realización de ejercicios de miembro inferior, el material audiovisual se obtuvo de dos fuentes principales, de la pagina oficial de la revista **Women´s Health** [43], y del centro de aprendizaje **Hinge Health** [46].

## 5.1. Captura de movimiento

### 5.1.1. Evaluación de ruido con análisis de posiciones cuasi-estáticas

En esta evaluación, los programas de captura analizaron videos de personas que realizan movimientos lentos y tratan de mantener una postura durante un corto periodo de tiempo. Debido a la complejidad de mantener ciertas posiciones junto a la realización de movimientos involuntarios, como la respiración, las posiciones inmóviles ejecutadas en los videos se consideraron cuasi-estáticas.

Para su análisis visual los resultados de ambos sistemas de captura son presentados en gráficas, adicionalmente se utiliza la raíz cuadrada de las diferencias sucesivas o por sus siglas en ingles *RMSSD*, para una evaluación más precisa sobre la estabilidad de los sistemas de captura. Esta métrica se cálculo utilizando los resultados de los momentos de baja movilidad de cada una de las pruebas, un valor *RMSSD* bajo (cercano a cero) en los segmentos mencionados es un indicativo directo de una mínima variabilidad en las mediciones, lo que se asocia con un alto grado de estabilidad y un bajo nivel de ruido en los datos capturados.

**Primer análisis de estabilidad.**

En esta primer prueba se utilizó un video en el cual una mujer realiza el ejercicio *Dead Bug*, Figura 5.1, el cual consiste en mantener flexionadas ambas piernas en el aire, mientras se extiende alternadamente una pierna y el brazo opuesto.

Figura 5.1: Ejercicio dead bug [46].

En todas las gráficas que muestran los resultados del análisis de movimiento de las articulaciones de interés, el eje ”*y*” corresponde a los grados de movimiento alcanzados y el eje ”*x*” al tiempo transcurrido, también se identifican en color azul los resultados del sistema basado en *MediaPipe* y en color rojo los resultados del sistema basado en *MMPose*.

En los resultados de la cadera, Figura 5.2, ambos sistemas de captura siguen la tendencia del cambio de posición de forma correcta, pero en momentos de poca movilidad existe una mayor presencia de ruido por parte de *MMPose*, aunque en una menor medida los resultados de la rodilla, Figura 5.3, muestran este mismo comportamiento.

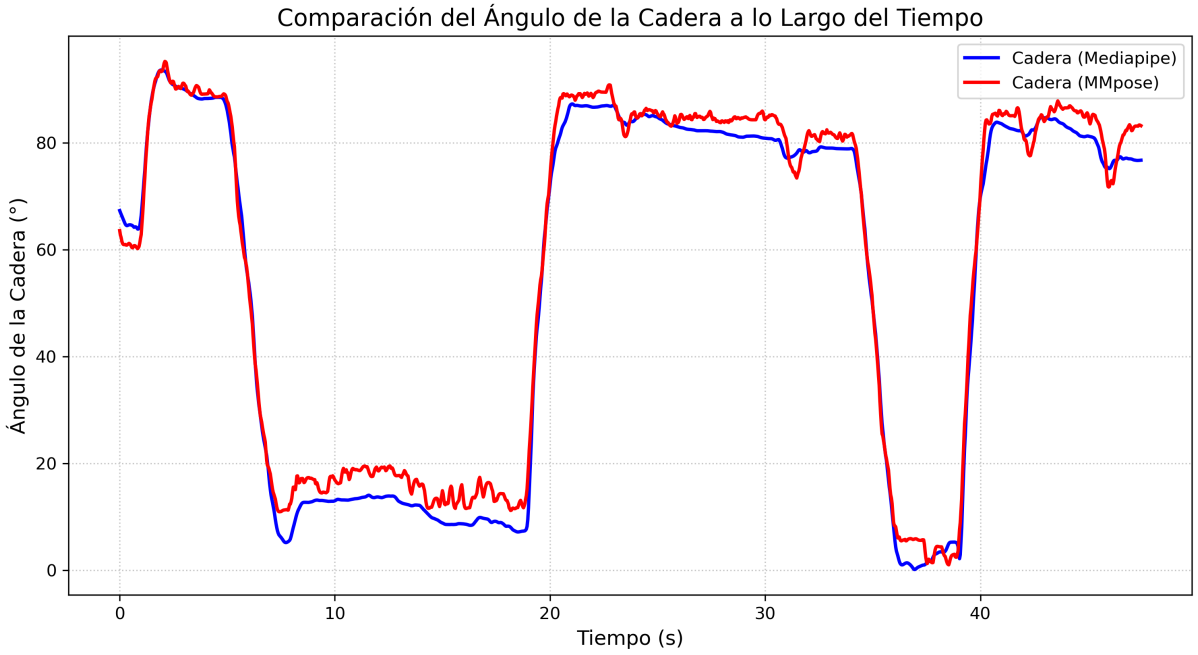



Figura 5.2: 1er Gráfica de estabilidad de resultados del análisis de la cadera.

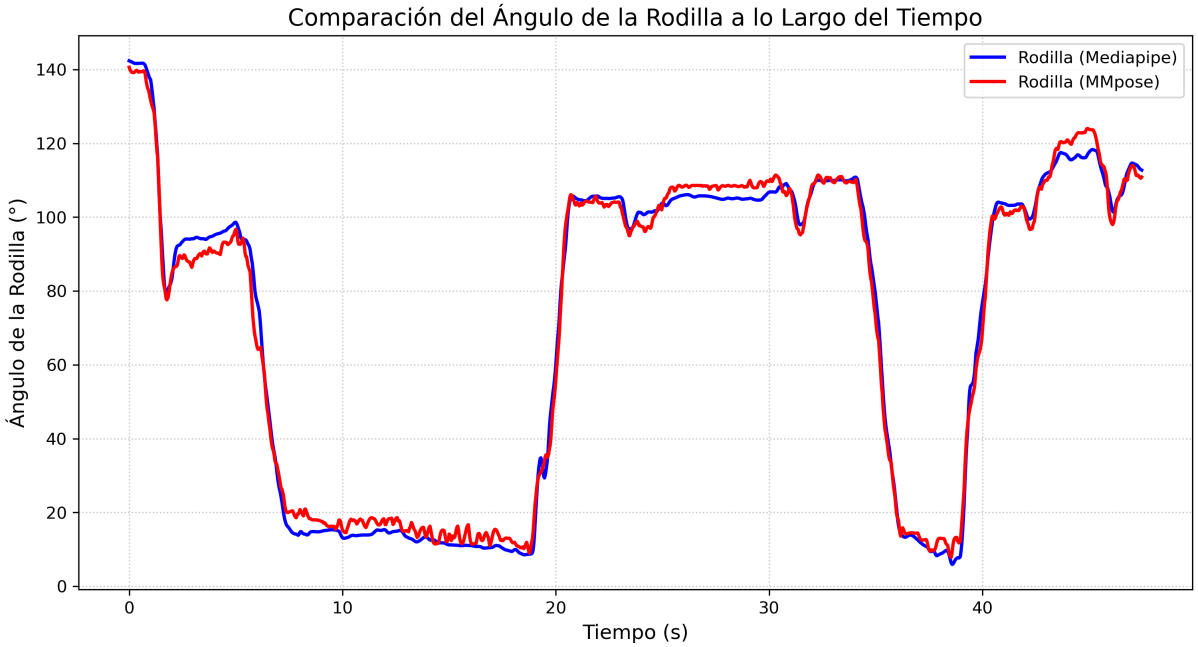



Figura 5.3: 1er Gráfica de estabilidad de resultados del análisis de la rodilla.

Para la articulación del tobillo, Figura 5.4, los resultados de ambos sistemas son visiblemente más erráticos, siguen una tendencia de movimiento ascendente pero con múltiples fluctuaciones, los cuales son más notables en el sistema basado en *MMPose*.

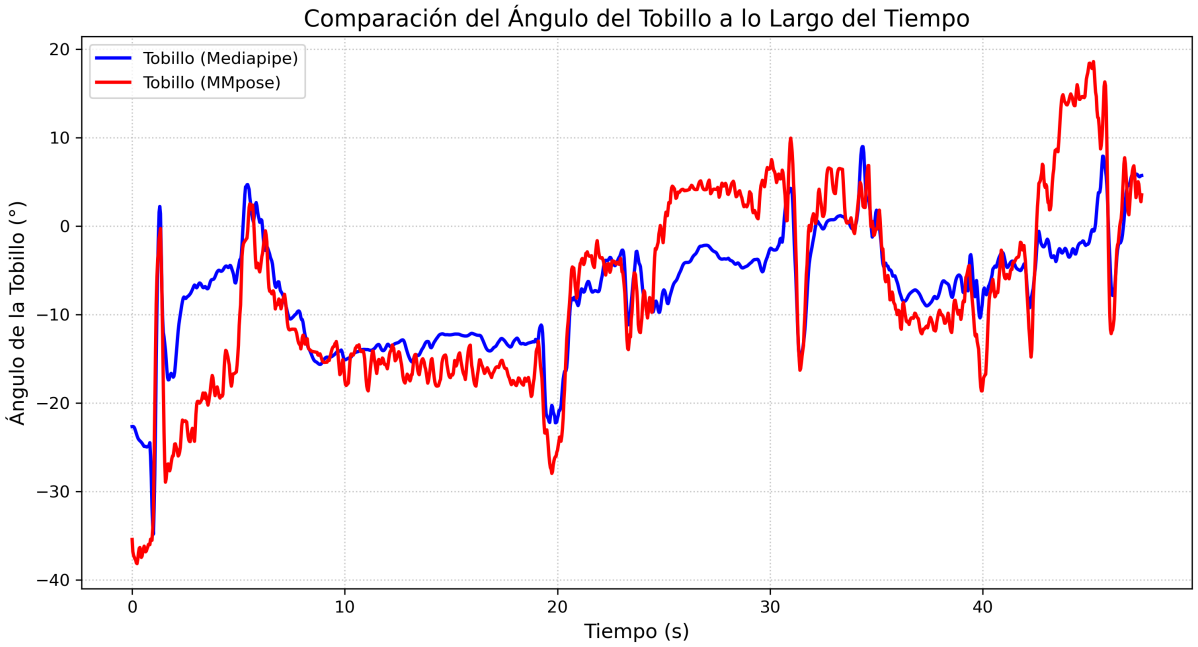



Figura 5.4: 1er Gráfica de estabilidad de resultados del análisis del tobillo.

**Segundo análisis de estabilidad.**

El ejercicio de movimiento analizado en esta prueba consta del levantamiento momentáneo de la pierna izquierda, manteniendo en una sola posición las articulaciones de la rodilla y tobillo, Figura 5.5.

Figura 5.5: Ejercicio de elevación de pierna recta [46].

Las figuras 5.6, 5.7 y 5.8, muestran las gráficas de comparación de resultados de ambos sistemas. La diferencia de estabilidad es más notoria en los resultados del sistema de *MMPose*, aún cuando los datos de la cadera y tobillo muestran tendencias similares con respecto a los resultados de *MediaPipe*, hay presencia de ruido en el movimiento angular de la rodilla.

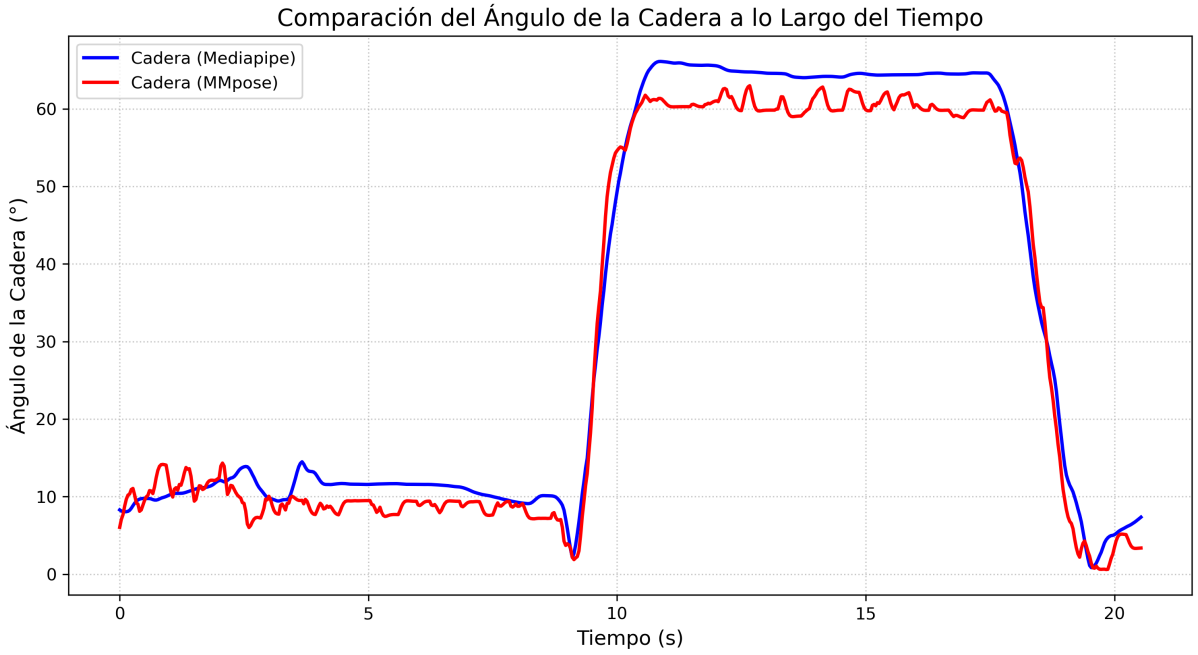



Figura 5.6: 2da Gráfica de estabilidad de resultados del análisis de la cadera.

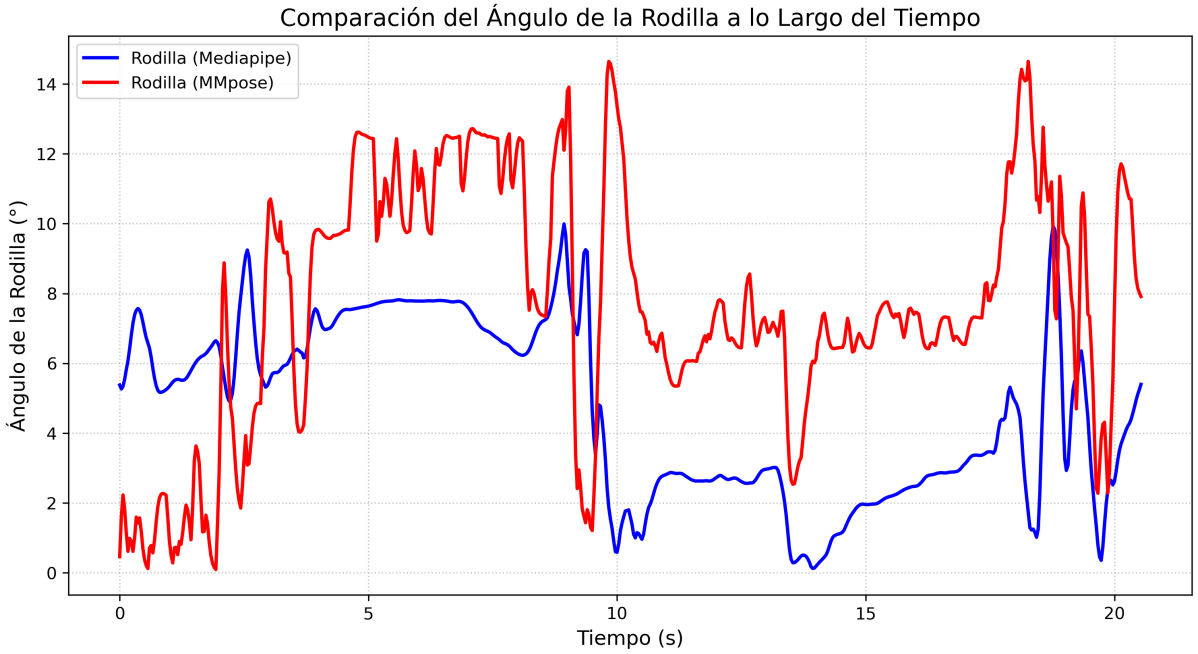



Figura 5.7: 2da Gráfica de estabilidad de resultados del análisis de la rodilla.

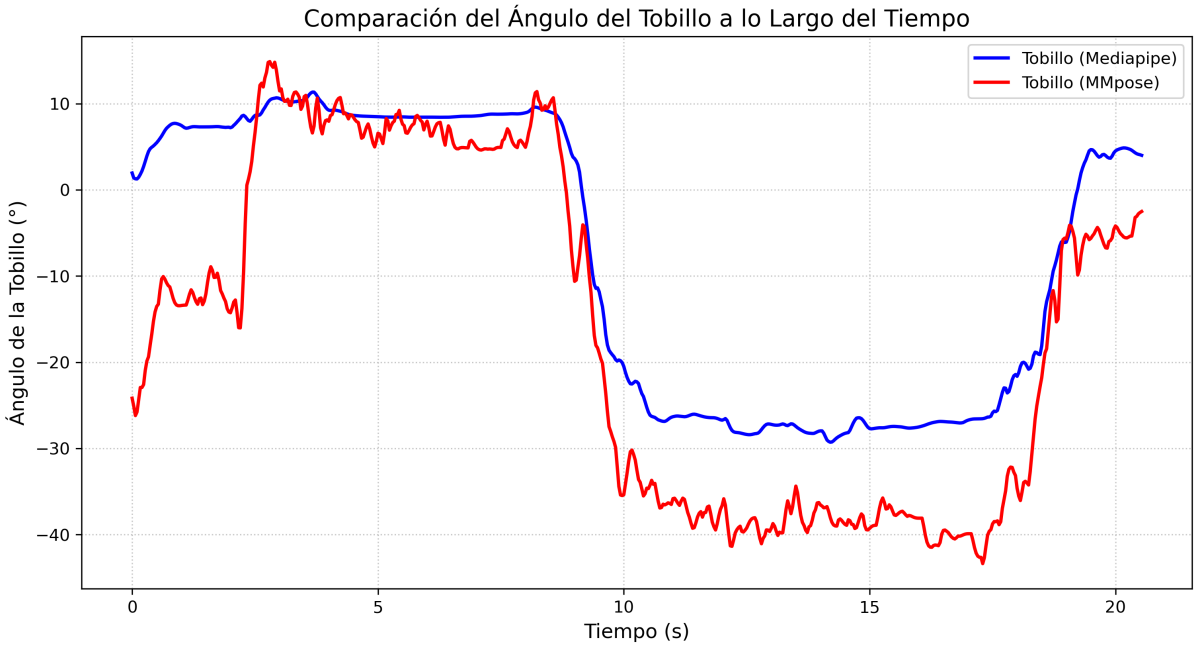



Figura 5.8: 2da Gráfica de estabilidad de resultados del análisis del tobillo.

En los resultados del tobillo, durante el ejercicio de levantamiento el sujeto realiza un movimiento de dorsiflexión, ambos sistemas detectan este ligero cambio pero es notable que *MMPose* carece de estabilidad, ya que sus resultados son visiblemente más erráticos.

**Tercer análisis de estabilidad.**

En esta prueba se utiliza un video que muestra el proceso de elevación de una silla, Figura 5.9, una persona comienza estando sentada y con movimientos suaves lentamente se pone de pie, y mantiene esa posición por unos segundos antes de volver a sentarse.

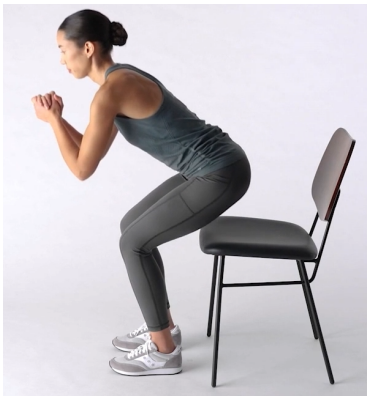


Figura 5.9: Ejercicio de elevación desde una silla [46].

Los resultados de la cadera y rodilla mostrados en las figuras 5.10 y 5.11, son consistentes entre ambos sistemas, aunque *MediaPipe* continua mostrando una mejor estabilidad en todo momento.

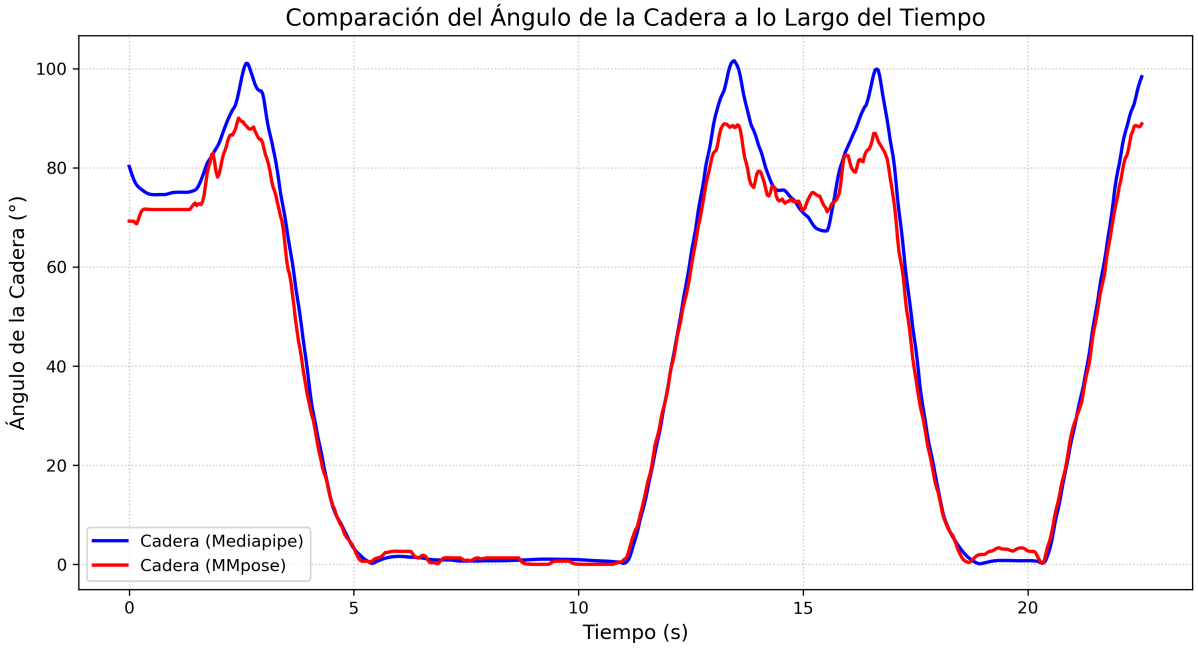



Figura 5.10: 3er Gráfica de estabilidad de resultados del análisis de la cadera.

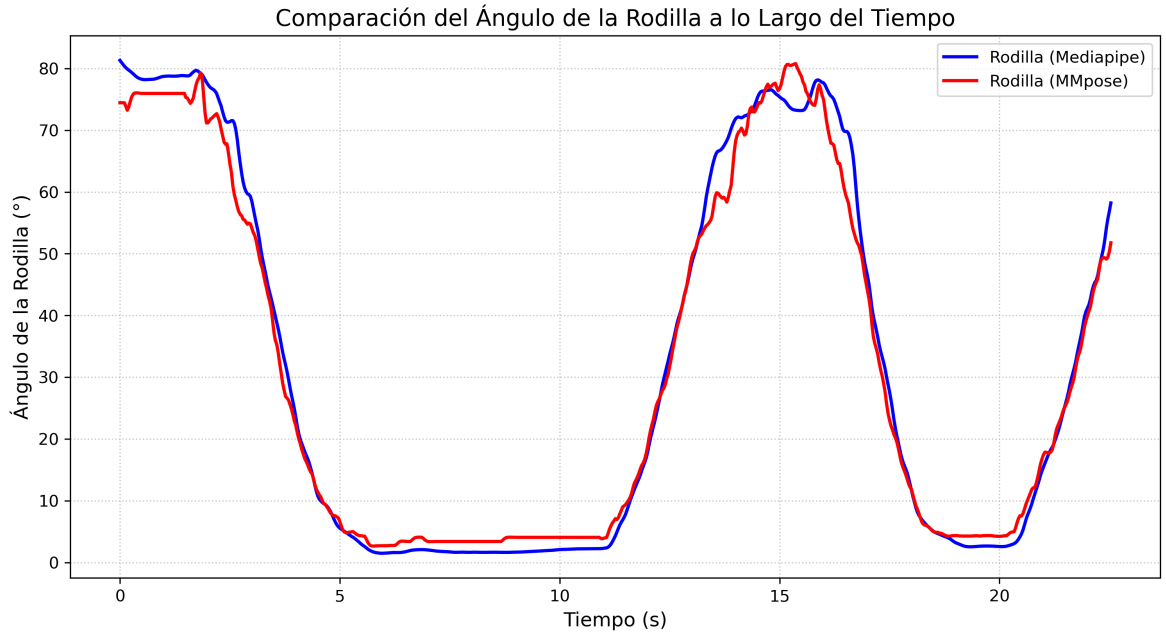



Figura 5.11: 3er Gráfica de estabilidad de resultados del análisis de la rodilla.

Los sistemas de captura detectan casi de forma sincronizada los momentos de flexión y extensión de la cadera y rodilla, sin embargo fluctuaciones en el sistema *MMPose* son notables. En el tobillo, Figura 5.12, el problema de inestabilidad en *MMpose* es más evidente.

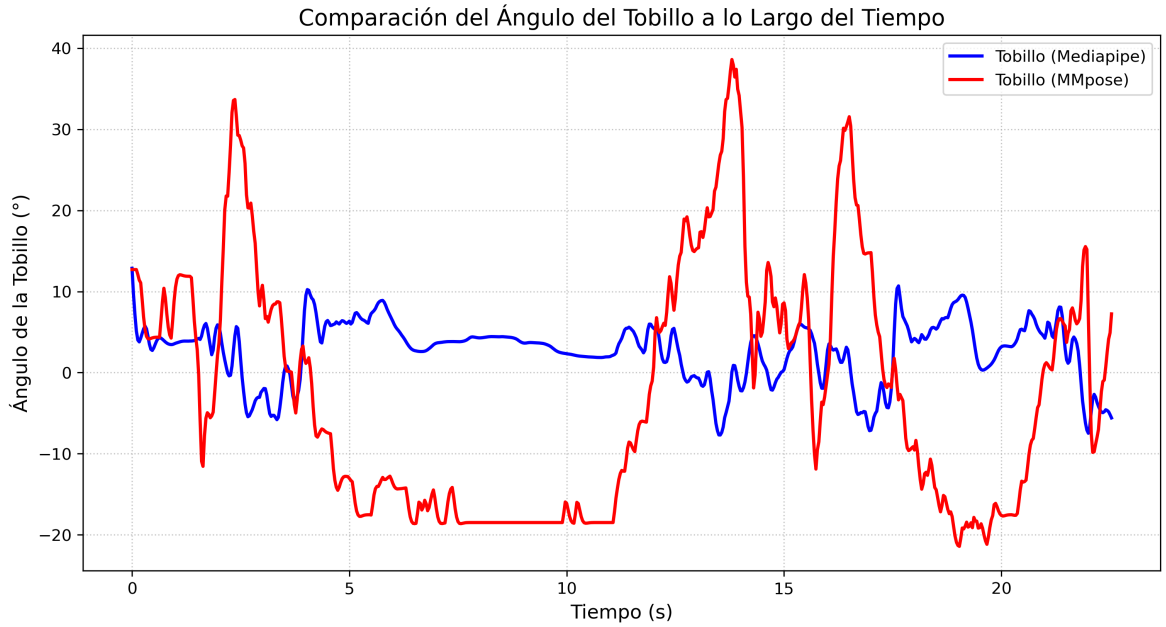



Figura 5.12: 3er Gráfica de estabilidad de resultados del análisis del tobillo.

**Cuarto análisis de estabilidad.**

En esta prueba se analizó un video de una persona realizando el ejercicio de bisagra de cadera [46], Figura 5.13, el ejercicio comienza de pie y se flexiona la cadera junto a las rodillas lentamente, sin llegar a juntar el torso y los muslos por completo.

Figura 5.13: Ejercicio de bisagra de cadera [46].

En las figuras 5.14, 5.15 y 5.16, se muestran los resultados del análisis, se puede destacar la gran coincidencia de ambos sistemas en sus resultados salvo por los momentos cuasi-estáticos, en los cuales el sistema basado en *MMPose* sigue una tendencia más errática.

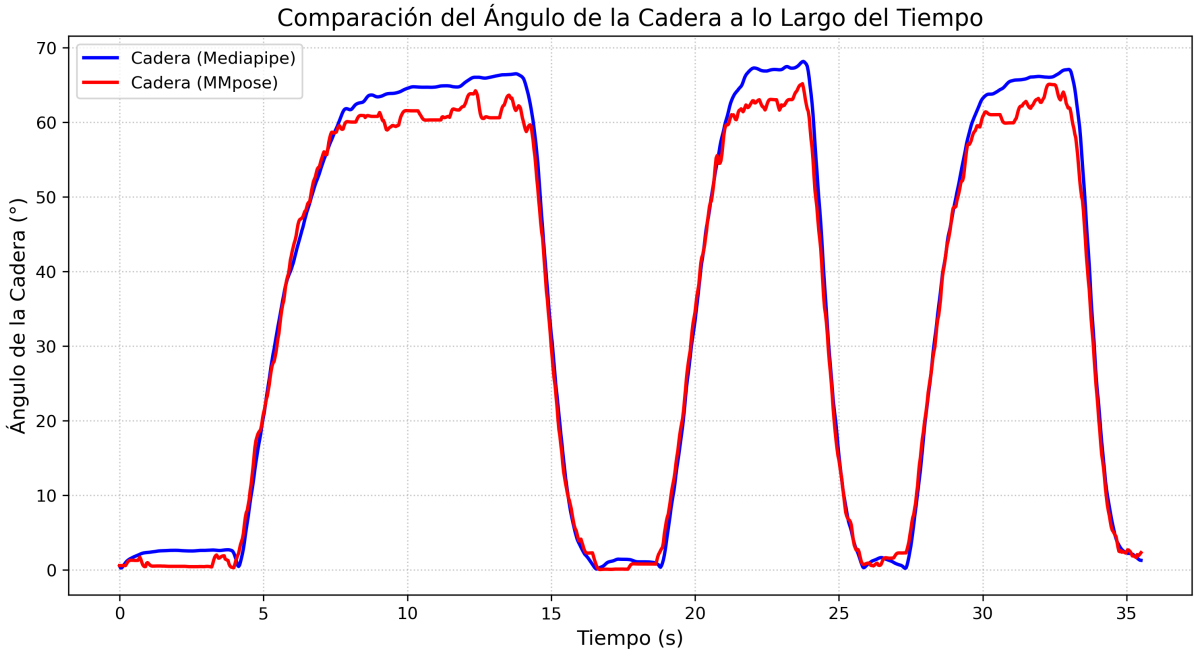



Figura 5.14: 4ta Gráfica de estabilidad de resultados del análisis de la cadera.

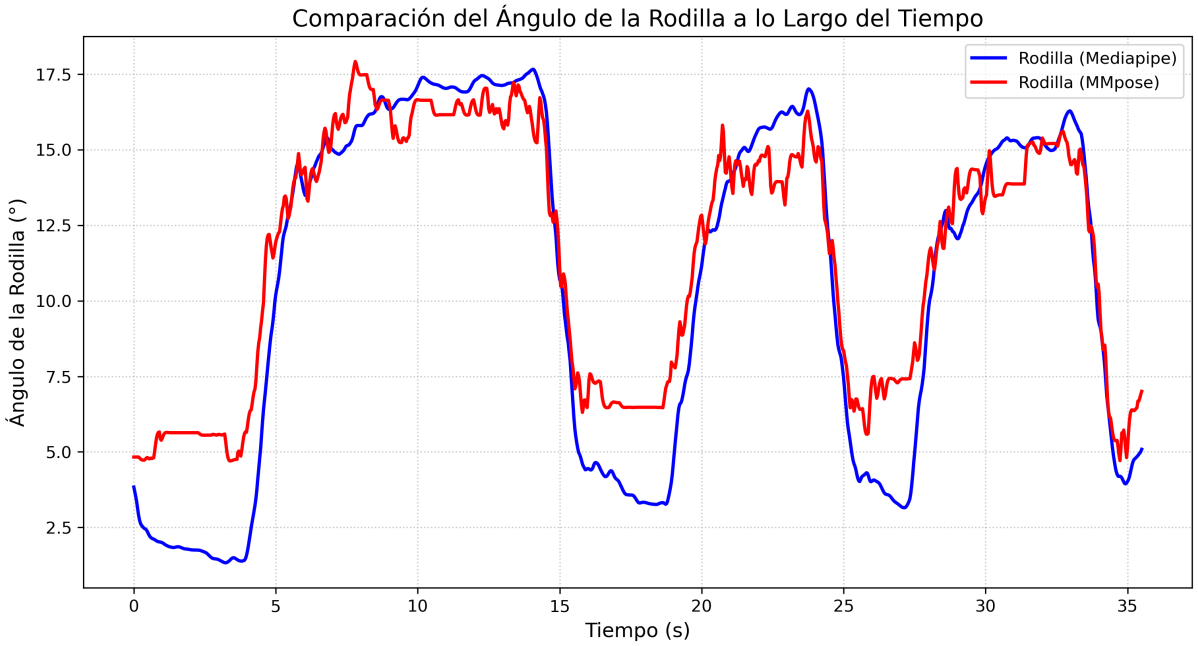



Figura 5.15: 4ta Gráfica de estabilidad de resultados del análisis de la rodilla.

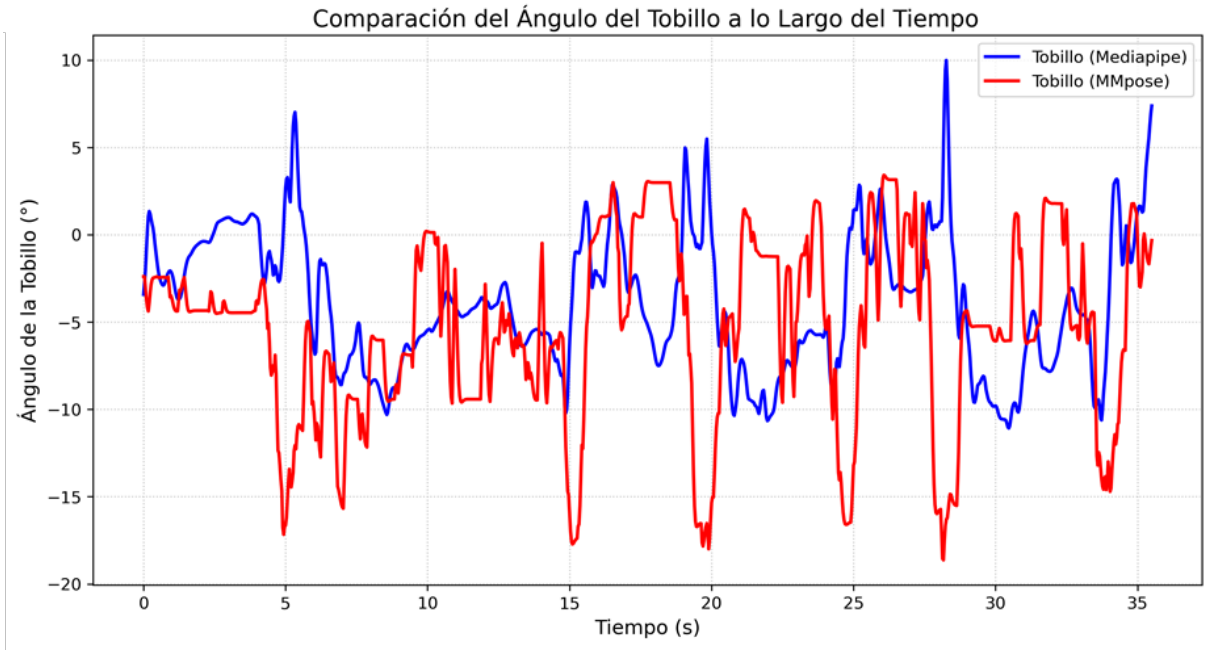



Figura 5.16: 4ta Gráfica de estabilidad de resultados del análisis del tobillo.

Los resultados de la rodilla, muestran que ambos sistemas siguen una tendencia de movimiento, las curvas suben y bajan de manera casi sincronizada, aunque nuevamente el sistema basado en *MediaPipe* mantiene una mejor estabilidad durante los momentos de poca

movilidad. Para la articulación del tobillo, los datos capturados carecen de concordancia gran parte del tiempo, hay presencia de picos y valles en momentos distintos y aún cuando *MediaPipe* mantiene una mejor estabilidad, en general los datos capturados visualmente son erráticos en ambos sistemas.

**Quinto análisis de estabilidad.**

Para esta prueba se analizó una rutina en la que se realizan sentadillas medias y completas, Figura 5.17, en ambas técnicas se mantiene la posición por un breve momento antes de levantar el cuerpo.

Figura 5.17: Ejercicio de sentadillas [46].

Los resultados del análisis de muestran en las figuras 5.18, 5.19 y 5.20, en esta comparación visual al igual que en varios resultados anteriores se aprecia una sincronización en el cambio de posición de las articulaciones por parte de los dos sistemas.

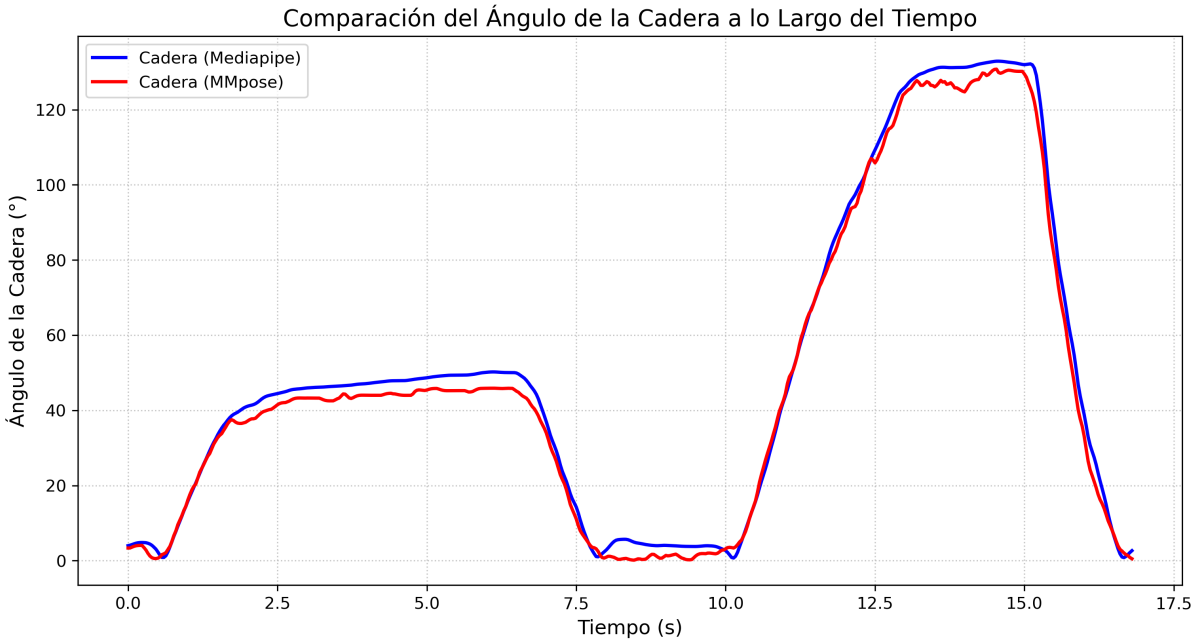



Figura 5.18: 5ta Gráfica de estabilidad de resultados del análisis de la cadera.

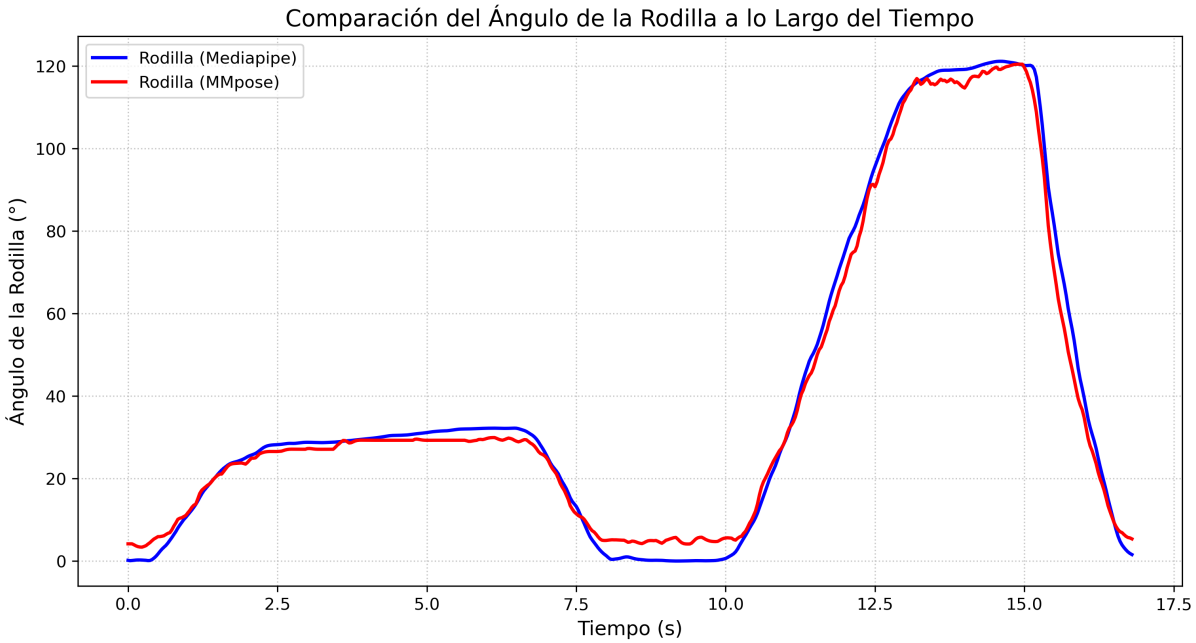



Figura 5.19: 5ta Gráfica de estabilidad de resultados del análisis de la rodilla.

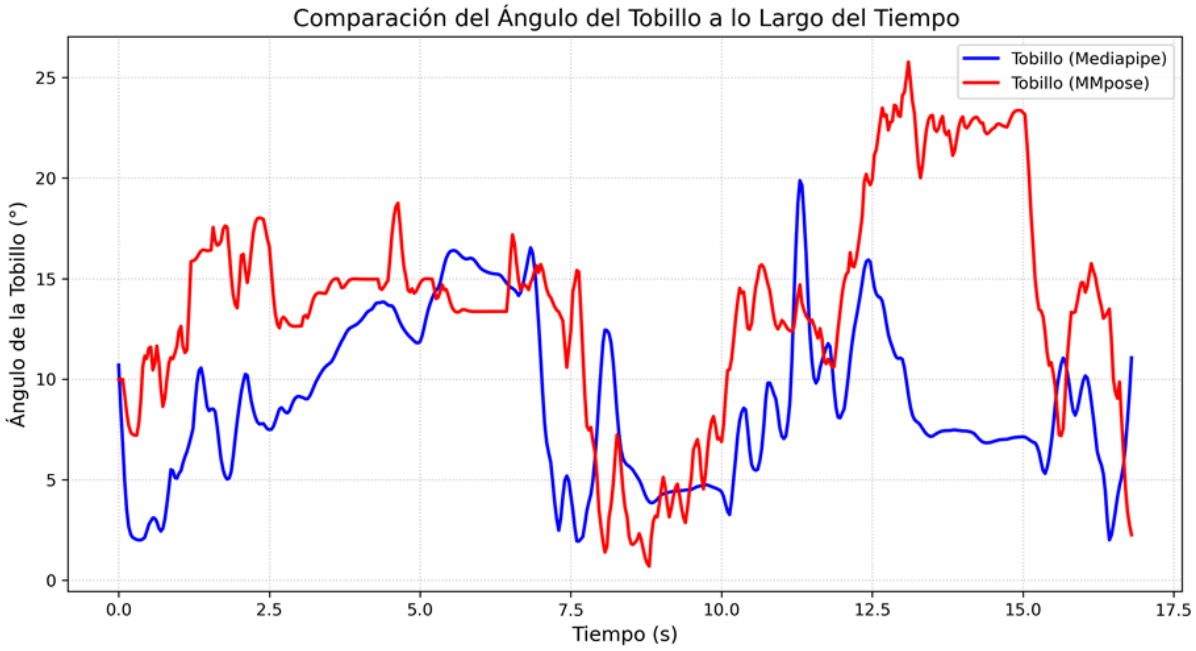



Figura 5.20: 5ta Gráfica de estabilidad de resultados del análisis del tobillo.

Nuevamente los resultados del tobillo, que deberían mantenerse estables por la técnica del ejercicio, presentan diversas fluctuaciones y datos erráticos en ambos sistemas, aunque el rango de variación de los datos en el sistema basado en *MediaPipe* es menor.

### 5.1.2. Evaluación de estabilidad mediante RMSSD

Para la evaluación de estabilidad mediante la raíz cuadrada de la media de diferencias sucesivas, se aislaron los resultados de los momentos cuasi-estáticos de las pruebas anteriores, y mediante el uso de la ecuación 5.1, se realizó el cálculo de la métrica. Los resultados se dividen de acuerdo a la prueba de la cual fueron extraídos los datos, distinguiendo los momentos de baja movilidad con lapsos de tiempo.

$$\textbf{RMSSD} = \sqrt{\frac{1}{N-1}\sum_{i=2}^{N}(x_i - x_i - 1)^2} \tag{5.1}$$

En determinadas pruebas debido a que no todas las articulaciones cambian de posición o realizan movimiento significativo, no se distinguen momentos de estabilidad, por lo que se aplica la métrica **RMSSD** a todo el conjunto de datos, complementando este análisis con el cálculo de la desviación estándar para estos casos.

**Primera prueba.**

De los resultados de esta prueba se tomaron en cuenta los tres momentos de baja movilidad o cuasi-estáticos más significativos para ser evaluados, los resultados de la aplicación de la RMSSD son los mostrados en la Tabla 5.1.

Tabla 5.1: Resultados de métrica RMSSD de la primer prueba de estabilidad.

| **Segmento** | **Articulación** | **RMSSD** *MediaPipe* | **RMSSD** *MMPose* |
|---|---|---|---|
| Segundos 9-19 | Cadera | 0.1369 | 0.4345 |
| | Rodilla | 0.2786 | 0.5110 |
| Segundos 21-33 | Cadera | 0.1680 | 0.3167 |
| | Rodilla | 0.3228 | 0.4515 |
| Segundos 41-47 | Cadera | 0.1496 | 0.4566 |
| | Rodilla | 0.4307 | 0.6768 |

La articulación del tobillo por contar con un mínimo movimiento durante toda la rutina de ejercicio, su métrica se calculó utilizando todo el conjunto de datos, el resultado es mostrado en la Tabla 5.2 junto con el resultado de la desviación estándar en esta articulación para ambos sistemas.

Tabla 5.2: RMSSD y desviación estándar del tobillo en la primer prueba de estabilidad.

| **Sistema** | **RMSSD** | **Desviación estándar** |
|---|---|---|
| **MediaPipe** | 0.6150 | 6.4083 |
| **MMPose** | 0.9582 | 10.2476 |

**Segunda prueba.**

En esta rutina de movimiento se distinguen dos momentos de mínima movilidad, los resultados del cálculo de la métrica para los datos de estos segmentos se muestran en la Tabla 5.3.

Tabla 5.3: Resultados de métrica RMSSD de la segunda prueba de estabilidad.

| **Segmento** | **Articulación** | **RMSSD** *MediaPipe* | **RMSSD** *MMPose* |
|---|---|---|---|
| Segundos 0-8 | Cadera | 0.1606 | 0.5315 |
| | Tobillo | 0.1285 | 1.3933 |
| Segundos 11-17 | Cadera | 0.1065 | 0.3523 |
| | Tobillo | 0.1494 | 0.7151 |

Por la naturaleza del ejercicio, en esta prueba la rodilla no cambia de posición pues se mantiene extendida en todo momento, por lo que se utilizó el lote completo de resultados de esta articulación en el cálculo de la RMSSD, Tabla 5.4.

Tabla 5.4: RMSSD y desviación estándar de la rodilla en la segunda prueba de estabilidad.

| **Sistema** | **RMSSD** | **Desviación estándar** |
|---|---|---|
| **MediaPipe** | 0.2870 | 2.4982 |
| **MMPose** | 0.6327 | 3.5144 |

**Tercera prueba.**

De los resultados de este análisis de levantamiento de una silla, se tomaron en cuenta dos momentos breves de poca movilidad, los resultados de la RMSSD de esta rutina se muestran en la Tabla 5.5

Tabla 5.5: Resultados de métrica RMSSD de la tercera prueba de estabilidad.

| **Segmento** | **Articulación** | **RMSSD** *MediaPipe* | **RMSSD** *MMPose* |
|---|---|---|---|
| Segundos 6-11 | Cadera | 0.0387 | 0.1625 |
| | Rodilla | 0.0349 | 0.0973 |
| Segundos 18-20 | Cadera | 0.2001 | 0.2495 |
| | Rodilla | 0.0898 | 0.1154 |

En esta prueba la articulación del tobillo no realiza cambios de posición significativos por lo que se aplico la métrica a todos los ángulos capturados. La Tabla 5.6 contiene los resultados de la RMSSD y la desviación estándar del tobillo para esta prueba.

Tabla 5.6: RMSSD y desviación estándar del tobillo en la tercera prueba de estabilidad.

| **Sistema** | **RMSSD** | **Desviación estándar** |
|---|---|---|
| **MediaPipe** | 0.7189 | 9.0931 |
| **MMPose** | 0.9210 | 3.7297 |

**Cuarta prueba.**

Para la rutina de bisagra de cadera, se aislaron los datos de tres momentos cuasi-estáticos para el cálculo de la métrica, la Tabla 5.7 contiene los resultados obtenidos.

Tabla 5.7: Resultados de métrica RMSSD de la cuarta prueba de estabilidad.

| **Segmento** | **Articulación** | **RMSSD** *MediaPipe* | **RMSSD** *MMPose* |
|---|---|---|---|
| Segundos 8-14 | Cadera | 0.0862 | 0.2365 |
| | Rodilla | 0.0535 | 0.1793 |
| Segundos 21-24 | Cadera | 0.2819 | 0.3104 |
| | Rodilla | 0.0670 | 0.2495 |
| Segundos 30-34 | Cadera | 0.1350 | 0.2307 |
| | Rodilla | 0.1442 | 0.1613 |

Debido a que el ejercicio analizado se realiza de pie en todo momento, el tobillo no ejecuta movimientos significativos por lo que se calculó la RMSSD a todo el conjunto de datos de esta articulación. La Tabla 5.8, muestra los resultados obtenidos de la métrica y la desviación estándar del tobillo en esta prueba.

Tabla 5.8: RMSSD y desviación estándar del tobillo en la cuarta prueba de estabilidad.

| **Sistema** | **RMSSD** | **Desviación estándar** |
|---|---|---|
| **MediaPipe** | 0.4857 | 3.9073 |
| **MMPose** | 0.8904 | 5.2151 |

**Quinta prueba.**

De los resultados de la rutina de sentadillas se aislaron los resultados de tres momentos de poca movilidad, en la Tabla 5.9 se presentan los resultados del cálculo de la métrica RMSSD con esos datos.

Tabla 5.9: Resultados de métrica RMSSD de la quinta prueba de estabilidad.

| **Segmento** | **Articulación** | **RMSSD** *MediaPipe* | **RMSSD** *MMPose* |
|---|---|---|---|
| Segundos 2-7 | Cadera | 0.2448 | 0.2265 |
| | Rodilla | 0.1036 | 0.1332 |
| Segundos 8-10 | Cadera | 0.0599 | 0.2275 |
| | Rodilla | 0.0460 | 0.2114 |
| Segundos 13-15 | Cadera | 0.3566 | 0.7595 |
| | Rodilla | 0.3588 | 0.4814 |

Por la baja movilidad del tobillo en el ejercicio analizado, se aplicó el cálculo de la métrica a todo su conjunto de resultados complementando el análisis de estabilidad del ejercicio con el cálculo de la desviación estándar, Tabla 5.10.

Tabla 5.10: RMSSD y desviación estándar del tobillo en la quinta prueba de estabilidad.

| **Sistema** | **RMSSD** | **Desviación estándar** |
|---|---|---|
| **MediaPipe** | 0.5792 | 4.137 |
| **MMPose** | 0.6629 | 5.4619 |

### 5.1.3. Evaluación de coherencia biomecánica en los resultados

El propósito de esta evaluación fue comprobar si los datos obtenidos de los sistemas de captura se mantienen dentro de los rangos de movimiento anatómicamente posibles. Se analizaron los resultados de captura a través de un programa de python diseñado para detectar y resaltar los resultados que excedan los rangos biomecánicos de cada articulación. En esta evaluación se utilizaron los rangos de movimiento descritos en la sección del **Marco Teórico** en el **Capítulo 2**.

Tabla 5.11: Rangos de movimiento angular para articulaciones de miembro inferior.

| **Movimiento** | **Rango** |
|---|---|
| **Cadera** | $-20\ a\ 140$ |
| **Rodilla** | $-10\ a\ 140$ |
| **Tobillo** | $-50\ a\ 30$ |

**Evaluación con datos de análisis anteriores.**

La Tabla 5.12, contiene los resultados de la evaluación de rangos de movimiento utilizando los resultados de las pruebas de movimiento lento para el sistema basado en *MediaPipe* y la Tabla 5.13 contiene los resultados para el sistema basado en *MMPose*.

Tabla 5.12: Resultados del análisis de rangos de movimiento en resultados de movimiento lento de MediaPipe.

| **Prueba** | **Resultado** | **Porcentaje con base en los datos totales** |
|---|---|---|
| **1** | Rodilla: 25 datos fuera de rango | **1 %** |
| | Cadera: Datos dentro de rango | **-** |
| | Tobillo: Datos dentro de rango | **-** |
| **2** | Todos los datos dentro de rango | **-** |
| **3** | Todos los datos dentro de rango | **-** |
| **4** | Todos los datos dentro de rango | **-** |
| **5** | Todos los datos dentro de rango | **-** |

Tabla 5.13: Resultados del análisis de rangos de movimiento en resultados de movimiento lento de MMPose.

| Prueba | Resultado | Porcentaje con base en los datos totales |
|---|---|---|
| **1** | Rodilla: 1 dato fuera de rango | **0.04 %** |
| | Cadera: Datos dentro de rango | **-** |
| | Tobillo: Datos dentro de rango | **-** |
| **2** | Todos los datos dentro de rango | **-** |
| **3** | Todos los datos dentro de rango | **-** |
| **4** | Todos los datos dentro de rango | **-** |
| **5** | Todos los datos dentro de rango | **-** |

En este conjunto de datos, ambos sistemas de captura en su mayoría mantienen sus resultados dentro de los rangos de movimiento angular, y aunque *MediaPipe* es el sistema con más datos que exceden el limite, un análisis más detallado sobre los valores de la rodilla que están fuera de rango revela que los datos se mantuvieron por debajo de los **145** grados.

### 5.1.4. Selección del sistema de captura de ángulos de movimiento

Con un análisis de los resultados de cada uno de los sistemas de captura, es innegable la superioridad del sistema de captura basado en *MediaPipe*, mantiene una mejor estabilidad en las cinco pruebas realizadas, tal como lo resalta la evaluación con **RMSSD**, y aunque este sistema es el que ha entregado más valores fuera del rango anatómicamente correcto también es el que mejores resultados mantiene en la articulación del tobillo, que como se puede notar en los resultados presentados es una de las articulaciones más complicadas de analizar sin la utilización de marcadores. Esta combinación de estabilidad general y buen rendimiento en articulaciones complejas refuerza su eficiencia en el cumplimiento de requerimientos de este proyecto.

Fuera de los resultados obtenidos a través de pruebas, la comodidad que ofrece *MediaPipe* con su instalación y manejo, dejan por debajo a la ardua tarea de la búsqueda de un modelo ideal para su uso en *MMPose*, además, un punto que se tomó cuenta para la elección de un sistema sobre otro han sido los errores de detección, si bien *MMPose* es catalogado muchas veces como un sistema de detección de articulaciones con alta precisión, esta característica depende mucho del recurso visual que se analiza, aún cuando los videos utilizados eran de una alta calidad y el sujeto de prueba destacaba en el entorno, el sistema basado en *MMPose* tiende a errar la detección de las articulaciones de interés de tal forma que era notorio a simple vista, mientras que el sistema *MediaPipe* no sólo mejoraba sus resultados analizar videos con buena resolución si no que es capaz de mantener buen seguimiento y detección de articulaciones con frames de baja resolución. Por las características demostradas en las pruebas realizadas así como durante su desarrollo, el sistema basado en la detección de articulaciones con *MediaPipe* es el que mejor cumple con los requerimientos de este proyecto, un sistema de fácil modificación y manipulación, capaz de entregar el movimiento angular de las tres articulaciones de miembro inferior.

## 5.2. Réplica de movimiento

Con el objetivo de validar el funcionamiento y evaluar el desempeño de los sistemas de réplica de movimiento, se llevaron a cabo una serie de pruebas específicas centradas en el análisis del proceso de réplica. Conforme a los resultados de la sección anterior, en estas pruebas se emplearon únicamente los datos de análisis de movimiento del sistema basado en *MediaPipe*.

Respecto al sistema de réplica virtual, ya que cuenta con los modos de envío de información simultánea y posterior al análisis de movimiento, las pruebas realizadas permitieron identificar el método de transmisión de datos más eficiente para el proyecto. En lo referente al sistema físico, se evaluó la precisión con la que los motores ejecutaron los movimientos capturados, extrayendo la información de los encoders sin que los motores estuvieran ensamblados en el modelo.

### 5.2.1. Réplica virtual

Para discernir el método de envío de datos angulares más óptimo para el proyecto, se utilizaron los resultados de tres análisis de movimiento con *MediaPipe* en la reproducción de movimiento utilizando el modelo virtual. El análisis de resultados se basa en la comparación gráfica de las pociones angulares capturadas y las alcanzadas por los actuadores virtuales, teniendo en cuenta la precisión de las posiciones alcanzadas por el modelo virtual y el tiempo de ejecución de cada proceso de réplica en comparación con el tiempo real de la rutina.

**Primer proceso de réplica.**

El video analizado para esta prueba contiene la realización del ejercicio patada invertida [43], Figura 5.21, la duración de esta rutina de movimiento en el video es de únicamente siete segundos.

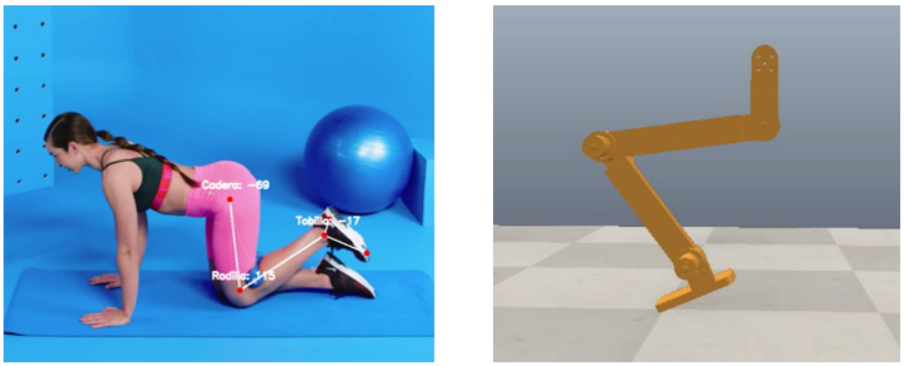



Figura 5.21: Análisis y réplica virtual de patada invertida.

Los resultados de las posiciones angulares calculadas durante el análisis de movimiento, y las alcanzadas por los actuadores virtuales utilizando el envío de datos simultáneo, se muestran en las figuras 5.22, 5.23 y 5.24.

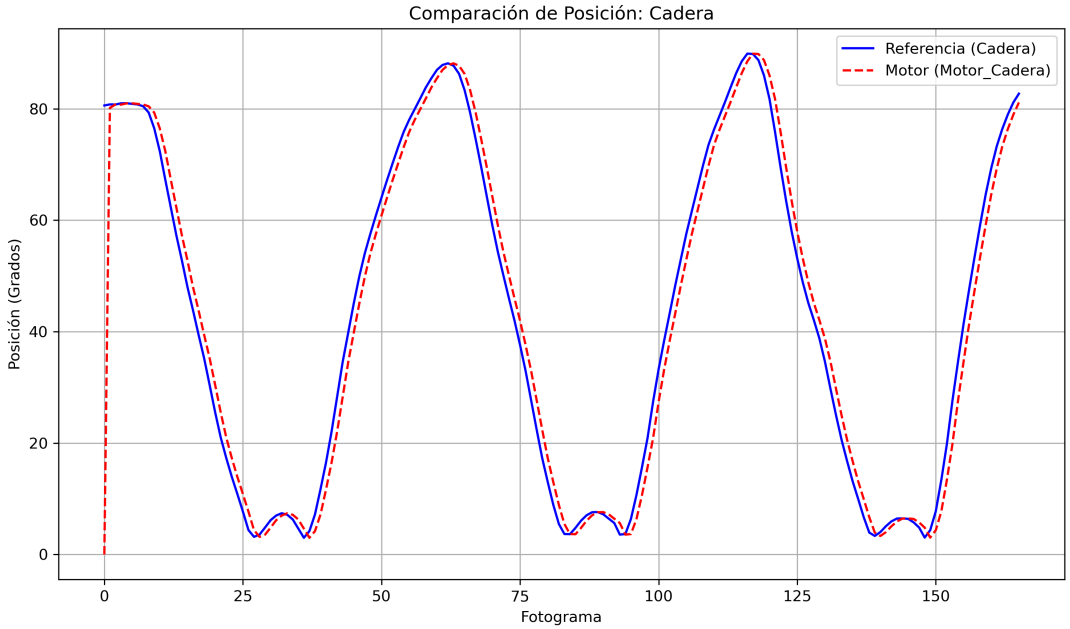



Figura 5.22: 1er Comparación de posiciones angulares de cadera con envío simultáneo.

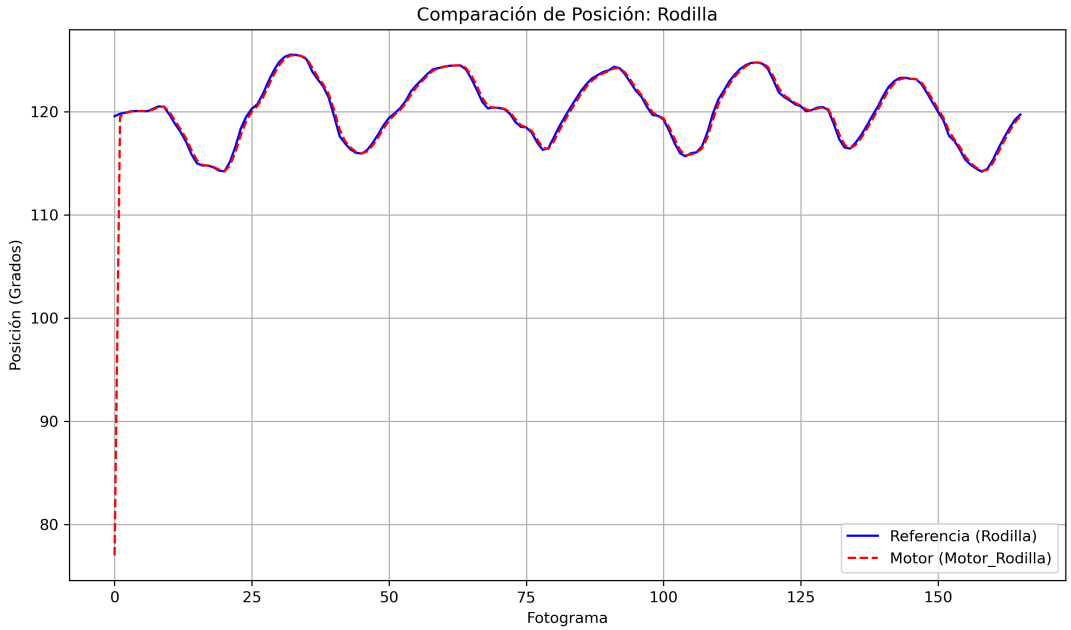



Figura 5.23: 1er Comparación de posiciones angulares de rodilla con envío simultáneo.

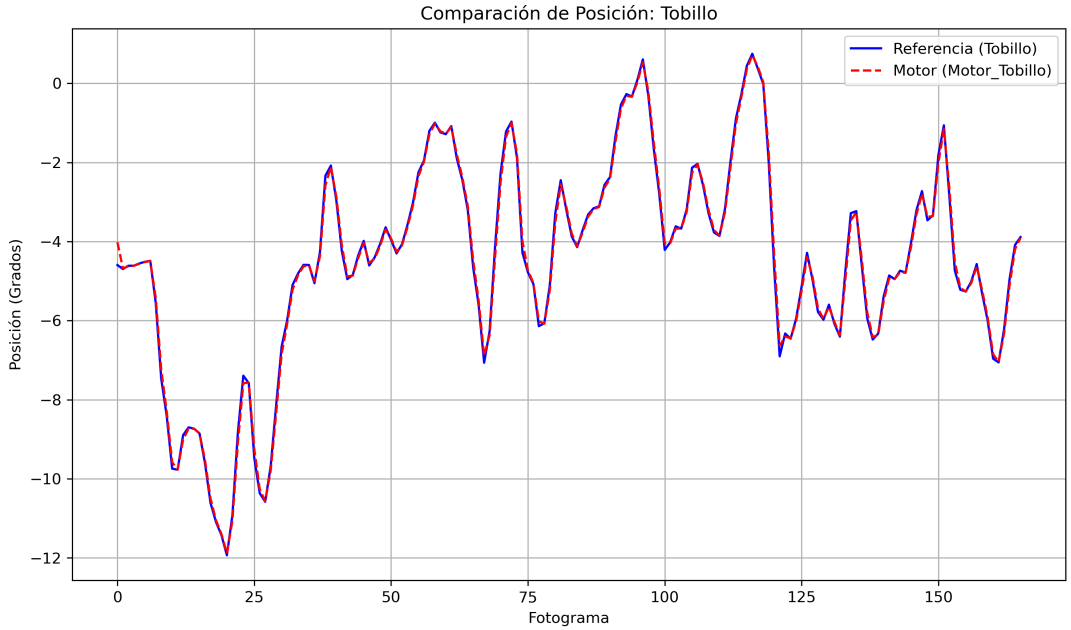



Figura 5.24: 1er Comparación de posiciones angulares de tobillo con envío simultáneo.

En las gráficas se aprecia un posicionamiento previo de los motores virtuales, el cual genera un desplazamiento en los valores de la gráfica, sin afectar de forma directa las valores angulares de los actuadores. Comparando todas las posiciones angulares capturadas con las alcanzadas

por los actuadores, sin tomar en cuenta el posicionamiento inicial, se obtuvieron los errores de posición para cada articulación, Tabla 5.14, estos resultados demuestran que hay una alta precisión en la réplica de movimiento, con un error máximo del **1.89 %**.

Tabla 5.14: Errores de posicionamiento de actuadores virtuales (Prueba 1, envío de datos simultáneo al análisis de movimiento ).

| **Articulación** | **Error** |
|---|---|
| **Cadera** | 0.52 % |
| **Rodilla** | 1.89 % |
| **Tobillo** | 0.66 % |

La medición del tiempo de ejecución del movimiento en el modelo virtual, realizada con el comando %%time desde el programa en Jupyter Notebook, arrojo un tiempo de **43.7 segundos**, una duración prolongada que resulta en un movimiento pausado y lento, poco realista respecto a la rutina del video analizado.

Utilizando el segundo método de envío de datos, mediante el archivo de resultados post-análisis de movimiento, se obtuvieron las gráficas de las figuras 5.25, 5.26 y 5.27.

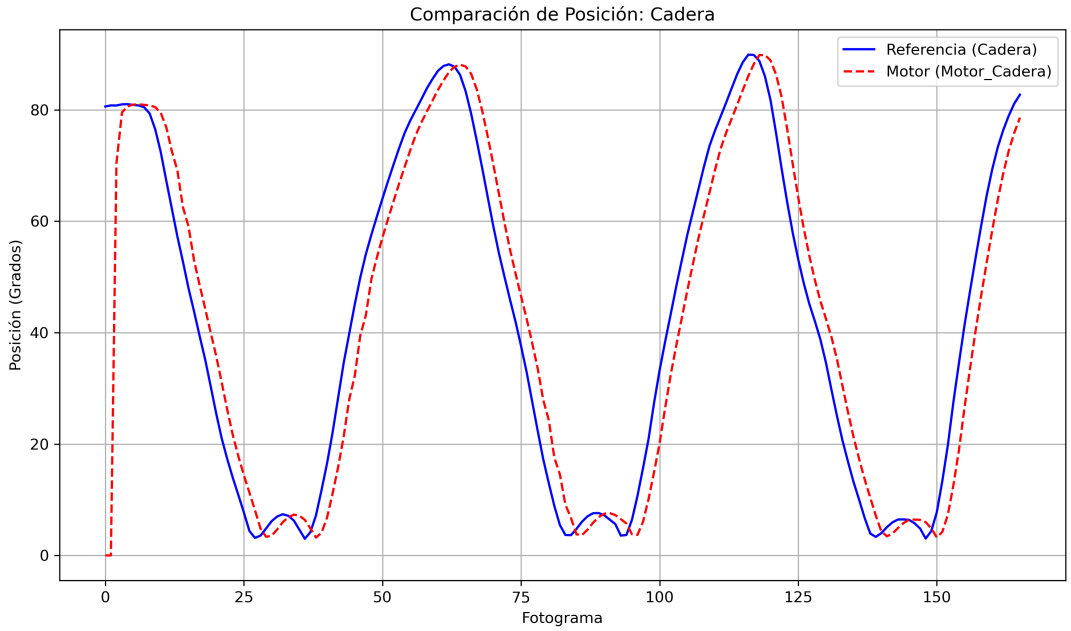



Figura 5.25: 1er Comparación de posiciones angulares de cadera con Envío post-análisis.

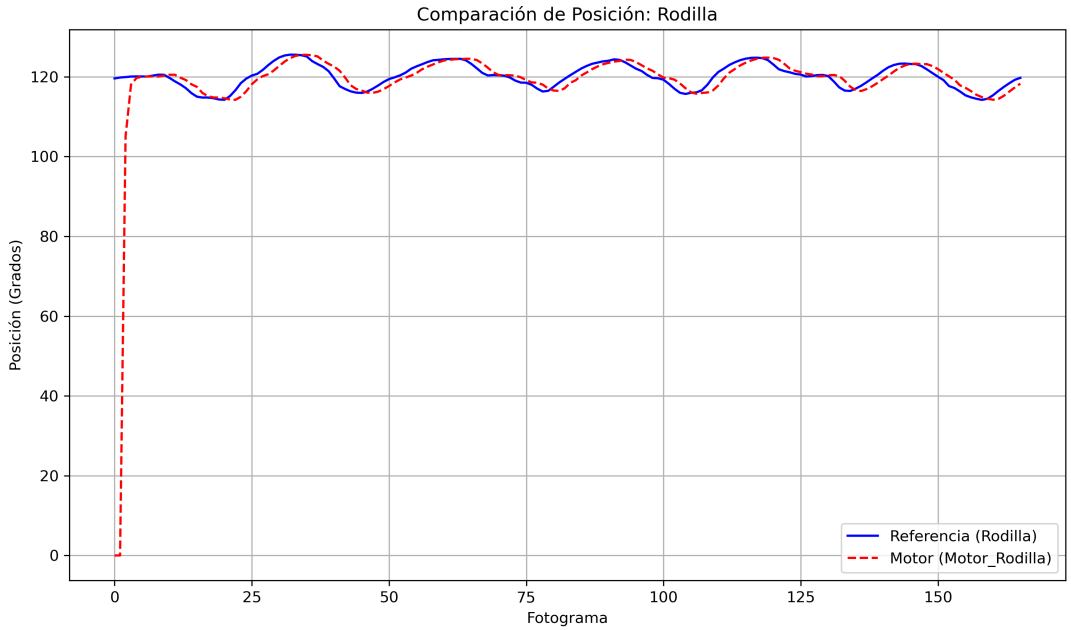



Figura 5.26: 1er Comparación de posiciones angulares de rodilla con envío post-análisis.

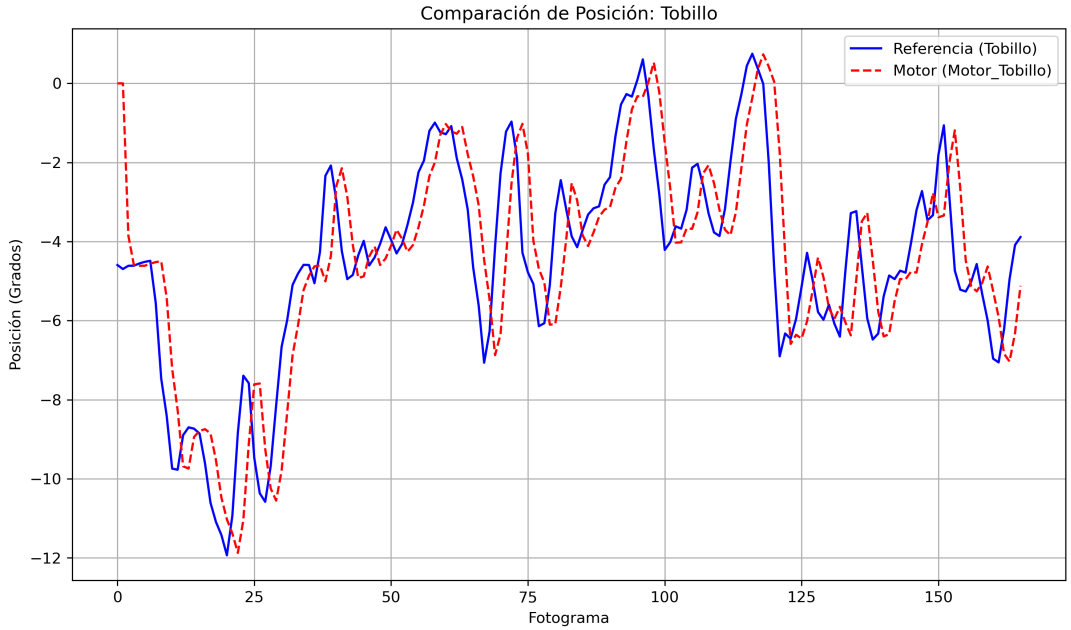



Figura 5.27: 1er Comparación de posiciones angulares de tobillo con envío post-análisis.

Con este método, en todas las gráficas se puede ver el posicionamiento inicial de los motores virtuales, lo que nuevamente provoca un desplazamiento en los valores de las gráficas. Al aplicar la comparación dato por dato, se obtuvieron los resultados de la Tabla 5.15, donde se aprecia

una disminución en el error del actuador de la rodilla así como un aumento en el error del tobillo.

Tabla 5.15: Errores de posicionamiento de actuadores virtuales (Prueba 1, Envío de datos post-análisis de movimiento).

| **Articulación** | **Error** |
|---|---|
| **Cadera** | 0.51 % |
| **Rodilla** | 0.83 % |
| **Tobillo** | 0.79 % |

Una característica mejorada al utilizar este método de transmisión de datos, fue el tiempo de ejecución de la rutina de movimiento el cual fue de solamente **7.8 segundos**, valor cercano al tiempo real de siete segundos, lo que a su vez resultó en una réplica de movimiento fluida sin pausas.

**Segundo proceso de réplica.**

El análisis de movimiento de esta prueba utilizó un video en donde se realizan escaladas de montaña [43], Figura 5.50, los movimientos analizados son fluidos pero con momentos cuasi-estáticos, lo cual fue ideal para la validación del proceso de réplica en momentos de mínima motricidad.

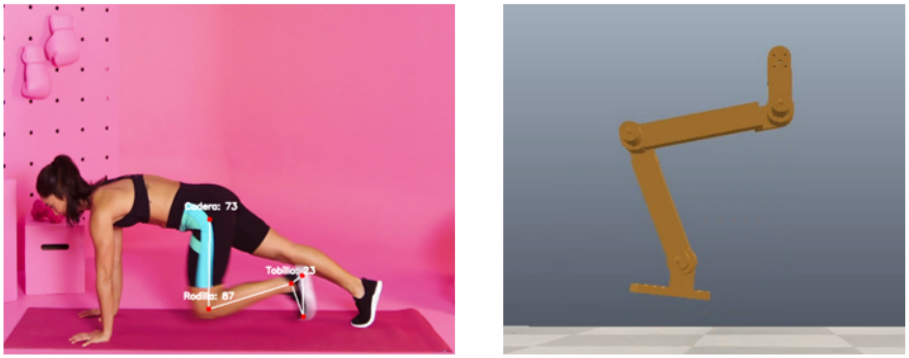



Figura 5.28: Análisis y réplica virtual de rutina de escalada.

Las gráficas de comparación resultantes utilizando el envío de datos simultáneo se muestran en las figuras 5.25, 5.26 y 5.27.

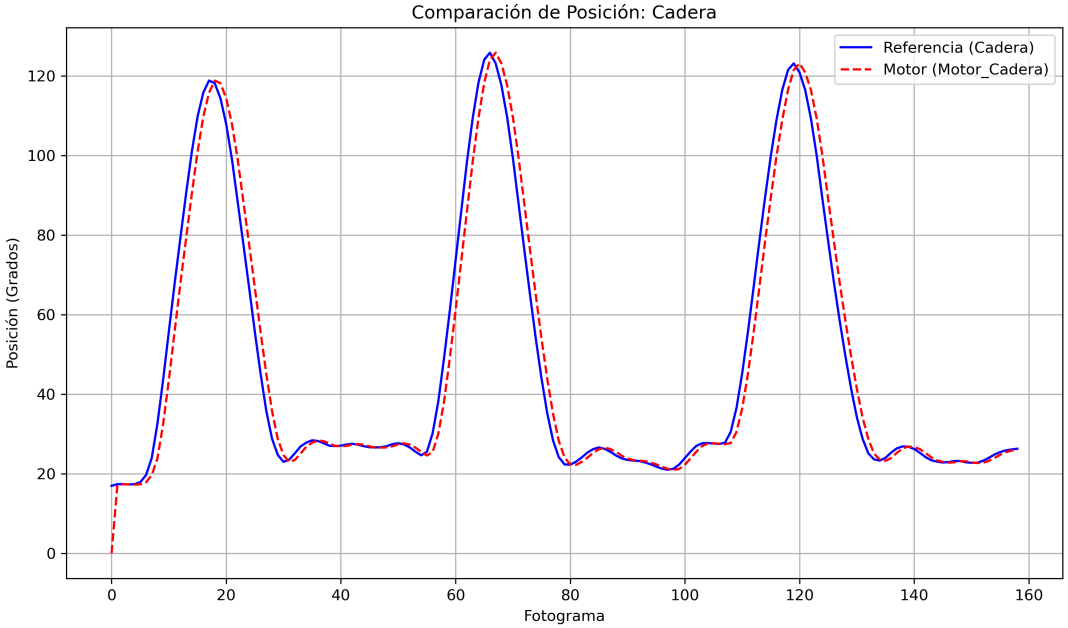



Figura 5.29: 2da Comparación de posiciones angulares de cadera con envío simultáneo.

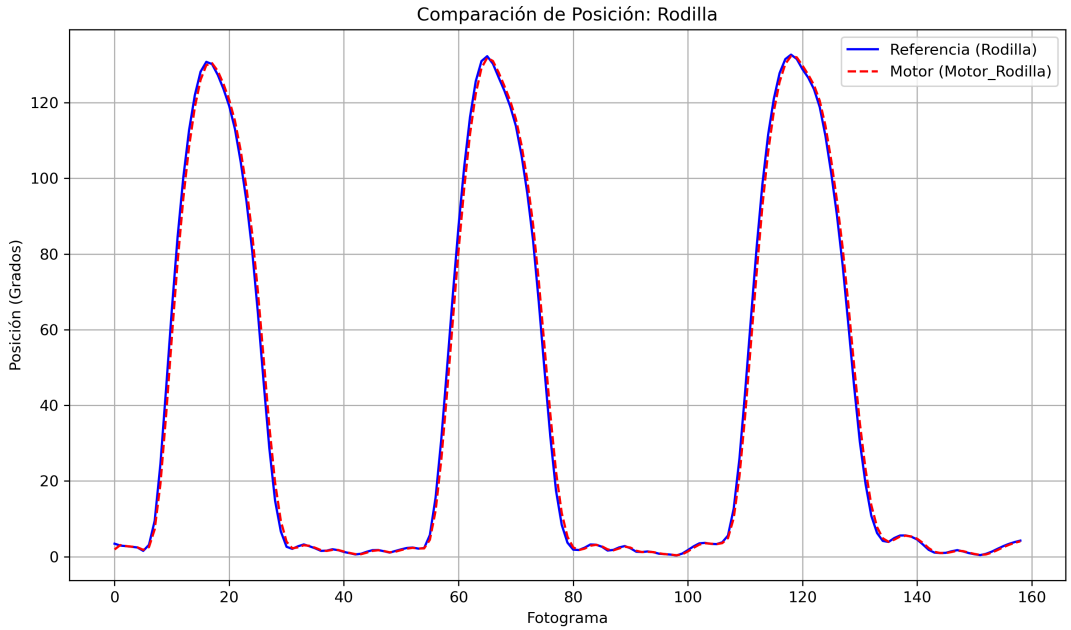



Figura 5.30: 2da Comparación de posiciones angulares de rodilla con envío simultáneo.

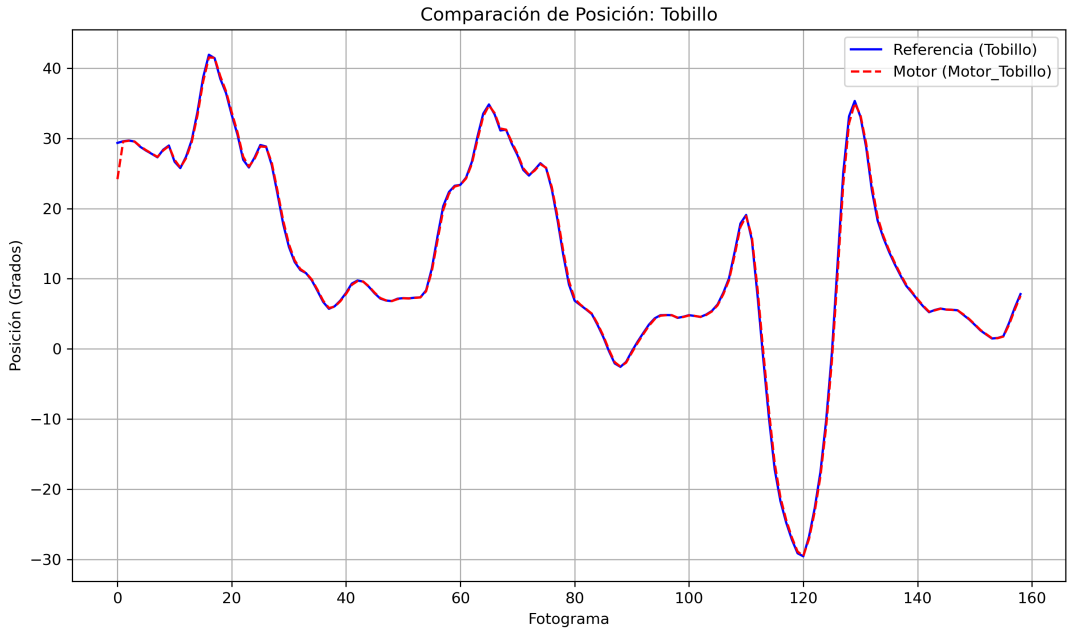



Figura 5.31: 2da Comparación de posiciones angulares de tobillo con envío simultáneo.

En esta ocasión, la rutina de movimiento comenzó cerca de un estado de reposo, por lo que el posicionamiento de los actuadores no produjo un desplazamiento significativo en los valores de las gráficas. En la Tabla 5.16, se muestra la precisión de cada motor virtual respecto a los

datos capturados, estos resultados denotan una alta precisión en la réplica de esta rutina de movimiento pausada.

Tabla 5.16: Errores de posicionamiento de actuadores virtuales (Prueba 2, Envío de datos simultáneo al análisis de movimiento ).

| Articulación | Error |
|---|---|
| **Cadera** | 0.58 % |
| **Rodilla** | 1.54 % |
| **Tobillo** | 0.64 % |

El tiempo de ejecución de la réplica de movimiento en esta prueba fue de **47.1 segundos**, una duración extendida en comparación con el tiempo real de la rutina en el video de sólo ocho segundos, lo que en consecuencia produjo una simulación de movimiento lenta nuevamente.

Con la transmisión de datos post-análisis, los resultados son los mostrados en las gráficas de las figuras 5.32, 5.33 y 5.34.

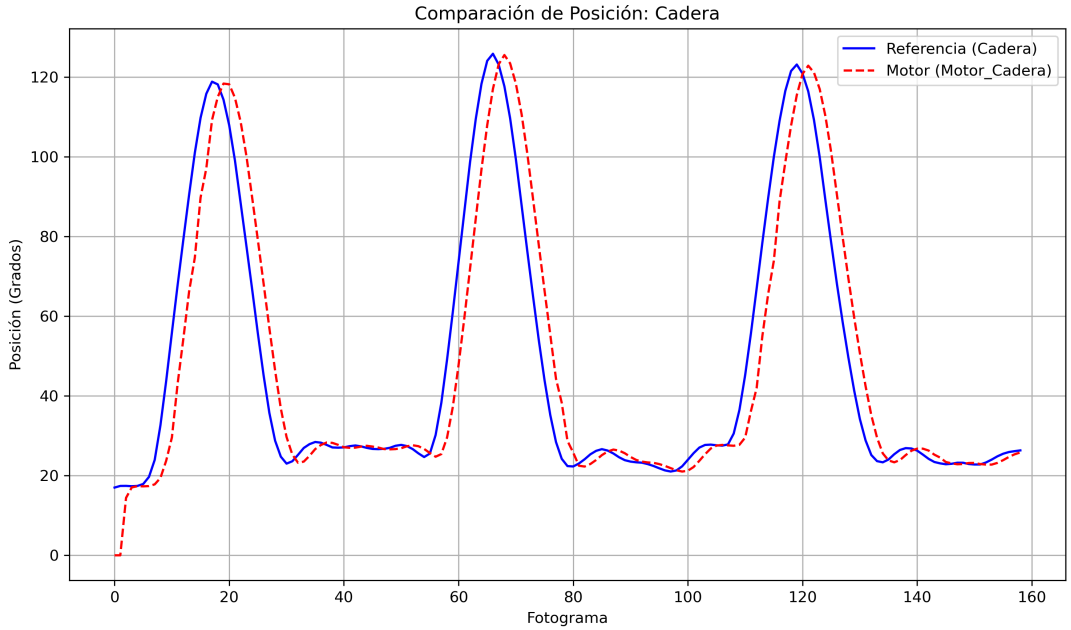



Figura 5.32: 2da Comparación de posiciones angulares de cadera con envío post-análisis.

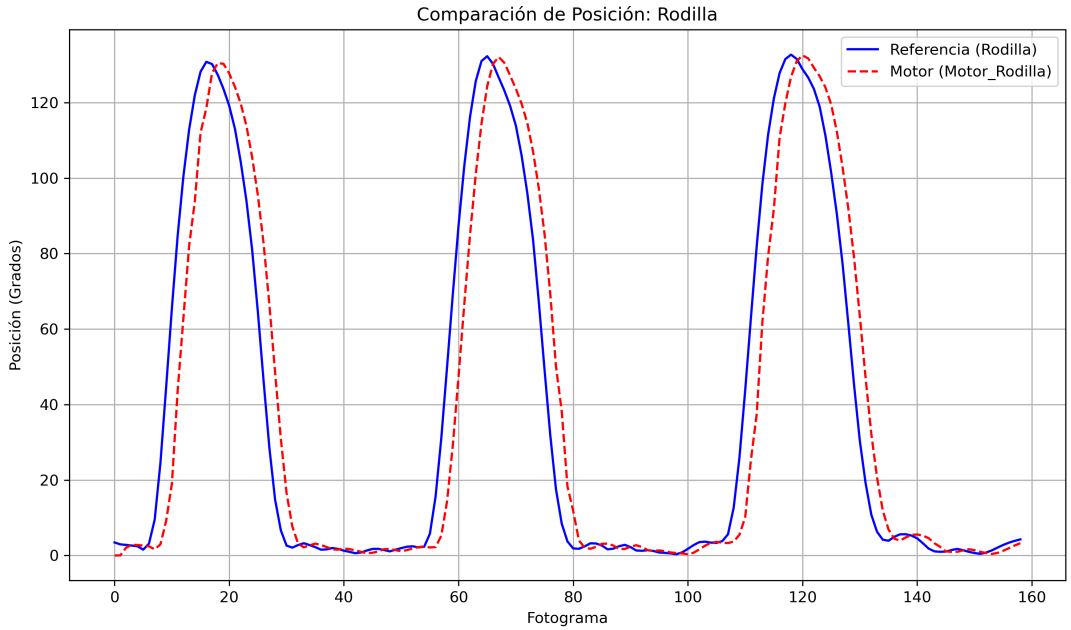



Figura 5.33: 2da Comparación de posiciones angulares de rodilla con envío post-análisis.

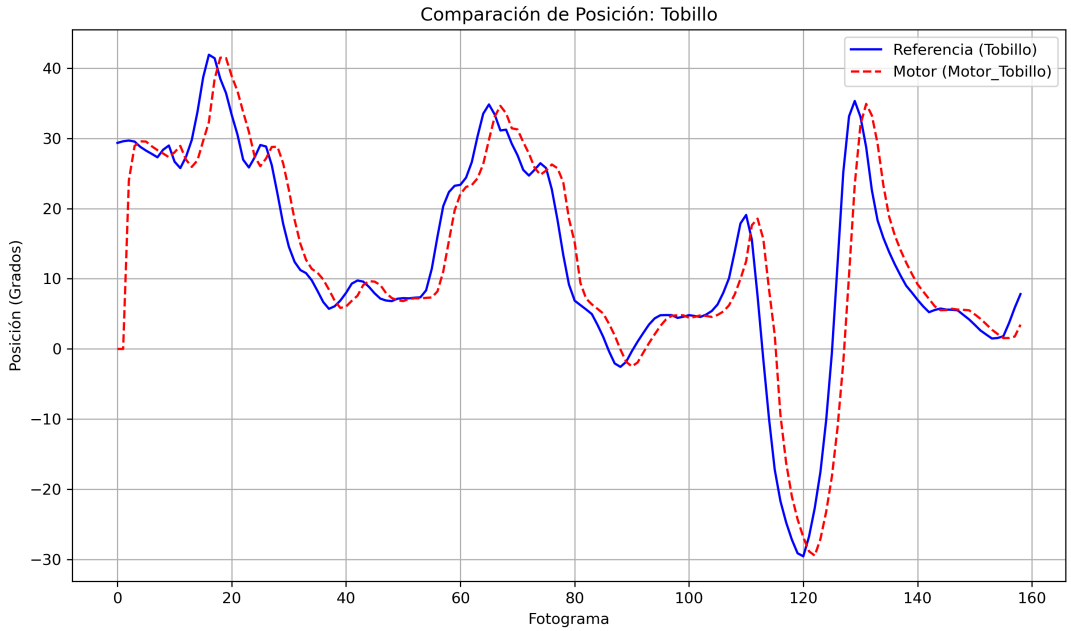



Figura 5.34: 2da Comparación de posiciones angulares de tobillo con envío post-análisis.

Al realizar la comparación dato por dato se obtuvieron los errores mostrados en la Tabla 5.17, los cuales demuestran la alta precisión alcanzada en el proceso de réplica del movimiento analizado.

Tabla 5.17: Errores de posicionamiento de actuadores virtuales (Prueba 2, Envío de datos post-análisis de movimiento ).

| **Articulación** | **Error** |
|---|---|
| **Cadera** | 0.56 % |
| **Rodilla** | 0.61 % |
| **Tobillo** | 0.47 % |

Con una reducción en el tiempo de ejecución de la rutina de movimiento a solamente **8.9 segundos** y una simulación fluida, el método de transmisión post-analisis supera nuevamente a la transmisión simultánea.

**Tercer proceso de réplica.**

La realización de este proceso de réplica utilizó los datos del análisis del ejercicio *Dead Bug* [46], Figura 5.35, debido a que se trata de una rutina de movimiento extendida de cuarenta y siete segundos de duración.

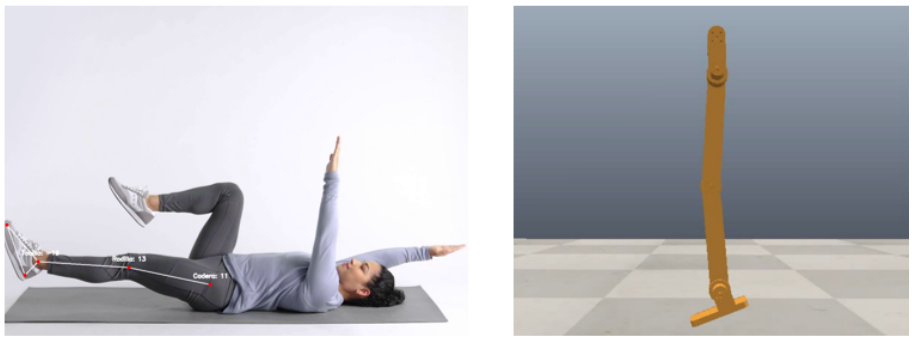


Figura 5.35: Análisis y réplica virtual de ejercicio dead bug.

Los resultados al utilizar el envío de datos simultáneamente con el análisis, son los mostrados en las figuras 5.36, 5.37 y 5.38.

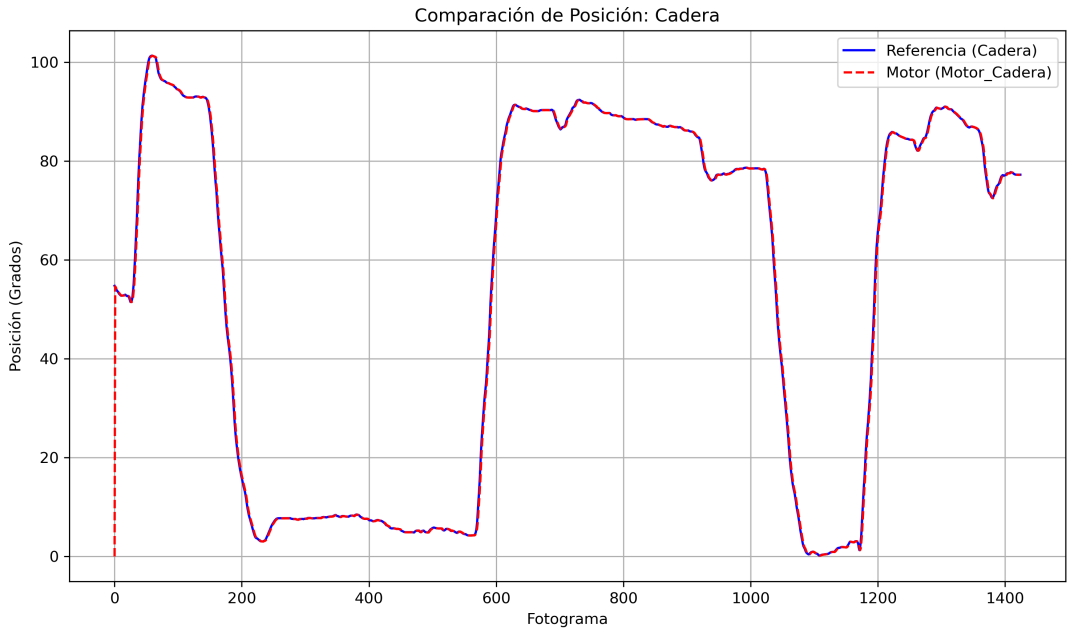



Figura 5.36: 3er Comparación de posiciones angulares de cadera con envío simultáneo.

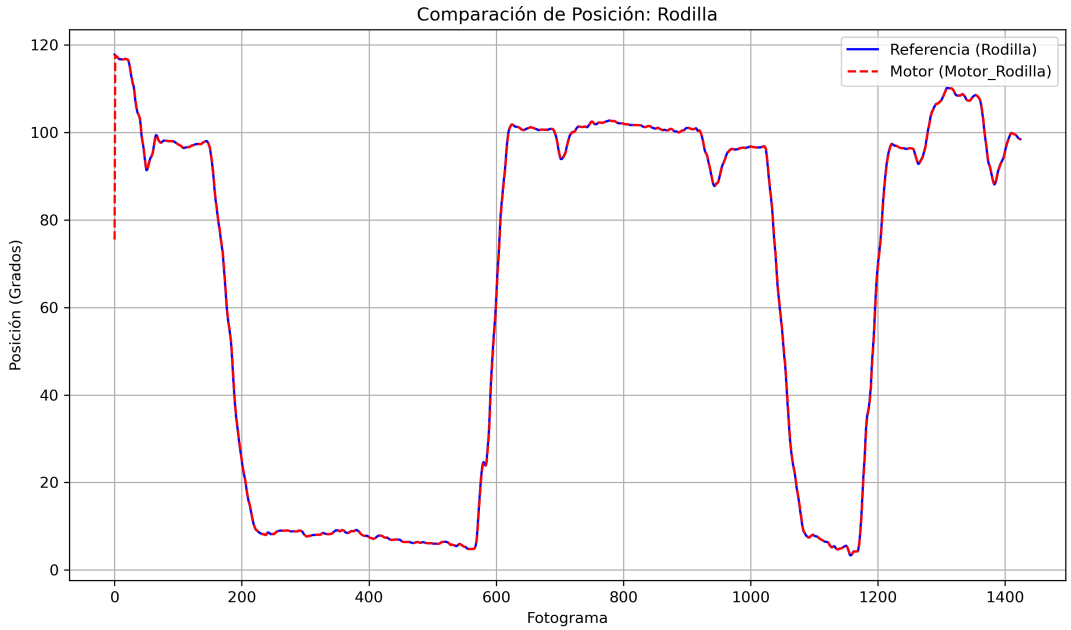



Figura 5.37: 3er Comparación de posiciones angulares de rodilla con envío simultáneo.

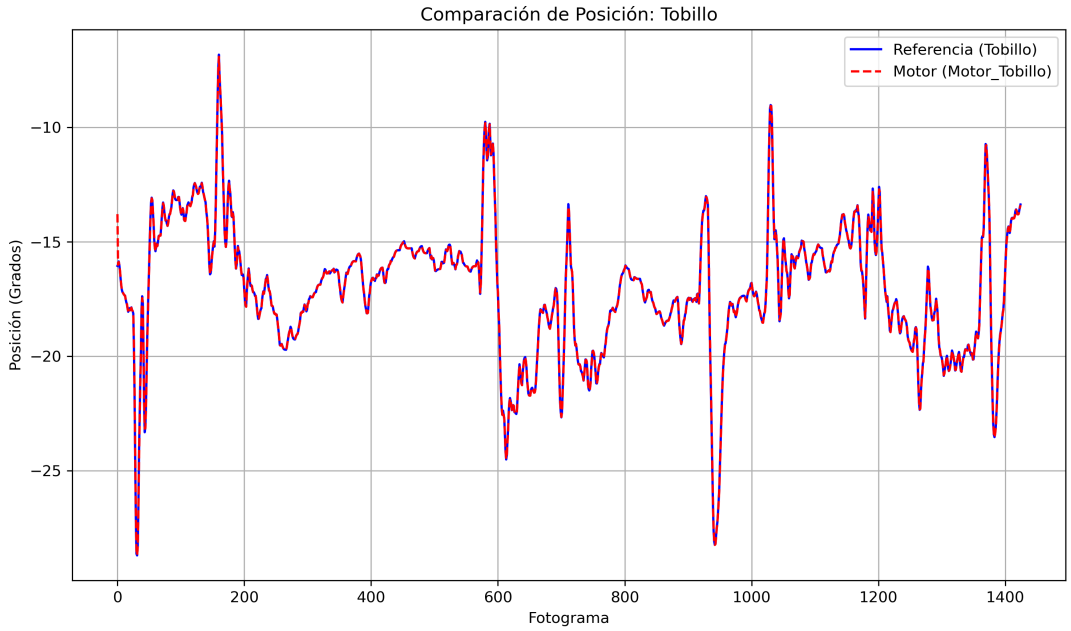



Figura 5.38: 3er Comparación de posiciones angulares de tobillo con envío simultáneo.

Los resultados de las gráficas como en las anteriores pruebas, muestran una gran precisión en el proceso de réplica de movimiento aún incluso considerando el posicionamiento inicial de los motores. Los errores calculados para esta prueba, en la Tabla 5.18, comprueban la exactitud del proceso de réplica de esta rutina.

Tabla 5.18: Errores de posicionamiento de actuadores virtuales (Prueba 3, Envío de datos simultáneo al análisis de movimiento).

| Articulación | Error |
|---|---|
| **Cadera** | 0.03 % |
| **Rodilla** | 0.22 % |
| **Tobillo** | 1.07 % |

Sin embargo la precisión alcanzada se atribuye al extenso tiempo de ejecución de la rutina de movimiento de **7.9 minutos**, al ser la réplica pausada y lenta permitía que los actuadores se posicionaran correctamente en todo momento.

Con el envío de datos posterior al análisis, los resultados se muestran en las figuras 5.39, 5.40 y 5.41.

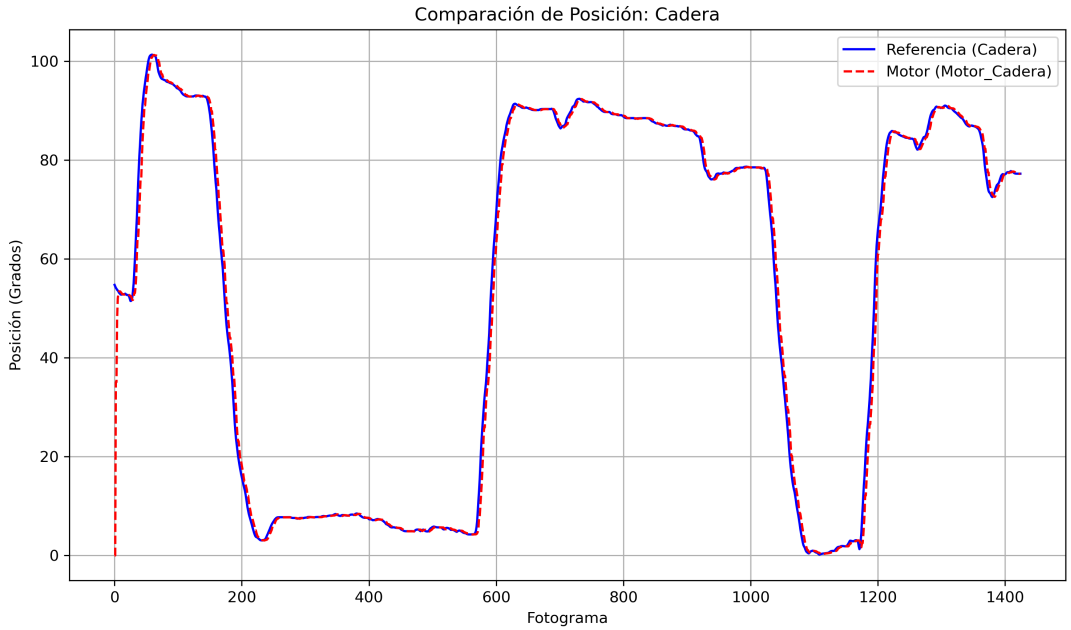



Figura 5.39: 3er Comparación de posiciones angulares de cadera con envío post-análisis.

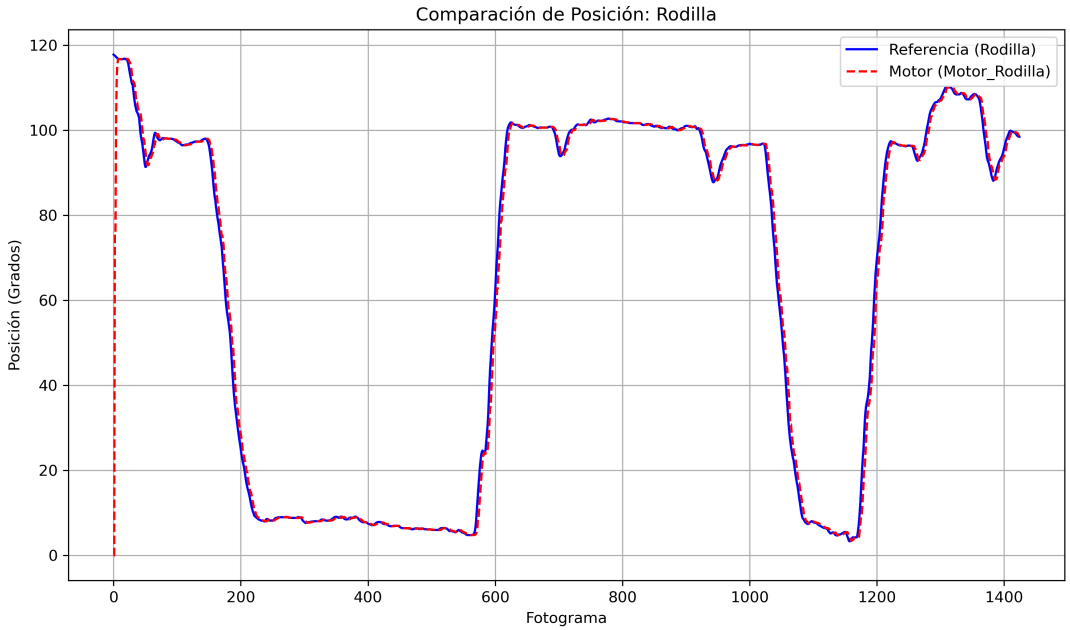



Figura 5.40: 3er Comparación de posiciones angulares de rodilla con envío post-análisis.

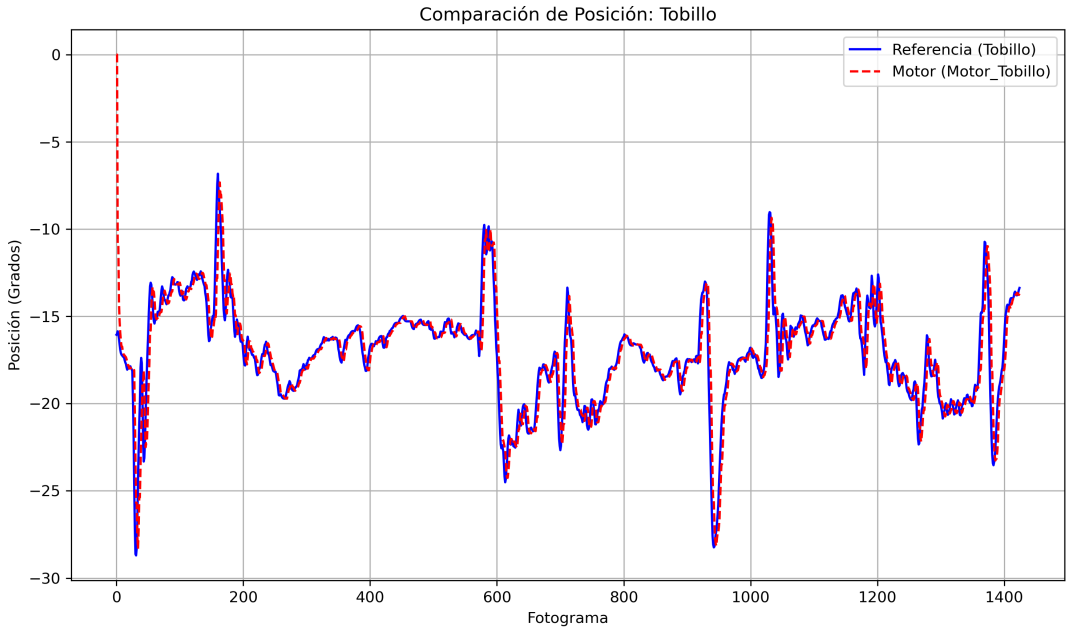



Figura 5.41: 3er Comparación de posiciones angulares de tobillo con envío post-análisis.

Los resultados de los errores de posición calculados, Tabla 5.19, denotan una menor precisión pero disminuyendo considerablemente el tiempo de ejecución de los movimientos a **49 segundos**.

Tabla 5.19: Errores de posicionamiento de actuadores virtuales (Prueba 3, Envío de datos post-análisis de movimiento)

| **Articulación** | **Error** |
|---|---|
| **Cadera** | 0.55 % |
| **Rodilla** | 0.51 % |
| **Tobillo** | 1.46 % |

Aún con la realización de más pruebas, el principal problema detectado de la transmisión de datos simultánea al análisis, es el tiempo de ejecución de la rutina de movimiento, pues como se ha podido observar este tiempo llega a ser del triple o más. Este retraso significativo en cada uno de los procesos de réplica se debe a la suma de procesos que ocurren en el análisis, pues para que una sola posición sea enviada al modelo virtual primero debe ser detectada, pasar por el filtro Kalman, ser calculada utilizando la ley de cosenos y después pasar por el filtro Gaussiano.

La precisión en ambos medios de transmisión es excelente, aún cuando el error es considerado alto este no supera el **2 %**, por esa parte cualquiera de los métodos de envío de posiciones es útil, pero para los propósitos de este proyecto no se puede ignorar el tiempo de ejecución, por lo que la transmisión de datos post-análisis se considera la mejor opción para hacer uso del sistema de réplica de movimiento virtual.

### 5.2.2. Réplica de movimiento físico

Con el objetivo de verificar la precisión del sistema de réplica de movimiento físico, se llevaron a cabo tres procesos de reproducción de posiciones, durante los cuales se registraron las posiciones alcanzadas por los motores sin que estuvieran colocados en el modelo impreso, para su comparación con las posiciones obtenidas durante la captura.

Al igual que con la verificación del sistema de réplica virtual, cada una de las pruebas de réplica física utilizó únicamente los datos de análisis del sistema de captura basado en *MediaPipe*.

**Primer proceso de réplica de movimiento físico.**

Los datos de movimiento utilizados en este proceso son del ejercicio de escalada rápida, Figura 5.42, una rutina de movimiento dinámica, en la que las tres articulaciones de interés están activas en todo momento.

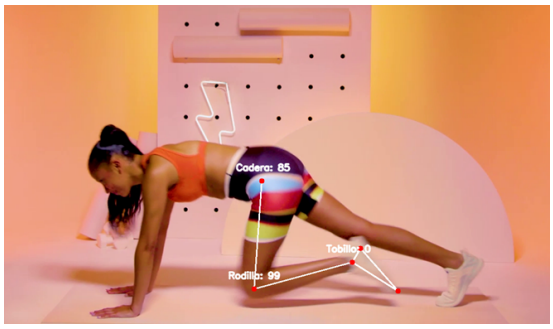



Figura 5.42: Primer proceso de réplica de movimiento físico [43].

La gráfica de posiciones alcanzadas por los motores con los datos de la cadera en la Figura 5.43, a simple vista muestra una alta coincidencia en los ángulos de movimiento ejecutados, pero con un ligero desfase entre las posiciones, fenómeno atribuido nuevamente al posicionamiento inicial de los motores, como lo visto en el proceso de réplica virtual. Un comportamiento similar se observa con los las posiciones alcanzadas por los motores de la rodilla, Figura 5.44, y del tobillo, Figura 5.45, los cuales siguen de cerca el movimiento angular real que se ha capturado

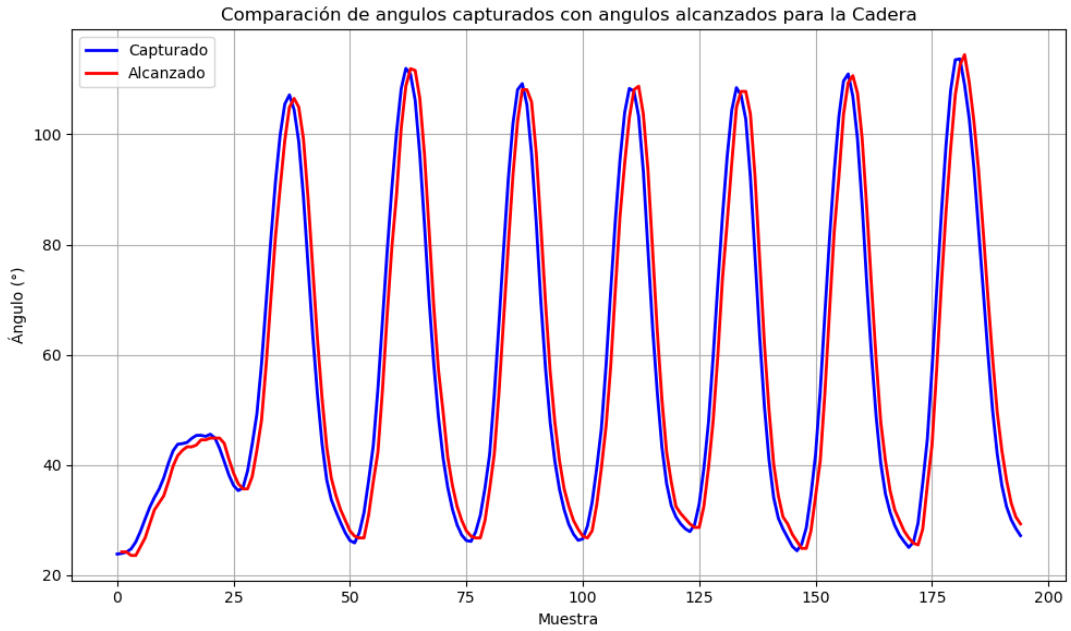



Figura 5.43: 1er Gráfica de comparación de posiciones angulares de la cadera con sistema de réplica físico.

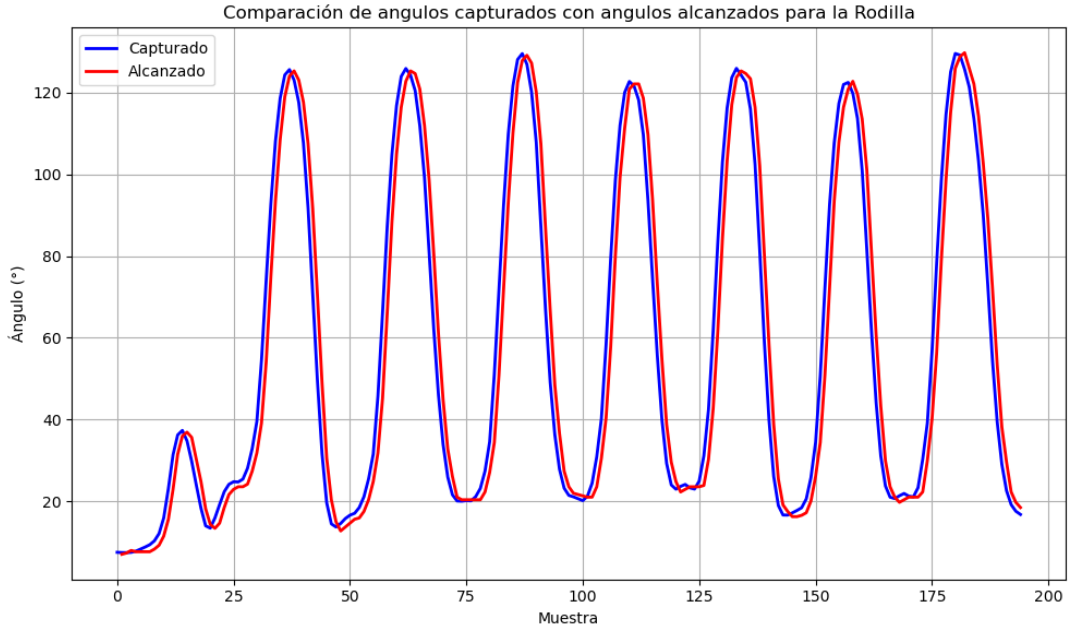



Figura 5.44: 1er Gráfica de comparación de posiciones angulares de la rodilla con sistema de réplica físico.

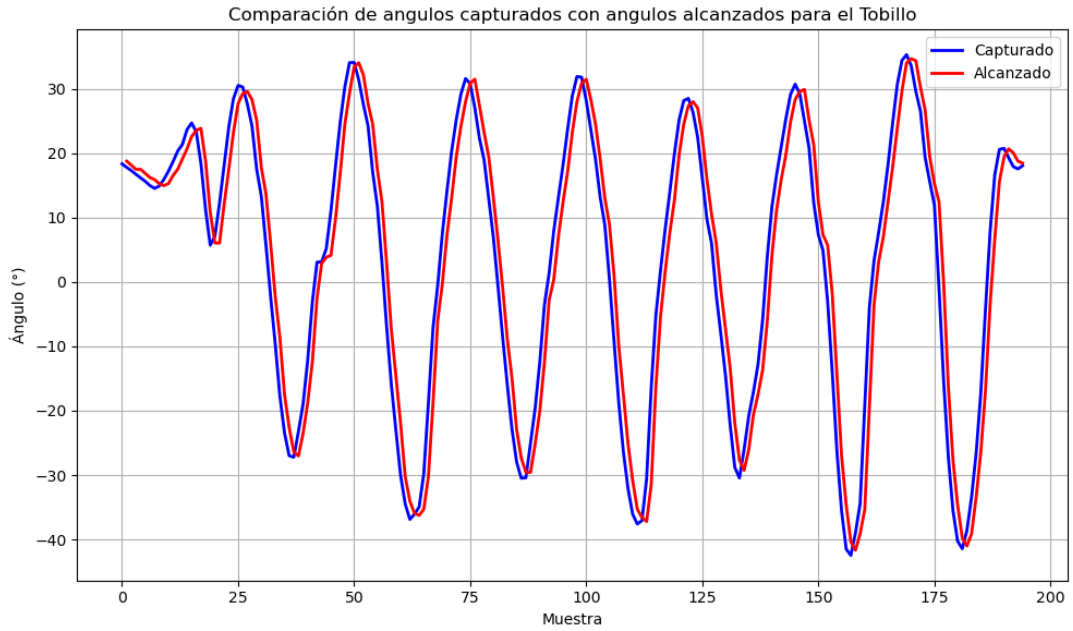



Figura 5.45: 1er Gráfica de comparación de posiciones angulares del tobillo con sistema de réplica físico.

Se realizó un análisis más detallado comparando dato por dato las posiciones alcanzadas por los motores con las capturadas, calculando el error absoluto promedio junto con el error porcentual promedio, Tabla 5.20. Se omitió el posicionamiento inicial de los motores, para evitar el desfase entre los datos y centrar la atención únicamente en el movimiento angular.

Tabla 5.20: Errores absolutos promedio de posicionamiento (Prueba 1, sistema de réplica físico)

| **Articulación** | **Error absoluto promedio en grados** | **Error porcentual promedio** |
|---|---|---|
| **Cadera** | 0.63° | 1.49 % |
| **Rodilla** | 0.58° | 1.88 % |
| **Tobillo** | 0.59° | 1.44 % |

**Segundo proceso de réplica de movimiento físico.**

En este proceso de réplica se hizo uso de los datos de análisis de una rutina de estocada inversa, Figura 5.46, un ejercicio de movimiento constante.


Figura 5.46: Segundo proceso de réplica de movimiento físico [43].

La comparación de posiciones se muestra en las figuras 5.47, 5.48 y 5.49, donde se observa que, en cada caso los motores presentan un ligero error de posición respecto al movimiento capturado, más notorio en el motor de la cadera, en los puntos de baja variabilidad de movimiento.

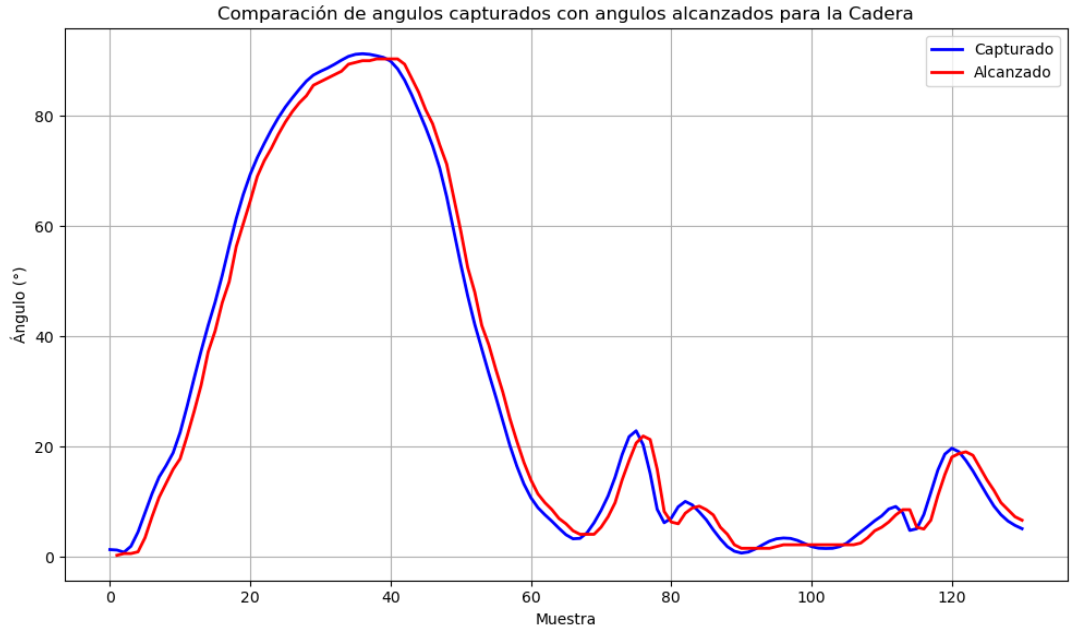



Figura 5.47: 2da Gráfica de comparación de posiciones angulares de la cadera con sistema de réplica físico.

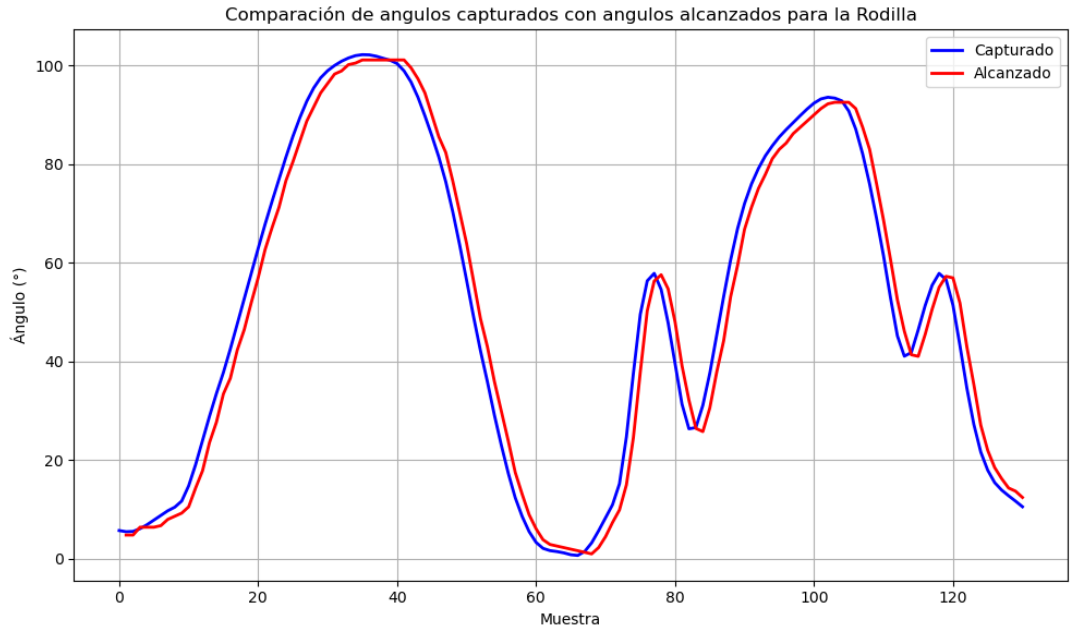



Figura 5.48: 2da Gráfica de comparación de posiciones angulares de la rodilla con sistema de réplica físico.

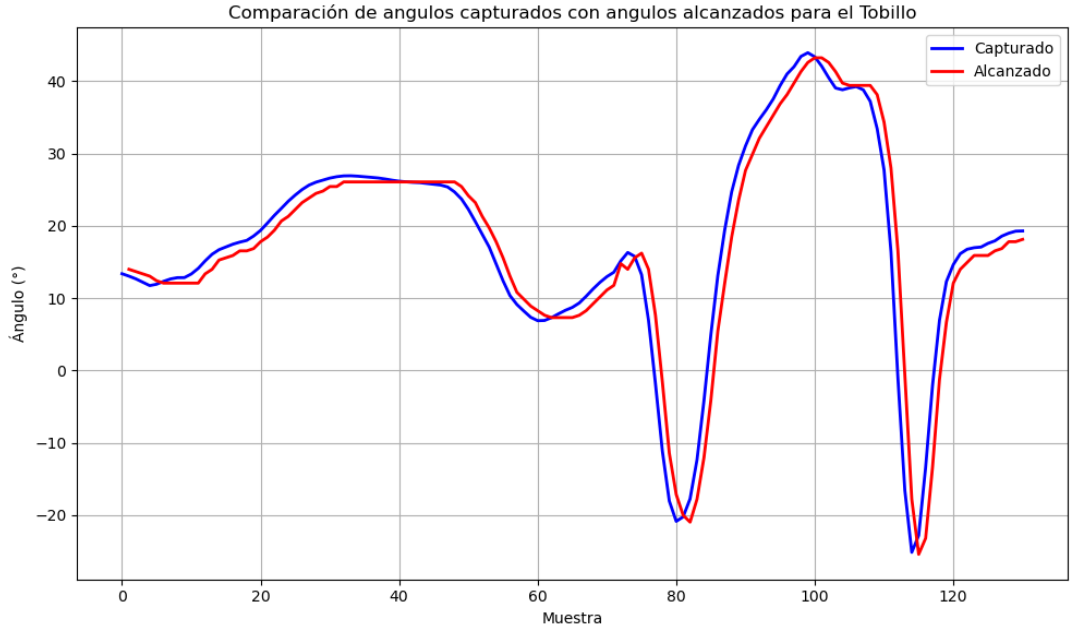



Figura 5.49: 2da Gráfica de comparación de posiciones angulares del tobillo con sistema de réplica físico.

Aunque el promedio del error calculado para las posiciones de las articulaciones esta por debajo del 1 %, Tabla 5.21, se observa un aumento en el error porcentual promedio con respecto a la primer prueba de réplica de movimiento.

Tabla 5.21: Errores absolutos promedio de posicionamiento (Prueba 2, sistema de réplica físico)

| **Articulación** | **Error absoluto promedio en grados** | **Error porcentual promedio** |
|---|---|---|
| **Cadera** | 0.70° | 7.12 % |
| **Rodilla** | 0.65° | 4.63 % |
| **Tobillo** | 0.54° | 4.99 % |

**Tercer proceso de réplica de movimiento físico.**

Para la tercera prueba de réplica de movimiento físico se utilizaron los resultados del análisis de un ejercicio flexor de cadera, Figura 5.50, la cual es una rutina de movimiento pausado, con el fin de verificar más en profundidad el comportamiento del sistema con rutinas de este tipo.

Figura 5.50: Tercer proceso de réplica de movimiento físico [46].

En las gráficas de comparación de resultados, figuras 5.51, 5.52 y 5.53, se observa una gran diferencia entre ambos conjuntos de datos, lo cual es más notorio en los momentos de poca movilidad.

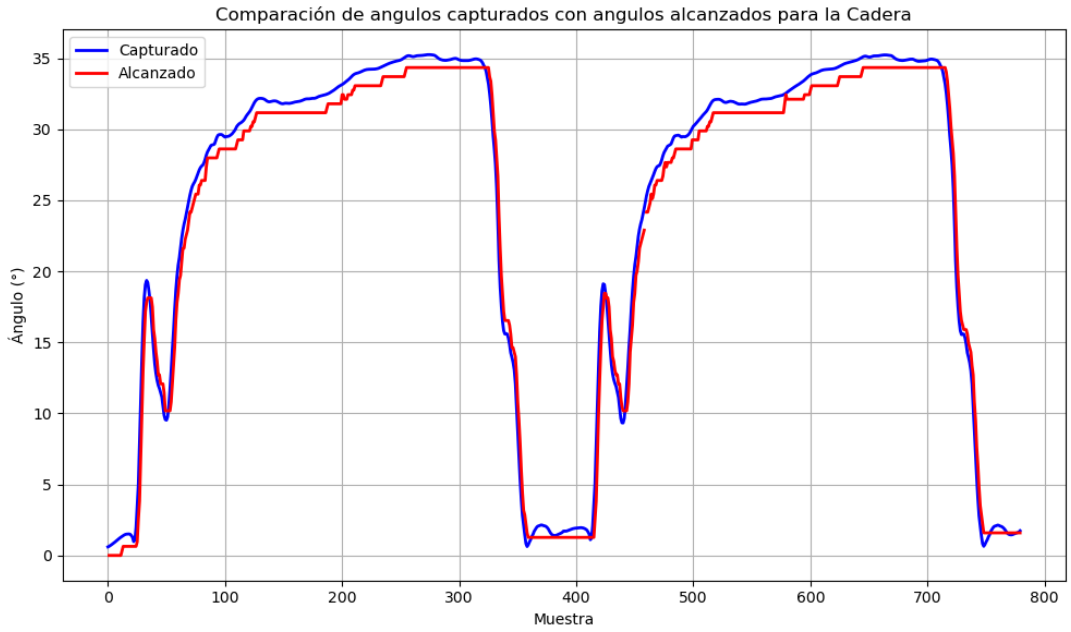



Figura 5.51: 3ra Gráfica de comparación de posiciones angulares de la cadera con sistema de réplica físico.

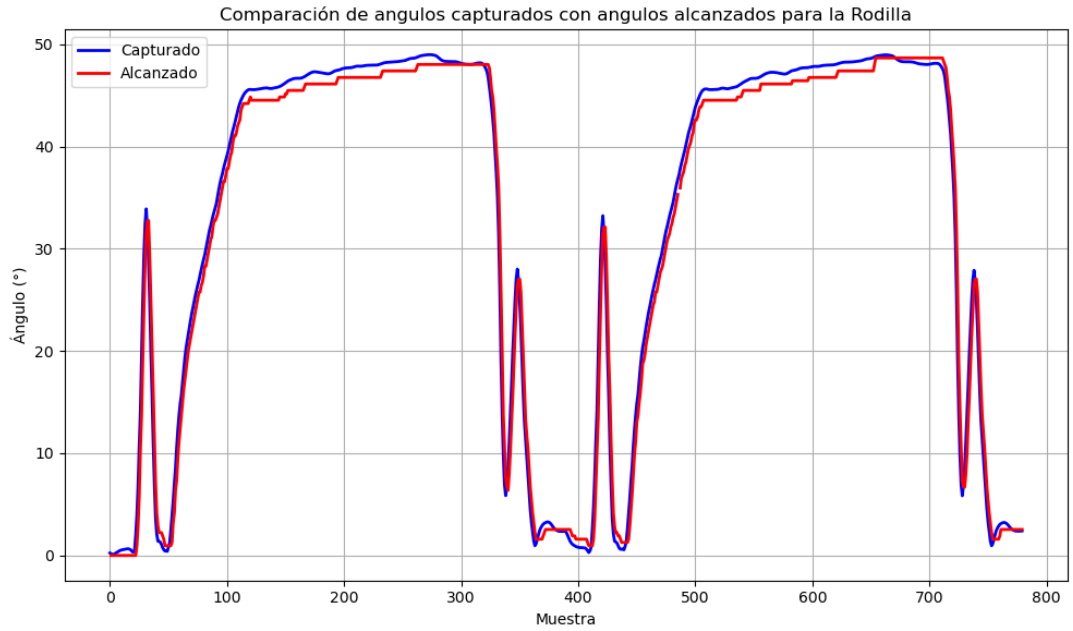



Figura 5.52: 3ra Gráfica de comparación de posiciones angulares de la rodilla con sistema de réplica físico.

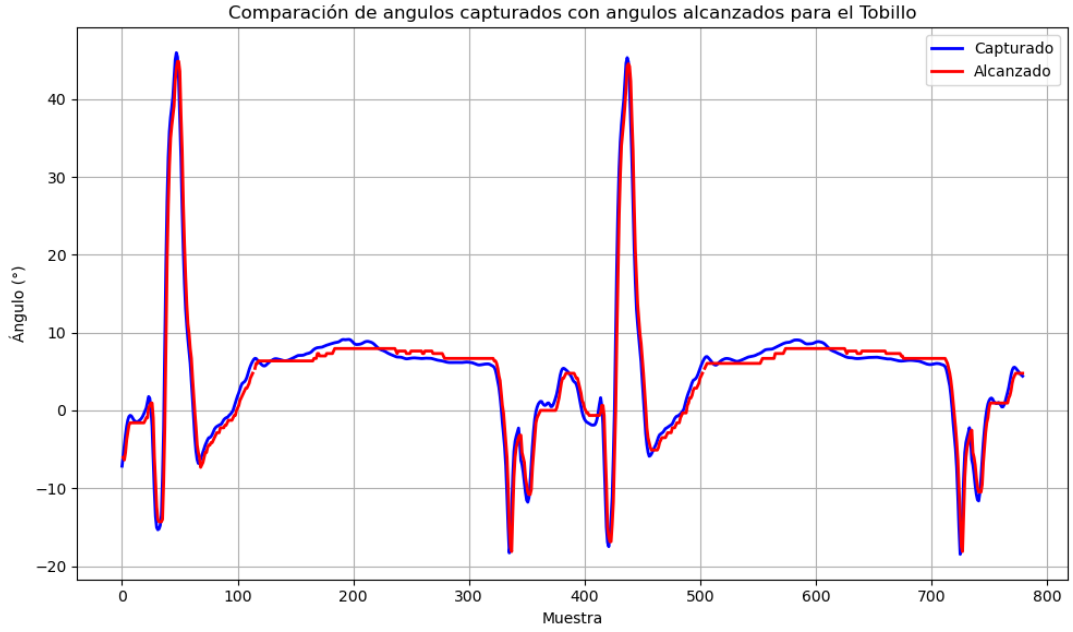



Figura 5.53: 3ra Gráfica de comparación de posiciones angulares del tobillo con sistema de réplica físico.

En general, el error observado entre los datos de esta prueba se puede atribuir a la capacidad de respuesta de los motores ante variaciones de baja magnitud, provocando que en diversos puntos o momentos los actuadores mantuvieran su posición aún recibiendo nuevas ordenes. Los resultados de los errores absolutos, Tabla 5.22, confirman la gran diferencia entre resultados.

Tabla 5.22: Errores absolutos promedio de posicionamiento (Prueba 3, sistema de réplica físico)

| **Articulación** | **Error absoluto promedio en grados** | **Error porcentual promedio** |
|---|---|---|
| **Cadera** | 0.75° | 9.13 % |
| **Rodilla** | 0.82° | 11.2 % |
| **Tobillo** | 1.24° | 35.73 % |

**Comparación de errores con el sistema de réplica virtual.**

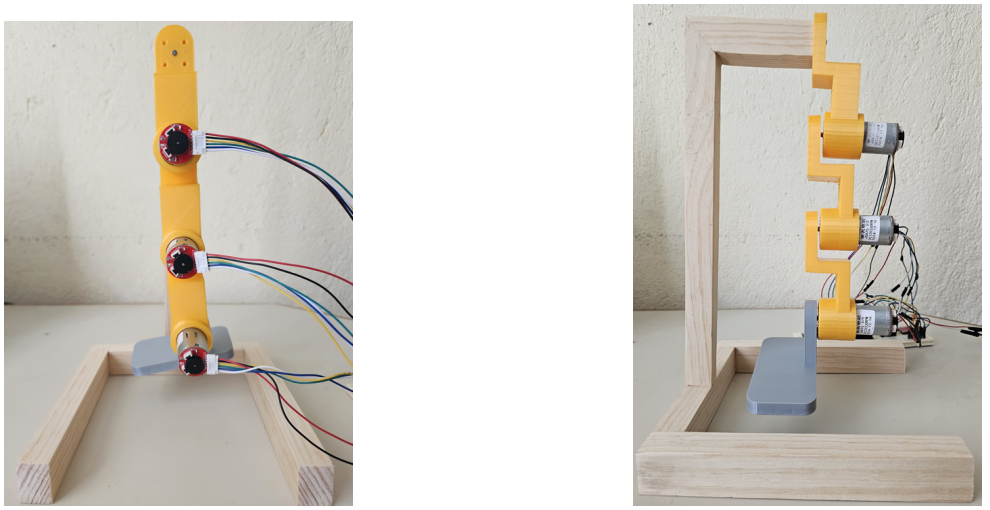
Como parte del proceso de verificación de la precisión del sistema de réplica físico, se realizó la comparación de errores con el sistema virtual usando el envío de posiciones post-análisis. Para esto se ejecutaron las mismas pruebas que se llevaron a cabo con el modelo virtual en el sistema de réplica físico. Los errores absolutos en porcentaje de cada sistema de réplica se muestra en la Tabla 5.23.

Tabla 5.23: Error absoluto medio de sistemas de réplica de movimiento.

| **Proceso** | **Articulación** | **Sistema virtual** | **Sistema físico** |
|---|---|---|---|
| **Prueba 1** | Cadera | 0.51 % | 3.64 % |
| | Rodilla | 0.83 % | 5.28 % |
| | Tobillo | 0.79 % | 1.06 % |
| **Prueba 2** | Cadera | 0.56 % | 3.87 % |
| | Rodilla | 0.61 % | 3.96 % |
| | Tobillo | 0.47 % | 3.11 % |
| **Prueba 3** | Cadera | 0.55 % | 1.0 % |
| | Rodilla | 0.51 % | 1.43 % |
| | Tobillo | 1.46 % | 2.03 % |

Los resultados de cada una de las pruebas de réplica de movimiento confirman una serie de detalles sobre este sistema, el más importante es que aunque el sistema consigue replicar rutinas de movimiento con un error absoluto bajo de menos del 2 %, el elevado error porcentual sugiere que en promedio, la precisión disminuye considerablemente ante la réplica de movimientos pequeños, como los capturados durante ejercicios con posiciones cuasi-estáticas. Sin embargo este comportamiento es totalmente previsible debido al sistema electrónico actual pues el uso de motores DC implica restricciones en cuanto a precisión, especialmente a escalas pequeñas con variaciones menores a dos grados.

Otro inconveniente presente en este sistema ocurre al momento de colocar los motores en el modelo impreso para la reproducción visual de los movimientos. El motor seleccionado para este sistema, modelo GM25-370, cuenta con un par nominal o torque de 5 $kg \cdot cm$ que resulta ser insuficiente para mantener las posiciones de las rutinas de movimiento, si bien esto no representa un problema critico en el motor del tobillo, en los motores de la cadera y la rodilla si se vuelve un factor limitante, debido al mayor peso que deben cargar. Como consecuencia de esto algunas posiciones no son alcanzadas o no se mantienen correctamente, lo que afecta la fidelidad de la réplica del movimiento aún si se realiza una reducción en la longitud de las piezas del modelo, Figura 5.54.



(a) Vista frontal. (b) Vista lateral.

Figura 5.54: Modelo físico de representación de miembro inferior.

Para alcanzar una precisión, y una visualización similar a lo conseguido en el sistema de réplica virtual, Tabla 5.23, se debe considerar la implementación de modificaciones a nivel de hardware y de software, optimizando el control PID del programa y empleando motores con mayor par nominal y con encoders de mayor resolución, especialmente en la articulación de la cadera. Modelos como el *Cytron PGM* 131 : 1 con un torque nominal de 15 $kg \cdot cm$, o el *Pololu* 37*D* 100 : 1 con un par nominal de 10 $kg \cdot cm$, son alternativas viables para su implementación en el sistema de réplica de movimiento físico, ambos son compatibles con controladores PID, cuentan con encoders integrados y gracias a su alto troque tendrían la capacidad de mover el peso actual del modelo sin problemas cumpliendo con las posiciones de cualquier tipo de rutina que se desee reproducir.

Aún con los inconvenientes descritos, para los propósitos de este proyecto, el sistema de réplica físico cumple con una función muy importante, ya que demuestra que los datos obtenidos mediante los sistemas de captura desarrollados pueden ser utilizados en procesos de reproducción de movimiento con una fidelidad aceptable. Los componentes seleccionados para conformar el sistema electrónico, junto con el programa desarrollado para su control, permiten ejecutar rutinas de movimiento estables que se desarrollan con una latencia mínima.

# Capítulo 6

# Conclusiones y trabajos futuros

## 6.1. Conclusiones

En este trabajo, y con el fin de cumplir con el objetivo del proyecto, se desarrollaron dos sistemas de captura de movimiento enfocados en el registro del movimiento angular de las principales articulaciones del miembro inferior, reproduciendo el movimiento capturado mediante dos sistemas de réplica, en un modelo virtual y en un sistema físico.

Para la selección de cada recurso o herramienta requerida en el desarrollo de ambos sistemas de captura, se compararon y analizaron las opciones disponibles, destacando ventajas y desventajas que ofrecían las alternativas. En cada sistema de captura implementado, la identificación de las articulaciones se llevó a cabo mediante visión artificial, haciendo uso de herramientas distintas en cada sistema, *MediaPipe* en un caso y *MMpose* en el otro. Para el cálculo de cada ángulo de movimiento se utiliza la ley de cosenos, incorporando además filtros Kalman y Gaussiano en distintas etapas de los programas de captura para la obtención de mejores resultados.

Mediante la realización de diversas pruebas y el análisis de resultados de cada sistema de captura, se determinó la superioridad del sistema de captura basado en *MediaPipe*, ya que los datos sobre el movimiento angular que ofrece son más estables y confiables,

destacando también la facilidad de su uso y modificación. Si bien *MMPose* ofrece resultados aceptables, carece de estabilidad e inclusive de precisión, pues tiende a errar la detección de las articulaciones de interés.

El sistema de réplica virtual utiliza el programa CoppeliaSim como entorno de simulación, diseñando un sistema de péndulo doble de tres eslabones para su uso como modelo de representación de miembro inferior para ambos sistemas de réplica de movimiento. Tras considerar las diferentes formas de posicionar actuadores virtuales utilizando los resultados del programa de captura, se optó por emplear el envió de ángulos de posición una vez a finalizado el análisis de movimiento, ya que esta alternativa ofrece una reproducción más fluida, y con una diferencia mínima respecto al tiempo real de la rutina analizada. Debido a las implicaciones propias de un proceso de réplica mediante el uso de motores DC, el sistema de reproducción física emplea un envió secuencial de posiciones, gestionando el movimiento de cada motor mediante un control tipo PID integrado en el programa que opera sobre la tarjeta programable utilizada. A través de estos métodos se consiguió una reproducción más precisa y coherente del movimiento que se analiza.

## 6.2. Trabajo futuro

- ✓ Rediseñar el sistema de captura de movimiento para la obtención de resultados más precisos y estables en la articulación del tobillo.

- ✓ Implementar una interfaz de usuario a los programas de captura y réplica de movimiento, con el fin de facilitar su uso y permitir una interacción más accesible e intuitiva.

- ✓ Implementar en los diferentes programas medidas de seguridad contra errores como lo pueden ser, lectura fallida del video, video sin una persona realizando movimientos, errores de conexión entre programas o sistemas, etc.

- ✓ Rediseñar el sistema de réplica de movimiento físico para mejorar la precisión de la reproducción del movimiento.
- ✓ Diseñar un dispositivo de rehabilitación física funcional para su uso en conjunto con los sistemas de captura de movimiento.

# Referencias

[1] D. Sierra, G. Sánchez Brizuela, J. Pérez Turiel y J. C. Fraile, "Control de exoesqueleto de rehabilitación de mano con Leap Motion Controller," en *XLIII Jornadas de Automática*, Universidade da Coruña. Servizo de Publicacións, 2022, págs. 141-147.

[2] IBM, *¿Qué es la visión artificial?* En linea, Disponible: `https://www.ibm.com/mx-es/topics/computer-vision`. Fecha de acceso: 19 Abril 2024, Diciembre de 2022.

[3] H. Hubel y N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *J Physiol*, vol. 3, n.º 148, págs. 574-591, 1959. DOI: 10.1113/jphysiol.1959.sp006308.

[4] I. García y V. Caranqui, "La visión artificial y los campos de aplicación," *Tierra infinita*, vol. 1, n.º 1, págs. 98-108, 2015.

[5] C. Luaces Vela, "Diseño e implementación de un entorno virtual de ejercicios físicos, basados en captura de movimiento," 2018.

[6] G. Azcárate Hernández, "Sistema de seguimiento visual para la rehabilitación," Diciembre de 2007.

[7] A. Condo y A. Liliana, "Desarrollo de un sistema de visión artificial a través de una tarjeta de desarrollo para controlar el Brazo Robótico Mitsubishi," B.S. Tesis, 2017.

[8] R. J. Moreno, F. A. E. Valcárcel y D. A. Hurtado, "Control de movimiento de un robot humanoide por medio de visión de máquina y réplica de movimientos humanos," *INGE CUC*, vol. 9, n.º 2, págs. 44-51, 2013.

[9] D. Bravo, C. Rengifo y W. Agredo, "Comparación de dos Sistemas de Captura de Movimiento por medio de las Trayectorias Articulares de Marcha," *Revista mexicana de ingeniería biomédica*, vol. 37, n.º 2, págs. 149-160, 2016.

[10] J. E. Muñoz-Cardona, O. A. Henao-Gallo y J. F. López-Herrera, "Sistema de Rehabilitación basado en el Uso de Análisis Biomecánico y Videojuegos mediante el Sensor Kinect," *TecnoLógicas*, 2013.

[11] J. Poza, "Ciclo de vida de los proyectos: la nueva aproximación de PMBOK," *Medium*, vol. 6, 2018, Disponible: https://medium.com/blog-de-astanapm/ciclo-de-vida-de-los-proyectos-la-nueva-aproximacin-de-pmbok-6-edicin-acbb0f91661e. Fecha de acceso: 15 Mayo 2024.

[12] L. L. G. Echeverry, A. M. J. Henao, M. A. R. Molina, S. M. V. Restrepo, C. A. P. Velásquez y G. J. S. Bolívar, "Sistemas de captura y análisis de movimiento cinemático humano: una revisión sistemática," *Prospectiva*, vol. 16, n.º 2, págs. 24-34, 2018.

[13] Google, *Mediapipe*, En linea, Disponible: https://ai.google.dev/edge/mediapipe/solutions/guide?hl=es-419. Fecha de acceso: 24 julio 2024, nov. de 2023.

[14] X. Teira, A. Guerra, G. Santamarina, L. Muños y V. Villareal, "Detección de mascarillas utilizando reconocimiento facial," *Tecnologia en Marcha*, vol. 36, págs. 57-65, oct. de 2023, Disponible: https://dialnet.unirioja.es/servlet/articulo?codigo=9270596.

[15] D. Plua, "Aplicación móvil de lenguaje de señas para personas con problemas auditivos mediante MediaPipe con acceso a cámara en tiempo real y dar a conocer la muestra correcta de las señas," sep. de 2021.

[16] Y. Inoue, Y. Kuroda, Y. Yamanoi, Y. Yabuki y H. Yokoi, "Development of Separate Exoskeleton Socket of Wrist Joint on Myoelectric Prosthetic Hand for Congenital Defects with Symbrachydactyly," en *2022 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, IEEE, 2023, págs. 164-169.

[17] G. Boesch, *The complete guide to OpenPose 2024*, En linea, Disponible: https://viso.ai/deep-learning/openpose/. Fecha de acceso: 05 Mayo 2024.

[18] Z. Cao, T. Simon, S. Wei e Y. Sheikh, *Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*, En linea, Disponible: https://ar5iv.labs.arxiv.org/html/1611.08050. Fecha de acceso: 10 Abril 2024.

[19] Vicon, *Valkyrie*, En linea, Disponible: https://www.vicon.com/hardware/cameras/valkyrie/. Fecha de acceso: 8 Abril 2024.

[20] Microsoft, *Kinect SDK*, En linea, Disponible: https://learn.microsoft.com/es-es/windows/apps/design/devices/kinect-for-windows. Fecha de acceso: 01 Mayo 2024.

[21] Microsoft, *Especificaciones Kinect*, En linea, Disponible: https://learn.microsoft.com/es-es/azure/kinect-dk/hardware-specification. Fecha de acceso: 11 Abril 2024.

[22] A. Nogueras, J. Arenillas, J. Rodríguez, F. Iglesias y C. Sánchez, "Fases de la marcha humana," *Revista Iberoamericana de Fisioterapia y Kinesiologia*, vol. 2, págs. 44-99, Enero de 1999, Disponible: https://www.elsevier.es/es-revista-revista-iberoamericana-fisioterapia-kinesiologia.

[23] F. H. Netter, *Atlas of Human Anatomy*, 5.ª ed. Philadelphia, PA, USA: Saunders, 2010, págs. 387-389.

[24] C. Schyke, D. Girela, F. Zurita Ortega y R. García, "Efectos de un programa de educacion en salud y entrenamiento de la fuerza en adultos mayores con artrosis de cadera de leve a moderada," Tesis doct., dic. de 2012.

[25] C. Panesso, "Biomecanica clinica de la rodilla," *Universidad del Rosario*, vol. 39, págs. 13-25, Diciembre de 2008.

[26] B. MD, "Anatomia normal de la articulación de la rodilla," Abril de 2024, Disponible:https://berkmanmd.com/anatomia-normal-de-la-articulacion-de-la-rodilla.. Fecha de acceso: 09 Mayo 2024.

[27] "Biomecánica de la Marcha Humana," *Desarrollo de soluciones de formacion innovadoras en el campo de la evaluacion funcional destinadas a actualizar los planes de estudio de las escuelas de ciencias de la salud*, págs. 32-33, 2024.

[28] F. Valencia, X. Lima, D. Ojeda y D. Ortiz, "Prótesis de rodilla externa mecatrónica," *Biomecánica*, pág. 35, Enero de 2015. DOI: 10.5821/sibb.23.1.4821.

[29] Fissioterapia, *La Importancia de la Dorsiflexión de Tobillo*, Disponible: https://fissioterapia.blogspot.com/2013/11/la-importancia-de-la-dorsiflexion-de.html. Fecha de acceso: 29 Abril 2024, 2013.

[30] C. Maximo, "Diseño de un Mecanismo de Movimiento Paralelo con dos Grados de Libertad para la Rehabilitación Pasiva Postquirúrgica de la Rodilla," *Universidad Señor de Sipán*, mar. de 2023, Disponible: https://repositorio.uss.edu.pe/bitstream/handle/20.500.12802/10899.

[31] C. J. Payton y R. M. Bartlett, "Biomechanics: Basics and Applied Research," en *Encyclopedia of Earth Sciences Series*, V. S. Williams, ed., Dordrecht: Springer, 2014. DOI: 10.1007/978-94-007-6046-2_82. dirección: https://link.springer.com/referenceworkentry/10.1007/978-94-007-6046-2_82.

[32] M. Johansson, *3D Human Pose and Shape Estimation Using Motion and Shape Priors*, https://www.diva-portal.org/smash/get/diva2:120188/FULLTEXT01.pdf, 2012.

[33] OpenMMLab, *MMPose: Documentation*, https://mmpose.readthedocs.io/en/latest/, Último acceso: marzo 2025, 2024.

[34] DeepLabCut, *DeepLabCut: Implementación oficial en GitHub*, Fecha de acceso: 27 Julio 2024, 2024. dirección: https://github.com/DeepLabCut/DeepLabCut.

[35] P. J. Soto-Pedraza et al., "Demostración de la Ley de Cosenos," *Vida Científica Boletín Científico de la Escuela Preparatoria No. 4*, vol. 9, n.º 17, págs. 21-23, 2021.

[36] A. O. Agbailu, A. Seno y O. O. Clement, "Kalman filter algorithm versus other methods of estimating missing values: time series evidence," *Studies*, vol. 4, n.º 2, págs. 1-9, 2020.

[37] V. A. O. Bravo, M. A. N. Arias y J. A. C. Cardenas, "Análisis y aplicación del filtro de Kalman a una señal con ruido aleatorio," *Scientia et technica*, vol. 18, n.º 1, págs. 267-274, 2013.

[38] Z. Meyer y A. Pretorius, "Design of a low-cost optical motion capture system using a multi-camera configuration and an asynchronous extended Kalman filter," *MATEC Web of Conferences*, vol. 406, pág. 04 010, dic. de 2024. DOI: 10.1051/matecconf/202440604010.

[39] Á. Solera-Ramírez, "Filtro de Kalman," Banco Central de Costa Rica, inf. téc., 2003.

[40] Coppelia Robotics, *CoppeliaSim Features*, En linea. Ultimo acceso 10/01/2025, Coppelia Robotics, 2023. dirección: https://manual.coppeliarobotics.com/en/coppeliaSimFeatures.htm.

[41] uElectronics, *GM-25-370 - Motor con Encoder 12V DC (140RPM - 330RPM)*, https://uelectronics.com/producto/gm-25-370-motor-con-encoder-12v-dc-140rpm-330rpm/, 2023.

[42] N. Mechatronics. "Driver Puente H TB6612FNG." En linea. Ultimo acceso 13/01/2025. (s.f.), dirección: https://naylampmechatronics.com/drivers/200-driver-puente-h-tb6612fng.html.

[43] Women's Health Magazine, *Women's Health: Fitness, Nutrition, Health, Weight Loss*, 2023. dirección: https://www.womenshealthmag.com/.

[44] T. Jiang, P. Lu, L. Zhang et al., *RTMPose: Real-Time Multi-Person Pose Estimation based on MMPose*, 2023. arXiv: 2303.07399 [cs.CV]. dirección: https://arxiv.org/abs/2303.07399.

[45] OpenMMLab, *RTMPose: A Real-Time Multi-Person Pose Estimation Framework*, https://github.com/open-mmlab/mmpose/tree/main/projects/rtmpose, Último acceso: marzo 2025, 2024.

[46] Hinge Health, *Rehabilitation exercises*, Video, Accedido: Mayo-20-2025, 2023. dirección: https://es.hingehealth.com/resources/articles/.

# Anexos

## A. Códigos utilizados

### A1. Programa de MediaPipe

https://drive.google.com/drive/folders/166nFuAAGTAcytq7hl

### A2. Programa de MMPose

https://drive.google.com/drive/folders/1Ob8nD5EcfsnGoA9F

### A3. Programa de envió de posiciones para la réplica de movimiento virtual

https://drive.google.com/drive/folders/1HBO7TnaxvUPvWOxDHNR