UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Desarrollo de un Sistema Avanzado de Reconocimiento de Intención y Fases de Marcha para Exoesqueletos de Miembros Inferiores utilizando Datos de IMUs

Trabajo Fin de Máster

Máster Universitario en Ingeniería Biomédica

AUTOR/A: Martín Molina, Javier

Tutor/a: Belda Lois, Juan Manuel

CURSO ACADÉMICO: 2023/2024

Agradecimientos

A mi tutor, Juan Manuel Belda Lois por su dedicación, paciencia y sabiduría a lo largo de este proyecto. Sin sus valiosos conocimientos y sugerencias siempre constructivas no estaría aquí hoy. Le estoy enormemente agradecido por dedicarme su tiempo y compromiso para ayudarme a alcanzar mis metas académicas.

A mis amigos, no tengo palabras suficientes para expresar cuánto valoro vuestras palabras de aliento, vuestras ideas y vuestra disposición para escucharme en momentos de duda han sido un verdadero salvavidas emocional. Gracias por estar ahí hasta el final.

Y finalmente, a mis padres, mi gratitud es eterna. Vuestra confianza inquebrantable en mí, vuestro amor incondicional y vuestro apoyo constante son los pilares que han sostenido mi espíritu a lo largo de los años. Vuestras palabras de aliento, vuestros abrazos cálidos y vuestra presencia en cada logro y desafío son mi mayor fuente de fortaleza.

RESUMEN

0.1 Resumen

Introducción: Los exoesqueletos de las extremidades inferiores presentan una vía prometedora para ayudar a las personas con problemas de movilidad a lograr mejores patrones de marcha y funcionalidad, si bien los sistemas de control desarrollados aún dejan que desear. Este trabajo propone una investigación exhaustiva sobre la detección de la fase de la marcha y el reconocimiento de intenciones para los exoesqueletos de extremidades inferiores, concretamente en pacientes hemipléjicos, utilizando unidades de medición inercial (IMU) junto con un algoritmo iterativo de predicción de puntos sigma.

Metodología: El estudio implica la recopilación de datos de una cohorte diversa de individuos sanos (n = 8) que abarca tanto hombres como mujeres para garantizar la solidez y generalizabilidad del modelo propuesto. Tres IMU colocadas unilateralmente en segmentos de las extremidades inferiores en pelvis, muslo y tibia capturan los patrones de movimiento complejos de la marcha normal. Para cada sujeto, se toman cuatro mediciones, incluyendo dos calibraciones y dos paseos. Tras la calibración de los datos, se diseña e implementa un filtro bayesiano no lineal para la identificación en tiempo real del de inicio de la marcha y el porcentaje de fase.

Objetivos: El proyecto pretende lograr varios objetivos: (I) Desarrollar una metodología coherente para sincronizar los datos de IMU con los principales eventos de la marcha; (II) Diseñar e implementar un algoritmo iterativo de predicción de puntos sigma capaz de detectar y predecir el inicio, fin y porcentaje de fase de la marcha; (III) Investigar la viabilidad de detectar el inicio y fin de la marcha; y (IV) Validar el enfoque propuesto con datos de sujetos reales.

Resultados: Los resultados de este trabajo muestran un rendimiento favorable de la metodología empleada, indicando que esta propuesta puede ayudar a sentar las bases para el desarrollo de sistemas de control de exoesqueletos inteligentes que se sincronicen perfectamente con los movimientos del usuario, promoviendo así una mayor independencia y calidad de vida.

Palabras clave: Exoesqueletos de miembros inferiores, Detección de fases de marcha, Reconocimiento de la marcha, IMU, *Unscented Kalman Filter*, Marcha sana, Marcha hemipléjica.

0.2 Resum

Introducció: Els exoesquelets de les extremitats inferiors presenten una via prometedora per ajudar les persones amb problemes de mobilitat a aconseguir millors patrons de marxa i funcionalitat, si bé els sistemes de control desenvolupats encara deixen de desitjar. Aquest treball proposa una investigació exhaustiva sobre la detecció de la fase de la marxa i el reconeixement d'intencions per als exoesquelets d'extremitats inferiors, concretament en pacients hemiplègics, utilitzant unitats de mesura inercial (IMU) juntament amb un algorisme iteratiu de predicció de punts sigma.

Metodologia: : L'estudi implica la recopilació de dades d'una cohort diversa d'individus sans (n=8) que abasta tant homes com dones per garantir la solidesa i la generalitzabilitat del model proposat. Tres IMU col·locades unilateralment en segments de les extremitats inferiors a pelvis, cuixa i tíbia capturen els patrons de moviment complexos de la marxa normal. Per a cada subjecte, es prenen quatre mesures, incloent dos calibratges i dues passejades. Després del calibratge de les dades, es dissenya i s'implementa un filtre bayesià no lineal per a la identificació en temps real del d'inici de la marxa i el percentatge de fase.

Objetius: El projecte pretén assolir diversos objectius: (I) Desenvolupar una metodologia coherent per sincronitzar les dades d'IMU amb els esdeveniments principals de la marxa; (II) Dissenyar i implementar un algorisme iteratiu de predicció de punts sigma capaç de detectar i predir l'inici, la fi i el percentatge de fase de la marxa; (III) Investigar la viabilitat de detectar l'inici i el final de la marxa; i (IV) Validar l'enfocament proposat amb dades de subjectes reals.

Resultats: Els resultats d'aquest treball mostren un rendiment favorable de la metodologia emprada, indicant que aquesta proposta pot ajudar a establir les bases per al desenvolupament de sistemes de control d'exosquelets intel·ligents que se sincronitzin perfectament amb els moviments de l'usuari, promovent així una independència més gran i qualitat de vida.

Paraules clau: Exoesquelets de membres inferiors, Detecció de fases de marxa, Reconeixement de la marxa, IMU, *Unscented Kalman Filter*, Marxa sana, Marxa hemiplègica.

0.3 Summary

Introduction: Lower extremity exoskeletons present a promising avenue to help people with mobility problems achieve better walking patterns and functionality, although the control systems developed still leave something to be desired. This work proposes a comprehensive investigation on gait phase detection and intention recognition for lower extremity exoskeletons, specifically in hemiplegic patients, using inertial measurement units (IMU) together with an iterative sigma point prediction algorithm.

Methodology: The study involves collecting data from a diverse cohort of healthy individuals (n = 8) encompassing both men and women to ensure the robustness and generalizability of the proposed model. Three IMUs placed unilaterally on lower extremity segments in the pelvis, thigh, and tibia capture the complex movement patterns of normal walking. For each subject, four measurements are taken, including two calibrations and two walks. After calibration of the data, a non-linear Bayesian filter is designed and implemented for the real-time identification of the start of gait and the percentage of phase.

Objectives: The project aims to achieve several objectives: (I) Develop a coherent methodology to synchronize IMU data with major gait events; (II) Design and implement an iterative sigma point prediction algorithm capable of detecting and predicting the start, end and percentage of the gait phase; (III) Investigate the feasibility of detecting the beginning and end of gait; and (IV) Validate the proposed approach with data from real subjects.

Results: The results of this work show a favorable performance of the methodology used, indicating that this proposal can help lay the foundations for the development of intelligent exoskeleton control systems that synchronize perfectly with the user's movements, thus promoting greater independence and quality of life.

Keywords: Lower limb exoskeletons, Gait phase detection, Walking recognition, IMU, Unscented Kalman Filter, Healthy gait, Hemiplegic gait.

Índice general

Índice de figuras

Índice de tablas

Parte I

Memoria

Capítulo 1

INTRODUCCIÓN

1.1 CONTEXTO Y JUSTIFICACIÓN DEL TRABAJO

1.1.1 El Movimiento Humano

En la gentil sinfonía del movimiento humano, el caminar se erige como una atributo fundamental y elegante en que la gravedad y los músculos se enfrascan en un rítmico compás de equilibrio e impulso. Sin embargo, debajo de este movimiento aparentemente sin esfuerzo se encuentra una compleja armonía biomecánica que da forma a nuestra marcha: el patrón único de movimiento que define cómo caminamos.

Durante la marcha, músculos, tendones, huesos y ligamentos de todo el cuerpo, trabajan en perfecta sincronía gracias al sistema nervioso para crear el movimiento fluido y natural que todos conocemos (Duan et al. 2017). La optimización de esta arquitectura neuro-muscular, permite a las personas mantener cierto balance y estabilidad a la par que se desplazan hacia delante. En efecto, es un hecho notable el que esta serie de movimientos coordinados que llamamos marcha mantiene unas proporciones áureas en su devenir (Iosa et al. 2013). No obstante, en ciertas ocasiones este patrón puede verse alterado por condiciones como la hemiparesia, hemiplejia, ataxia, amputación, artroplastia, enfermedad de Parkinson o como consecuencia misma de la edad (Baker 2018) como se explicará en la Sección 1.1.3.

Es de estos casos de donde surge la necesidad de diseccionar este armónico proceso empleando la diversa gama de técnicas y herramientas presentes en los análisis de marcha (Arellano-González et al. 2021). El objetivo principal de estos estudios reside habitualmente en identificar desviaciones respecto a la norma a través de distintas variables. Algunas de estas son numéricas como la longitud de paso, el centro de presiones o la pendiente del terreno, mientras que otras son cualitativas como la fase de marcha o el modo de marcha.

El objetivo de este primer capítulo pues, es llegar a comprender las principales estrategias empleadas en el análisis de la marcha para sentar los cimientos sobre los cuales se erigirán las siguientes secciones de este Trabajo Fin de Máster. Para ello, en esta primera sección se muestra una visión general de la estructura de la marcha junto a sus posibles alteraciones. En la segunda sección se comentan los métodos tradicionalmente empleados para el análisis de la marcha y, por lo tanto, aquellos con mayor bagaje empírico. Las siguientes tres secciones van destinadas a ofrecer una descripción de la gran variedad de nuevos sensores y algoritmos que, cada vez más van desplazando a los métodos tradicionales dado el interés y las posibilidades que presentan. Una vez presentados todos los actores, se han dedicado tres secciones más para explicar en mayor detalle los modelos de detección y/o predicción tanto para las fases como para los diversos patrones de la marcha. Para concluir este capítulo y dar paso a la metodología se indican el objetivo, la propuesta y la hipótesis de este Trabajo Fin de Máster.

1.1.2 La Estructura de la Marcha

La complejidad de la marcha queda patente una vez se comienza a intentar abordar el problema en cuestión. Para facilitar su análisis emplearemos una de las herramientas más ampliamente usadas en problemas físico-mecánicos: la segmentación por partes. Siendo así, se definen una serie de eventos que dividen la marcha en sus diferentes fases y sub-fases.

Los dos principales eventos son el contacto inicial del talón (*Heel-Strike* o HS) y el despegue del antepié del suelo (*Toe-Off* o TO). A partir de estos se pueden definir las dos principales fases de la marcha: Apoyo (*Stance* o ST) y Balanceo (*Swing* o SW). La primera, empieza con HS y acaba en TO, ocupando cerca del 60 % del ciclo, mientras que la segunda empieza en TO y acaba en HS, ocupando el 40 % restante (Taborri et al. 2016). Estas fases representan en qué puntos de la marcha el pie está en contacto con el suelo y en cuáles no, respectivamente. Además, si bien se suele estudiar la marcha tomando como referencia uno de los dos miembros inferiores, en realidad hay un solapamiento entre estas fases que hace que la marcha sea el movimiento harmónico que naturalmente todos reconocemos.

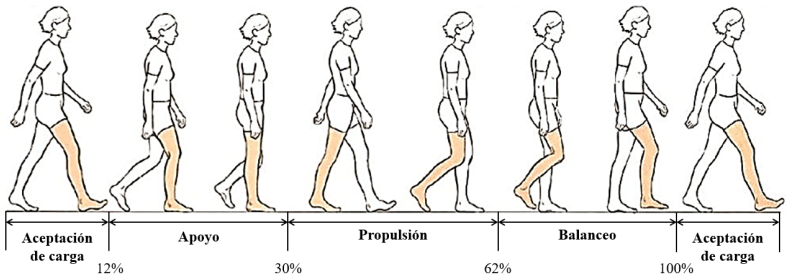


Figura 1.1: Modelo de cuatro fases de marcha (Chen et al. 2021b)

Para captar la idea general del ciclo es suficiente con las dos fases y eventos mencionados. Sin embargo, cuando se ven involucradas las alteraciones de la marcha, este fraccionamiento a menudo es incapaz de capturar completamente la complejidad del movimiento, impidiendo su correcto análisis. Estos inconvenientes han llevado a la incorporación de nuevos eventos y sub-fases. Específicamente, los eventos más empleados, a excepción de los antedichos, son el contacto completo del pie con el suelo (*Foot-Flat* o FF) y el despegue del talón del suelo (*Heel-Off* o HO). Estos permiten dividir las dos macro-fases en cuatro sub-fases. Por un lado, la fase de apoyo se divide en una sub-fase inicial de aceptación de la carga (*Loading-Response* o LR) que ocupa el primer 12 % del ciclo aproximadamente, la (sub-)fase de apoyo (ST) que continúa hasta el 30 % y una sub-fase de propulsión (*Push-Off* o PO) al final de la fase de apoyo que termina de completar el 62 %. Esta división permite mayor claridad a la hora de identificar los intervalos de tiempo en que se transfiere la carga del peso del cuerpo desde la pierna opuesta (LR), se lleva la carga hacia delante adelantando la pierna opuesta (ST) y se empieza a despegar el pie a la par que se transfiere la carga de vuelta a la pierna opuesta (PO) para empezar de nuevo el ciclo. Por otro lado, la fase de balanceo que sigue a PO permanece inalterada coincidiendo temporalmente con la sub-fase de balanceo (SW) en el modelo del ciclo de marcha de cuatro fases (Figura 1.1).

Existen numerosas formas de clasificar los eventos y sub-fases de la marcha que pueden ir desde las dos fases simples hasta un total de ocho sub-fases (Taborri et al. 2016). Sin embargo, a partir de las cuatro fases descritas en el párrafo previo, es posible extraer con precisión el periodo de la marcha más afectado ante la presencia de anomalías motrices: la sub-fase de doble soporte (DS). Esta etapa sucede durante PO que coincide con HS de la pierna contraria, haciendo que ambos pies estén en contacto con el suelo durante este breve lapso temporal que ocupa aproximadamente el 10 % de la marcha. El hecho de tener que compensar una mayor cantidad de fuerzas hace que DS sea el intervalo temporal con potencial para mayor inestabilidad. Por este motivo, con el objetivo de lograr un buen balance entre precisión temporal y viabilidad del esudio y diseño, este trabajo se centrará principalmente en el modelo de cuatro sub-fases descrito.

Tabla 1.1: Definiciones de las fases del ciclo de marcha (Taborri et al. 2016; Liu et al. 2014)

No. Fases	Fases de la marcha							
2	Apoyo					Balanceo		
3	Contacto inicial	Apoyo				Balanceo		
4	-	Aceptación de carga	Apoyo	Propulsión		Balanceo		
5	Contacto inicial	Aceptación de carga	Apoyo	Propulsión		Balanceo		
6a	Contacto inicial	Aceptación de carga	Apoyo medio	Apoyo final	Pre-Balanceo	Balanceo		
6b	-	Aceptación de carga	Apoyo medio	Apoyo final	Pre-Balanceo	Balanceo inicial	Balanceo final	
7	-	Aceptación de carga	Apoyo medio	Apoyo final	Pre-Balanceo	Balanceo inicial	Balanceo medio	Balanceo final
8	Contacto inicial	Aceptación de carga	Apoyo medio	Apoyo final	Pre-Balanceo	Balanceo inicial	Balanceo medio	Balanceo final
Porcentaje	2 %	12 %	30 %	50 %	62 %	75 %	88 %	100 %

1.1.3 Alteraciones de la Marcha

Como se mencionaba en la Sección 1.1.1 el análisis de la marcha nos permite conocer gran cantidad de información no tan aparente a simple vista sobre el estado funcional de un individuo. Además de la segmentación temporal comentada en la Sección 1.1.2, otros aspectos espaciales como la longitud y ancho de paso, el ancho de la base de soporte o la propia velocidad de marcha son habitualmente empleados para examinar la marcha complementando a su contrapartida.

En casos fisiológicos, la duración de las fases ST y SW es estable, manteniendo una proporción áurea entre sus duraciones (Iosa et al. 2013). No obstante, para las sub-fases HS, ST, PO y SW, independientemente de la edad, la altura o el peso, sí existe una alteración lógica y natural de hasta el 24.31 % en el periodo como consecuencia de un aumento o disminución de la velocidad de marcha (Liu et al. 2014). Además, bien por la edad, bien por inestabilidad intrínseca (e.g., vértigo, mareo, etc.) o extrínseca (e.g., superficie inestable), es normal que se produzca una reducción de la velocidad y la longitud de paso, así como un incremento de la sub-fase DS (i.e., marcha cautelosa) (Baker 2018).

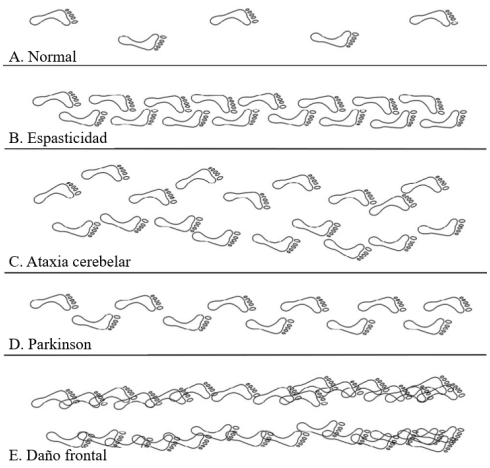


Figura 1.2: Morfología de paso en afectaciones de la marcha (Baker 2018)

A parte de estas alteraciones fisiológicas de la marcha, existen aquellas etiológicamente patológicas que constituyen el mayor ámbito de interés desde el punto de vista clínico. Patologías ortopédicas como la osteoartritis o deformaciones oseas impiden la marcha normal al reducir el tiempo que pasa el individuo en la fase de apoyo del miembro afectado debido al dolor, manifestándose en un patrón de marcha asimétrico (i.e., marcha antálgica) (Pirker y Katzenschlager 2017). Siguiendo esta línea, el otro tipo de desórdenes que afecta a la marcha son aquellos de origen neurológico. Estos incluyen la ataxia cerebelar y/o sensorial (18 % de los casos), la enfermedad de Parkinson (16.2 % de los casos), daño a las regiones del córtex frontal (7.7 % de los casos), paso cauteloso anormalmente excesivo (6 % de los casos), paresia (5.1 % de los casos), espasticidad (5.1 % de los casos), otros (4.3 % de los casos) o una combinación entre estos (30.8 % de los casos) (Baker 2018). La Figura 1.2 muestra una comparativa de las características de la marcha normal con la de aquellos individuos afectados por alguno de los trastornos de la marcha mencionados (Tabla 1.2).

1.2 HIPÓTESIS Y OBJETIVOS

El presente Trabajo Fin de Máster tiene como objetivo principal estudiar la dinámica de la marcha en varios sujetos de ensayo sanos, con el propósito último de desarrollar un algoritmo robusto capaz de predecir y detectar el inicio y el fin de la marcha, así como las dos principales fases de cada ciclo de marcha. Asimismo, la concepción se inspira en la idea de aplicar los resultados del mismo en el control de exoesqueletos de miembros inferiores para pacientes hemipléjicos.

Para el estudio se recopilarán datos de marcha de 8 sujetos voluntarios, se procesarán y se extraerán las variables de interés, y en base a estas se estimarán los parámetros del modelo predictivo. Se tomará como hipótesis principal que el *Unscented Kalman Filter* constituye una opción viable para el pronóstico y la identificación de los eventos clave de la marcha en tiempo real dada la naturaleza no lineal de este proceso y la notable influencia del ruido en la medida.

Los objetivos secundarios de este trabajo serán los detallados en los siguientes puntos:

1. Estudio del estado del arte para situar debidamente el trabajo y obtener una visión completa del contexto en que se pretende englobar.
2. Concepción de un diseño experimental adecuado que permita capturar los eventos fundamentales de la marcha de forma precisa.
3. Extracción de los datos en bruto a través de los medios adecuados
4. Procesar correctamente los datos en bruto y extraer las variables de interés teniendo en cuenta el modelo de la marcha humana.
5. Desarrollo de un algoritmo para la segmentación de los periodos de marcha.
6. Extracción manual de las etiquetas de las fases de marcha en cada periodo.
7. Desarrollo de un algoritmo para la predicción de las fases de la marcha.
8. Evaluación del rendimiento de cada algoritmo.
9. Esclarecer la efectividad y precisión de los algoritmos diseñados y extraer las debidas conclusiones.

En este TFM se contribuye al ODS 3: Salud y Bienestar, a través de su Meta 3.4: Reducción de las enfermedades no transmisibles y salud mental, ya que el objetivo del proyecto es desarrollar un algoritmo de predicción y detección del inicio y fin de la marcha y de las fases principales de cada ciclo con la intención de ser aplicado a un exoesqueleto robótico para miembros inferiores en pacientes hemipléjicos, aspirando a mejorar la movilidad de aquellas personas que padecen dicha patología motriz y mejorar, así, su calidad de vida.

Además, se contribuye al ODS 9: Industria, innovación e infraestructura, a través de su Meta 9.5: Aumento de la investigación científica, capacidad tecnológica, mediante el desarrollo de dicho algoritmo en el cual no sólo se emplean datos obtenidos por medio de sensores de vanguardia como son los IMUs, sino que también se ponen en práctica metodologías como el *Unscented Kalman Filter* que han dado resultados consistentes a lo largo de su historia y ofrecen un excelente grado de calidad a los resultados presentados.

1.3 ESTADO DEL ARTE: SISTEMAS DE MEDIDA

1.3.1 Sistemas de Captura Cinemática

Sistemas ópticos de captura tridimensional

Dada la importante contribución del análisis de la marcha al diagnóstico temprano y rehabilitación específica de una gran gama de patologías neurodegenerativas, es de vital importancia emplear métodos de análisis cuantitativos tan precisos y objetivos como sea técnicamente posible. Uno de los primeros métodos de análisis de la marcha dentro de esta rama es la captura óptica tridimensional del movimiento, pertenecientes a los sistemas de medida optoelectrónicos o OMSs (Andrés Diaz et al. 2009). Los OMSs se pueden diferenciar en dos categorías según la necesidad del uso de marcadores físicos.

Tabla 1.2: Características de las afectaciones de la marcha (Baker 2018, Pirker y Katzenschlager 2017)

Desorden	Prevalencia	Características motoras	Características de la marcha
Ataxia sensorial	18 %	Déficit propioceptivo Compensación visual parcial Romberg positivo	Menor longitud de paso Mayor base en el apoyo Menor calidad del contacto inicial
Enfermedad de Parkinson	16.2 %	Temblor Rigidez muscular Bradicinesia Falta de estabildad	Menor longitud y altura de paso Menor base en el apoyo Menor balanceo del brazo Giros en bloque
Daño a las regiones del córtex frontal	8 %	Ansiedad Rigidez muscular Bradicinesia Movimientos no coordinados y descontrolados	Menor longitud de paso Mayor base en el apoyo Menor balanceo del brazo Mayor extensión del tronco Congelación Impedimento del inicio de la marcha
Ataxia cerebelar	6 %	Inestabilidad Romberg negativo Movimientos no coordinados e irregulares	Longitud de paso variable Mayor base en el apoyo Desviación lateral Dificultad en patrones complejos (giros, marcha en tándem, superficie irregular...)
Paso cauteloso anormalmente excesivo	6 %	-	Menor longitud de paso Lentitud y cautela
Paresia	5.2 %	Debilidad muscular unilateral o bilateral	-
Espasticidad	5.2 %	Rigidez muscular (Aumento del tono muscular)	Lentitud Mayor base en el apoyo Menor apoyo en el lado afectado y circundicción en el balanceo (Asimetría)
Otros	4.4 %	-	-
Combinación	31 %	-	-

- Basados en marcadores

Los OMSs basados en marcadores operan bajo la premisa de la detección de la luz reflejada en los marcadores de seguimiento, la cual se utiliza para triangular su posición empleando el tiempo de vuelo (Kruk y Reijne 2018). Entre estos, se diferencian: sistemas con marcadores activos, donde es el propio marcador el que actúa como fuente de luz (e.g., LED) para poder ser identificado fácilmente por la cámara (Matthew, Seko y Bajcsy 2017, Lee y Yoo 2017); sistemas con marcadores pasivos, donde el marcador en sí no emite luz, pero sí es capaz de reflejar la que le llega (e.g., radiación infrarroja) de modo que sea captada por la cámara sin mayor dificultad (Linnell 2015). El principal beneficio de un marcador activo frente a uno pasivo es la robustez de las medidas. No obstante, debido a su naturaleza, estos marcadores necesitan cables y baterías adicionales que limitan el movimiento y están limitados en número puesto que las frecuencias de emisión entre estos deben diferir lo suficiente para poder ser reconocidos por el sistema óptico de captura (Kruk y Reijne 2018).

Como norma general, los OMSs basados en marcadores presentan un bajo error de medida de aproximadamente 0.1 mm para las traslaciones y 0.1 ° para las rotaciones, aunque estos valores pueden oscilar notablemente en función del diseño experimental, incluyendo: volumen de captura, número y posición relativa de las cámaras, distancia cámara-marcador, número y posición de los marcadores y tipo y velocidad de los movimientos a realizar (Maletsky, Sun y Morton 2007). Se debe también tener en cuenta que la resolución de las imágenes y el campo de visión (FOV) guardan una relación inversamente proporcional con la frecuencia de muestreo por lo que es normal encontrar OMSs basados en marcadores con frecuencias de muestreo (fs) entre 100 y 1000 Hz (Tabla 1.3). Además, en caso de emplear marcadores activos, el número de estos también guarda una relación inversamente proporcional con fs.

Si bien la instalación de estos sistemas es relativamente compleja e invasiva sobre el espacio de trabajo, su vida de servicio es muy alta (Figura 1.3). Normalmente, entre 6 y 12 cámaras son empleadas en los laboratorios y su coste unitario va desde los 20 hasta los 11,000 €/ud. según la gama (e.g. Flex 3 y $Prime^x 120$ de Optitrack). Con este número de unidades, cada una de las cuales puede llegar a un rango de medida de 90 m, se pueden capturar volúmenes desde 9x9x6 m^3 hasta 45x45x6 m^3 con excelente precisión (Tabla 1.3), aunque en el caso de emplear marcadores pasivos, el volumen de captura se puede ver reducido hasta un 40 %. El tipo de marcador también influye sobre el uso de cualquier OMS basado en marcadores ya que con marcadores pasivos funcionará por definición indefinidamente, mientras que con marcadores activos su tiempo de funcionamiento se verá limitado a la vida de la batería de los marcadores (normalmente entre 12 y 72 horas).

Un punto favorable de los OMSs basados en marcadores es que al ser uno de los sistemas de medida más clásicos, cuentan con un mayor desarrollo del software necesario para las mediciones, haciendo que en la mayoría de los casos la intervención del usuario investigador en la toma y preprocesado de los datos sea mínima (Menolotto et al. 2020). Empleando marcadores pasivos, además, es un sistema mínimamente invasivo para el individuo que participa en el experimento, evitando artefactos de la marcha derivados de la intrusión del sistema de medida (e.g., cables, baterías, etc.). Por contra, los OMSs basados en marcadores son sistemas fijos, o como mucho de portabilidad limitada (Begon et al. 2009), y necesitan de una línea de visión constante sobre los marcadores, por lo que cualquier obstrucción de la cámara y/o de los marcadores puede suponer un error en la medida. Sumado a esto, son altamente sensibles a las interferencias por exceso de iluminación y a las vibraciones o micro-movimientos de la cámara. Es por este motivo que a menudo es necesario recalibrar estos sistemas para garantizar una medida correcta (Tabla 1.3).

En la literatura, los OMSs basados en marcadores (especialmente aquellos que emplean infrarrojos) son considerados el *gold standard* en la captura cinemática dada su robustez en relación al resto de métodos (Kruk y Reijne 2018; Nakano et al. 2020). De los 76 estudios de análisis de marcha incluidos para la revisión del estado del arte, sólo un 6.6 % emplearon un OMS basado en marcadores como sistema de recogida de datos (Tabla 1.3) y, de estos, sólo Dinovitzer, Shushtari y Arami 2023 se basó exclusivamente en un OMS basado en marcadores para obtener los datos. No obstante, estos sistemas son la principal referencia para el etiquetado de los datos en el 25 % de los estudios que indicaron explícitamente el uso de una 'verdad fundamental' (aproximadamente el 48 %) (e.g., Sharifi-Renani, Mahoor y Clary 2023; Romijnders et al. 2022; Zhao et al. 2016).

El uso de un sistema basado en marcadores para el análisis de la marcha se suele limitar mayoritariamente a la predicción cinemática (Tabla 1.3), siendo también posible el obtener (de forma no excluyente) la detección y/o la predicción de la cinética y la detección de las fases de la marcha. Conjuntamente, este tipo de sistemas se suele emplear también en el campo de la biomecánica para el modelado y la simulación. Además, dada la naturaleza del movimiento, las posiciones más habituales de los marcadores suelen ser la cadera y la rodilla, seguidos del tobillo y el sacro, ya sea unilateral o bilateral. En algunos casos se pueden añadir otros marcadores como son los del torso, brazos o cabeza.

La disparidad en el uso de los OMSs basados en marcadores como sistema de medida principal frente a como sistema de referencia para el etiquetado, indica que la mayor parte de la investigación sobre estos sensores ya se

ha llevado a cabo y el consenso es bastante claro, por lo que cada vez más se busca indagar las posibilidades que el resto de sensores ofrecen en comparación con el, a día de hoy indiscutible, *gold standard* en espacios cerrados del OMS basado en marcadores. Es evidente que la investigación directa sobre los OMSs prosigue, si bien cada vez se centra más en la combinación de estos con otros sensores como las unidades de medida inercial o las plataformas dinamométricas (Tabla 1.3).

- Sin marcadores

Los OMSs sin marcadores emplean el mismo fundamento que aquellos basados en estos, pero suelen hacer uso de la luz visible y no se basan en marcadores externos para detectar los puntos clave en la imagen, sino que los extraen bien mediante la extracción de un modelo 3D, bien mediante la implementación de algoritmos basados en herramientas de tratamiento de imágenes, como pueden ser filtros u operadores morfológicos. El paso de un planteamiento centrado en el sensor a uno sin sensores basado en algoritmos de procesado de imagen, les ofrece ciertas ventajas frente a los OMSs basados en marcadores. Algunas de las principales son una mayor comodidad para el sujeto de estudio, una implementación más simple, rápida y flexible y una mayor adaptabilidad a distintos entornos de captura de imagen sin necesidad de una extensa calibración (Moro et al. 2022; Kruk y Reijne 2018). Otra de las ventajas respecto a los OMSs basados en marcadores es que, a pesar de contar con cámaras con menor rango de medida (unos 5.5 m desde la lente), pueden llegar a ofrecer un mayor rango operacional gracias a la mayor facilidad para mover las cámaras, especialmente si se emplean dispositivos como los *smartphones*, aunque dependerá del FOV y la profundidad que pueda detectar la cámara, así como de la vida de la batería (entre 1 y 8 horas). Además, son dispositivos con una vida de servicio considerable y son más asequibles pudiendo ir desde los 0 € si se emplea un *smartphone* propio hasta los 400 €/ud si se buscan comprar cámaras comerciales especializadas. Como mínimo se emplean entre 1 y 4 cámaras aunque lo ideal serían 8 dispositivos para poder mantener la línea de visión que sigue siendo necesaria (Tabla 1.3). Sumado a estas ventajas, hay que tener en cuenta que este tipo de sistemas, virtualmente, tienen nula invasividad sobre el sujeto ya que no es necesario que este interactúe con sensor alguno (Figura 1.3). Del mismo modo, el no contar con marcadores que colocar, elimina el error humano de la medida. Esto no significa que el diseño experimental no influya en los resultados, pero sí que excluye las desventajas del uso de marcadores (Moro et al. 2022; Kruk y Reijne 2018).

Sin embargo, la exactitud de estos dispositivos no es tan elevada como los OMSs basados en marcadores con errores entre los 5 y los 50 mm en función de la distancia a la cámara y la morfología del fondo de la imagen. Por otro lado, las frecuencias de muestreo de las cámaras empleadas en un OMS sin marcadores son inferiores a las de las cámaras para la detección de marcadores; entre 30 y 120 Hz es común en la mayoría de aplicaciones. Junto a esto, un OMS sin marcadores necesita del desarrollo de un software propio por parte del usuario para lograr detectar los puntos clave del sujeto en la imagen adaptando el *software* al entorno experimental concreto (Tabla 1.3). Sin embargo, es esta flexibilidad la que precisamente confiere a estos sistemas todos los aspectos positivos previamente mencionados (Kruk y Reijne 2018). Asimismo, a la hora de emplear esta técnica, es importante poner especial atención en eliminar o reducir las interferencias por radiofrecuencias, interferencias por infrarrojos (sobre todo en caso de emplearlos en combinación con la luz visible) y reflejos en la superficie observada. Se debe considerar, también, que cualquier obstrucción de la cámara puede errar la medida, al igual que el seguimiento de objetos muy claros u oscuros o, en general, muy similares con el fondo de la imagen (Menolotto et al. 2020).

Aún más evidente que en el caso de los OMSs basados en marcadores, la literatura del uso de estos sistemas sin marcadores para el análisis de marcha indica una clara tendencia hacia la detección de la cinemática del movimiento, seguida de otros propósitos como el análisis espacio-temporal o la detección de las fases de la marcha (Tabla 1.3). En otros casos, estos sistemas se pueden emplear también para el modelado y la simulación biomecánica. Por otro lado, aunque en este caso no se empleen marcadores físicos, de alguna forma los puntos de interés representan marcadores virtuales, los cuales se suelen colocar normalmente en los mismos puntos que los OMSs basados en marcadores (i.e., cadera, rodilla, tobillo y sacro) y, frecuentemente, se añaden el torso, los brazos y la cabeza.

El volumen de estudios encontrados que emplean esta tecnología es relativamente bajo respecto al resto de sensores. Sólo un 7.9 % de los estudios se valieron de un OMS sin marcadores para analizar los aspectos de la marcha y sólo 3 de estos los emplearon exclusivamente (Moro et al. 2022; Gu et al. 2018; André et al. 2020); el resto utilizaron también datos de unidades de medida inercial para obtener los resultados (Tabla 1.3).

Actualmente, Microsoft $Kinect^{TM}$ Xbox $360^{(R)}$ continua siendo el OMS más empleado para la captura sin marcadores gracias a su bajo coste, capacidad de captura del cuerpo completo, uso en tiempo real y legado, poniendo en entredicho al método tradicional basado en marcadores. Poco a poco los nuevos algoritmos o sensores implementados buscan compensar las carencias de este sistema (e.g., baja sensibilidad a movimientos finos), las cuales dan lugar a errores superiores al 5 % en la captura cinemática (Pfister et al. 2014) dificultando su implementación en la práctica clínica donde el error máximo aceptable es de entorno al 2 %. Entre los más exitosos algoritmos de

estimación de posición se encuentran OpenPose y PoseNet (Nakano et al. 2020. La ventaja que ofrecen es que potencian los puntos débiles del uso de OMSs sin marcadores mejorando la precisión (a unos 5-50 mm de error) y mejoran los puntos fuertes en su aplicación mediante cámaras de dispositivos *smartphones*, permitiendo lograr una mejor representación de los patrones de movimiento del individuo al estar en un entorno familiar (Lam, Tang y Fong 2023). Del mismo modo, una de las limitaciones de los OMSs basados en marcadores actuales es que, por encima de una frecuencia de muestreo de 1000 Hz, el FOV se vuelve significativamente estrecho, dificultando todo tipo de medidas de movimientos amplios o rápidos. Por contra, los OMSs sin marcadores, dado que están basados en software, son capaces de superar esta limitación (Nakano et al. 2020).

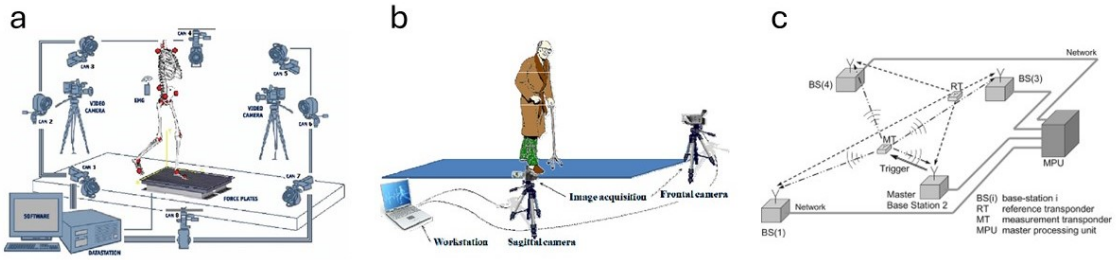


Figura 1.3: Sistemas de captura cinemática (a) OMS basado en marcadores, (b) OMS sin marcadores, (c) EMS (Stelzer, Pourvoyeur y Fischer 2004; Kovač, Medved y Ostojić 2010)

Sistemas electromagnéticos de captura cinemática

Existe un tercer tipo de sistema de captura cinemático conocido como sistema de medida electomagnético (EMS) que emplea las distorsiones de un campo electromagnético de baja frecuencia (radiofrecuencia) para localizar los segmentos en movimiento (Rahul 2018). La posición es hallada por medio del registro de los tiempos de vuelo de las ondas electromagnéticas desde el transmisor hasta el receptor colocado sobre el cuerpo del sujeto similar a un marcador (Stelzer, Pourvoyeur y Fischer 2004). Son sistemas que utilizan una mayor frecuencia de muestreo que los anteriores aunque siendo menos precisos que los OMSs basados en marcadores, pero más precisos que los OMSs sin marcadores. Además, se ven afectados por las interferencias electro y ferromagnéticas y el ruido sobre la medida crece rápidamente con la distancia (Kruk y Reijne 2018). Si bien el EMS no es un sistema que se emplee tan a menudo como los antedichos, es interesante comentarlo pues permite la captura de movimientos dinámicos en grandes volúmenes espaciales sin necesidad de mantener la línea de visión sobre los puntos de interés (Figura 1.3).

Sistemas de medida inercial

Un tipo cada vez más común de sistema de medida cinemática dado su reducido tamaño, bajo peso y simplicidad de uso, son los sistemas de medida inercial (IMS) tales como acelerómetros, giroscopios e, incluso, magnetómetros. Colocados sobre los segmentos corporales de interés, estos sistemas permiten tomar medidas fundamentales de cada uno en un espacio tridimensional como son la aceleración lineal, velocidad angular y orientación respecto al campo magnético terrestre (Prasanth et al. 2021).

Las mediciones de los IMS se ven afectadas por diferentes ruidos en la señal que pueden ser deterministas (o sistemáticos) o estocásticos. Los primeros incluyen el sesgo constante del sensor y errores de factor de escala, mientras que los segundos incluyen el ruido blanco termo-mecánico y la inestabilidad temporal del sesgo (Aggarwal 2010; Woodman 2007). La corrección del ruido sistemático es bastante sencilla y normalmente hay alguna indicación del fabricante. Por ejemplo, para corregir el sesgo constante se puede utilizar una calibración inicial en un entorno estático (i.e., con valor esperado nulo) para estimar y así compensar este valor (Pasciuto et al. 2015). De manera similar, el factor de escala (de no ser proporcionado por el propio fabricante) se puede extraer comparando la salida del sensor ante una entrada conocida con la salida teórica (Ferraris, Grimaldi y Parvis 1995). Por otro lado, el ruido estocástico es más difícil de compensar ya que es una cuestión de probabilidad que depende de factores como el instante de tiempo, la presión, las deformaciones mecánicas, las interferencias electromagnéticas, la humedad y, sobre todo, la temperatura (Qi et al. 2022). El ruido blanco es una señal aleatoria que contiene muchas frecuencias con intensidades iguales, por lo que se considera un proceso de paseo aleatorio (o *random walk*) de media cero con una desviación estándar proporcional a la raíz cuadrada del tiempo (Elbes, Al-Fuqaha y Rayes 2012; Titterton y Weston 2004). Además, el la inestabilidad del sesgo (también llamada velocidad o tasa

de deriva) proviene del sesgo residual que permanece sin compensar tras la calibración, dando lugar también a un error que varía con el tiempo (Woodman 2007).

Es habitual estimar otros parámetros cinemáticos como la posición, la velocidad o la orientación relativa mediante derivación o integración de la señal original, por lo que cualquier ruido estocástico se acumula con el tiempo afectando considerablemente a estas pseudo-medidas (Qi et al. 2022; Thong et al. 2002). Por ende, eliminar o reducir el ruido estocástico suele ser el principal foco de atención en los IMS con enfoques desde más generales hasta más específicos del sensor. Un ejemplo común de enfoque general es modelarlos como un proceso de paseo aleatorio gaussiano, donde se asume que el ruido que afecta a estos dispositivos sigue dicha distribución y que la desviación estándar del ruido aumenta de forma lineal con el tiempo («IEEE standard specification format guide and test procedure for single-axis interferometric fiber optic gyros» 1998; Pasciuto et al. 2015). Sin embargo, cabe la posibilidad de que no se logran capturar completamente las características de este ruido, lo que sugiere que enfoques más complejos y específicos de sensores pueden ser más adecuados para resolver este problema.

- Acelerómetros

Los acelerómetros fueron una de las primeras alternativas y continúan siendo de las más ampliamente utilizadas en el análisis de la marcha humana. Estos sensores de escaso tamaño y excelente movilidad permiten medir la aceleración lineal o fuerza específica de un cuerpo por un bajo coste haciendo accesible el análisis de la marcha humana a muchos más investigadores (Figura 1.4(a)). Además, sirviéndose de la aceleración como únicos datos de trabajo, es posible obtener una mayor granularidad (i.e., la escala o nivel de detalle de una medida), tal como pasar de un modelo con las dos fases básicas de la marcha (ST y SW) a un modelo que contemple cuatro o incluso más fases dependiendo del algoritmo de clasificación que se emplee (Taborri et al. 2016).

Sin embargo, al tratarse de un sensor inercial, se debe considerar el ruido estocástico al momento de obtener la velocidad lineal o posición del sensor integrando o doble integrando la señal (Qi et al. 2022). Con el paso del tiempo han ido surgiendo varios planteamientos que han tratado de corregir este error en la medida de los acelerómetros. Algunos de los más notables son emplear el acelerómetro como un inclinómetro, utilizar el análisis de frecuencia por transformada Wavelet, implementar un algoritmo de actualización de velocidad cero o aplicar todo tipo de filtros, especialmente paso-alto (Bai et al. 2022; Djurić-Jovičić, Jovičić y Popović 2011). No obstante, estos métodos dejan que desear en términos de precisión si los comparamos con otros sensores. Recientemente, Cui et al. 2024 implementaron un algoritmo de fusión basado en la combinación de unidades recurrentes cerradas, un perceptrón multicapa y un mecanismo de atención, logrando reducir las desviaciones en la medida un 96.11 % y avivando las esperanzas sobre el uso independiente de los acelerómetros.

Además, en cualquier instancia en que se empleen acelerómetros es necesaria la compensación de la aceleración gravitatoria en caso de desear obtener el valor de la aceleración libre. Otro aspecto a tener en cuenta sobre los acelerómetros es la necesidad de calibración para corregir el sesgo como efecto del posicionamiento del sensor previamente a tomar cualquier medida (Thong et al. 2002). De la misma forma, existe la necesidad de un buen pre- y post-procesado de la señal para eliminar ruidos y artefactos debido a las vibraciones producidas durante el transcurso de la marcha y suavizar la señal, lo que aumenta el coste computacional (Taborri et al. 2016).

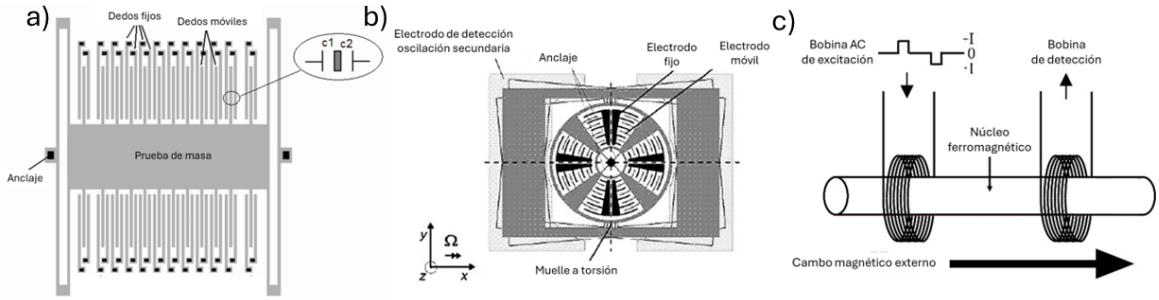


Figura 1.4: Diagramas básicos de (a) acelerómetro capacitivo, (b) giroscopio capacitivo, y (c) magnetómetro *fluxgate* (Sharma et al. 2008; Liu et al. 2009; Miles 2017)

- Giroscopios

Pese a que los acelerómetros pueden ser empleados como inclinómetros, la forma más común de medir orientaciones son los giroscopios (Figura 1.4 (b)). Estos sensores inerciales permiten tomar una medición directa de la velocidad angular del segmento corporal donde se coloquen y han ido ganando popularidad en las últimas décadas debido a sus ventajas. La primera es que los entornos dinámicos o las vibraciones sobre el sensor no influyen sobre la estabilidad de la señal. Además, al medir rotaciones, los giroscopios son menos sensibles a su colocación en el segmento que los acelerómetros, dando más 'manga ancha' en este aspecto (Taborri et al. 2016).

Finalmente, un aspecto de interés en estos sensores es que son menos sensibles a la acción de fuerzas externas (i.e., aceleraciones lineales) tales como las fuerzas de contacto con el suelo o la gravedad, siendo especialmente cierto cuanta mayor sea la velocidad de giro (Usubamatov 2019; Usubamatov 2020). Esto se debe al conocido como 'efecto giroscópico' que reduce la precesión del IMS por fuerzas externas en proporción a la velocidad angular. No obstante, por pequeño que sea, el efecto de las aceleraciones lineales externas se debe tener en cuenta en la calibración y corregir el error de deriva si se desean obtener medidas lo más exactas posible.

Algunas de las estrategias comúnmente planteadas para solucionar los errores estocásticos en los giroscopios incluyen la estimación del error (i.e., in-estabilidad de la frecuencia) por medio de la varianza Allan (o varianza de dos muestras) (Diao et al. 2013; Yafei, Xizheng y Yijie 2007), la estimación de los coeficientes de la ecuación de un modelo lineal dependiente de la temperatura para predecir el error en función de esta (Prikhodko, Trusov y Shkel 2013), la integración de la señal acotada por umbrales adaptativos (Hoang y Pietrosanto 2021) o el uso de resonadores ópticos con 2 o más modos de resonancia (Howard, Behbahani y M'Closkey 2018) entre otras (Mansoor et al. 2019).

- Magnetómetros

A pesar de no ser un IMS como tal, los magnetómetros constituyen un complemento fundamental tanto para acelerómetros como para giroscopios. Existen aquellos capaces de detectar cambios en la magnitud del campo magnético terrestre o aquellos capaces de detectar cambios en la orientación del sensor respecto al vector del campo magnético terrestre. En el campo del análisis de la marcha humana, se emplean mayoritariamente los últimos (Edelstein et al. 2008). En concreto, los más habituales son aquellos basados en el efecto Hall (i.e., generación de un voltaje Hall), la magnetorresistencia (i.e., cambios en la resistencia del material magnetorresistivo) o el efecto *fluxgate* (Figura 1.4 (c)) (i.e., cambios en la permeabilidad) (Emmerich y Schofthaler 2000; Yeh, Duan y Chung 2019). Sin embargo, la necesidad de reducir, e idealmente eliminar, el consumo energético de estos sensores hace que el uso de materiales piezoeléctricos sea cada vez más atractivo (Yeh, Duan y Chung 2019).

Estos pequeños dispositivos son capaces de proveer información de la orientación y la dirección del movimiento en ausencia de señal GPS o demás sistemas de geolocalización (Bennett et al. 2021). No obstante, su frecuencia de muestreo es menor comparados con otros IMSs y son extremadamente sensibles a las interferencias electro y ferromagnéticas (Ning et al. 2021). Utilizados independientemente su precisión puede dejar que desear para movimientos rápidos en comparación con acelerómetros o giroscopios, pero no poseen ruido estocástico que genere error de deriva con el tiempo. Dado que esto compensa las características de los dos anteriores, es habitual emplearlos en métodos de fusión con estos otros sensores.

- Unidades de medida inercial (IMUs)

Las técnicas de fusión de datos son ampliamente empleadas con estos sensores dada la gran potenciación de sus rendimientos compensando mutuamente sus carencias. El pragmatismo del concepto y las mejoras técnicas en el diseño y fabricación de los sistemas micro-electromecánicos (MEMS) han dado lugar al advenimiento de la unidad de medida inercial (IMU) que conocemos actualmente (Figura 1.5).

Como se verá más adelante en la Sección 1.4.1, la fusión de estas variables inerciales, logra una evaluación de los eventos de la marcha más robusta a ruidos, interferencias y artefactos derivados de las distintas patologías neuro-motrices. Sin embargo, no todos los todos los sensores y/o ejes necesitan ser empleados al mismo tiempo, se pueden tomar para el análisis aquellas combinaciones que no se vean afectadas; y no todos las IMUs que existen poseen acelerómetro, giroscopio y magnetómetro. Los primeros modelos utilizaban exclusivamente acelerómetro y giroscopio, cada uno con 2 o 3 grados de libertad (DOFs) para un total de 4 a 6 DOFs (Figura 1.5), mientras que los modelos más contemporáneos ya añaden el magnetómetro a la mezcla y la mayoría cuentan con 9 DOFs (Ahmad et al. 2013). Contar con los tres sensores en un mismo sensor en lugar de tenerlos separados, reduce el tiempo de desfase en un orden de magnitud hasta unos 10 ms, arroja un mayor flujo de información sobre la medida (e.g., la dirección del movimiento) y mejora la precisión de esta corrigiendo el ruido estocástico y el error de deriva asociado a través de una mejor calibración y técnicas de fusión de datos (e.g., filtro Kalman) (Taborri et al. 2016). De este modo, es posible lograr análisis cinemáticos más exactos y llevar a cabo análisis espacio-temporales (e.g.,

longitud, anchura y cadencia de paso). Por contra, al incluir el magnetómetro, la IMU de 9 DOF es más susceptible a las interferencias electro y ferromagnéticas, pudiendo verse afectada por un mayor grado de ruido, y necesita de una calibración más meticulosa que una IMU de 6 DOF (Zhu y Zhou 2004). Dado que ambas poseen sus fortalezas y debilidades, aún es común el uso de ambas según el contexto de aplicación.

Las IMUs comerciales como el Xsens MTW Awinda o la gama de Xsens MTi-6x0 logran exactitudes elevadas dentro de los sensores portátiles con un error angular máximo entre 0.2 y 1.5 ° y un error de desplazamiento máximo de entre 50 y 100 mm. Su coste, entre 50 y 325 €/ud., es relativamente bajo y asequible, y se requieren tan solo 2 unidades para tomar una medida. Se deben calibrar siempre antes de cualquier medida para garantizar la mayor precisión posible, pero una vez hecho su implementación es muy simple y ocupan el mínimo espacio del laboratorio. Además, muchas cuentan con sistemas de sujeción $Velcro^(R)$ que minimizan la incomodidad del sujeto que realiza la prueba. Por otro lado, una de las limitaciones de un OMS basado en marcadores es que necesitan una continua línea de visión de estos, haciéndolo sensible a cambios de iluminación y a oclusiones, así como limitando el volumen de captura. Una ventaja fundamental de emplear una IMU es que no hace falta preocuparse tanto por la línea de visión, haciendo que el volumen de captura pueda ser tan grande como se desee, siempre y cuando el dispositivo se mantenga dentro de un rango de unos 50 m desde la estación de medida y la vida de la batería (i.e., 6 horas aproximadamente) lo permita. Conjuntamente, la vida de servicio de una IMU es alta y, aunque lo normal es emplear entorno a 120 Hz, pueden llegar a muestrear a frecuencias de entre 40 y 400 Hz según el número de sensores y el modelo (Tabla 1.3).

En contraposición, aunque lo habitual es que estos sensores traigan incorporado un software comercial para un preprocesado automático básico de la señal (incluyendo la fusión de datos), es necesario que el usuario incorpore un algoritmo propio para procesar la señal si se pretenden emplear en aplicaciones específicas como el análisis de la marcha humana. A parte, en todo uso que se pretenda hacer de una IMU, es imperativo contemplar las posibles fuentes de ruido como las perturbaciones por interferencias electro y ferromagnéticas cerca de equipos electrónicos o materiales ferromagnéticos, por variaciones de temperatura o por variaciones de presión atmosférica (Tabla 1.3). Si bien suelen ser corregidas con sensores complementarios, especialmente en las IMU con 9 DOF, se debe revisar pues no siempre será el caso.

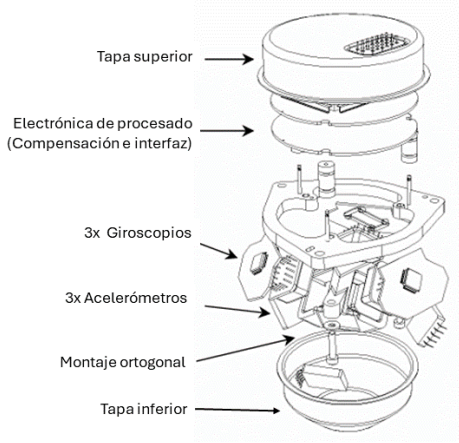


Figura 1.5: Unidad de medida inercial con 6 DOFs (NATO 2017)

En lo referente al campo de la marcha humana, las IMUs son, sin duda, los sensores más populares en estos últimos años protagonizando el 81.6 % de los 76 artículos revisados, en 47 de los cuales (61.8 %) fue el único sensor empleado (e.g., Chinimilli et al. 2019; Sharifi-Renani, Mahoor y Clary 2023; Liu et al. 2023b). En la mayoría de casos en que se emplearon en conjunción con otros sensores, estos fueron OMSs, sensores de fuerza resistivos o sistemas de electromiografía. Las aplicaciones más comunes de una IMU para el análisis de la marcha son la detección de las fases (GPD) y la detección de los patrones de marcha (LMD). Además de estas también es común el uso de IMUs para la predicción de la cinemática (GKP) y la cinética (GXP). Otros casos minoritarios incluyen la detección de la cinemática (GKD), el análisis de parámetros espacio-temporales (STA), la predicción de las fases (GPP) y los patrones (LMP) y la detección y/o predicción de las transiciones de un patrón a otro (LTD y LTP). Asimismo, los lugares más habituales para colocar una IMU en este campo son la pierna, el muslo y el pie, por orden, seguidos de la pelvis (sacro). Alternativamente, existen otros puntos complementarios donde a veces se colocan como el tobillo, la rodilla, la cadera, el torso o el brazo (Tabla 1.3).

1.3.2 Sistemas de Captura Cinéticos

Plataformas dinamométricas

Una plataforma de fuerza (FP) o dinamométrica es una clase particular de transductor de fuerza capaz de convertir las fuerzas y momentos aplicados a una placa en diferencias de potencial que se amplifican y registran digitalmente para obtener los parámetros cinéticos específicos (Figura 1.6) como la fuerza de reacción con el suelo (GRF), la distribución de presiones y magnitudes derivadas (i.e., el centro de presión (CoP) y/o el centro de masa (CoM)) o los momentos respecto al centro del plano (Taborri et al. 2016). Toda placa de este tipo consta bien de 1 sensor central, bien de 3 sensores dispuestos triangularmente, o bien de 4 sensores repartidos por las esquinas, donde estos sensores pueden ser capacitivos, piezoeléctricos o piezorresistivos (i.e., galgas extensiométricas). De estos, los piezoeléctricos poseen una excelente linealidad en los resultados y mayor sensibilidad, pero pecan de necesitar un mayor espacio para ser implementados en la FP. Por contra, las galgas extensiométricas se pueden fusionar a la superficie de la plataforma, ocupan un menor espacio y proveen de resultados absolutos y más estables a largo plazo, lo que las ha convertido con el tiempo en la opción más escogida en el diseño de FPs (Silva, Moreira y Rocha 2017).

El error de las FPs comerciales se encuentra entre los 0.25 y 1 N, permitiendo un nivel de exactitud en las medidas cercano al de los OMSs. El coste de estas plataformas va desde los 3000 hasta los 20,000 €/ud. que es superior a los vistos previamente, pero sólo se suelen emplear 1 o 2 unidades en total. Añadido a esto, las FPs permiten trabajar con frecuencias de muestreo entre los 10 y hasta 50,000 Hz, poseen una vida de servicio considerablemente alta y, debido a su naturaleza, no es necesario mantener una línea de visión constante sobre los puntos de interés (i.e., los pies en este caso). Sumado a esto, la mayoría de FPs se comercializan junto a un software que procesa automáticamente los datos de los 3 o 4 sensores de las plataformas, facilitando notablemente la interpretación y el procesado necesarios) Además, si bien la mayoría de modelos son fijos (e.g., Kistler 9281E y AMTI Optima-BMS), también existen modelos portátiles que pueden cambiarse de lugar bajo ciertas premisas (e.g., BiosignalsPlux). Por lo general se asumen que el volumen de captura se corresponde con el rango de medida que, a su vez, se corresponde con el área de la plataforma la cual oscila entre los 0.45x0.45 m^2 y 1x1 m^2 (Tabla 1.3). Estos valores suelen ser suficientes para aplicaciones simples en análisis de marcha que requieran unos pocos pasos, pero en casos más complejos donde haya que analizar un mayor número de pasos consecutivos, normalmente se emplean otro tipo de sensores portátiles y/o con mayor volumen de captura. Alternativamente, la instalación de una única FP bajo el raíl de una cinta de correr es una buena estrategia para aumentar indefinidamente su volumen de captura (aunque sea en una sola dirección) y poder aprovechar al máximo este excelente sensor.

Por otra parte, el sujeto de ensayo puede sentir una ligera incomodidad puesto que el tener que pisar en las placas hará que ponga especial atención en el espacio que delimitan y reajuste en mayor o menor medida sus movimientos para atenerse a este (Figura 1.6). El cambio de las propiedades de la superficie de contacto pie-suelo es otro aspecto que puede influir sobre la marcha normal del sujeto, alterándola, por lo que tanto las propiedades de la placa como las de la suela del calzado empleado se deben tener en cuenta. Además de esta limitación, el uso de toda FP conlleva una responsabilidad sobre el tratamiento del ruido estocástico, el cual normalmente se cuela en las medidas de las FPs originado por interferencias electromagnéticas, térmicas, ruido mecánico o vibraciones, artefactos de movimiento presentes en la marcha o el fenómeno de diafonía (o *cross-talk*) (Tabla 1.3).

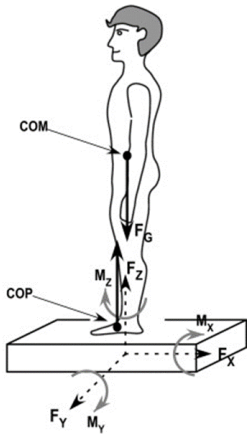


Figura 1.6: Medidas cinéticas de una plataforma dinamométrica (Latash 2012)

Aun siendo un sensor con tales prestaciones, su presencia en la literatura revisada es mejorable cuanto menos, logrando aparecer únicamente en un 4 % de las entradas. Además sólo 1 de los 76 artículos incluidos utilizó una FP como único sensor (Jani et al. 2022). A pesar de que su uso en la práctica clínica continua siendo crucial y frecuente, da la impresión de que hay una tendencia, similar a los OMSs basados en marcadores, en que el foco en la investigación de este campo cada vez más se dirige hacia los sensores portátiles dadas las potenciales ventajas que ofrece una valoración menos constreñida y/o un seguimiento extra-clínico (Mason et al. 2023). Por otra parte, dentro de los usos más comunes para una FP en el análisis de marcha, la predicción cinemática es la más empleada, seguida de la predicción cinética y la detección del patrón de marcha (Tabla 1.3). Asimismo, todos los casos recogidos que no emplearon únicamente FPs, las combinaron con OMSs basados en marcadores para resolver las carencias de información de la cinemática global que los primeros presentan por sí solos.

Sensores de fuerza resistivos

Los sensores de fuerza resistivos (FSR) consisten en una fina lámina de poliéster con un material piezorresistivo semiconductor (e.g., silicio policristalino) o conductor (e.g., nanotubos de carbono) en su interior (Figura 1.7) de modo que, ante una fuerza aplicada, se produce un incremento en la resistencia inversamente proporcional a la magnitud de esta (Shaikh, Salcic y Wang 2015; Yuan et al. 2010; Amjadi, Kim y Park 2014). Se trata de dispositivos con una sensibilidad muy alta ante cualquier esfuerzo con error bajo respecto a la magnitud (entre 65 y 245 nN), pero con error alto en la posición de la fuerza (entre un $pm2$ % y $pm3$ %), por lo que, al igual que las IMUs, será necesaria una calibración previa a cualquier uso (Hirasawa, Okada y Shimojo 2008; Nadvi et al. 2012). No obstante, su vida de servicio es alta (i.e., entre 1 y 10 millones de ciclos de histéresis) y su velocidad de respuesta es también excepcionalmente alta, tomando tan solo entre 1 y 2 ms en generar un resultado (Rana 2009). Asimismo, al no necesitar mantener constantemente la línea de visión y ser dispositivos pasivos (i.e., no necesitan batería) y portátiles, logran un rango de medida y un volumen de captura extremadamente elevados, únicamente limitados por la batería del almacenamiento de memoria portátil donde se guarden los datos. Sumado a esto, el precio de los modelos comerciales es considerablemente inferior al resto de sensores mencionados hasta el momento rondando entre los 4 y 30 €/ud. para un mínimo de 1 unidad (e.g., gama 400 de Interlink electronics o modelo A502 de Tekscan) (Tabla 1.3). Otro aspecto positivo de los FSRs es que su frecuencia de muestreo es continua dada su naturaleza por lo que la frecuencia de muestreo del conversor analógico digital (ADC) será el factor más limitante en este aspecto.

Aunque presentan aspectos muy positivos, emplear FSRs requiere que el investigador sea capaz de elaborar un software propio para el preprocesado y procesado de la señal en bruto ya que lo normal es que no se incluya con el propio producto. Otra cuestión relevante es que, aunque no necesitan instalarse en el laboratorio, siguen teniendo un efecto sobre el individuo de magnitud similar o ligeramente menor al de las FPs pues, si bien no modifica la superficie del suelo, sí modifica el calzado pudiendo hacer que la marcha normal del sujeto de ensayo se vea ligeramente afectada. Asimismo, la integridad de la interfase mecánica puede verse dañada dada la elevada susceptibilidad al desgaste de estos sensores, mas este se debe procurar reducir al mínimo para garantizar el menor error posible (Shaikh, Salcic y Wang 2015). Por otro lado, se deben considerar y tratar las interferencias electromagnéticas, los cambios de humedad, las deformaciones mecánicas y el ruido mecánico (i.e., vibraciones) como fuentes de ruido estocástico en este tipo de sensores (Tabla 1.3).

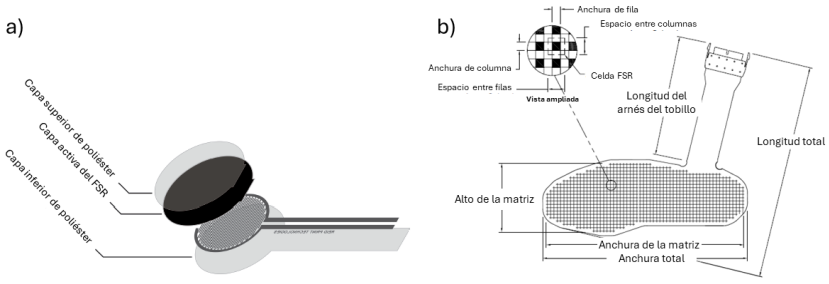


Figura 1.7: Sensor de fuerza resistivo (a) simple, y (b) en matriz de sensor de presión de plantilla (Tekscan)

Finalmente, dentro de los artículos revisados en la Tabla 1.3, estos sensores suponen el 13.2 %, alcanzando el tercer mayor porcentaje de uso tras las IMUs y los OMSs basados en marcadores con hasta 4 artículos que hacen uso exclusivo de FSRs para extraer los datos (Yang et al. 2019; Drouot y Jarrassé 2022; Michał et al. 2017; Jiang et al. 2018); y en los casos restantes se combinan con IMUs o electromiografía. Asimismo, la implementación más común y sencilla es en la suela del zapato y los usos principales son la GPD, seguida de la LMD y, en casos menos comunes, la GKD y la GXP.

- Sensor de presión de plantilla

En buena parte de los casos, los FSRs son aplicados al análisis a través del diseño y fabricación de sensores de presión de plantilla (IPSs). EL diseño de estos a menudo se basa en colocar una membrana elastomérica para repartir mejor la presión bajo la cual se sitúan FSRs en la configuración del puente de Wheatstone de 1, 2 o 4 galgas para mitigar el ruido térmico y aumentar la sensibilidad a la presión y la linealidad de la respuesta, manteniendo un amplio rango de medida y un bajo coste relativo (Yuan et al. 2010; Amjadi, Kim y Park 2014; Hoffmann 1974). Este tipo de sensores pueden contar con 64 o hasta 966 de estos bloques de medida (Figura 1.7). Para que funcione debidamente, es crucial que los 4 FSRs estén en una posición que les permita resistir una cantidad similar de presión, que su sensibilidad a la presión y temperatura sean similares y que tengan un nivel de error sistemático parecido (Yuan et al. 2010). Por otro lado, aunque menos comunes, también existen otros IPSs tales como los basados en sensores de fuerza capacitivos, compuestos por una capa de dieléctrico entre un par de electrodos, y los basados en sensores optoelectrónicos, compuestos por una matriz de pares LED-fotodiodo conectada a una membrana de elastómero. Los primeros cuentan con una buena linealidad y menor histéresis que los FSRs, pero suelen estar limitados en sus sensibilidad y rango de respuesta (Lipomi et al. 2011; Wei y Xu 2015). Los segundos presentan mejor sensibilidad, pero están limitados en el rango de respuesta y su error de medida es directamente proporcional a la velocidad de la marcha (Martini et al. 2020; Wei y Xu 2015).

En cuanto a sus características, dado su diseño, el IPS comparte buena parte de estas con el FSR, pero hay matices que varían. Por ejemplo, aunque los hay que utilizan un sistema de almacenamiento de memoria portátil como los FSRs, la mayoría de los nuevos IPSs emplean trasmisión inalámbrica para guardar la información en el propio ordenador directamente. Esto facilita su uso, pero limita el rango de medida a unos 10 m desde la estación. Además, la vida de su batería suele durar entre los 90 y 120 minutos y suelen operara a frecuencias de muestreo entre 100 y 500 Hz. Por otro lado, su precio se encuentra en el rango de los 25 a 1,250 €/ud. aunque aquí dependerá del número y tipo de sensores y de la calidad de los materiales empleados en la suela (i.e., 1 unidad). Sumado a esto, la mayoría de IPSs comercialmente disponibles incluyen un software de preprocesado e, incluso, procesado de la señal, simplificando su uso para los análisis más básicos (e.g., F-Scan 64 2.0 y F-Scan Go de Tekscan). Finalmente, comparado con una FP que obtiene una medición del valor total de la fuera de contacto, el IPS (o un FSR por extensión) propicia medidas puntuales del contacto pie-suelo, logrando una mayor precisión en la GPD (Taborri et al. 2016).

- Interruptor de pisada

A pesar de que los IPSs obtienen un registro continuo y con buena resolución espacial de la presión en la planta del pie, existen sensores más simples y prácticos para la GPD como los interruptores de pisada (FSW). Estos interruptores se pueden basar en el mismo principio de funcionamiento que los IPSs (e.g., FSRs, capacitivos u optoeléctricos) o en métodos mecánicos y su salida es binaria, indicando únicamente si hay contacto o no. Gracias a su simplicidad, son de los más empleados como 'verdad fundamental en GPD (41.6 % de los estudios que reportaron una referencia basal en la Tabla 1.3). Además, son accesibles, su precisión es excelente y el preprocesado necesario es nulo o mínimo. Sin embargo, el número de fases de la marcha que son capaces de detectar es limitado pues no pueden detectar las sub-fases del SW y la precisión de la detección es inversamente proporcional al aumento de granularidad (i.e., número de fases). Sumado a esto, los FSWs no son capaces de aislar la fuerza relacionada con la marcha frente a la relacionada con el movimiento del CoM y su vida de servicio es reducida (Taborri et al. 2016).

1.3.3 Sistemas Complementarios

Electromiografía

A parte de los sistemas de medida mencionados, en varios casos se ha incluido a modo de complemento la electromiografía (EMG) (e.g., Su, Smith y Farewik 2020; Hu et al. 2021; Park y Kim 2022). Se trata, efectivamente, de una técnica útil para la GPD basada en la actividad muscular de los miembros inferiores, la cual se ha comprobado que sigue un patrón periódico a lo largo de la marcha (Ivanenko, Poppele y Lacquaniti 2004; Taborri et al. 2016). Por medio de la EMG se pueden llegar a identificar hasta 8 fases de la marcha, lo que supone la máxima granularidad recogida en la literatura hasta la fecha (Joshi, Lahiri y Thakor 2013), y permite reducir el lapso de tiempo entre la intención y la acción en aplicaciones de control de prótesis y órtesis al tratar directamente con la actividad muscular. Asimismo, son sensores portátiles, con un rango de medida de entre 20 y 40 m desde la estación, con entre 4 y 8 horas de autonomía de la batería, una frecuencia de muestreo alta (máximo 4,000 Hz), con errores muy bajos y con un precio de entre 1,500 y 6,000 €/pack. Por contra, la variabilidad intersujeto de la EMG y la posición de los electrodos pueden alterar la medida y comprometer la repetitibilidad de la metodología y, además, son altamente sensibles a defectos en la interfase, variaciones de humedad, sudoración, artefactos de movimiento, interferencias electromagnéticas y diafonía. Por todo esto, la EMG no es tan común como el resto de sensores en el análisis de marcha. Aún así, los sensores junto a los que se ha empleado en la literatura revisada han sido IMUs y/o FSRs (e.g., Su, Liu y Gutierrez-Farewik 2021; Krausz y Hargrove 2021).

Sensores de proximidad

Para terminar esta sección, en determinados casos se emplean sensores poco frecuentes como los sensores de proximidad, los cuales normalmente se basan en el tiempo de vuelo para detectar si hay y a qué distancia o ángulo se halla cierto obstáculo. Algunos ejemplos serían los sensores basados en ultrasonidos o los sensores basados en láser (Sahoo et al. 2020).

Tabla 1.3: Características de los sistemas de captura de movimiento en el análisis de la marcha humana.

	Sensores						
	IMU	**Ópticos con marcadores (MK)**	**Ópticos sin marcadores (ML)**	**Plataformas de fuerza (FP)**	**Sensores de fuerza (FSR)**	**Sensores de presión (IPS)**	**EMG**
Exactitud (RMS)	Alta (entre 0.2 y 1.5 ° o entre 50 y 100 mm)	Muy alta (0.1 mm o 0.1 °); depende del número y posición de las cámaras	Media (entre 5 y 50 mm); depende de la distancia a la cámara, fondo...	Muy alta (entre 0.25 y 1 N)	Limitada/baja en posición (entre ±2 y ±3 %); Alta en magnitud (entre 65 y 245 nN)		Muy alta (<1 mV)
Implementación	Calibración previo uso; Implementación muy simple	Múltiples calibraciones anuales según uso; Implementación compleja	Múltiples calibraciones según el uso; Implementación simple	Calibración mensual/anual; implementación compleja	Calibración previo uso; implementación simple		Calibración previo uso; implementación tediosa
Volumen de captura	Depende del rango	Entre 9x9x6 m^3 y 45x45x6 m^3; depende del tipo, número y posición de las cámaras, así como del tipo de marcador	FOV: 90(°) x 60(°) ; 5.5 m profundidad	Entre 0.45x0.45 m^2 y 1x1 m^2	Depende del rango		Depende del rango
Coste	Entre 50 y 325 €/ud; Mín 2 uds.	Entre 20 y 11,000 €/ud.; Mín 6 uds.	Entre 0 y 400 €/ud; Mín. 1 ud.	Entre 3,000 y 20,000 €/ud; Mín. 1 ud.	Entre 4 y 30 €/ud; Mín 1 ud.	Entre 25 y 1,250 €/ud; Mín 1 ud.	Entre 1,500 y 6,000 €/pack; Mín 1 pack
Preprocesado	Automático en general; pueden hacer falta filtrado, segmentación, sincronización, fusión de datos y/o corrección de la deriva	Automático en general; pueden hacer falta filtrado, segmentación, sincronización, etiquetado de marcadores, relleno de huecos y/o reconstrucción de la trayectoria	Por cuenta propia en general; pueden hacer falta filtrado, segmentación, reconstrucción de superficies y/o reconstrucción de la trayectoria	Automático en general; pueden hacer falta filtrado, segmentación, y/o algoritmos matemáticos	Por cuenta propia en general; pueden hacer falta filtrado, segmentación, y/o interpolación	Automático en general; pueden hacer falta filtrado y/o segmentación	Automático en general; pueden hacer falta filtrado, segmentación, rectificación, normalización, corrección de la línea base y/o transformadas de Fourier
Invasividad (individuo)	Mínima (Velcro)	Mínima (Pegatina)	-	Media (Plataforma)	Media (Calzado)		Mínima (Electrodo)
Invasividad (laboratorio)	Mínima (Estación)	Alta (entre 6 y 12 cámaras es normal)	Media (entre 1 y 4 cámaras, lo ideal son 8)	Alta (entre 1 y 2 es lo normal)	-		Mínima

Tabla 1.3 *Cont.*

	IMU	**Ópticos con marcadores (MK)**	**Ópticos sin marcadores (ML)**	**Plataformas de fuerza (FP)**	**Sensores de fuerza (FSR)**	**Sensores de presión (IPS)**	**EMG**
Linea de visión necesaria	No, pero puede reducir el rango	Sí	Sí	No	No		No
Portátil	Sí	Limitada	Sí (smartphone)	Limitada	Sí		Sí
Rango de medida	Hasta 50 m desde la estación	Hasta 90 m desde la cámara	Hasta 5.5 m desde la cámara	La plataforma o lo que permita el raíl en caso de ser móvil	Ilimitado	Hasta 10 m desde la estación	Entre 20 y 40 m desde la estación
Vida de la batería	∼6 h	Entre 12 y 72 horas para los activos; los pasivos no tienen	Entre 1 y 8 horas	12 h (sólo si es portátil)	Entre 10 y 40 horas (por el dispositivo de memoria)	Entre 1.5 y 2 horas	Entre 4 y 8 h
Vida de servicio	Alta	Alta	Alta	Alta	Alta; entre 1 y 10 millones de ciclos		Alta
Frecuencia de muestreo	Entre 40 y 400 Hz; normalmente no más de 120 Hz (según número y modelo)	Entre 100 y 1000 Hz; depende de la resolución y el FOV	Cualquier rango: Entre 30 y 120 Hz es lo habitual	Entre 10 y 50,000 Hz; normalmente entre 10 y 100 Hz.	Contínua	Entre 100 y 500 Hz máximo	4,000 Hz máximo
Software	Software comercial básico y/o propio según necesidad	Software comercial completo	Software propio	Software comercial completo	Software propio	Software comercial completo	Software comercial básico
Fuentes de ruido e interferencias del medio	Perturbaciones ferromagnéticas, variaciones de temperatura y variaciones de presión atmosférica; normalmente corregidas con sensores complementarios	Interferencias por exceso de iluminación y alta sensibilidad a vibraciones o movimientos de la cámara	Interferencia por radiofrecuencias, interferencias por infrarrojos y reflejos en la superficie observada	Interferencia electromagnética, variaciones de temperatura, ruido mecánico o vibraciones, artefactos de movimiento, diafonía (cross-talk)	Interferencia electromagnética, cambios de humedad, deformación mecánica, ruido mecánico/vibraciones		Interferencia electromagnética, variaciones de temperatura, variaciones de humedad, artefacto de movimiento, diafonía (cross-talk)
Otras limitaciones	Error de deriva y falta de medida global de la posición (sólo para 6 DOF)	Obstrucciones de la cámara o del márcador	Obstrucciones de la cámara y seguimiento de objetos claros u oscuros	Influencia del calzado	Dependencia de la interfase mecánica, susceptibilidad elevada al desgaste		Resolución espacial limitada e influencia de la fatiga muscular, las diferencias anatómicas intersujeto y la colocación en la medida

Tabla 1.3 *Cont.*

	IMU	Ópticos con marcadores (MK)	Ópticos sin marcadores (ML)	Plataformas de fuerza (FP)	Sensores de fuerza (FSR)	Sensores de presión (IPS)	EMG
Aplicaciones en análisis de marcha	[1] GKD - 3.2 % [2] GKP - 17.7 % [3] GXP - 9.7 % [4] STA - 3.2 % [5] GPD - 61.3 % [6] GPP - 3.2 % [7] LMD - 45.2 % [8] LMP - 3.2 % [9] LTD - 3.2 % [10] LTP - 1.6 %	[11] GKP - 60 % [12] GXD - 20 % [13] GXP - 20 % [14] GPD - 20 % * Modelado y simulación biomecánica	[15] GKD - 100 % [16] GPD - 25 % [17] STA - 50 % * Modelado y simulación biomecánica	[18] GKP - 66.6 % [19] GXP - 33.3 % [20] LMD - 33.3 %	[21] GKD - 10 % [22] GXP - 10 % [23] GPD - 70 % [24] LMD - 60 %		[25] GKP - 40 % [26] GXP - 60 % [27] GPD - 40 % [28] LMD - 60 % [29] LTP - 20 %
Colocación	Brazo: 2 Torso: 6 Pelvis: 13 Muslo: 42 Pierna: 44 Pie: 38 Cadera: 2 Rodilla: 4 Tobillo: 5	Pelvis: 1 Muslo: 4 Pierna: 3 Pie: 1	Pelvis: 1 Muslo: 4 Pierna: 3 Pie: 1	-	Pierna: 1 Tobillo: 1 Suela/Pie: 9		-
Sinergias	MK, ML, FSR, EMG	IMU, FP	IMU	MK	IMU, EMG		IMU, FSR
Porcentaje de estudios	Solo: 61.8 % Mix: 81.6 %	Solo: 1.3 % Mix: 6.6 %	Solo: 3.9 % Mix: 7.9 %	Solo: 1.3 % Mix: 4 %	Solo: 5.26 % Mix: 13.2 %		Solo: 0 % Mix: 6.6 %

[1]Chinimilli et al. 2019; David Li y Hsiao-Wecksler 2013

[2]Chinimilli et al. 2019; Sharma y Rombokas 2022; Karakish, Fouz y ELsawaf 2022; Nozaki y Watanabe 2019; Park y Kim 2022; Shariffi-Renani, Mahoor y Clary 2023; Romero-Hernandez, Lope Asiain y Grana 2019; Renani et al. 2021; Jung et al. 2021; Yi et al. 2021; Zhang et al. 2022

[3]Chinimilli et al. 2019; Park y Kim 2022; Yi et al. 2021; Zhang et al. 2022; David Li y Hsiao-Wecksler 2013; Zhang et al. 2023a

[4]Weigand et al. 2022; Wang et al. 2021

[5]Sharma y Rombokas 2022; Sanchez Manchola et al. 2019; Zhao et al. 2016; Karakish, Fouz y ELsawaf 2022; Maqbool et al. 2016; Zhen, Yan y Kong 2020; Wei et al. 2023; Zhen, Kong y Yan 2020; Schuy et al. 2015; Arshad et al. 2022; Su, Liu y Gutierrez-Farewik 2021; Wahjudi y Lin 2019; Park y Kim 2022; Yu et al. 2023; Hu et al. 2021; Song, Fernandes y Nordin 2023; Weigand et al. 2022; Yan et al. 2020; Romijnders et al. 2022; Ding et al. 2018a; Jung et al. 2021; Grimmer et al. 2019; Zhang et al. 2023b; Lu et al. 2023; Liu et al. 2023a; Yang et al. 2023; Su, Smith y Farewik 2020; Ding et al. 2018b; Nguyen, La y Duong 2015; Novak et al. 2013; Gonçalves et al. 2018; David Li y Hsiao-Wecksler 2013; Liu et al. 2023b; Xu et al. 2023; Huo et al. 2018; Zhang et al. 2023a; Cheng, Bolívar-Nieto y Gregg 2021; Martínez-Hernandez, Rubio-Solis y Dehghani-Sanij 2018

[6]Yu et al. 2023; Martinez-Hernandez, Rubio-Solis y Dehghani-Sanij 2018

[7]Prigent et al. 2023; Brard et al. 2022; Yu et al. 2022; Nozaki y Watanabe 2019; Semwal, Gupta y Lalwani 2021; Wahjudi y Lin 2019; Gao et al. 2019; Park y Kim 2022; Yu et al. 2023; Hu et al. 2021; Weigand et al. 2022; Bartlett y Goldfarb 2018; Chattopadhyay y Nandy 2018; Potluri et al. 2019; Oh y Hong 2023; Ershadi et al. 2022; Sherratt, Plummer e Iravani 2021; Zhang et al. 2023b; Shakya, Taparugssanagorn y Silpasuwanchai 2023; Wang et al. 2021; Chen et al. 2021a; Novak et al. 2013; David Li y Hsiao-Wecksler 2013; Liu et al. 2023b; Zhang et al. 2023a; Cheng, Bolívar-Nieto y Gregg 2021; Krausz y Hargrove 2021; Martínez-Hernandez, Rubio-Solis y Dehghani-Sanij 2018

[8]Chinimilli et al. 2019; Schuy et al. 2015

[9]Chinimilli et al. 2019; Oh y Hong 2023

[10]Su, Liu y Gutierrez-Farewik 2021

[11]Renani et al. 2021; Mundt et al. 2020; Sunny et al. 2023

[12]Dinovitzer, Shushtari y Arami 2023

[13]Mundt et al. 2020

[14]Grimmer et al. 2019

[15]Gu et al. 2018; André et al. 2020

[16]André et al. 2020

[17]Gu et al. 2018; André et al. 2020

[18]Mundt et al. 2020; Sunny et al. 2023

[19]Mundt et al. 2020

[20]Jani et al. 2022

[21]David Li y Hsiao-Wecksler 2013

[22]David Li y Hsiao-Wecksler 2013

[23]Drouot y Jarrassé 2022; Yang et al. 2019; Novak et al. 2013; Michal et al. 2017; David Li y Hsiao-Wecksler 2013; Cheng, Bolívar-Nieto y Gregg 2021; Jiang et al. 2018

[24]Potluri et al. 2019; Novak et al. 2013; David Li y Hsiao-Wecksler 2013; Sahoo et al. 2020; Cheng, Bolívar-Nieto y Gregg 2021; Krausz y Hargrove 2021

[25]Park y Kim 2022; Yi et al. 2021

[26]Su, Liu y Gutierrez-Farewik 2021; Park y Kim 2022; Hu et al. 2021

[27]Park y Kim 2022; Yi et al. 2021

[28]Park y Kim 2022; Hu et al. 2021; Krausz y Hargrove 2021

[29]Su, Liu y Gutierrez-Farewik 2021

1.4 ESTADO DEL ARTE: TÉCNICAS DE DETECCIÓN Y/O PREDICCIÓN

Llegados a este punto, queda clara la importancia de elegir el sensor y, por lo tanto, los datos adecuados, pero es, como mínimo, de igual importancia el tratamiento que hagamos a estos datos si el objetivo es lograr resultados óptimos. Por esto, en esta Sección se discutirán las técnicas más empleadas en el análisis de la marcha humana, cubriendo su principio de funcionamiento, ventajas y desventajas, aplicaciones más comunes, datos más empleados, algoritmo de entrenamiento más habitual y precisión de cada una. Antes de comentar los métodos, empero, se explicarán brevemente el preprocesado típico de las señales y las técnicas de fusión de datos.

1.4.1 Técnicas Complementarias

Preprocesado de la señal

Todas las señales que se utilicen en los distintos métodos descritos aquí, pasan por un preprocesado que, más o menos exhaustivamente según el autor, prepara los datos para el modelo de clasificación (i.e., detección) o regresión (i.e., predicción).

La primera de las tareas del preprocesado consiste en calibrar los sensores a través de una medición o patrón de referencia para obtener las matrices de rotación, desfases iniciales y demás parámetros de calibración; y, en caso necesario, sincronizar o alinear los datos de los distintos sensores. La eliminación de valores atípicos o perdidos también es uno de los primeros pasos del preprocesado aunque no siempre se implementa, al menos explícitamente. Las estrategias habituales son eliminar estos valores (Mundt et al. 2020; Jani et al. 2022) y, a veces, remuestrearlos (i.e., interpolación) (Yang et al. 2019).

En segundo lugar, es muy habitual emplear segmentación por ventana deslizante (SWS) con o sin solapamiento para segmentar los datos. Se emplean todo tipo de estrategias para determinar el ancho de la ventana como emplear un valor constante, utilizar la información de otro sensor o usar un valor adaptable a la función, segmentando los intervalos entre las intersecciones de la función con el eje de abscisas o segmentando de tal forma que se maximice la auto-correlación entre segmentos.

A continuación, el siguiente paso suele consistir en la eliminación del ruido y el suavizado de la señal a través de la umbralización, habitualmente heurística (i.e., manual); y el filtrado, principalmente por medio de filtros de tipo Butterworth (i.e., cero desfase) desde orden 2 hasta 5, aunque lo habitual es orden 4. Alternativamente, se emplean filtros de paso banda, de promediado, de mediana, exponenciales, etc. Otro tipo de filtros como los filtros de Kalman, Madwick o Mahony son empleados, pero su uso suele estar restringido mayoritariamente a la fusión de datos a nivel de sensor en los IMUs (ver Sección 1.4.1).

En cuarto lugar, según el modelo puede ser beneficiosa la extracción de características a través del cálculo de estadísticos descriptivos (e.g., media, mediana, varianza, kurtosis etc.), aplicación de matemática vectorial (e.g., cálculo de la norma, del valor resultante, del valor cuadrático medio, etc.), cálculos cinemáticos por derivadas o integrales (e.g., posición, orientación, velocidad, aceleración, etc.) y cálculos cinéticos (e.g., fuerza total, CoP, momentos, etc.).

Seguidamente, en caso de que se empleen señales con diferente resolución en el eje temporal (e.g., IMU y EMG), o bien la señal con menor fs se sobremuestrea interpolando sus valores a la fs de la señal con mayor fs o esta última se submuestrea a la fs de la primera diezmando sus valores.

Finalmente, otro paso importante y muy habitual en el preprocesado es la normalización de las variables, que se puede llevar a cabo por medio de la Z-score o en base al valor máximo y mínimo absoluto de la señal, seguida de la concatenación de los vectores de variables de el/los sensores empleados. Asimismo, en algunos casos se incluye un paso intermedio de selección de características antes de la concatenación que puede emplear técnicas como las de reducción de la dimensionalidad (e.g., PCA y LDA), el método de filtro o el método de envoltura.

Fusión de datos

La marcha humana es un proceso tan complejo que son casos singulares los que logran un discernimiento completo de sus características por vía de un único sensor. Para conseguir un modelo lo suficientemente relevante para analizar debidamente un proceso de este estilo hace falta un mayor número de DOFs, los cuales no se obtienen sino por medio de incrementar el número de fuentes de las que recabar información y ser capaz de integrarla, es decir, por medio de un proceso conocido como fusión de datos (Lahat, Adali y Jutten 2015).

Como concepto general, la fusión de datos no es un algoritmo de detección y/o predicción; más bien consiste en un proceso automático o semi-automático donde se asocian, refinan, combinan y transforman los datos de una o más fuentes (e.g., un sensor) y/o instantes de tiempo hasta conseguir una representación más completa del sistema analizado que permita un apoyo consistente y efectivo a la toma de decisiones humana o automática (i.e., detección y/o predicción) (Meng et al. 2020).

Según la clasificación de Luo y Kay 1988, existen cuatro principales niveles de fusión de datos: señal, píxel, característica y símbolo. De estos, señal y característica suelen ser los niveles más comunes en el análisis de la marcha, mientras que píxel y símbolo suelen ser menos empleados o, en el caso del primero, más destinados a un tipo concreto (i.e., OMS). El nivel de señal se centra más en la combinación de los datos en crudo que en la transformación de estos. En este nivel se incluyen las técnicas como la alineación o sincronización de la señal (espacial o temporal), los filtros de fusión (e.g., Kalman, Madwick, Mahony) y el promediado de la señal segmentada. Es habitual previamente emplear técnicas de preprocesado como la calibración de los sensores y el filtrado y suavizado de la señal. El nivel de píxel toma la misma idea que el nivel de señal, pero la aplica a los píxeles de las imágenes. El nivel de característica está más enfocado a transformar o crear nuevas características a partir de las existentes. En este nivel se incluyen las técnicas de fusión como algunos métodos de extracción de características (e.g., análisis de componentes principales, PCA), la concatenación de vectores de características y algunos métodos de ensamblaje (e.g., promediado, *bagging*), aunque estos últimos no se suelen considerar. Es habitual emplear algún tipo de preprocesado previo a este nivel como podría ser normalizar las características, eliminar valores faltantes y/o atípicos, la extracción de características (e.g., estadísticos descriptivos) y la selección de características (e.g., método de filtro y método de envoltura). Por último, el nivel de símbolo (también llamado nivel de decisión) ayuda a integrar la información categórica de las clasificaciones obtenidas (i.e., símbolos) para optimizar la toma de decisiones (Luo y Kay 1988). En este nivel, tienen sentido técnicas de voto mayoritario y toma de decisiones consensuada y no suele haber preprocesado dado que trata con decisiones de los clasificadores empleados, no con datos ni características.

En la mayor parte de casos, la fusión de datos se realiza tras hacer las medidas. Sin embargo, un notable ejemplo visto en la Sección 1.3.1 son las IMUs, que incorporan la fusión de datos a nivel de señal directamente a nivel de *hardware* en el sensor, agrupando en un único tándem de información a acelerómetro, giroscopio e, incluso, magnetómetro para reducir el ruido y mejorar la precisión recíprocamente. Sumado a esto, también se vio en las secciones anteriores relacionadas con los sensores que muchos autores los combinan habitualmente para tratar de mejorar sus resultados; la información cinemática de las propias IMUs se puede fusionar, por ejemplo, con la información cinética de los IPSs o de los FSWs para esgrimir un algoritmo de GPD más robusto y preciso, aumentar la granularidad y evitar valores atípicos contrastando los resultados de ambos (Taborri et al. 2016).

Gracias a la fusión de datos, se consigue una aproximación unificada, global y más exacta a la respuesta del sistema analizado (e.g., la marcha humana) y se logra una mayor valía y calidad de los datos, mejorando la calidad de las decisiones tomadas. Asimismo, este tipo de estrategia facilita hacer pruebas exploratorias con distintas combinaciones de sensores hasta encontrar la más apropiada para la aplicación concreta (Lahat, Adali y Jutten 2015). Por otro lado, la fusión de datos se emplea ante sistemas complejos como la marcha humana, donde hay un considerable número de variables que se desconocen, por lo que hay que tener claro de antemano el número, tipo y extensión de problemas que se desean abordar.

No todos los casos en que se emplee la fusión son garantía de éxito; es necesario que el usuario que lo implementa sea lo suficientemente ducho para plantear la fusión de datos al nivel correcto y de la forma adecuada y no empeorar el rendimiento del modelo (Lahat, Adali y Jutten 2015). Al llevar a cabo la fusión de datos es capital contar con sensores no redundantes, evitar inconsistencias en las medidas, la buena calidad de los datos, ordenarlos y sincronizarlos correctamente, decidir el/los niveles de fusión e integrar debidamente la información de todos los sensores evitando el conflicto en los datos (Meng et al. 2020).

En el análisis de la marcha humana, el concepto de fusión de datos a nivel de señal ha aparecido en casi todos los artículos revisados en el preprocesado de la señal (Sección 1.4.1), mas la fusión a nivel de características y/o decisiones no es tan común. Ejemplos de la aplicación de la fusión a nivel de características son la fusión de características por concatenación (Krausz y Hargrove 2021), vía red neuronal (Zhen, Kong y Yan 2020; Sharma y Rombokas 2022), a través de una estructura salto-conexión (Zhen, Yan y Kong 2020) o mediante análisis de correlación canónica (Hu et al. 2021).

En cuanto a sensores se refiere, combinaciones sensores como son la IMU (tomando 6 DOFs), el OMS sin marcadores y/o la EMG son la elección preferida de aquellos autores que se aventuran con este desafío. El foco central de la mayoría de estudio fue la GPD, pero también hubo espacio para la GKP y la LMD. El procedimiento más común es preprocesar los datos de entrada siguiendo lo descrito en la Sección 1.4.1, aplicar la fusión de datos y entrenar el modelo correspondiente (generalmente de *deep learning* o DL) por medio del descenso de gradiente estocástico (SGD) con o sin criterio de parada temprana (ESC). La precisión lograda aplicando la fusión de datos es del 96.7 % (Tabla 1.4), calculada como la mediana de los resultados de los estudios revisados, situando el rendimiento de la fusión de datos en novena posición por debajo de las máquinas de soporte de vectores, lo cual, si bien es un excelente resultado, dista mucho del potencial que posee la aplicación de esta estrategia y revela cuán importante es poner atención a los detalles de la fusión.

Sistemas de inferencia difusa

La lógica difusa es una herramienta matemática que permite manejar información incierta, imprecisa o poco detallada, representando los valores numéricos exactos (’*crisp*’) por medio de múltiples conceptos lingüísticos (conjuntos difusos). Esto se lleva a cabo, por medio de las denominadas funciones de pertenencia que mapean cada elemento en el dominio *crisp* a un valor (difuso) entre 0 y 1 (Zadeh 1988; Sabri et al. 2013).

Un sistema de inferencia difusa (FIS) ese caracteriza por una serie de reglas difusas (IF-THEN) definidas, a través de conjuntos difusos, lógica difusa e inferencia difusa, empleando el conocimiento y la experiencia de expertos (humanos) en la materia (Figura 1.8). El mecanismo (deductivo) de inferencia difusa consiste en una ’fuzzificación’ de los datos de entrada, la evaluación y agregación de las reglas difusas y la ’defuzzificación’ de los resultados de la inferencia (Hellmann 2001). La fuzzificación es un proceso matemático que convierte cada elemento *crisp* de cada variable en un valor (grado) de pertenencia a los distintos conjuntos difusos (dominio difuso) de entrada. Tras fuzzificar las entradas, se evalúan las condiciones del antecedente (IF) de cada regla difusa en base a los grados de pertenencia de cada dato de entrada a los conjuntos difusos de entrada y las relaciones lógicas entre estos. En función del consecuente (THEN) de cada regla, se determina el grado de pertenencia de cada valor de la variable a los conjuntos difusos de salida. Una vez evaluadas las reglas, todos los conjuntos difusos de salida creados son combinados en un único conjunto difuso. El último paso de la inferencia es la conversión de este conjunto difuso global de salida a un resultado cuantitativo *crisp* mediante combinación matemática (e.g., por cálculo del centroide, por máxima pertenencia) (Sabri et al. 2013).

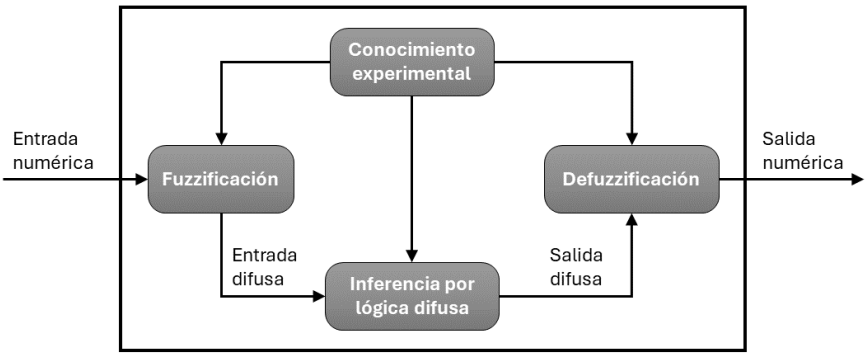


Figura 1.8: Esquema de funcionamiento de un sistema de inferencia por lógica difusa

Gracias a la consideración de las imprecisiones en las entradas y salidas del sistema en la formulación del propio FIS, este es capaz de modelar y razonar adecuadamente con la información imprecisa proporcionada, aumentando su robustez a datos poco precisos. Estos sistemas son especialmente útiles en el contexto de sistemas cuya respuesta es poco conocida o en situaciones donde una solución aproximada, pero rápida, es preferible (Ross 2009). En el marco del análisis de la marcha, los FISs rara vez se emplean. Aún así hay casos como Schuy et al. 2015 que entrenaron un FIS por medio de Levenberg- Marquardt mínimos cuadrados (Levenberg-Marquardt) empleando la información de velocidad angular triaxial de varias IMUs para lograr GPD y LMD con una precisión de 96.1 %. De forma similar, Michał et al. 2017 implementó una red neuronal basada en FIS que entrenó por retropropagación con descenso de gradiente estocástico (SGD) con los datos de un IPS para GPD con un error del 24.6 %. Esto demuestra que los FISs son efectivos en el análisis de marcha implementados por sí solos, pero aún se debe mejorar su implementación en el aprendizaje profundo.

1.4.2 Modelos Basados en Reglas

Modelos heurísticos

Los modelos heurísticos que emplean el método basado en reglas (RBM) son el método más simple, intuitivo y rápido para obtener, por ejemplo, la clasificación de las fases de la marcha o el patrón de marcha empleado. El principio fundamental de este método son los enunciados condicionales conectados entre sí. La implementación más habitual de este método es establecer uno o varios umbrales de decisión a los datos en crudo a través de enunciados lógicos *if-else* conectados vía operadores lógicos (e.g., AND o OR). Los FSWs vistos en la Sección 1.3.2 son un ejemplo perfecto de sensor donde se aplica directamente el RBM en el *hardware* para umbralizar la señal con uno o varios umbrales de decisión. El RBM también incluye técnicas de detección de picos, pero es más común emplear esta estrategia en el preprocesado de los datos (Chinimilli et al. 2019).

La puesta en práctica del RBM (e.g., Cheng, Bolívar-Nieto y Gregg 2021; Zhao et al. 2016; Maqbool et al. 2016) ha demostrado robustez y precisión especialmente cuando se trata de detectar las dos fases o eventos principales de la marcha (i.e., las fases ST y SW o los eventos HS y TO), pero también pueden volverse rápidamente complejos al aumentar la granularidad del análisis (e.g., Ding et al. 2018a; Drouot y Jarrassé 2022; David Li y Hsiao-Wecksler 2013).

De todos los métodos que aquí se van a presentar, el RBM, que apareció en 20 de los 76 estudios, es el más utilizado de todos. Además, todo tipo de sensores se pueden emplear con este; IMUs, OMSs con o sin marcadores, FSRs o IPSs, EMG y sensores de proximidad han sido empleados sólos o en combinación con el RBM, convirtiéndolo uno de los métodos más versátiles. Añadido a esto, en un 75 % de los casos fue usado en GPD y en un 15 % en LMD. En menor proporción se empleó también para LTD, STA, GKD y GXP. La precisión mediana lograda fue de 97.35 %, con un rango entre 63 % y 99.8 %, demostrando que, salvo algún caso concreto, los modelos basados en RBM logran un alto porcentaje de acierto en tareas de GDP (e.g., Schuy et al. 2015; Zhao et al. 2016; Maqbool et al. 2016) y LMD (e.g., Bartlett y Goldfarb 2018; Chen et al. 2021b; Oh y Hong 2023), al menos mientras la granularidad sea baja (Tabla 1.4).

Árboles de decisión

Un árbol de decisión (DT) es un modelo no paramétrico de aprendizaje automático supervisado que consiste en una estructura arboriforme capaz de clasificar un conjunto de muestras mediante una secuencia de enunciados condicionales o reglas sobre las variables independientes o características. La optimización de los umbrales de decisión de la secuencia se lleva a cabo con los datos de entrenamiento en base a medidas de la entropía relacionadas con la información o impureza asociada a las variables. Estos parámetros (e.g., ganancia de información o índice de Gini) se basan en la frecuencia y correlación de las variables. La construcción de un DT se realiza generando “ramas“ o divisiones basadas en reglas desde la “raíz“ o nodo inicial hacia las “hojas“ o nodos finales, donde la prioridad de inclusión de las variables en el modelo es inversamente proporcional a su entropía. En la práctica, a la hora de construir un modelo como este se deben considerar dos hiper-parámetros principales: el número de ramas (o el tamaño de hoja) y el nivel de poda o *pruning* (Iyer, Rajurkar y Gudivada 2020).

Existen varias aproximaciones para la implementación de los DTs como ID3, C4.5 o CART. El algoritmo C4.5 es una versión mejorada de ID3 donde se incorporan nuevos aspectos en la medida de ganancia de información basada en la entropía permitiendo el uso de variables dependientes e independiente continuas o discretas, el mejor manejo de datos perdidos y la reducción del sesgo hacia aquellas características con excesivas ramificaciones (i.e., sobreajuste) (Mienye, Sun y Wang 2019). Similar a C4.5, el algoritmo CHAID también ofrece divisiones múltiples en los nodos, pero no permite el uso de variables dependientes continuas al estar basado en el test Chi-cuadrado. La otra aproximación es el algoritmo CART, el cual se basa mayoritariamente en el índice de Gini para medir la impureza asociada a las variables y decidir los niveles por donde dividir y las reglas o umbrales de decisión. Al igual que C4.5, CART es capaz de operar con variables dependientes e independientes que sean continuas o discretas. Sin embargo, se diferencia también en que su división no es múltiple sino binaria. Un algoritmo que genera divisiones binarias como CART, pero que emplea Chi-cuadrado es QUEST (Song y Ying 2015).

Las principales ventajas del uso de un DT son la capacidad de simplificación de relaciones complejas entre las variables de entrada y salida, la facilidad de comprensión e interpretación de estas relaciones y del modelo en general, la aproximación no paramétrica (i.e., sin asumir una distribución estadística concreta), la eficacia en el manejo de datos sesgados sin necesidad de transformar los datos, el manejo eficaz de datos faltantes sin necesidad de imputar o eliminar y la robustez frente a valores atípicos (Mienye, Sun y Wang 2019; Song y Ying 2015). Por otro lado, los modelos de DTs son fáciles de sobre- o sub-ajustar, especialmente si se emplea un conjunto de datos pequeño en relación la complejidad del modelo; y son muy sensibles a la falta de causalidad entre variables

altamente correlacionadas que puede dar lugar a un modelo muy preciso, pero errado en su fundamento (Song y Ying 2015).

La aplicación de DTs al análisis de la marcha (Tabla 1.4) se centra en el algoritmo CART aplicado a datos de aceleración y velocidad angular en los tres ejes (i.e., 6 DOFs) recogidos de IMUs para LMD. Los resultados son mejores en la clasificación de los patrones de marcha que en la segmentación de estos con una precisión mediana de 84 % y un rango entre 55 % y 84 %. A nivel global, muestran potencial para clasificar los patrones de marcha, puede que incluso las fases, pero necesitan un mayor desarrollo y refinamiento para mejorar su precisión y consistencia.

1.4.3 Modelos de Estados

Modelos de máquinas de estado finitos

Una máquina de estados finitos (FSM) es un modelo determinista (aunque puede darse el caso de no serlo) donde diferentes estados están conectados entre sí por ciertos enunciados condicionales, de tal modo que el modelo sólo cambiará del estado actual a otro si se cumple la condición específica que los interrelacione. Asimismo, toda FSM debe cumplir la condición de que todos los estados deben estar conectados al menos a otro estado de modo que se pueda alcanzar cualquier estado existente independientemente del estado en que se inicie la máquina de cumplirse las condiciones adecuadas (Jagdale 2021). En esencia, una máquina de estados está formada por seis parámetros principales: el estado inicial, las variables de entrada, los estados internos, las funciones de transición, las funciones de emisión (salida) y las variables de salida. Sin embargo, al ser un modelo determinista, las funciones de salida no se suelen considerar en la FSM ya que las salidas, de haber, están predeterminadas directamente por los estados del modelo. De esta forma, el siguiente estado y la salida del modelo son únicamente dependientes del estado actual y la entrada (Barkalov et al. 2024; Bourlard y Bengio 2001).

La representación de una FSM habitualmente consiste en un gráfico o una tabla que represente las reglas de transición entre estados. En el gráfico los estados se representan como nodos o vértices y las relaciones interestados (transiciones) como arcos o líneas conectando dichos nodos (Jagdale 2021; Barkalov et al. 2024). Las ventajas de emplear el modelo de FSM son su simplicidad, modularidad, eficiencia y predictibilidad, todo en tiempo real. Por contra, es un modelo rígido en donde relaciones complejas entre variables son difícilmente incluidas, especialmente si se requieren decisiones basadas en el contexto, y pueden llevar a un crecimiento exponencial del número de estados o la complejidad de las funciones de transición.

Las máquinas de estado no se emplean tan a menudo en el análisis de marcha como en el ámbito de teoría del lenguaje (Bourlard y Bengio 2001). Tan sólo 2 de los 76 artículos revisados implementaron este modelo empleando alguna combinación de IMUs de 9 DOFs, OMSs sin marcadores, IPSs, EMG y goniómetros (Tabla 1.4). La aplicación mayoritaria en ambos casos fue la GPD, tomando las fases de la marcha como los estados del modelo y los datos de los sensores, las variables de entrada (observaciones). De este modo, se logró una granularidad de la marcha de hasta 8 fases (Drouot y Jarrassé 2022) con una precisión mediana del 95 % y un rango entre el 94 % y el 96 %.

Modelos ocultos de Markov

Un modelo oculto de Markov (HMM) es un modelo de aprendizaje automático empleado para el modelado estocástico de secuencias. Un HMM consiste en un modelo o cadena de Markov o, lo que es lo mismo, una FSM donde las transiciones entre estados siguen una función estocástica (i.e., probabilística) y en la que los estados no son directamente observables a través de las entradas, es decir, se encuentran “ocultos“ (Eddy 1996). Esto significa que se trata de una FSM doblemente estocástica donde no sólo las transiciones entre estados siguen una función estocástica, sino que las emisiones (salidas medibles) también lo hacen, por lo que deja de ser cierto que exista una única salida para cada estado ante una determinada entrada. En su lugar, cada estado produce una de las posibles salidas (símbolos) con una cierta probabilidad que depende de la función de emisión del estado actual del modelo que, a su vez, depende de la función de transición y las entradas del modelo (Bourlard y Bengio 2001). Además, al estar basado en una cadena de Markov, HMM también sigue denominada ’propiedad de Markov’ que implica que estos procesos “no tienen memoria“ en tanto que las transiciones de un estado a otro no dependen de estados pasados o futuros, sino exclusivamente del presente estado. Esto es especialmente cierto cuando se trata de un HMM de primer orden (Figura 1.9), pero puede dejar de serlo en variantes del mismo como los HMMs de orden superior que generalizan al HMM de primer orden y extienden la dependencia a los *n* estados anteriores (Mor, Garhwal y Kumar 2021).

Análogamente a la FSM, el HMM tiene como parámetros la función del estado inicial, las variables de entrada, los estados internos, las funciones de transición, las funciones de emisión (salida) y las variables de salida. Entrenar un HMM consiste en hallar las tres funciones mencionadas a partir de el algoritmo de Baum-Welch (BW), un

algoritmo de Esperanza-Maximización (EM) para hallar la estimación de los parámetros más probable a partir de las frecuencias observadas (MacKay 1997). Además, el modelo se puede entrenar considerando la probabilidad de ocurrencia de todas las posibles secuencias de estados o únicamente la secuencia más probable (i.e., algoritmo de Viertbi).

La flexibilidad, la fácil implementación, el manejo eficiente de secuencias temporales de datos de longitud variable y/o con relaciones complejas, hacen del HMM una opción ideal para aplicaciones de la vida real (Mor, Garhwal y Kumar 2021). Sin embargo, un modelo estadístico como HMM funciona mejor con series temporales discretas y estacionarias que con secuencias continuas y no estacionarias. La marcha humana como tal es un proceso continuo y no estacionario, pero al medir con un sensor estamos muestreándola, es decir, transformándola en una serie discreta. Por otro lado, aunque el aspecto continuo de la marcha se podría llegar a debatir bajo ciertas circunstancias como ir a velocidad constante (Luo y Kay 1988), en general, e incluso en el caso de intentar ir a velocidad constante, esto no se cumple dada la variabilidad y el dinamismo del movimiento. Sin embargo, aun en el caso de un proceso no estacionario, se puede aplicar HMM satisfactoriamente si se asumen las observaciones como un proceso estacionario a trozos. De este modo, este tipo de procesos, como la marcha, puede ser modelado como una sucesión de estados discretos estados con transiciones instantáneas entre ellos (Bourlard y Bengio 2001).

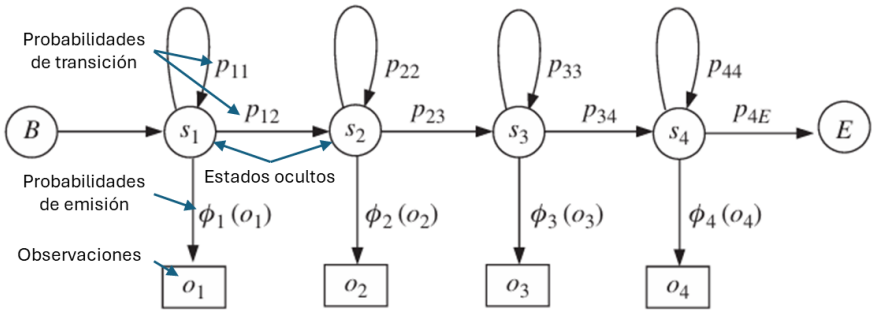


Figura 1.9: Modelo oculto de Markov de primer orden (Ibe 2013)

En la revisión de la literatura llevada a cabo, el concepto de HMM apareció en varias ocasiones; en concreto, se mencionaba en 6 artículos, tantos como la fusión de datos. Los autores emplearon principalmente IMUs de 9 DOFs sólos o junto a IPSs para obtener la GPP y/o la LTP según el caso, aunque GPP fue doblemente común. Por otra parte, la precisión mediana entre las distintas metodologías se halló en 81.44 % con un rango entre el 63 % y el 97.5 % (Tabla 1.4). Aunque la precisión mediana parece indicar una falta de rendimiento de HMM considerable en comparación con el resto de modelos, el rango nos muestra hay esperanza ya que realmente hay casos donde se lograron precisiones excelentes, que son los que se deben explorar (Martinez-Hernandez, Rubio-Solis y Dehghani-Sanij 2018).

1.4.4 Modelos de Aprendizaje Automático

El campo del aprendizaje automático (ML) se relaciona íntimamente con métodos estadísticos de aprendizaje vistos anteriormente, los cuales hacen uso de funciones paramétricas directamente estimadas a partir de los datos de entrada. En ML los modelos buscan, además, aprender las relaciones existentes entre los datos de tal forma que sean más eficientes a la hora de generalizar en situaciones noveles (Salcedo-Sanz et al. 2014).

Análisis discriminante lineal

Una de las técnicas más básicas y efectivas de ML supervisado es el análisis discriminante lineal (LDA). Se trata de una técnica estadística de subespacios que busca maximizar la ratio (discriminante) de Fisher o, lo que es lo mismo, maximizar la varianza inter-clase y minimizar la varianza intra-clase (Lee et al. 2009). Los clasificadores basados en LDA buscan hallar la frontera de decisión entre las distintas clases del conjunto de datos, la cual extraen de los datos de entrenamiento en forma de función discriminante (Bandos, Bruzzone y Camps-Valls 2009).

La desventaja de estos modelos es que son incapaces de operar con problemas mal planteados donde el número de características sea mayor que el número de muestras debido a que las matrices de las matrices de dispersión en que se basa el algoritmo de LDA estándar deben ser invertibles y, por lo tanto, no singulares (i.e., solución única) (Bandos, Bruzzone y Camps-Valls 2009). Además, en en el caso de necesitar aplicarlo a un problema de clasificación con múltiples etiquetas, el clasificador basado en LDA no es capaz de obtener regiones de decisión

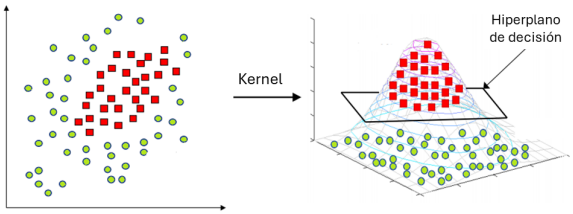


Figura 1.10: Efecto de la aplicación de un kernel sobre un conjunto de datos no separables linealmente

que no solapen, haciendo que las fronteras de decisión se vuelvan ambiguas (Wang, Ding y Huang 2010). No obstante, existen alternativas que tratan de poner solución a estos problemas con cierto grado de éxito (Wang, Ding y Huang 2010; Bandos, Bruzzone y Camps-Valls 2009; Fabiyi et al. 2021).

Entre los artículos revisados en el ámbito del análisis de la marcha humana hubo dos autores que emplearon un clasificador basado en LDA. Krausz y Hargrove 2021 empleó una combinación de los datos de IMUs de 9 DOFs, FSRs, EMG, goniómetros y OMSs sin marcadores para entrenar un clasificador basado en LDA con el objetivo de obtener la LMD y lo logró con un error máximo del 0.94 %. En comparación, Jiang et al. 2018 empleó únicamente los datos de miografía de fuerza de varios FSRs colocados en una banda del tobillo para entrenar el clasificador con el objetivo de lograr la GPD y su valor máximo del error fue del 10.9 % (Tabla 1.4). Esto nos puede indicar que este tipo de clasificación podría ser favorable en el análisis de la marcha para LMD, si bien no tanto para GPD, o al menos basándose en FSR de forma exclusiva. No obstante, si bien los resultados de Krausz y Hargrove 2021 son bastante buenos, la cantidad de datos de la que tuvo que proveer al clasificador puede ser un factor limitante a considerar de la metodología.

Máquinas de vectores de soporte

La máquina de vectores de soporte (SVM) es una ingeniosa solución del ML supervisado para aprovechar el buen rendimiento de clasificadores simples a datos complejos y no linealmente separables. Una SVM se define como un clasificador por maximización del margen inter-clases; ya que se fundamenta en utilizar la función de un hiper-plano como frontera de decisión lineal para maximizar la distancia entre las distintas clases presentes en los datos de entrada (Salcedo-Sanz et al. 2014). En esencia, SVM mapea (i.e., asigna) el vector de entradas a un nuevo espacio de características de alta dimensionalidad (Figura 1.10) por medio de algún mapeo no lineal elegido *a priori* conocido como 'kernel' (Cortes y Vapnik 1995). La elección del kernel es uno de los pasos fundamentales en toda SVM ya quede esto depende la separabilidad lineal de las clases en el nuevo espacio de características. Los kernels más comunes con el lineal, el polinómico y el exponencial, también conocido como función de base radial (RBF). Dada su simplicidad y estabilidad, la RBF es la preferida en la mayor parte de las aplicaciones. En realidad, la RBF (i.e., kernel exponencial) se puede aproximar por un kernel polinómico con infinitos DOFs aplicando la expansión de la serie de Taylor, lo que quiere decir que los datos son mapeados en un espacio dimensional infinito (i.e., continuo) (Salcedo-Sanz et al. 2014). Sumado al 'truco del kernel' para favorecer la separabilidad lineal de las clases (Figura 1.10), las SVM cuentan con otro aspecto que las hace muy eficientes, pues resulta que, para construir los hiper-planos que separen óptimamente las clases, no es necesario tener todos los puntos de los datos en cuenta, sino que basta con tomar los puntos más cercanos a la frontera de decisión, también conocidos como 'vectores de soporte' (Cortes y Vapnik 1995).

Al entrenar una SVM se buscan los coeficiente de la función de aquel hiper-plano capaz de separar los vectores de soporte y maximizar el margen, lo que implica que el valor esperado del error está acotado (i.e., tiene un máximo) debido a que el margen reduce la probabilidad de cometer errores de clasificación de nuevas muestras (Figura 1.11). Sin embargo, esto dependerá de la distribución de los datos y la selección y parametrización adecuada del kernel (Figura 1.10). Si sumada a la premisa antedicha se cumple que el hiper-plano pueda definirse a partir de un número reducido de vectores de soporte (i.e., el modelo captura eficientemente la estructura subyacente de los datos), la habilidad de la SVM de generalizar para cualquier dato será significativamente alta independientemente de las dimensiones del espacio al que se mapeen los datos,lo que incluye el caso de un espacio de infinitas dimensiones (e.g., RBF) (Cortes y Vapnik 1995). El entrenamiento de las SVMs se puede realizar priorizando la maximización del margen sobre cualquier otro parámetro, incluido el error de clasificación (SVM de margen duro) o incluyendo variables de holgura para controlar el balance entre maximización del margen y minimización error de clasificación (SVM de margen blando). En este segundo tipo, la relajación de la maximización del margen suele atribuirse al denominado (hiper-)parámetro C o factor de penalización, el cual es inversamente proporcional a la anchura del margen (en el espacio transformado) y al error de clasificación. Asimismo, un

valor elevado del hiper-parámetro C aumenta la complejidad de la frontera de decisión (en el espacio original) y la granularidad de la clasificación (Salcedo-Sanz et al. 2014). En resumen, un mayor valor de C reducirá el error disminuyendo el ancho del margen entorno al hiper-plano en el espacio de alta dimensionalidad, lo que hará que la frontera de decisión se vea mucho más compleja en el espacio original de los datos (i.e., más DOFs) y que se puedan identificar un mayor número de clases en la clasificación (i.e., mayor granularidad).

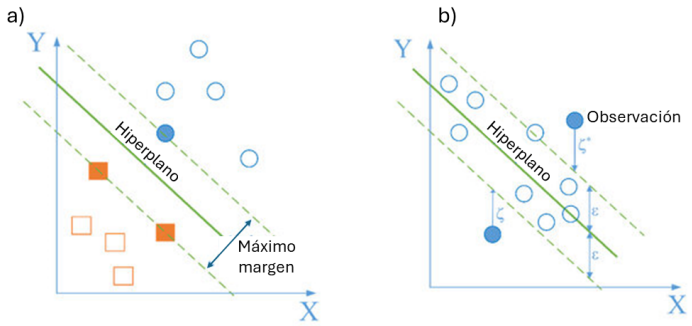


Figura 1.11: Clasificación en el espacio transformado de una máquina de vectores de soporte (Chou, Truong y Tsai 2021)

Las SVMs reducen la mayoría de problemas de ML a problemas de optimización, principalmente convexos. Además permiten una representación matemática clara, una intuición geométrica simple y un gran capacidad de generalización. No obstante, son muy sensibles al sobreajuste, en particular ante el uso de fronteras de decisión complejas (i.e., parámetro C demasiado elevado), conjuntos de datos desbalanceados y/o bajo tamaño muestral (Tian, Shi y Liu 2012).

En el análisis de marcha es común emplear las SVMs en combinación con otros algoritmos aunque existen casos que integran este modelo si mezclarlo con otros. Yang et al. 2019 empleó una SVM para la detección de de fases de la marcha con una precisión del 99 %. Por otro lado, Chinimilli et al. 2019 utilizó una SVM como clasificador de bajo nivel para la LMD, con una precisión del 95.2 %, combinando el resultado con un HMM como clasificador de alto nivel para obtener la LMP, con una precisión del 94.9 %. Wang et al. 2021 y Liu et al. 2023b emplearon un DT basado en SVM donde las reglas de decisión en los nodos del árbol dependían de la frontera de clasificación de la SVM y lograron precisiones del 100 % y 99.65 %, respectivamente. Todos los estudios mencionados, salvo Yang et al. 2019 que empleó un IPS, emplearon como datos la velocidad angular obtenida a partir de un IMU, si bien no todos emplearon los tres DOFs. Lo aquí mencionado nos indica que la SVM es una técnica útil en el análisis de marcha para la LMD y GPD con capacidad de alcanzar altos niveles de precisión combinada adecuadamente o por sí sola (Tabla 1.4).

Aprendizaje por ensamblaje de modelos

A menudo una de las estrategias de las que se hace uso en ML supervisado cuando se pretende mejorar el rendimiento de la clasificación sin aumentar drásticamente la complejidad de el/los modelo/s empleados son los métodos de aprendizaje por ensamblaje de modelos. Estos consisten en entrenar múltiples modelos para resolver un mismo problema y combinar sus decisiones individuales de algún modo para clasificar los nuevos datos que se presenten. Mediante esta técnica, se crea un clasificador 'fuerte' a partir de clasificadores 'débiles', mejorando su precisión y habilidad de generalización individual, siempre y cuando los clasificadores débiles sean diversos y su precisión sea superior al caso aleatorio (P>0.5). Si esto se cumple, el modelo ensamblado podrá clasificar erróneamente sólo si más de la mitad de los clasificadores débiles erran también, lo cual se demuestra que es como mínimo un orden de magnitud menos probable que el caso individual (Zhou 2012; Dietterich 2000).

Existen varios métodos para agregar las decisiones tomadas por los modelos básicos. Para tareas de clasificación se emplea voto o alguna variante de este como voto mayoritario o voto ponderado. Para tareas de regresión se emplea la media simple o la media ponderada. Según los clasificadores básicos empleados, los métodos de ensamblaje se dividen en homogéneos, si emplean el mismo tipo, o heterogéneos. Según como se ensamblen los clasificadores débiles, se distinguen el método de ensamblaje secuencial y el método de ensamblaje paralelo. El primero busca la dependencia entre los clasificadores básicos para mejorar el rendimiento reduciendo el error sucesivamente y el segundo busca la independencia entre los clasificadores básicos para reducir significativamente el error al combinarlos. Sumado a esto, el método de ensamblaje paralelo es inherentemente ideal para la computación en paralelo, permitiendo acelerar el proceso de entrenamiento (Zhou 2012).

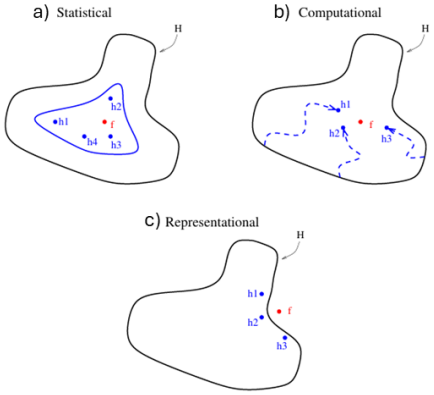


Figura 1.12: Representación gráfica del problema (a) estadístico, (b) computacional, y (c) de representación en aprendizaje automático (Dietterich 2000)

Entre las ventajas de aplicar la metodología de ensamblaje, una muy importante es que, por medio de la combinación, se pueden reducir la varianza y el sesgo de los algoritmos de ML individuales empleados en los clasificadores débiles (Zhou 2012). Esto es consecuencia de la liquidación de tres problemas claves en ML que son el problema estadístico, el problema computacional y el problema de representación (Figura 1.12). Al ensamblar se combinan hipótesis (de cada clasificador débil) con cierto grado de precisión y se logra una mejor aproximación de la verdadera hipótesis incluso cuando no hay datos suficientes para que un clasificador débil logre una buena aproximación por sí solo (i.e., problema estadístico, Figura 1.12(a)) (Dietterich 2000). Seguidamente, el iniciar el algoritmo de búsqueda local desde múltiples puntos (i.e., clasificadores débiles) permite lograr una minimización de la función de pérdida más robusta al estancamiento en un mínimo local (i.e., problema computacional, Figura 1.12(b)) (Zhou 2012). Y, por último, el método de ensamblaje hace posible expandir el espacio finito de funciones representables y representar la función verdadera que no sería representable de otra forma pues nada garantiza que toda la solución estuviera contenida en el espacio de funciones de un único clasificador por sí solo (i.e., problema de representación, Figura 1.12(c)) (Dietterich 2000).

- Boosting

Dos técnicas muy empleadas, *Boosting* y *Bagging*, pertenecen a los métodos de ensamblaje que modifican los datos de entrenamiento en cada iteración y además, por lo general, suelen funcionar mejor con clasificadores básicos que empleen algoritmos inestables (e.g., RBM, DT, redes neuronales). La primera, el *Boosting* (Figura 1.13(a)), consiste en obtener una predicción final de la combinación de los resultados de una serie de clasificadores básicos entrenados secuencialmente de tal forma que cada iteración el entrenamiento del nuevo clasificador básico prioriza los errores cometidos por los anteriores (Sagi y Rokach 2018).

Adaptative Boosting (AdaBoost) es el algoritmo de *Boosting* más representativo, el cual emplea un método homogéneo de ensamblaje y utiliza la función de pérdidas exponencial para minimizar el error. En cada iteración del entrenamiento, AdaBoost computa el error de la última hipótesis y actualiza una serie de pesos que impone sobre cada uno de las muestras de entrada, aumentando la ponderación de aquellas que hayan sido clasificadas incorrectamente (Dietterich 2000). Al finalizar, la decisión final se toma generalmente por voto ponderado en función de la precisión de cada clasificador débil durante el entrenamiento. Una gran ventaja de AdaBoost es su alta robustez al sobreajuste, pero también tiene el contra que es muy sensible al ruido dada la naturaleza exponencial de su función de pérdida que hace que las muestras con ruido (más probables a clasificarse erróneamente) sean las que más peso tengan en la ponderación. Algunas alternativas como MadaBoost o FilterBoost reducen el efecto del ruido mediante un límite superior para los pesos que puede ser rígido o suave (logarítmico) (Zhou 2012).

- Bagging

A diferencia del anterior, el método de *Bootstrap AGGregation* o *Bagging* remuestrea un subconjunto a partir del conjunto original de datos de entrenamiento extrayendo aleatoriamente cierto porcentaje de muestras(Figura 1.13(b)). Este remuestreo incluye, en promedio, el 63.2 % de las muestras originales al menos una vez y se lleva a cabo con reemplazo, es decir, se repiten las muestras (*Bootstrap*) para lograr un mismo tamaño muestral que el conjunto original y garantizar la efectividad y robustez del clasificador básico entrenado. Para la segunda parte, la agregación, se emplean los métodos más comunes de combinación: voto para los problemas de clasificación y promediado para los de regresión. Como consecuencia, especialmente en el caso de clasificadores débiles inestables, el *Bagging* genera inherentemente un efecto de suavizado al reducir considerablemente la varianza de los clasificadores débiles (Dietterich 2000).

Random Forest (RF) es el algoritmo de *Bagging* más representativo, el cual emplea un método homogéneo de ensamblaje y, al ser un método de *Bagging*, no utiliza función de pérdidas como tal, aunque evidentemente sus clasificadores básicos sí puede que usen. La característica diferenciadora de RF frente al *Bagging* por defecto es que RF, al general los subconjuntos en cada submodelo, no utiliza todas las características sino que incorpora una selección de características aleatoria. La aleatoriedad de este proceso viene determinada por el parámetro 'K' que indica el número de características aleatorias a escoger. Además, gracias a esta cualidad, RF logra mayor flexibilidad en las fronteras de decisión de la hipótesis final, mayor habilidad de generalizar de los clasificadores débiles que lo conforman y menor tasa de error con el conjunto de test. Asimismo, RF tiende a ser más eficiente que el *Bagging* bajo la definición tradicional dado que logra mejores resultados con menos datos al escoger un subconjunto de características y no todas (Parmar, Katariya y Patel 2019).

- Stacking

Finalmente, el *Stacking* es un procedimiento que consiste en la fusión de clasificadores débiles (clasificadores de primer nivel) mediante un meta-clasificador (clasificador de segundo nivel). En la fusión de clasificadores, el clasificador de segundo nivel recibe como entradas las etiquetas originales de los datos y las decisiones tomadas por los clasificadores de primer nivel a partir de los datos originales(Figura 1.13(c)). Seguidamente, el clasificador de segundo nivel asigna un peso a cada una de las predicciones de los clasificadores de primer nivel en base a la precisión de estas y, finalmente, extrae el resultado final optimizando el rendimiento predictivo del conjunto (Zhou 2012). El *Stacking* es habitualmente un método heterogéneo de ensamblaje para aprovechar las ventajas de distintos tipos de clasificadores básicos, pero un planteamiento homogéneo también es posible. Por otro lado, para la validación del meta-clasificador, se suele emplear validación cruzada, bien por K iteraciones (K-Fold) bien dejando uno fuera (Leave-one-Out), para la selección del subconjunto de predicciones de primer nivel (Sagi y Rokach 2018).

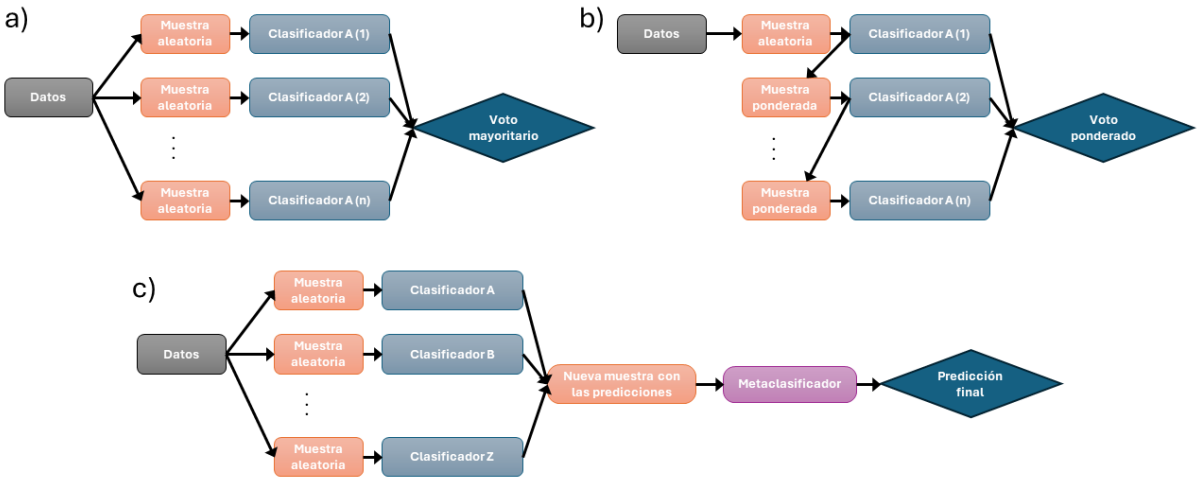


Figura 1.13: Esquemática general de las técnicas de ensamblaje principales: (a) *Bagging*, (b) *Boosting*, y (c) *Stacking*

Los modelos de ensamblaje son muy útiles en el análisis de marcha y aparecen en 11 de los estudios de la Tabla 1.4. En estos se emplean todo tipo de combinaciones de IMUs de 9 DOFs, OMSs sin marcadores, FPs, EMG y/o sensores de profundidad. Los métodos de ensamblaje más empleados son el *Bagging* (RF) y el *Boosting* (AdaBoost, CatBoost, LSBoost), e.g., Yang et al. 2023; Jani et al. 2022. Asimismo, la LMD y GPD son las aplicaciones mayori-

tarias, con algún caso excepcional de GKP. Entre todos los casos, la precisión mediana obtenida es del 95.75 % con un rango entre 71 % y 99.3 %, lo que indica que, por lo general, el ensamblaje de modelos es una técnica que ofrece una muy notable precisión y robustez al ruido aunque dependiendo de su implementación puede ser contraproducente.

1.4.5 Modelos de Redes Neuronales Artificiales

El siguiente nivel al aprendizaje automático, es el aprendizaje profundo (DL), el cual se fundamenta en las redes neuronales artificiales (ANNs), sistemas computacionales de procesado de datos inspirados en el funcionamiento de los sistemas nerviosos biológicos. Manteniendo la analogía, una ANN está compuesta por un gran número de nodos interconectados ('neuronas'), los cuales se hallan interconectados trabajando de forma distribuida, permitiendo aprender las características de los datos de entrada y optimizar las predicciones obtenidas, ya sea clasificación o regresión.

Redes neuronales prealimentadas

- Red neuronal completamente conectada

La red neuronal completamente conectada o perceptrón multicapa (MLP) es la forma tradicional de las redes neuronales prealimentadas (FNN) y el tipo más común de ANN donde la información se propaga por una red de nodos completamente interconectada en un único sentido, de entrada a salida, sin bucles en los nodos, por lo que la salida de cada nodo no afecta al propio nodo (Ramchoun et al. 2016; Rosenblatt 1958). Los nodos de la red se encuentran dispuestos en distintas capas y según el orden en que se encuentren, se puede considerar que cierta capa es una capa de entrada, oculta o de salida(Figura 1.13(b)). Independientemente del número de capas intermedias en la red, la primera capa y la última capa de la red siempre forman parte de esta y se conocen como la capa de entrada y la capa de salida, respectivamente. La función de estas capas es más simple que las intermedias, aunque quizás la capa de entrada sea la más simple de todas pues su función es exclusivamente transmitir la señal de entrada a la siguiente capa, sin preprocesarla. La capa de salida, en cambio, es algo más compleja ya que se encarga de producir las predicciones finales del modelo a partir de los patrones aprendidos por la red. Por otro lado, ajenas al contacto con el mundo físico exterior, se encuentran las capas intermedias, también llamadas capas 'ocultas' (Popescu et al. 2009). En estas, los nodos de la misma capa no están interconectadas, mientras que los nodos de distintas capas sí lo están, llevando la información desde las capas de bajo nivel a las de alto nivel (Rosenblatt 1958).

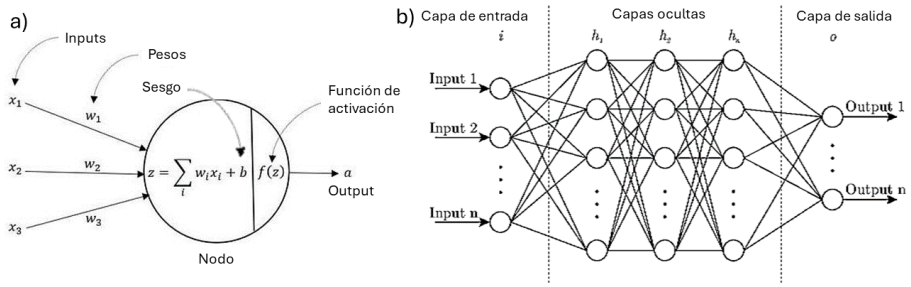


Figura 1.14: Estructura básica de (a) un nodo neuronal, y (b) de una red neuronal completamente conectada (Agarwal 2016)

A excepción de la capa de entrada, las conexiones entre los nodos poseen unos pesos asociados que sirven para ponderar la importancia de las entradas de la capa anterior al nodo y, asimismo, en cada uno de los nodos se lleva a cabo una operación matemática conocida como 'función de activación' que determina el grado de activación del nodo (Figura 1.13(a)) (Delashmit, Manry et al. 2005). La selección de dicha función de activación para cada capa es uno de los (hiper)parámetros fundamentales de un MLP. Lo ideal es que, dado que las conexiones ponderadas de la red son operaciones lineales, se introduzcan no linealidades por medio de funciones de activación no lineales para aumentar el rendimiento y la capacidad de aprendizaje del modelo y evitar el problema de desvanecimiento de gradiente (Popescu et al. 2009). Por lo general, para tareas de clasificación tanto los nodos de las capas ocultas como los de la capa de salida poseen funciones de activación no lineales (e.g., tangente hiperbólica, sigmoidea o *softmax*), mientras que en tareas de regresión la capa de salida utiliza la función de activación lineal ($f(x) = x$) o rectificadoras (ReLU) (O'shea y Nash 2015). Sumado a esto, la capa de salida para una tarea de clasificación tiene tantos nodos como clases haya en los datos, mientras que para una tarea de regresión sólo tiene un único nodo.

Dentro de las funciones de activación comunes en tareas de clasificación, la sigmoidea es una función principalmente enfocada al uso en clasificación binaria ya que transforma la salida en crudo de la red a un único valor de probabilidad en el rango [0, 1] para una de las (dos) clases, por lo que habitualmente se emplea con un cierto umbral de decisión (e.g., 0.5) para determinar la clase predicha(Figura 1.15(d)). Por contra, la *softmax* se usa en la clasificación multiclase y transforma la salida en crudo de la red a una distribución normalizada de probabilidades para todas las clases, la suma de cuyos valores es igual a 1. Otra función empleada para clasificación es la tangente hiperbólica (tanh), una función con forma de S igual que *softmax*, pero con salidas en el rango [-1, 1](Figura 1.13(c)), que ayuda a regular el flujo de datos en las redes neuronales y dificulta el fenómeno de gradiente explosivo (i.e., lo opuesto al desvanecimiento) (Yamashita et al. 2018; Szandała 2021). En cuanto a las funciones de activación para tareas de regresión, la función lineal ofrece un método simple de transformar continuamente las salidas en crudo de la red a valores predichos de regresión(Figura 1.13(a)). La función ReLU es una versión alternativa de la lineal empleada tanto en regresión como en clasificación, aunque mayoritariamente en esta última(Figura 1.13(b)). A diferencia de las funciones lineales, es capaz de resolver en gran medida el problema de desvanecimiento de gradiente al establecer todos los valores negativos a cero y mantener una relación lineal para los valores positivos (i.e., $f(x) = max0, x$). Asimismo, su simplicidad y eficiencia la sitúan por delante de otras funciones no lineales (Yamashita et al. 2018; Sharma, Sharma y Athaiya 2017).

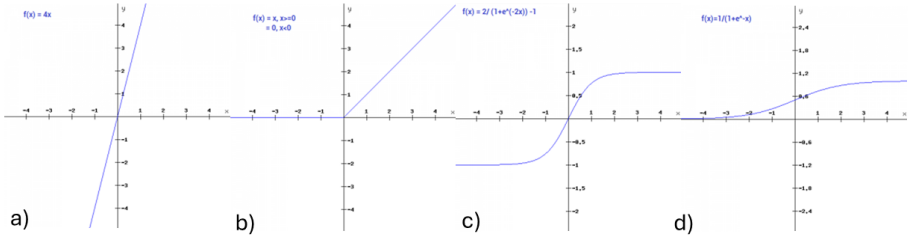


Figura 1.15: Funciones de activación más comunes: (a) lineal, (b) ReLU, (c) tanh, y (d) sigmoidea (Sharma, Sharma y Athaiya 2017)

- Red neuronal convolucional (CNN)

Uno de los aspectos críticos para evitar el sobreajuste en una FNN es la complejidad y el tamaño de la red. A mayor número de parámetros, mayor probabilidad hay de que el modelo aprenda completamente los datos de entrenamiento y sobreajuste, reduciendo su capacidad de generalizar para nuevos datos. Los datos de entrada unidimensionales no suponen problemas para un MLP, mas los datos multidimensionales como las imágenes atacan directamente las limitaciones de un MLP al requerir un número de parámetros (nodos) demasiado elevado como para que el modelo rinda adecuadamente. Es por esto mismo que en el ámbito de reconocimiento de patrones de imagen, se emplea un tipo de FNN que extiende al MLP para permitir el tratamiento de datos multidimensionales (O'shea y Nash 2015).

Las redes neuronales convolucionales (CNN) introducen una forma alternativa de tratar con estos tipo de datos complejos como entrada tal que, en lugar de una conexión completa que requiera un nodo por cada píxel y sus pesos asociados, se disgregue la matriz de la imagen (también llamada 'tensor') en regiones locales (conocidas como 'campos receptivos') sobre las cuales operen una serie de pesos (agrupados en una matriz conocida como 'kernel'). Puesto que típicamente se emplea el mismo kernel para todas las regiones de la imagen, su aplicación se asemeja a una ventana deslizante sobre los datos de entrada que mapea los píxeles del campo receptivo a su correspondiente lugar en la salida de la capa (también llamada 'mapa de características' o 'mapa de activación') (Albawi, Mohammed y Al-Zawi 2017). El resultado mapeado se obtiene por medio de la operación de convolución del kernel con el correspondiente campo receptivo, involucrando operaciones de multiplicación elemento a elemento y suma, motivo por el cual este tipo de capas (y la propia red) reciben el nombre de capas 'convolucionales'.

La aplicación repetida de la convolución a través de las distintas capas convolucionales con sus respectivos kernels da lugar a un número arbitrario de mapas de características que representan las diferentes características del tensor de entrada (O'shea y Nash 2015). El kernel, pues, es un parámetro optimizable del modelo que representa un extractor de características y permite detectar y reconocer la presencia características de la imagen independientemente de su localización en la imagen, aumentando la eficiencia de la CNN respecto al MLP (Yamashita et al. 2018). La optimización por parte del investigador de las capas convolucionales depende principalmente de tres hiper-parámetros: la profundidad (i.e., el número de neuronas en la capa para una misma región), el paso (i.e., el desplazamiento espacial del kernel sobre la imagen) y el relleno con ceros de los bordes de la imagen (O'shea y Nash 2015).

Tras la convolución, se coloca una capa que introduzca no linealidades en la estructura y sature o limite la salida. Para ello, nodos con funciones de activación ReLU (o sigmoidea o tanh en cierta medida) aplicadas elemento a elemento, son comúnmente empleadas (Albawi, Mohammed y Al-Zawi 2017). Asimismo, si bien la arquitectura de una CNN está centrada en las capas convolucionales, estas no son su único elemento, sino que también incluyen capas de agrupación (o *pooling*) y capas completamente conectadas, también llamadas 'capas densas' (i.e., perceptrones).

El objetivo principal de una capa de *pooling* es reducir la complejidad de sus mapa de activación para las capas siguientes sub-muestreando los datos de entrada en la dimensión espacial (i.e., no reduce la dimensión de color si es una imagen a color) (O'shea y Nash 2015). Además, su uso mejora la robustez del modelo a pequeños desplazamientos o distorsiones en el tensor de entrada. Por contra, el sub-muestreo del tensor por medio de esta técnica no preserva la posición de las características por lo que se emplea en casos en que lo que importe sea sólo la presencia y no la localización de las características (Yamashita et al. 2018; Albawi, Mohammed y Al-Zawi 2017). Uno de los métodos más comunes es *Max-Pooling*, el cual divide la matriz de la imagen de entrada (tensor) en sub-regiones rectangulares y devuelve únicamente el valor máximo de cada región. Ciertos tipos de *pooling* más agresivos como el promediado global (i.e., devuelve un sólo valor: la media de todos los valores) sólo se pueden llevar a cabo una vez (Albawi, Mohammed y Al-Zawi 2017).

Una arquitectura típica para una CNN serían sendas repeticiones de capas convolucionales, funciones de activación ReLU y capas de *pooling*, seguidas de al menos una capa densa (Figura 1.16). Debido a su diseño, una CNN es capaz de ir extrayendo características más complejas con cada capa, aprender automáticamente la jerarquía de características espaciales (i.e., de menor a mayor complejidad) y obtener predicciones en base a esta. En concreto, las capas de convolución y *pooling* llevan a cabo la extracción de características, mientras que las capas densas se encargan de transformar los mapas de características resultantes a la predicción final (Yamashita et al. 2018). Análogo a un MLP, las capas densas emplean funciones de activación para mejorar el rendimiento (Figura 1.15).

Añadido a esto, el nexo entre la última capa convolucional o de *pooling* y la primera capa densa se caracteriza por una capa de aplanamiento (*flatten*) que realiza la transición de mapas de características estructurados espacialmente (multidimensionales) a vectores unidimensionales compatibles con las capas densas (O'shea y Nash 2015). No obstante, las capas densas tienen la desventaja de requerir un gran número de parámetros, llevando a menudo a aumentar la complejidad computacional del modelo (Albawi, Mohammed y Al-Zawi 2017). Para mitigar este problema, es común intercalar una o varias capas de expulsión (*dropout*), que reducen el número de nodos y conexiones de la red de forma aleatoria, mejorando no solo la complejidad del modelo, sino también su robustez frente al sobreajuste (ver Sección 1.4.5) (Guo et al. 2016).

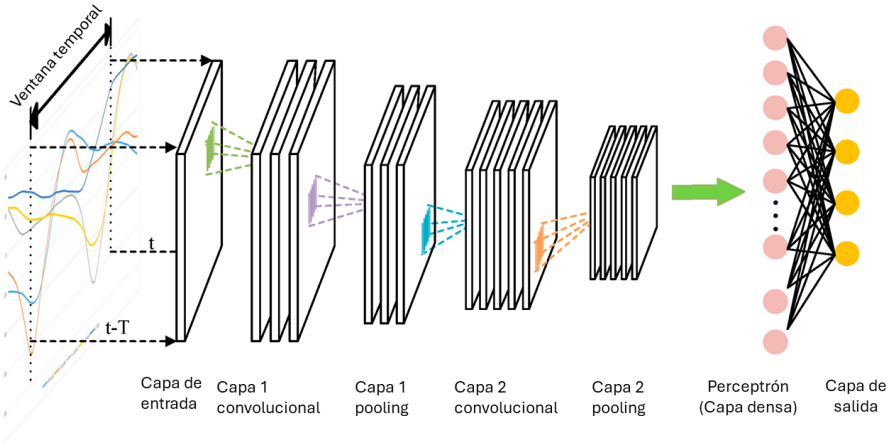


Figura 1.16: Esquemática de una red neuronal convolucional (Liu et al. 2023a)

Como técnica, la FNN, ya sea por medio de un MLP o una CNN, es sin duda la técnica de predicción y/o detección más empleada de todas las revisadas con 24 artículos de los 76 totales incluidos. Su implementación se ha llevado a cabo haciendo uso de todo tipo de sensores, desde IMUs (Karakish, Fouz y ELsawaf 2022) hasta FPs (Mundt et al. 2020). Los modelos diseñados, por otro lado, se han entrenado principalmente con SGD (Nozaki y Watanabe 2019) o con GD adaptativo (Adam) (Vu et al. 2022); y en algunos casos, además, con ESC (Chen et al. 2021a). Del conjunto de estudios, un 50 % tenían como objetivo GPD, un 33.3 % LMD y un 29.2 % GKP. Estos tres son, pues, las tres áreas más exploradas mediante FNN en los estudios encontrados. Además, otras áreas también exploradas

son GKD, GXD, GPP y STA, por lo que prácticamente se ha probado de todo con FNN. Aún mejor, su rendimiento ha sido excepcional, con un 97.5 % de precisión mediana, siendo el peor de los rendimientos de un 90 % de precisión y el mejor de un 100 % (Tabla 1.4). Esto deja bastante claro que, comparado con modelos anteriores, los modelos de DL logran alcanzar rendimientos excelentes en una gran variedad de tareas en el análisis de la marcha humana, haciéndolos especialmente prometedores.

- Entrenamiento de redes neuronales prealimentadas

El aprendizaje o entrenamiento de una FNN es comparable a un proceso de optimización de los pesos de las conexiones en función de la minimización del error entre la predicción de la red y el resultado observado. No sólo aplicable al MLP sino también a la CNN, el algoritmo más empleado para entrenarlas es la propagación hacia atrás de errores o retropropagación (BP), que permite ajustar los pesos de la red en base a la técnica de optimización por descenso de gradiente (GD) (Ramchoun et al. 2016).

En general, el primer paso es inicializar los pesos con valores aleatorios y propagar la información a través de la red hacia la salida para obtener las predicciones iniciales. Seguidamente, se estima el rendimiento del modelo con los parámetros actuales en base al valor de la función de pérdida o función de error elegida (generalmente error cuadrático medio para regresión continua y entropía cruzada para clasificación multiclase) (Popescu et al. 2009). De esta forma, según el sentido del gradiente del error, se ajustan los pesos en el sentido negativo del gradiente de la función en un proceso iterativo que, por lo general, necesita de varios ciclos o 'épocas' para optimizar debidamente los parámetros del modelo (Yamashita et al. 2018).

Por otro lado, en la práctica, las actualizaciones de los parámetros de cada época suelen llevarse a cabo considerando todo el conjunto de datos, de muestra en muestra o por 'minilotes' (o *minibatch*) de los datos. De estos, el primer método es el GD por lotes de todos los datos, un método que alcanza el mínimo global de forma precisa, pero lenta; bueno con pocos datos (Figura 1.17 (a)). El segundo es el 'descenso de gradiente estocástico' (SGD) que, al contrario que el anterior, converge más rápidamente, pero no es tan preciso y no logra llegar a estabilizarse en el mínimo global; bueno con un gran número de datos (Figura 1.17 (c)). El tercero, el 'descenso de gradiente por minilotes' (MbGD), combina GD por lotes con SGD tomando una solución a medio camino que optimiza el compromiso entre velocidad y precisión(Figura 1.17 (b)). Asimismo, la velocidad y precisión de la convergencia al mínimo global de la función dependerá de otros hiper-parámetros a parte de la función de pérdida escogida como es la tasa de aprendizaje. La terminación de la BP debe ser definida por el usuario, bien sea por convergencia al mínimo global, por un máximo número de épocas, por un umbral máximo de rendimiento o por algún otro criterio de parada temprana que evite que el modelo sobreajuste a los datos (Yamashita et al. 2018; Ioffe y Szegedy 2015).

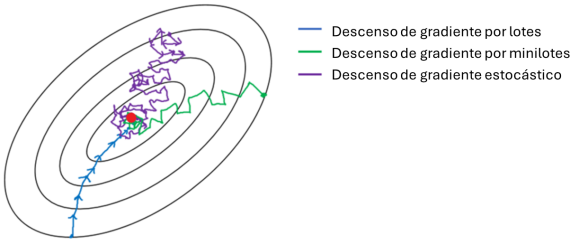


Figura 1.17: Métodos de descenso de gradiente (a) por lotes, (b) por minilotes, y (c) estocástico (Irby 2013)

Como se mencionó en modelos anteriores, el sobreajuste del modelo a los datos es un problema fundamental en el entrenamiento de modelos predictivos. En el ámbito de las FNNs (aunque en general aplica a todas las ANNs), la mejor solución a este problema es obtener más datos, ya sea por medio de un mayor número de pruebas o a través de aumento (artificial) de los datos por medio de transformaciones (Zhong et al. 2020). Sumado a esto, se pueden emplear técnicas de regularización de los datos para añadir restricciones o penalizaciones al entrenamiento del modelo para promover modelos más simples y evitar el sobreajuste.

La regularización L1 o regresión Lasso consiste en añadir un término de penalización a la función de pérdida proporcional a la suma del valor absoluto de los pesos del modelo, por lo que también es llamada 'decaimiento de la pérdida'. Como consecuencia, penaliza las desviaciones grandes de los pesos respecto a cero y favorece la dispersión en el vector de pesos. En su aplicación, la regularización L1 tiende a llevar hacia cero los coeficientes con menor grado de correlación con las etiquetas (i.e., con menor importancia), por lo que en esencia lleva a cabo una selección de características si eliminamos aquellas variables cuyos coeficientes sean los más cercanos a cero (Ng 2004).

Otra técnica similar, la regularización L2, regresión Ridge o 'decaimiento de los pesos', consiste también en añadir un término de penalización a la función de pérdida, pero en este caso proporcional a la suma de los valores al cuadrado de los pesos del modelo (i.e., la norma euclídea o norma L2). Esto logra, efectivamente, penalizar los valores más altos y suavizar el vector de pesos. A diferencia de la L1, la regularización L2 tiende a reducir todos los coeficientes más uniformemente, lo que lo hace especialmente conveniente ante la presencia de colinealidad en las variables. Asimismo, reducir la varianza del vector de pesos le permite regular la complejidad del modelo, lo que dificulta el sobreajuste sin perder tanta información (Ng 2004).

La 'expulsión' o *Dropout* es otra técnica de regularización común en cualquier FNN donde, durante el entrenamiento, una fracción de los nodos de determinada capa es seleccionada aleatoriamente y sus activaciones (i.e., salidas) anuladas. De esta manera, se obtiene un modelo menos sensible a pesos concretos de la red (Hinton et al. 2012).

Una cuarta técnica de regularización frecuentemente empelada es la normalización por lotes (*Batch Normalization* o BN) cuya implementación normaliza, escala y/o desplaza adaptativamente las activaciones que pasan de la capa anterior a la siguiente para igualar la distribución actual de los datos con la original. Por medio de esto, neutraliza en buena medida el cambio en la distribución de las activaciones de entrada de cada capa durante el entrenamiento debido a la aleatoriedad de los datos y de los pesos iniciales (referido como *internal covariate shift*). A la vez, la BN regulariza el modelo añadiendo algo de ruido, lo que permite utilizar tasas de aprendizaje más agresivas y, por ende, acelera el entrenamiento del modelo con BP y GD (Ioffe y Szegedy 2015).

Redes neuronales recurrentes

En aquellas áreas de investigación relacionadas con secuencias de datos como archivos de texto, archivos de audio y archivos de vídeo, las redes neuronales recurrentes (RNN) y sus derivadas son el paradigma de modelo a emplear. Esto se deriva del hecho que la RNN es un modelo capaz de manejar elegantemente aquellas secuencias de datos dependientes del tiempo que presenten correlación entre puntos próximos de la secuencia (Schuster y Paliwal 1997). Así, en lugar de emplear unas dimensiones fijas para los datos de entrada como en una FNN, una RNN se basa en conexiones cíclicas retroalimentadas que le permiten actualizar el estado (oculto) actual del modelo en función de los estados pasados y los datos de entrada actuales (Schuster y Paliwal 1997).

Añadido a esta información, estos modelos también pueden incluir información futura retrasando la salida un cierto número de muestras, pero en la práctica los nodos o celdas recurrentes estándar (i.e., la unidad mínima funcional de una RNN) no son capaces de manejar datos de entrada con dependencia a (muy) largo plazo. La razón fundamental de este hecho es que las ventanas de datos segmentados se asumen independientes por lo que harán falta ventanas demasiado grandes haciendo que las señales de error retropropagadas en el tiempo acaben desvaneciéndose. Por lo tanto, si se emplea un desfase, este será dependiente de la tarea y su valor se deberá hallar por ensayo y error (Yu et al. 2019).

En esencia, la RNN teórica es una secuencia con respuesta infinita al impulso. Sin embargo, en la práctica, para su implementación, la RNN debe aproximarse utilizando una secuencia con respuesta finita al impulso mediante un proceso conocido como 'desenrollado', donde se expande la estructura de la red para un número arbitrario de pasos (Figura 1.18 (a)). Una vez desenrollada, la RNN implementa el método de integración numérica hacia atrás (o método de Euler) sobre una ecuación diferencial ordinaria no lineal con retardo discreto (Sherstinsky 2020).

- Memoria larga a corto plazo

La arquitectura de memoria larga a corto plazo (*Long Short-Term Memory* o LSTM) es un tipo de RNN que, a diferencia de la RNN tradicional donde sólo hay un estado oculto del modelo que se emplea como memoria a corto plazo de la red, implementa un estado del nodo que actúa como memoria a largo plazo además de hacer uso de este estado oculto como memoria a corto plazo (o memoria de trabajo) (Zarzycki y Ławryńczuk 2021). En este sentido, el estado oculto transfiere la información del estado anterior al siguiente, mientras que la memoria a largo plazo transfiere la información aprendida a lo largo de las iteraciones (Figura 1.18 (b)). Por otro lado, para gestionar el flujo de información y saber cuál almacenar y cuál descartar, la LSTM emplea cuatro estructuras conocidas como 'puertas' que se integran en combinación con funciones de activación sigmoideas y tangente hiperbólica (Figura 1.15) que limitan los fenómenos de desvanecimiento y/o explosión del gradiente (Zarzycki y Ławryńczuk 2021).

Como se aprecia en la Figura 1.18 (b) a primera puerta, la 'puerta del olvido', aplica la función sigmoidea sobre la suma ponderada de la entrada actual y el estado oculto anterior, y toma la decisión respecto a cuánto del estado del nodo debe ser descartado y cuánto debe mantenerse en la memoria a largo plazo. La segunda puerta, la 'puerta de la entrada', aplica de nuevo la función sigmoidea sobre la suma ponderada de la entrada actual y el estado oculto anterior, y selecciona cuánto de los datos de entrada actuales y/o de los estados ocultos pasados

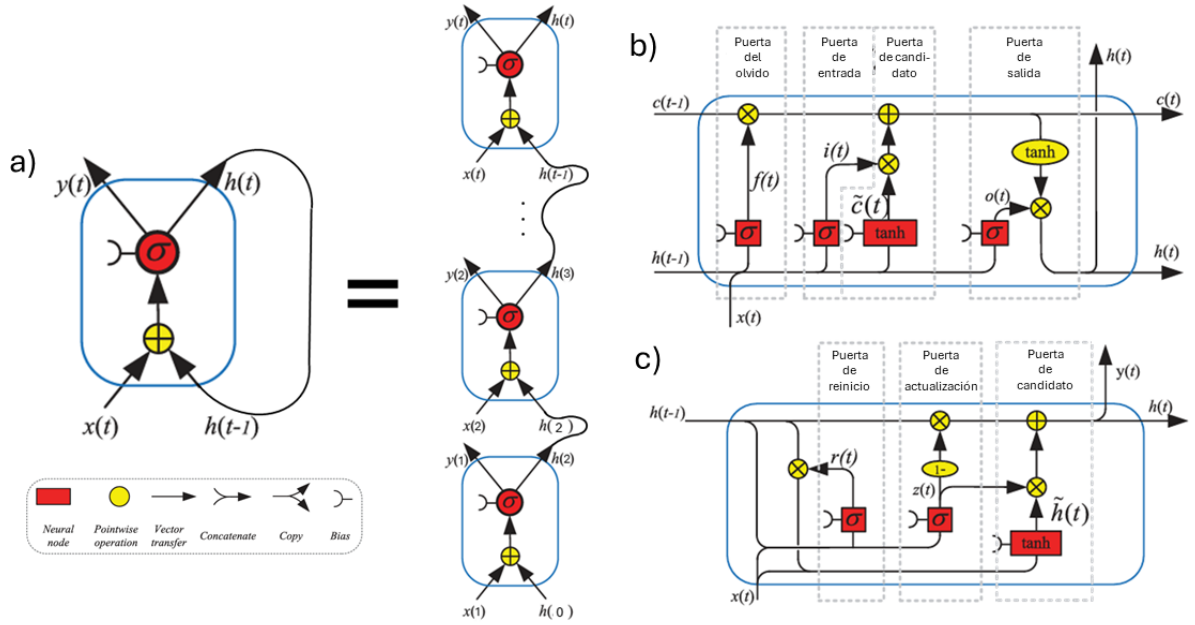


Figura 1.18: Arquitecturas de red neuronal recurrente (a) tradicional, (b) memoria larga a corto plazo, y (c) unidades recurrentes cerradas (Yu et al. 2019)

debe incorporarse al estado de la celda (i.e., memoria a largo plazo) (Schuster y Paliwal 1997). A continuación, la 'puerta del candidato para estado de la celda' aplica la función de tangente hiperbólica (Figura 1.15 sobre la suma ponderada de la entrada actual y el estado oculto anterior, y obtiene el valor candidato a ser añadido al estado del nodo (una vez multiplicado por el porcentaje resultante de la puerta de la entrada). Finalmente, la 'puerta de salida' lleva a cabo el producto elemento a elemento de la tangente hiperbólica del estado del modelo actualizado con la sigmoidea de la suma ponderada de la entrada actual con el estado oculto anterior, y así logra calcular el valor correspondiente al estado oculto actual. De todas las descritas, las puertas del olvido y de salida son los componentes más críticos para el aprendizaje exitoso del modelo. De hecho, aumentar el sesgo de la puerta del olvido, incrementa significativamente el rendimiento del modelo (Schuster y Paliwal 1997; Zarzycki y Ławryńczuk 2021).

En el aspecto práctico, el método más habitual y simple de utilizar la estructura de LSTM es apilar capas de nodos de LSTM, formando una estructura semejante a un MLP si se 'desenrrollan' los nodos recurrentes y añadiendo capacidad y profundidad a la red en general. Además, la implementación de una LSTM puede darse en una ANN diseñada con esta como elemento central que permita optimizar las conexiones internas de los nodos de la LSTM y mejorar las propiedades de la red gracias a esta. También puede integrarse en una ANN más general para que pueda contar con las ventajas asociadas con una LSTM (Yu et al. 2019).

- Unidades recurrentes cerradas

Para tratar de reducir el número de parámetros, la unidad recurrente cerrada (*Gated Recurrent Unit* o GRU) combina la puerta del olvido y la de entrada de la LSTM en una única 'puerta de actualización'. Por lo tanto, la arquitectura GRU contiene únicamente tres puertas: una puerta de reinicio, una puerta de actualización y una puerta de candidato a estado oculto (no a estado del nodo) (Yu et al. 2019). Además, en una estructura GRU, no es necesario emplear un estado de nodo, pues el estado oculto es capaz de actuar como memoria a largo y a corto plazo. Aunque estos cambios aumentan la eficiencia del modelo, también le restan potencia respecto a la LSTM original, lo que hace que la GRU no pueda resolver los mismos problemas.

La primera, la puerta de reinicio, aplica la función sigmoidea sobre la concatenación de la entrada actual y el estado oculto anterior, con lo que selecciona qué información descartar de ambos. La siguiente, la puerta de actualización, aplica la función sigmoidea de nuevo a la concatenación de la entrada actual y el estado oculto anterior, con lo que selecciona qué proporción de información del estado oculto anterior debe mantenerse y, por diferencia, qué proporción de información del estado oculto se actualizará. Por último, la puerta del candidato a estado oculto aplica la tangente hiperbólica sobre la concatenación de la entrada actual con el producto elemento a elemento del estado oculto anterior y la salida de la puerta de reinicio, con lo que determina qué información es

de potencial interés para ser añadida. Finalmente, el nuevo estado oculto se calcula por una suma ponderada (en base al resultado de la puerta de actualización) del estado oculto anterior con el candidato actual, así, logrando olvidar información poco importante del pasado y añadir nueva información de interés (Chung et al. 2014).

- Entrenamiento de redes neuronales recurrentes

Por lo general, toda RNN se entrena con el algoritmo de retropropagación a través del tiempo (BPTT), ya sea la arquitectura general descrita en los párrafos anteriores o las arquitecturas más avanzadas que se comentarán más adelante. La BPTT es una extensión del algoritmo de BP estándar empleado para las FNNs adaptado a las secuencias temporales que se emplean en las RNNs, donde al 'desenrollar' la red, esta se trunca a un número arbitrario de pasos, por lo que, en esencia, BPTT aplica repetidamente la regla de la cadena de la derivación (Sherstinsky 2020).

Dado que las RNNs convencionales no son capaces de hacer uso del contexto futuro de forma eficiente, Schuster y Paliwal 1997 introdujo el concepto de bidireccionalidad a la RNN clásica (BiRNN). Este se basa en una arquitectura capaz de ser entrenada en ambos sentidos temporales simultáneamente (i.e., sin retardo) empleando capas ocultas donde la información se propaga de pasado a futuro y capas ocultas donde la información se propaga de futuro a pasado separadas de las primeras (Figura 1.19). Desde este punto, Graves y Schmidhuber 2005 aplicó exitosamente este mismo concepto a las LSTM resultando en las LSTM bidireccionales (BiLSTM). Por otro lado, el entrenamiento de una estructura recurrente bidireccional se puede lleva a cabo a través de los mismos algoritmos que una estructura unidireccional (i.e., BPTT) ya que las capas son completamente independientes. Entrenar el modelo con el algoritmo de BPTT simplemente ejecutará un pase sobre la RNN en la dirección temporal positiva desenrollada y otro pase en la RNN en la dirección temporal negativa desenrollada (Schuster y Paliwal 1997).

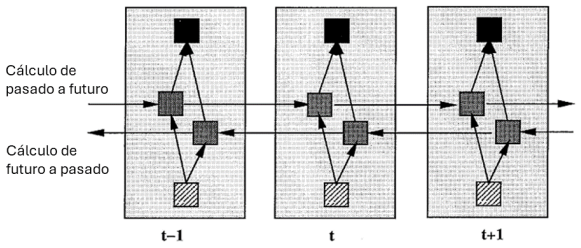


Figura 1.19: Esquema general de una red neuronal recurrente bidireccional desenrollada para tres pasos de tiempo (Schuster y Paliwal 1997)

Para el análisis de la marcha, la RNN es una técnica robusta y prometedora, pero que aún no parece haber logrado alcanzar a la FNN y sus variantes ni en rendimiento ni en popularidad. En concreto, sólo 4 de los 76 estudios incluidos en total implementaron una RNN, ya sea una arquitectura tradicional, LSTM o GRU. La variedad de sensores empleados, por ende, es obviamente menor, incluyendo IMUs (Yi et al. 2021), IPS (Potluri et al. 2019) y OMSs sin marcadores (Sharma y Rombokas 2022). Por otro lado, el entrenamiento por BP con GD para las RNN es aplicado por medio de SGD (Potluri et al. 2019) o Adam (Pazar et al. 2022). De este modo, de los 4 estudios un 50 % se centraron en GPD, otro 50 % en GKP y un 25 % también en LMD. La precisión obtenida osciló entre 85 % y 89 % con una mediana de 87 % que sitúa a los modelos de RNN incluidos por debajo de los modelos de FSM, superando únicamente a HMMs y DTs (Tabla 1.4). Esto nos muestra como, a pesar de lo prometedor de sus características temporales, las RNN y sus arquitecturas derivadas aún requieren de desarrollo para poder ser consideradas como una opción realmente viable en el análisis de marcha, si bien tampoco se hayan excesivamente desencaminadas.

Modelos combinados de redes neuronales

Una estrategia común en los artículos incorporados es la fusión de varios modelos con el objetivo de formar un modelo final cuyas características incluyan las de sus partes; idealmente se busca que se potencien los aspectos positivos de los modelos de base y que entre ellos se contrarresten mutuamente los aspectos negativos. Algún caso concreto como Sharifi-Renani, Mahoor y Clary 2023 combinó modelos explicados como la MLP con otros no abarcados en este texto como los transformadores BERT para GKP. Otro caso similar, Wei et al. 2023, también trató de conformar un modelo para GPD combinando LSTM con un campo aleatorio condicional (CRF), un modelo estadístico tampoco explicado en este texto. Sin embargo, la mayor parte de los autores que emplean modelos alternativos combinan algún tipo de FNN con algún tipo de RNN. Por ejemplo, Arshad et al. 2022 entrenó con Adam una CNN implementada en combinación con una BiGRU obteniendo una precisión de 94.1 % para GPD.

Tabla 1.4: Algoritmos para la detección y/o predicción en el análisis de la marcha humana.

#	Sensores	Modelos	Datos de entrada	Entrenamiento	Resultado	Precisión
[1]6	IMU ML EMG	DataFusion	Accel YXZ Gyro ZXY RGBD EMG	SGD (ESC)	GPD - 50 % GKP - 33.3 % LMD - 16.6 %	96.7 % [88.7, 99.3] %
[2]2	FSR	LDA	EMG, FMG	-	GPD - 100 %	-
[3]2	IMU IPS	FIS	Gyro XYZ GRF	SGD Levenberg - Marquardt	GPD - 50 % LMD - 50 %	96.1 %
[4]20	IMU MK ML IPS EMG LRS URS	RBM	Accel YXZ Gyro ZYX RGB GRF CoP EMG ToF	-	GPD - 75 % LMD - 15 % LTD - 10 % STA - 10 % GKD - 5 % GXP - 5 %	97.35 % [63, 99.8] %
[5]1	IMU	DT	Accel YXZ Gyro ZXY	-	LMD - 100 %	84 % [55, 84] %
[6]5	IMU IPS	HMM	Accel YXZ Gyro ZYX Magnet XYZ GRF CoP	Baum - Welch (EM)	GPP - 60 % LTP - 40 %	81.44 % [63, 97.5] %
[7]2	IMU ML IPS EMG Gonio-metro	FSM	Accel YXZ Gyro ZXY Magnet XYZ RGBD GRF CoP EMG Orientación	-	GPD - 100 %	95 % [94, 96] %
[8]1	IMU IPS	SVM	Gyro XYZ	-	LMD - 50 % LTD - 25 % GPD - 25 %	95.2 %
[9]1	FSR	HMM - SVM	GRF	-	GPD - 100 %	99.0 %
[10]2	IMU	DT - SVM	Gyro ZXY	-	LMD - 100 %	99.85 % [99.7, 100] %
[11]13	IMU ML FP SSR EMG	Ensamblaje	Accel XYZ Gyro ZXY Magnet XYZ RGBD GRF CoP Deformación EMG	Bagging (RF) Boosting (AdaBoost, CatBoost, LSBoost)	LMD - 50 % GPD - 43 % GKP - 7 %	95.75 % [71, 99.3] %
[12]24	IMU IPS MK ML FP EMG	FNN MLP CNN	Accel XYZ Gyro ZXY RGBD GRF CoP EMG	SGD Levenberg - Marquardt Adam (ESC)	GPD - 50 % LMD - 33.3 % GKP - 29.2 % GKD - 12.5 % GXD - 4.2 % GPP - 4.2 % STA - 4.2 %	97.5 % [90, 100] %

#	Sensores	Modelos	Datos de entrada	Entrenamiento	Resultado	Precisión
[13] 4	IMU ML IPS EMG SSR	RNN LSTM GRU	Accel XYZ Gyro XYZ Magnet XYZ RGB GRF CoP EMG Deformación	SGD Adam (ESC)	GPD - 40 % GKP - 40 % LMD - 20 %	87 % [85, 89] %
[14] 2	IMU	BERT - DNN LSTM - CRF	Accel XYZ Gyro XYZ	SGD Regularización L2	GPD - 50 % GKP - 50 %	-
[15] 13	IMU MK ML	DNN - LSTM CNN - BiGRU CNN - LSTM	Accel XYZ Gyro XYZ Magnet XYZ	Adam CGM Regularización L2 (ESC)	GPD - 61.5 % LMD - 30.8 % GKP - 23.1 %	96.55 % [73, 97.8] %

[1] Hu et al. 2021; Krausz y Hargrove 2021; Moro et al. 2022; Sharma y Rombokas 2022; Zhen, Yan y Kong 2020; Zhen, Kong y Yan 2020

[2] Jiang et al. 2018; Krausz y Hargrove 2021

[3] Michał et al. 2017; Schuy et al. 2015

[4] Bartlett y Goldfarb 2018; Chattopadhyay y Nandy 2018; Cheng, Bolívar-Nieto y Gregg 2021; David Li y Hsiao-Wecksler 2013; Ding et al. 2018a; Drouot y Jarrassé 2022; Grimmer et al. 2019; Huo et al. 2018; Jung et al. 2021; Karakish, Fouz y ELsawaf 2022; Maqbool et al. 2016; Moro et al. 2022; Nguyen, La y Duong 2015; Oh y Hong 2023; Park y Kim 2022; Prigent et al. 2023; Sahoo et al. 2020; Schuy et al. 2015; Wahjudi y Lin 2019; Zhao et al. 2016

[5] Brard et al. 2022

[6] Chattopadhyay y Nandy 2018; Chinimilli et al. 2019; Lu et al. 2023; Martinez-Hernandez, Rubio-Solis y Dehghani-Sanij 2018; Sanchez Manchola et al. 2019

[7] Drouot y Jarrassé 2022; Gonçalves et al. 2018

[8] Chinimilli et al. 2019

[9] Yang et al. 2019

[10] Liu et al. 2023b; Wang et al. 2021

[11] Ershadi et al. 2022; Hu et al. 2021; Jani et al. 2022; Moro et al. 2022; Novak et al. 2013; Rezaei et al. 2018; Semwal, Gupta y Lalwani 2021; Sharma y Rombokas 2022; Song, Fernandes y Nordin 2023; Wahjudi y Lin 2019; Yang et al. 2023; Zhang et al. 2023b; Zhang et al. 2023a

[12] D'Antonio et al. 2020; D'Antonio et al. 2021; Dinovitzer, Shushtari y Arami 2023; Gu et al. 2018; Karakish, Fouz y ELsawaf 2022; Mundt et al. 2020; Nozaki y Watanabe 2019; Oh y Hong 2023; Park y Kim 2022; Potluri et al. 2019; Romijnders et al. 2022; Schuy et al. 2015; Shakya, Taparugssanagorn y Silpasuwanchai 2023; Sharma y Rombokas 2022; Su, Smith y Farewik 2020; Su, Liu y Gutierrez-Farewik 2021; Sunny et al. 2023; Vu et al. 2022; Weigand et al. 2022; Yan et al. 2020; Yang et al. 2023; Yu et al. 2023; Zhang et al. 2022; Zhen, Kong y Yan 2020

[13] Pazar et al. 2022; Potluri et al. 2019; Sharma y Rombokas 2022; Yi et al. 2021

[14] Sharifi-Renani, Mahoor y Clary 2023; Wei et al. 2023

[15] Arshad et al. 2022; Chen et al. 2021a; Ding et al. 2018b; Gao et al. 2019; Liu et al. 2023a; Liu et al. 2023b; Renani et al. 2021; Romero-Hernandez, Lope Asiain y Grana 2019; Sharma y Rombokas 2022; Sherratt, Plummer e Iravani 2021; Xu et al. 2023; Zhen, Yan y Kong 2020; Zhen, Kong y Yan 2020

Capítulo 2

METODOLOGÍA

2.1 PLANTEAMIENTO DEL PROBLEMA

El capítulo anterior comenzó poniendo en contexto el trabajo mediante la exposición de la importancia de la marcha en la calidad de vida de las personas, la estructura que presenta y las potenciales afectaciones que puede sufrir. Seguido a esto, se presentaron los principales sensores y algoritmos empleados en el de análisis de marcha, especialmente en la predicción de fases de marcha (GPD) y la predicción de patrones de marcha (LMD).

El objetivo de este trabajo es diseñar un algoritmo para la detección y predicción del inicio y fin de la marcha, y de las fases de la marcha, apto para ser implementado en un sistema robótico de exoesqueleto para miembros inferiores en pacientes con hemiplejia. Para aproximar el problema, los datos inerciales de la marcha de varios participantes fueron recopilados y procesados, y se ha diseñado un sistema que cumple el objetivo principal del trabajo de forma robusta y precisa.

El presente capítulo detalla los materiales y el proceso mediante el cual se ha realizado el estudio para alcanzar el objetivo mentado, incluyendo las diferentes fases de este: revisión de la literatura, protocolo experimental, calibración anatómica de los sensores, extracción de las variables de interés, detección de los periodos de marcha, segmentación de los datos, etiquetado de los datos, cálculo de los porcentajes de cada ciclo

2.2 MATERIALES UTILIZADOS

Los materiales utilizados para la realización de este Trabajo Fin de Máster se listan a continuación:

- Gestor de referencias bibliográficas: JabRef
- Software de análisis bibliométrico y visualización: VOSViewer
- Software de configuración, calibración y manejo de sistemas Xsens: MT Manager 2022.0
- Editor de código fuente: Visual Studio Code
- Herramienta de cálculo de resultados y gráficas: Excel

2.3 REVISIÓN DE LA LITERATURA

2.3.1 Búsqueda de la Información

La recopilación de artículos se centró en las más comunes y fidedignas de las denominadas bases de datos de citas académicas (Haddaway et al. 2015) de las que se incluyeron *Web of Science* (WOS), *ScienceDirect* (SD) e *Institute of Electrical and Electronics Engineers eXplore* (IEEEXplore).

Para tratar de definir unos criterios de búsqueda más precisos acorde al objetivo del trabajo, se realizó una búsqueda preliminar (en WOS) con términos generales donde se recopilaron metadatos como título, resumen y palabras clave, los cuales posteriormente se emplearon para analizar más en detalle los resultados por medio del software VOSviewer (Van Eck y Waltman 2007). Esta herramienta nos permite visualizar gráficamente patrones, tendencias y clústeres dentro de grandes conjuntos de datos de forma simple mediante varios mapas, facilitando la identificación de aquellas palabras clave prioritarias para la búsqueda. Como se aprecia en la Figura 2.1 los conceptos se agrupan en 3-4 clústeres principales en los que hay ciertos conceptos que aparecen con mayor frecuencia que otros (mayor tamaño de los nodos). Esto nos lleva a pensar que será de nuestro máximo interés incluir palabras clave como '*gait*', '*analysis*', '*machine learning*', '*sensor*' o 'IMU' (*Inertial Measurement Unit*).

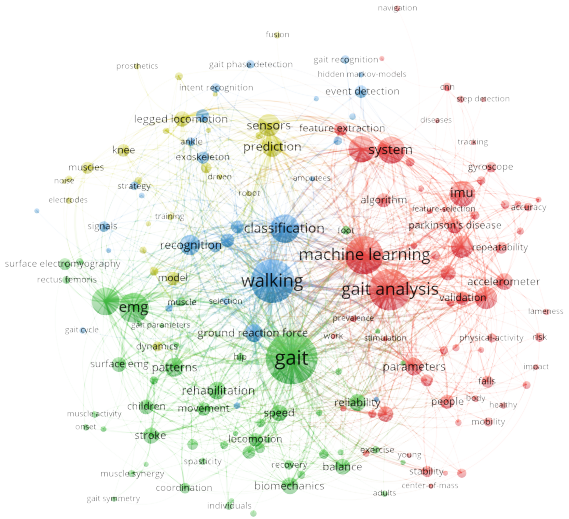


Figura 2.1: Nube de conceptos representando las palabras claves más empleadas entre los autores para los resultados de WOS. El tamaño de los nodos es proporcional a la frecuencia de ocurrencia de los términos entre los artículos incluidos.

Tras destilar las palabras clave que más se ajustaban al objetivo del trabajo, se procedió a crear las frases de búsqueda para cada uno de las bases de datos mencionadas empleando operadores booleanos (Tabla 2.1). Para introducir los 40 términos resultantes se emplearon las funciones de búsqueda avanzada en WOS, SD e IEEXplore, si bien en esta última fue por medio de búsqueda por comandos para poder trabajar con un mayor número de términos permisibles.

2.3.2 Selección Sistemática

Con el propósito de llevar a cabo la revisión convenientemente se siguió la estructura del estándar PRISMA (Figura 2.2). Dado que este trabajo está centrado en el ámbito del análisis de la marcha humana con datos de IMUs, se hizo especial hincapié en la búsqueda con estos sensores, aunque se trató de abarcar todo el abanico de sensores aplicables a este ámbito.

Partiendo de la primera búsqueda, se importaron un total de 1,778 artículos entre las 3 bases de datos, los cuales se importaron con la herramienta de gestión de referencias bibliográficas JabRef. A continuación, aplicando filtro se eliminaron aquellos publicados en otro idioma que no fuera inglés, las publicaciones previas a al año 2010 y los duplicados, quedando un total de 944 artículos. Una evaluación más exhaustiva de los títulos y resúmenes de cada artículo, permitió excluir aquellos artículos no centrados en humanos sino en animales, los no centrados en la marcha y aquellos artículos no originales en el sentido de que son obras que no presentan nuevos hallazgos experimentales, sino síntesis de resultados o metodologías (e.g., otras revisiones sistemáticas). Esto nos dejó con

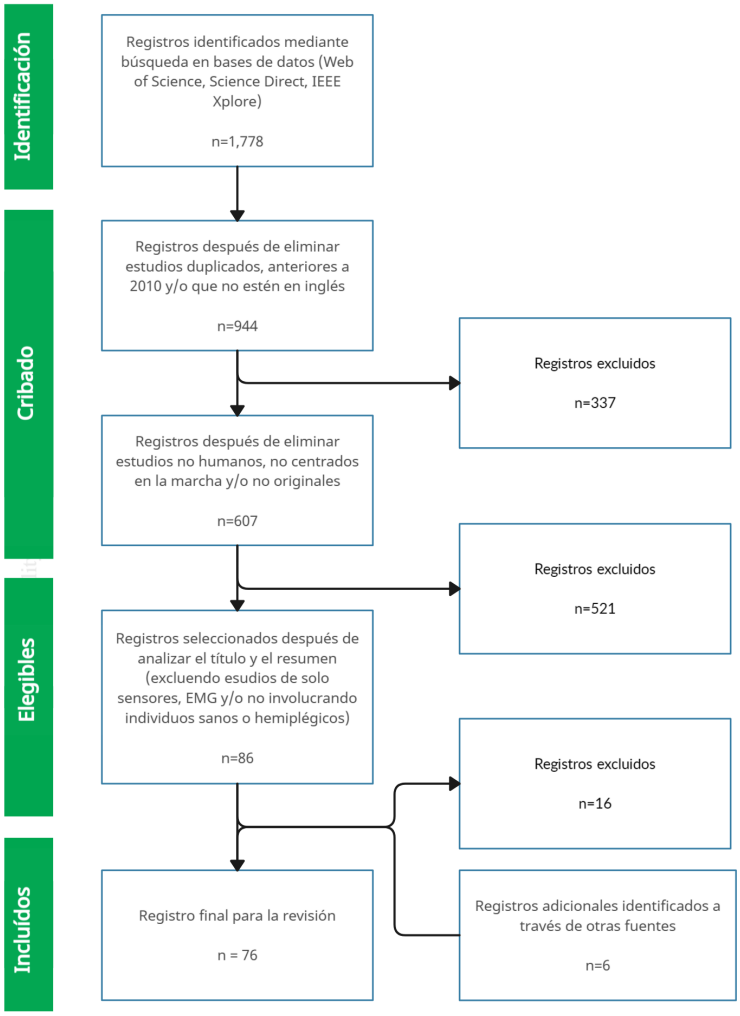


Figura 2.2: Diagrama de flujo PRISMA

607 artículos, que se redujeron a 86 tras excluir todos aquellos que se centraran exclusivamente en el desarrollo de sensores, los que trataran de EMG o EEG únicamente y los enfocados a patologías que afectan a la marcha que no fueran la hemiplejia. Tras un análisis manual del texto en sí, se eliminaron otros 14 más, quedando un total de 70 estudios. No obstante, tras evaluar otros artículos encontrados por otros medios, se decidió añadir 6 artículos extra para la revisión, para un total de 76. Todos estos se engloban dentro del total de artículos incluidos en este trabajo que alcanza las más de 209 referencias para complementar este registro de base y poder garantizar la calidad de la información ofrecida en este.

2.4 PROTOCOLO EXPERIMENTAL

2.4.1 Configuración de los Sensores

En base a la información recopilada sobre los múltiples tipos de sensores empleados en el análisis de la marcha, se decidió emplear 3 unidades de medida inercial (IMUs) para la recopilación de datos de marcha. En concreto, se recogieron datos de aceleración lineal, velocidad angular y orientación relativa de los segmentos de la pelvis, el fémur y la tibia para un total de 8 sujetos sanos voluntarios, incluyendo 4 hombres y 4 mujeres. Para ello se emplearon 3 IMUs del modelo MTw Awinda de Xsens que se configuraron empleando el software de MT Manager 2022.0 (Figura 2.3).

Tabla 2.1: Comandos empleados en la búsqueda bibliográfica

Bases de datos	Comandos de búsqueda
Web of Science	TS=((gait OR (gait AND phase)) AND (detection OR analysis or prediction)AND (IMU OR IPS OR FSR OR EMG OR ENG OR FMG OR GRF) AND (Machine Learning OR ML OR Deep learning OR Neural Network OR dcnn OR CNN OR ANN OR LDA OR PCA OR SVM OR HMM OR FNN OR NARX OR LSTM OR LSTM-DNN OR ED-FNN OR Exponentially delayed OR OR OR Hybrid OR Rule OR Rule-based OR Threshold OR Threshold-based OR Random Forest OR RF OR MLP OR multilayer perceptron))
ScienceDirect	TITLE-ABS-KEY ((gait OR (gait AND phase)) AND (detection OR analysis OR prediction) AND (imu OR ips OR fsr OR emg OR eng OR fmg OR grf) AND (narx OR deep-cnn OR ann OR lda OR pca OR svm OR hmm OR fnn OR dnn OR dlnn OR (deep AND learning) OR (machine AND learning) OR lstm OR lstm-dnn OR ed-fnn OR (exponentially AND delayed) OR hybrid OR rule OR (rule AND based) OR threshold OR (threshold AND based) OR (random AND forest) OR mlp OR (multilayer AND perceptron)))
IEEE Xplore	(”All Metadata”: ”gait” OR ”gait phase”) AND (”All Metadata”: ”detection” OR ”analysis” OR ”prediction”) AND (”All Metadata”: ”IMU” OR ”IPS” OR ”FSR” OR ”EMG” OR ”ENG” OR ”FMG” OR ”GRF”) AND (”All Metadata”: ”NARX” OR ”Deep CNN” OR ”ANN” OR ”LDA” OR ”PCA” OR ”SVM” OR ”HMM” OR ”FNN” OR ”DLNN” OR ”deep learning” OR ”machine learning” OR ”LSTM” OR ”LSTM-DNN” OR ”ED-FNN” OR ”exponentially delayed” OR ”hybrid” OR ”rule” OR ”rule-based” OR ”threshold” OR ”threshold-based” OR ”RF” OR ”Random forest” OR ”MLP” OR ”multilayer perceptron”)

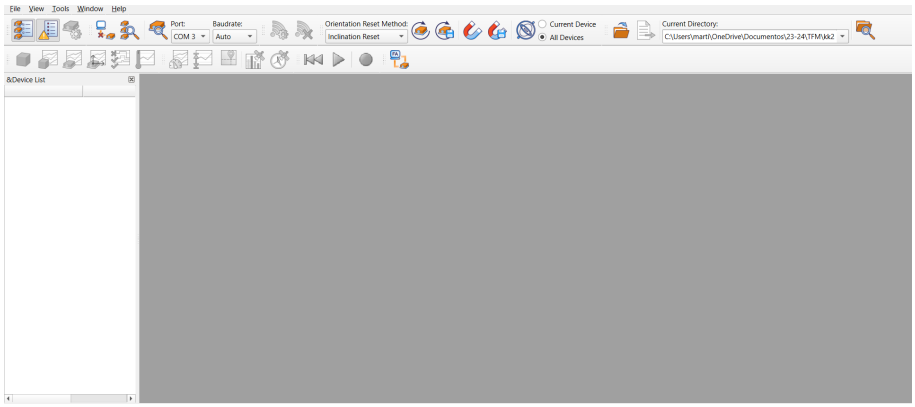


Figura 2.3: Interfaz inicial de MT Manager 2022.0.

Conexión de los sensores

Para la configuración de los sensores, en primer lugar estos se conectaron a la estación Awinda (el centro de sincronización y control de los datos recibidos) que a la vez estuvo conectada al ordenador personal investigador (Figura 2.4).

Figura 2.4: IMUs conectadas a la estación maestra de Awinda para la configuración inicial.

Activación del modo de medida

Seguidamente, se seleccionaron las opciones *Scan all ports >Wireless configuration >Enable all wireless masters* y, tras desconectar las IMUs de la estación, se seleccionó *Start measurement on all wireless masters* (Figura 2.5).

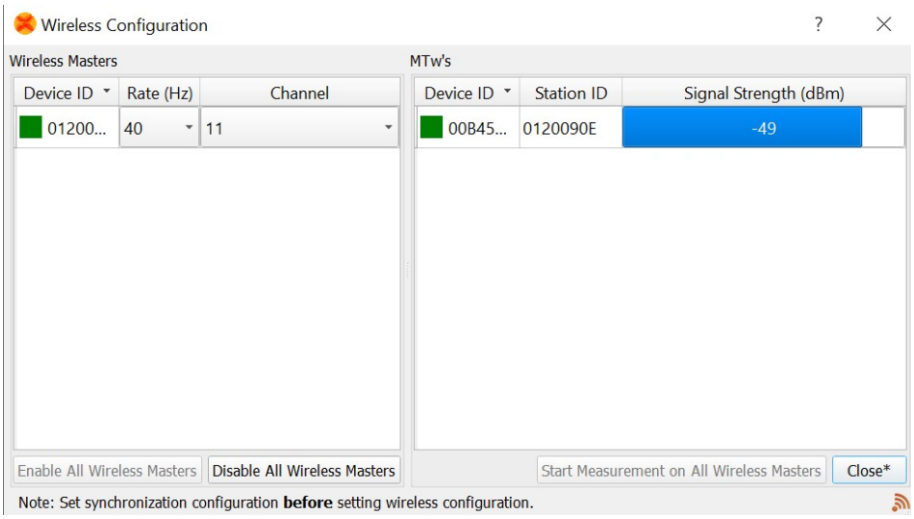


Figura 2.5: Activación del seguimiento de las medidas de los IMUs con MT Manager 2022.0.

Reinicio de la orientación

Tras esto, se comprobó que los sensores tuvieran batería suficiente y se reinició la orientación de estos con la opción *Reset orientation* con las IMUs colocadas sobre una superficie plana a mínimo 3 metros de cualquier posible fuente de interferencia electromagnética (e.g., equipos informáticos) con la parte delantera del sensor orientada hacia arriba (Figura 2.6).

Figura 2.6: Colocación de las IMUs durante el reinicio de la orientación.

Visualización de los sensores

A partir de este punto, durante toda la prueba se mantuvo activada la opción *3D Orientation* para visualizar en tiempo real la posición de cada uno de los IMUs (Figura2.7). Así se pudo detectar fácilmente cualquier anomalía que pudiera acontecer en el proceso como aquellas derivadas de un bajo nivel de batería del sensor, el movimiento o despegue del sensor o cualquier situación similar.

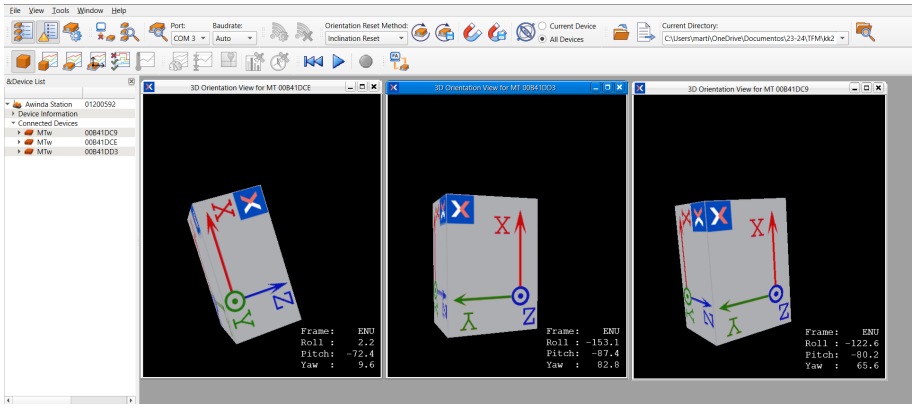


Figura 2.7: Visualización de la orientación 3D de las IMUs en MT Manager2022.0

Colocación de los sensores

Para garantizar la comparabilidad, consistencia y reproducibilidad de las medidas y evitar errores sistemáticos, la toma de medidas se llevó a cabo empleando exclusivamente las mismas 3 IMUs colocadas en el mismo orden. Los sensores se colocaron sobre la ropa en la cadera, el muslo izquierdo y la pierna izquierda por medio de cinchas, con el LED hacia abajo y asegurando en todos los casos que los bolsillos de los sujetos estuvieran vacíos (Figura 2.8).

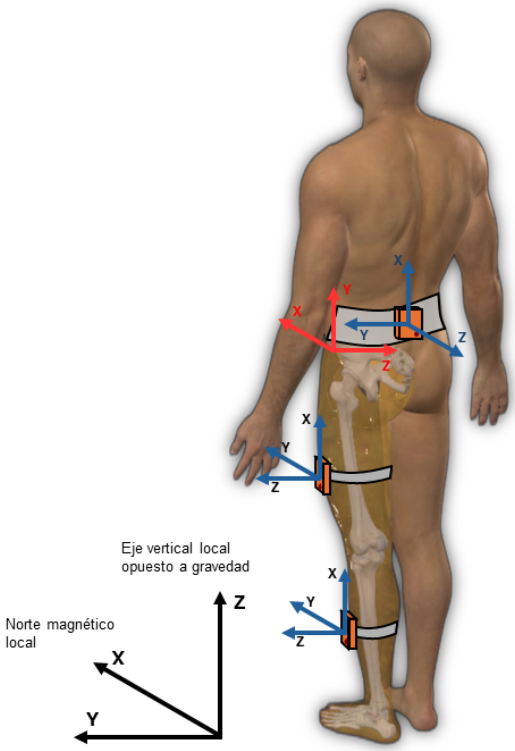


Figura 2.8: Esquema del diseño experimental con los sistemas de referencia de interés. En azul el sistema de referencia local de cada IMU, en rojo el sistema de referencia local de la pelvis según la Sociedad Internacional de Biomecánica (ISB) (D'Lima et al. 2000) y en negro el sistema de referencia global según el manual de Xsens (JKO s.f.). Nótese que la orientación de los sistemas de referencia de la imagen no necesariamente se corresponde con la realidad ya que el sujeto no tiene por qué comenzar orientado hacia el norte magnético.

2.4.2 *Registro Experimental*

Toma de datos

El experimento consistió de 4 tomas de muestras a 100 Hz de frecuencia de muestreo que se realizaron en el Laboratorio de Análisis Biomecánico del Instituto de Biomecánica de Valencia. La primera fue la que se tomó como calibración inicial, donde los participantes permanecieron 3 segundos aproximadamente con la mirada al frente, hombros relajados y pies a la altura de los hombros y apuntando al frente. Esto más adelante permitió tomar ciertas premisas como válidas, simplificando los cálculos para la calibración de los sensores (Sección 2.5.1). En la segunda prueba, se indicó a los sujetos para realizar una sesión de marcha en 4 bloques secuenciales donde cada bloque se inició con 3 segundos en reposo, seguidos de 7 pasos empezando a caminar con la pierna derecha y finalizando con 3 segundos en reposo, antes de dar voltear 180° y empezar el siguiente bloque. En este caso, las pausas de 3 segundos nos facilitarán más adelante, el procesado y la segmentación de los periodos de marcha (Sección 2.5.3). Para la tercera prueba, los sujetos recibieron las mismas indicaciones que la segunda, salvo que esta vez tuvieron que iniciar la marcha con la pierna izquierda (i.e., donde están colocados los IMUs), que es donde potencialmente sería implementado el exoesqueleto para los pacientes hemipléjicos. De nuevo, esta prueba contará con las pausas de 3 segundos por los motivos ya mencionados. Por último, la cuarta prueba consistió en exactamente lo mismo que la primera: una medida de calibración. En este caso, la medida de calibración final, nos sirvió para comprobar si la calibración realizada con las medidas iniciales fue válida para toda la prueba o si era necesario ajustar el *bias* o el *offset* del sensor (Sección 2.5.1).

Exportación de datos

Para registrar los datos de cada una de las pruebas se utilizó la opción *Record*, la cual generó un fichero .mtb en el directorio indicado, cuyos datos podemos visualizar por medio de la opción de *Inertial Data* que nos muestra las medidas del acelerómetro, giroscopio y magnetómetro (Figura 2.9).

Figura 2.9: Visualización de las medidas inerciales registradas en MT Manager 2022.0.

Finalmente, para obtener los datos en formato .txt, abrimos el fichero creado con MT Manager 2022.0 utilizando *File >Open File* y *File >File export* exportamos la aceleración, la tasa de giro (i.e., velocidad angular) y los cuaterniones (Figura 2.10).

Esto nos generó ficheros de texto de valores separados por espacios como los de la Tabla 2.2 con 12 filas de encabezado y 11 columnas de datos, incluyendo para cada medida el número de cuenta, la aceleración en los ejes X, Y y Z, la velocidad angular en los ejes X, Y y Z, la componente escalar del cuaternión y la componente vectorial del cuaternión en los ejes X, Y y Z.

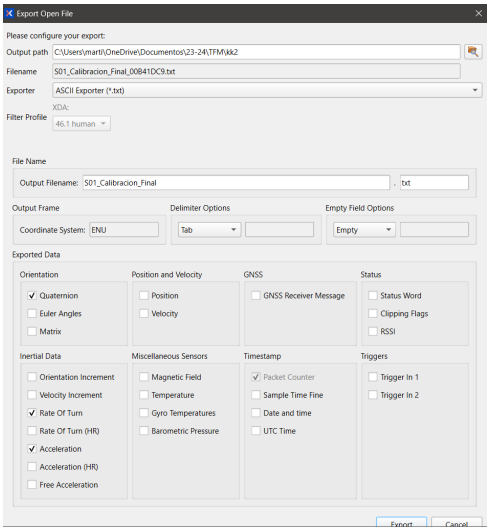


Figura 2.10: Ventana de exportación de las variables en MT Manager 2022.0.

Tabla 2.2: Cabecera y 10 primeros valores del archivo .txt generado para una prueba concreta de un sujeto específico

// General information:
// MT Manager version: 2022.0.0
// XDA version: 2022.0.0 build 7085 rev 119802 built on 2022-11-08
// Device information:
// DeviceId: 00B41DD3
// ProductCode: MTW2-3A7G6
// Firmware Version: 4.6.0
// Hardware Version: 1.1.0
// Device settings:
// Filter Profile: human(46.1)
// Option Flags: Orientation Smoother Disabled, Position/Velocity Smoother Disabled, Continuous Zero Rotation Update Disabled, AHS Disabled, ICC Disabled
// Coordinate system: ENU

PacketCounter	Acc_X	Acc_Y	Acc_Z	Gyr_X	Gyr_Y	Gyr_Z	Quat_q0	Quat_q1	Quat_q2	Quat_q3
45180	9.774903	-0.24429	-0.783422	0.005953	-0.001531	0.005471	-0.572026	0.421686	0.583182	0.393529
45181	9.745418	-0.227388	-0.779993	0.014198	0.000038	0.00847	-0.572081	0.421667	0.583185	0.393466
45182	9.752775	-0.2507	-0.774482	0.009683	-0.005778	0.005432	-0.572094	0.42167	0.583201	0.39342
45183	9.737963	-0.236824	-0.802347	0.005952	0.00025	0.006782	-0.572117	0.421678	0.583188	0.393398
45184	9.752626	-0.210199	-0.817248	0.002941	0.001715	0.008693	-0.572142	0.421695	0.583162	0.39338
45185	9.730657	-0.257863	-0.765151	-0.009823	0.003235	0.009216	-0.572151	0.421739	0.58311	0.393398
45186	9.745323	-0.261859	-0.796183	-0.010611	-0.001486	0.010075	-0.572141	0.421804	0.583065	0.393411
45187	9.767369	-0.257769	-0.805431	-0.006811	0.00177	0.007305	-0.572145	0.421842	0.583023	0.393425
45188	9.730575	-0.260416	-0.765321	-0.000141	-0.006107	0.012023	-0.572148	0.421893	0.583007	0.393391
45189	9.774643	-0.25155	-0.784376	-0.001337	-0.005623	0.005841	-0.572137	0.421928	0.582999	0.39338

2.5 PREPROCESADO DE LOS DATOS

Como se mencionó en la Sección 1.4.1, la primera parte en el planteamiento del problema a resolver es el preprocesado de los datos, donde un buen preprocesado puede suponer la diferencia entre el éxito o el fracaso del trabajo. En esta sección se explicarán los detalles sobre la gestión inicial de los datos, yendo desde la calibración de los sensores y la extracción de características hasta el etiquetado manual de la señal y la segmentación de los periodos y los ciclos de la marcha. Es importante recalcar que todos los pasos explicados a continuación se realizaron en Python con Visual Studio Code.

2.5.1 Calibración anatómica de las IMUs

Importación de los datos

La calibración de las IMUs es el primer paso para obtener las medidas que buscamos. En primer lugar, mediante la función *importar_datos* los datos se importaron desde el directorio específico donde se guardaron los ficheros exportados .txt con la función 'read_csv' de la librería Pandas. Se crearon 4 listas para adjuntar ('append') los *arrays* de datos en bruto, una para los datos de cada prueba: calibración inicial (ci), primer paseo empezando con la pierna derecha (d), segundo paseo empezando con la pierna izquierda (i) y calibración final (cf). Asimismo, las componentes $[q_0, q_1, q_2, q_3]$ de cada cuaternión extrajeron de los datos almacenados a través de la función *load_quat*, que internamente las convirtió de *arrays* a objetos de clase cuaternión ('Q') por medio de la función *__init__* que inicializa un objeto de la clase *quaternion* (Anexo 8.1), y se adjuntaron en otras 4 listas para facilitar su manejo. Por último, partiendo de una frecuencia de muestreo (fs) de 100 Hz, se obtuvieron los vectores de tiempo para cada uno de los sujetos en cada una de las pruebas y se adjuntaron a sus respectivas variables.

Calibración anatómica

Los cuaterniones extraídos del registro de las IMUs eran los cuaterniones que expresaban la rotación del sistema de referencia (SdR) local del sensor (i) al global (o): ${}^{o}q_{i\,s}$. Para calibrar el sensor anatómicamente fue necesario conocer los cuaterniones ${}^{s}q_{i\,s}$ que expresan la rotación del SdR local de cada IMU al SdR local del segmento (s) anexo (i.e., pelvis, fémur o tibia). Con este propósito se creó la función *alfa_beta_theta* (Anexo 8.1). La idea principal es obtener los cuaterniones ${}^{o}q_{s}$ en las condiciones estáticas de calibración, donde se hicieron varias asunciones, y calcular ${}^{s}q_{i\,s}$ a partir de estos. Lo interesante de este método, es que ${}^{s}q_{i\,s}$ no sólo es válido para las medidas estáticas de calibración, sino también para las dinámicas durante la marcha. Por lo tanto, los cuaterniones ${}^{s}q_{i\,s}$ extraídos en la calibración se emplearon más adelante para el cálculo de los cuaterniones ${}^{o}q_{s}$ durante la marcha (Ecuación 2.3). Al expresar la rotación del SdR local del segmento al SdR global, por medio de este podremos extraer los ángulos articulares de interés durante la marcha.

$$ {}^{o}q_{i\,s} = {}^{o}q_{s} \otimes {}^{s}q_{i\,s} \tag{2.1} $$

De la que puede despejarse ${}^{s}q_{i\,s}$ como:

$$ {}^{s}q_{i\,s} = {}^{o}q_{s}{}^{*} \otimes {}^{o}q_{i\,s} \tag{2.2} $$

Y del mismo modo ${}^{o}q_{s}$:

$$ {}^{o}q_{s} = {}^{o}q_{i\,s} \otimes {}^{s}q_{i}{}^{*} \tag{2.3} $$

Donde $q_1 \otimes q_2$ representa el producto de los cuaterniones q_1 y q_2 (Ecuación 2.4) y q^*, el conjugado del cuaternión q (Ecuación 2.5).

$$ q_1 \otimes q_2 = (a_1 \cdot a_2 - \vec{v}_1 \cdot \vec{v}_2, \quad a_1 \cdot \vec{v}_2 + a_2 \cdot \vec{v}_1 + \vec{v}_1 \times \vec{v}_2) \tag{2.4} $$

$$ q^* = (a, -\vec{v}) = (q_0, -q_1, -q_2, -q_3) \tag{2.5} $$

Donde a es la parte escalar y $\vec{v}$ la parte vectorial del cuaternión q original. Así, a la derecha, se define explícitamente los cuatro componentes del cuaternión $q_0 = a$ y $(q_1, q_2, q_3) = \vec{v}$

Para resolver esta tarea, se consideró que en el instante inicial (i.e., condiciones de calibración estáticas) el eje X del SdR local del sensor se hallaba alineado con el eje Z del SdR global (i.e., dirección vertical en sentido opuesto a la gravedad) con un giro θ desconocido entorno al eje Z global. Sumado a esto, también se consideró que, independientemente del instante de tiempo, el SdR local de las IMUs y el SdR local de los segmentos asociados tendrían desalineados sus ejes a causa de dos giros α y β desconocidos entorno a los ejes X antero-posterior y Z medio-lateral del SdR del segmento. Además, se asumió que θ debería ser al menos un orden de magnitud mayor que α y β. La Figura 2.8 muestra el sentido de los ejes del SdR local de las IMUs de Xsens tras su reinicio y el sentido de los ejes del SdR local de la pelvis según el estándar de la Sociedad Internacional de Biomecánica (ISB) que se consideró el SdR para todos los segmentos (D'Lima et al. 2000).

En un caso ideal, con el sujeto estático, perfectamente vertical y mirando hacia el norte y con los sensores perfectamente alineados con cada uno de los segmentos, los cuaterniones ${}^{o}q_{s}$, ${}^{s}q_{i\,s}$ y ${}^{o}q_{i\,s}$ se pueden definir por giros elementales de $\pi/2$ radianes deducibles de la Figura 2.8 que únicamente representan el cambio de ejes. En este caso, al considerar el mismo SdR para todos los segmentos, los ${}^{o}q_{s}$ son idénticos para pelvis, fémur y tibia (Ecuación 2.6). En cambio, los cuaterniones ${}^{s}q_{i\,s}$ no se pueden asumir idénticos para todos los segmentos ya que se empleó el SdR de la pelvis para definir el SdR de los tres segmentos mientras que los sensores no estaban igualmente orientados respecto a este. Por lo tanto, deben definirse dos casos: ${}^{p}q_{i}$ para la pelvis y ${}^{f}q_{i} = {}^{t}q_{i}$ para el fémur y la tibia (Ecuación 2.7). Por lo tanto, siguiendo la Ecuación 2.1, los ${}^{o}q_{i\,s}$ dependerán del segmento al que esté anexo la IMU (Ecuación 2.8).

$$ {}^{o}q_{p} = {}^{o}q_{f} = {}^{o}q_{t} = q_{x}(\pi/2) \tag{2.6} $$

$$ \begin{aligned} {}^{p}q_{i\,p} &= q_{x}(\pi/2) \otimes q_{y}(\pi/2) \\ {}^{f}q_{i\,f} = {}^{t}q_{i\,t} &= q_{x}(\pi/2) \otimes q_{y}(\pi/2) \otimes q_{x}(\pi/2) \end{aligned} \tag{2.7} $$

$$ \begin{aligned} {}^{o}q_{i\,p} &= {}^{o}q_{p} \otimes {}^{p}q_{i\,p} = q_{x}(\pi/2) \otimes [q_{x}(\pi/2) \otimes q_{y}(\pi/2)] \\ {}^{o}q_{i\,f} = {}^{o}q_{i\,t} &= {}^{o}q_{t} \otimes {}^{t}q_{i\,t} = q_{x}(\pi/2) \otimes [q_{x}(\pi/2) \otimes q_{y}(\pi/2) \otimes q_{x}(\pi/2)] \end{aligned} \tag{2.8} $$

Donde $q_{x}(\xi)$, $q_{y}(\xi)$ y $q_{z}(\xi)$ se definen como giros de ξ entorno a los ejes x, y, z, respectivamente:

$$ \begin{aligned} q_{x}(\xi) &= \left\langle \cos\frac{\xi}{2}, \sin\frac{\xi}{2}, 0, 0 \right\rangle \\ \mathsf{q}_{y}(\xi) &= \left\langle \cos\frac{\xi}{2}, 0, \sin\frac{\xi}{2}, 0 \right\rangle \\ \mathsf{q}_{z}(\xi) &= \left\langle \cos\frac{\xi}{2}, 0, 0, \sin\frac{\xi}{2} \right\rangle \end{aligned} \tag{2.9} $$

Por el contrario, si sólo el sensor estuviera perfectamente colocado, se debería considerar el giro θ para expresar la diferencia entre la orientación de los segmentos de la persona y el SdR global. Este giro se representó con el cuaternión $q_{z}(\theta)$. Nótese que, al emplear un mismo SdR para los tres segmentos, $q_{z}(\theta)$ será el mismo para los tres y, por ende, los ${}^{o}q_{s}$ también seguirán siendo iguales para pelvis, fémur y tibia. Los ${}^{p}q_{i\,s}$, en cambio, no varían en este caso:

$$ {}^{o}q_{p} = {}^{o}q_{f} = {}^{o}q_{t} = q_{z}(\theta) \otimes q_{x}(\pi/2) \tag{2.10} $$

$$ \begin{aligned} {}^{p}q_{i\,p} &= q_{x}(\pi/2) \otimes q_{y}(\pi/2) \\ {}^{f}q_{i\,f} = {}^{t}q_{i\,t} &= q_{x}(\pi/2) \otimes q_{y}(\pi/2) \otimes q_{x}(\pi/2) \end{aligned} \tag{2.11} $$

$$ \begin{aligned} {}^{o}q_{i\,p} &= {}^{o}q_{p} \otimes {}^{p}q_{i\,p} = [q_{z}(\theta) \otimes q_{x}(\pi/2)] \otimes [q_{x}(\pi/2) \otimes q_{y}(\pi/2)] \\ {}^{o}q_{i\,f} = {}^{o}q_{i\,t} &= {}^{o}q_{t} \otimes {}^{t}q_{i\,t} = [q_{z}(\theta) \otimes q_{x}(\pi/2)] \otimes [q_{x}(\pi/2) \otimes q_{y}(\pi/2) \otimes q_{x}(\pi/2)] \end{aligned} \tag{2.12} $$

No obstante, en realidad no es del todo suficiente con considerar θ, sino que es fundamental también tomar en consideración α y β. Dado que se consideran rotaciones sucesivas, se reflejaron con un único cuaternión incógnita $q_{xz}(\alpha, \beta)$. Además, la orientación del sensor relativa al segmento es específica del sensor y del segmento, por lo que ningún ${}^{s}q_{i\,s}$ coincidirá ya que los α y β serán diferentes:

$$ {}^{o}q_p = {}^{o}q_f = {}^{o}q_t = q_z(\theta) \otimes q_x(\pi/2) \tag{2.13} $$

$$ \begin{aligned} {}^{p}q_{ip} &= q_{xz}(\alpha_p, \beta_p) \otimes q_x(\pi/2) \otimes q_y(\pi/2) \\ {}^{f}q_{if} &= q_{xz}(\alpha_f, \beta_f) \otimes q_x(\pi/2) \otimes q_y(\pi/2) \otimes q_x(\pi/2) \\ {}^{t}q_{it} &= q_{xz}(\alpha_t, \beta_t) \otimes q_x(\pi/2) \otimes q_y(\pi/2) \otimes q_x(\pi/2) \end{aligned} \tag{2.14} $$

$$ \begin{aligned} {}^{o}q_{ip} &= {}^{o}q_p \otimes {}^{p}q_{ip} = [q_z(\theta) \otimes q_x(\pi/2)] \otimes {}^{p}q_{ip} \\ {}^{o}q_{if} &= {}^{o}q_f \otimes {}^{f}q_{if} = [q_z(\theta) \otimes q_x(\pi/2)] \otimes {}^{f}q_{if} \\ {}^{o}q_{it} &= {}^{o}q_t \otimes {}^{t}q_{it} = [q_z(\theta) \otimes q_x(\pi/2)] \otimes {}^{t}q_{it} \end{aligned} \tag{2.15} $$

Para el abordaje del problema, se computaron los ${}^{s}q_{is}$ con las ecuaciones del tercer caso (realista) a partir de una aproximación coherente de $q_z(\theta)$ con las ecuaciones del segundo caso. Para el cálculo de $\hat{q_z}(\theta)$ primero se despejó de la Ecuación 2.12 de la pelvis por ser más directa, lo cual no afectará al resultado de los otros dos segmentos pues ${}^{o}q_s$ es igual para todos bajo las condiciones de medida de la calibración.

$$ q_z(\theta) = {}^{o}q_i \otimes (q_x(\pi/2) \otimes q_x(\pi/2) \otimes q_y(\pi/2))^* \tag{2.16} $$

Seguidamente, como el giro descrito por $q_z(\theta)$ según la Ecuación 2.16 en realidad conocemos que se ve afectado por $q_{xz}(\alpha, \beta)$, no podemos tomar este resultado directamente como $\hat{q_z}(\theta)$. En su lugar, emplearemos la función *q2zxy* (Anexo 8.1) para expresar los giros del $q_z(\theta)$ calculado por medio de la secuencia de ángulos de Euler ZXY según las recomendaciones de Diebel et al. 2006 (Ecuación 2.24). De esta forma, tomaremos únicamente el giro $\hat{\theta}$ entorno al eje Z, donde conocemos que la componente principal es el verdadero giro θ, y calcularemos un $\hat{q_z}(\theta)$ como un cuaternión unitario describiendo un giro puro θ entorno a Z.

$$ \hat{q_z}(\theta) = q_z(\hat{\theta}) = \left\langle \cos\frac{\hat{\theta}}{2}, 0, 0, \sin\frac{\hat{\theta}}{2} \right\rangle \tag{2.17} $$

Desde este punto, despejando de la Ecuación 2.15 para la pelvis, fémur y tibia se obtiene:

$$ \begin{aligned} {}^{p}q_{ip} &= [\hat{q_z}(\theta) \otimes q_x(\pi/2)]^* \otimes {}^{o}q_{ip} \\ {}^{f}q_{if} &= [\hat{q_z}(\theta) \otimes q_x(\pi/2)]^* \otimes {}^{o}q_{if} \\ {}^{t}q_{it} &= [\hat{q_z}(\theta) \otimes q_x(\pi/2)]^* \otimes {}^{o}q_{it} \end{aligned} \tag{2.18} $$

Donde conocemos $\hat{q_z}(\theta)$ de la Ecuación 2.17, $q_x(\pi/2)$ de la Ecuación 2.9 y ${}^{o}q_{it}$ lo tenemos de los datos. Siguiendo estos pasos, se obtuvieron los cuaterniones ${}^{s}q_{is}$ para todos los sujetos a partir de las medidas de calibración. Una vez hecho esto, se empleó la Ecuación 2.4 con los datos (${}^{o}q_{is}$) de los periodos de marcha y se calcularon los ${}^{o}q_s$ para los tres segmentos en estos casos dinámicos a partir de los ${}^{t}q_{it}$ previamente calculados (Ecuación 2.21).

$$ \begin{aligned} {}^{o}q_p &= {}^{o}q_{ip} \otimes {}^{p}{q_{ip}}^* \\ {}^{o}q_f &= {}^{o}q_{if} \otimes {}^{f}{q_{if}}^* \\ {}^{o}q_t &= {}^{o}q_{it} \otimes {}^{t}{q_{it}}^* \end{aligned} \tag{2.19} $$

Comprobación de la calibración

Por último, antes de extraer los ángulos de cadera y rodilla, se comprobó que la diferencia entre la orientación ${}^{s}R_{i}$ calculada con los datos de la calibración inicial y la orientación ${}^{s}R_{i}$ calculada con los datos de la calibración final, fuera lo suficientemente pequeña como para atribuirla a un pequeño error de medida (JKO s.f.). Esto es, idealmente, que el máximo valor absoluto del ángulo de giro descrito por el cuaternión resultante q_{ci}^{cf} del producto de los ${}^{s}q_{i}{}^{ci}$ con ${}^{s}q_{i}{}^{cf}$ para cada uno de los segmentos fuera inferior a 3° o, en su defecto, no mayor de 6°:

$$q_{ci}^{cf} = {}^{s}q_{i}{}^{cf} \otimes {}^{s}q_{i}{}^{ci} \tag{2.20}$$

El resultado de la Ecuación 2.20 se implementó en la función *inicio_fin* (Anexo 8.1) donde se extrajo el ángulo de giro entre la orientación de la calibración inicial y la de la calibración final (θ_{ci}^{cf}) mediante la ecuación:

$$\theta_{ci}^{cf} = 2 \cdot \arctan\left(\frac{\|v\left(q_{ci}^{cf}\right)\|_2}{a\left(q_{ci}^{cf}\right)}\right) \tag{2.21}$$

Donde a y v representan la parte escalar y vectorial del cuaternión correspondiente y $\|\vec{v}\|_2$ es la norma euclídea del vector $\vec{v}$ obtenida mediante la función 'norm' del submódulo 'linalg' para álgebra lineal de la librería NumPy.

Los resultados de la función *inicio_fin* se muestran en la Tabla 2.3, donde constan los tres cuartiles de θ_{ci}^{cf} para cada sujeto. Esta primera visualización, nos mostró que durante algunas pruebas el deslizamiento del sensor fue superior al límite establecido. Específicamente, en los sujetos S01, S05 y S08 los valores del ángulo de giro θ_{ci}^{cf} son muy superiores al límite de 6° para pelvis, fémur y tibia, por lo cual fueron marcados *a priori* para el descarte a través de la variable booleana 'Desliz'.

Tabla 2.3: Cuartiles del ángulo diferencia entre la orientación del sensor respecto a cada segmento corporal en la calibración inicial frente a la calibración final.

Sujeto	Desliz	Cuartil	Pelvis	Fémur	Tibia
1	0	Q1	-0.60	-0.75	-0.67
		Q2	-0.53	-0.68	-0.60
		Q3	-0.46	-0.60	-0.52
2	0	Q1	1.00	0.94	0.93
		Q2	1.16	1.10	1.09
		Q3	1.37	1.31	1.31
3	1	Q1	7.91	8.30	5.99
		Q2	8.51	8.65	6.72
		Q3	8.54	8.72	6.76
4	0	Q1	1.15	0.92	1.21
		Q2	1.30	1.06	1.35
		Q3	1.43	1.19	1.48
5	0	Q1	-0.70	-0.66	-0.50
		Q2	-0.17	-0.12	0.04
		Q3	-0.06	-0.01	0.15
6	0	Q1	-0.47	-0.35	-0.57
		Q2	-0.14	-0.01	-0.23
		Q3	0.14	0.27	0.05
7	1	Q1	9.47	5.04	6.24
		Q2	9.75	5.31	6.51
		Q3	9.82	5.41	6.60
8	0	Q1	-0.23	-0.37	-0.35
		Q2	-0.08	-0.21	-0.20
		Q3	0.09	-0.05	-0.03

2.5.2 *Extracción de Variables*

Una vez se obtuvieron los cuaterniones de interés y se comprobó que las orientaciones al inicio y al final de las pruebas fueran lo suficientemente similares, se procedió a la obtención o cálculo de las variables de mayor interés.

Aceleración y velocidad angular

A modo de visualizar los datos obtenidos en crudo, se extrajeron los valores de la aceleración y la velocidad angular en cada uno de los tres ejes a partir de los valores de los registros exportados de las IMUs (Figuras 2.11 y 2.12) mediante las funciones *plot_acel* y *plot_gyro*.

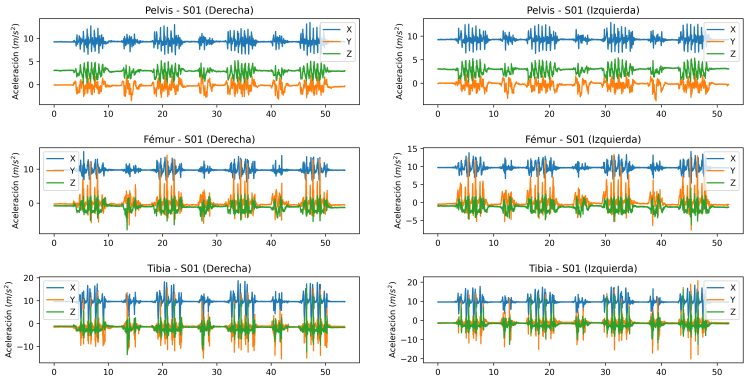


Figura 2.11: Valores de aceleración triaxial exportados de los registros de las IMUs durante la marcha para el sujeto S01.

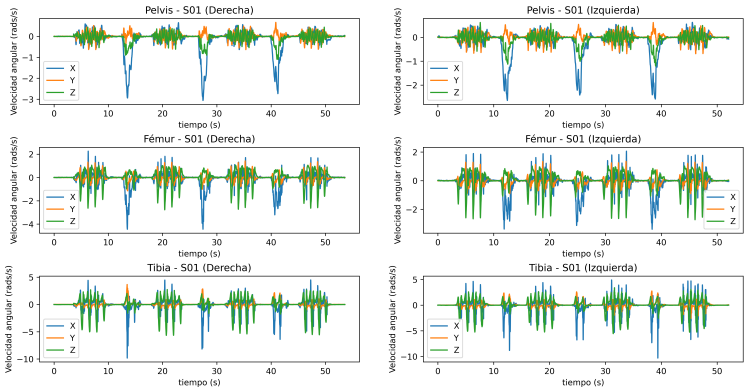


Figura 2.12: Valores de velocidad angular triaxial exportados de los registros de las IMUs durante la marcha para el sujeto S01.

Además, en estas funciones también se calcularon los valores de la norma de cada una de estas variables se calculó con la función 'norm'. En las Figuras 2.13 y 2.14 se pueden observar los valores de las normas de la aceleración y la velocidad angular para cada uno de los segmentos de un sujeto en concreto, las cuales se obtuvieron por medio de la función 'subplot' del submodulo 'pyplot' de la librería Matplotlib.

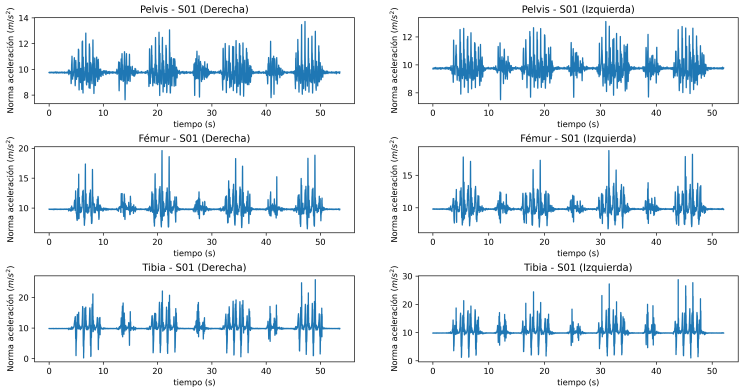


Figura 2.13: Valores de la norma Euclídea de la aceleración durante la marcha para el sujeto S01.

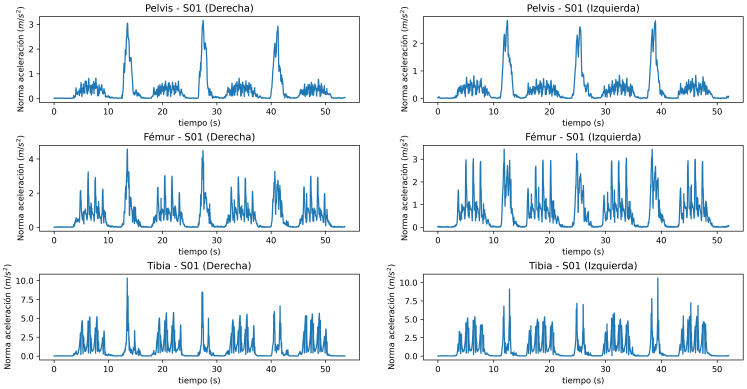


Figura 2.14: Valores de la norma Euclídea de la aceleración durante la marcha para el sujeto S01.

Ángulos de cadera y rodilla

Para lograr el objetivo de este trabajo, se emplearon las normas Euclídeas de los ángulos de la cadera y de la rodilla durante la marcha. El cálculo de estos se realizó en la función *euler_zxy* (Anexo 8.1), donde se calcularon los ángulos de Euler de la cadera y la rodilla empleando los cuaterniones ${}^{o}q_{p}$, ${}^{o}q_{f}$ y ${}^{o}q_{t}$ calculados en el apartado anterior (Ecuación 2.21). El giro correspondiente a dichos ángulos se definió por ${}^{p}q_{f}$ (Ecuación 2.22) que describe la rotación del SdR del fémur al SdR de la pelvis y por ${}^{f}q_{t}$ (Ecuación 2.23) que describe la rotación del SdR de la tibia al SdR del fémur. Tras obtener los cuaterniones de ambas articulaciones, los ángulos de la cadera y la rodilla durante la marcha se hallaron a partir de la matriz de rotación definida por la Ecuación2.24, implementada en la función .

$$ {}^{p}q_{f} = {}^{o}q_{p}{}^{*} \otimes {}^{o}q_{f} \tag{2.22} $$

$$ {}^{f}q_{t} = {}^{o}q_{f}{}^{*} \otimes {}^{o}q_{t} \tag{2.23} $$

Tras obtener los cuaterniones ${}^{p}q_{f}$ y ${}^{f}q_{t}$, se empleó la función llamada *q2zxy* (Anexo 8.1) para obtener secuencia de ángulos de Euler ZXY (Figura 2.15) de ambas articulaciones a partir de estos cuaterniones según las recomendaciones de Diebel et al. 2006 (Ecuación 2.24).

$$u_{zxy}(q) = \begin{bmatrix} u_z \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} atan2\left(\dfrac{2 \cdot q_1 \cdot q_2 + 2 \cdot q_0 \cdot q_3}{q_0^2 - q_1^2 + q_2^2 - q_3^2}\right) \\ -asin\left(2 \cdot q_2 \cdot q_3 - 2 \cdot q_0 \cdot q_1\right) \\ atan2\left(\dfrac{2 \cdot q_1 \cdot q_3 + 2 \cdot q_0 \cdot q_2}{q_0^2 - q_1^2 - q_2^2 + q_3^2}\right) \end{bmatrix} \tag{2.24}$$

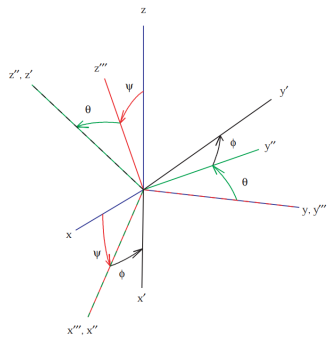


Figura 2.15: Secuencia ZXY de ángulos de Euler

De esta forma, se obtuvo el giro en cada uno de los ejes X, Y, Z en cada instante de tiempo para las dos articulaciones antedichas. A continuación, se tomó la componente del giro en el eje Z medio-lateral de los ángulos de cadera (ψ_c) y rodilla (ψ_r), que serán los datos empleados en las siguientes secciones del trabajo. La Tabla 2.4 a continuación muestra los parámetros estadísticos principales (media y desviación típica) y el número de muestras (N) para los ángulos de ψ_c y ψ_r en cada una de las dos pruebas. Además, volviendo a lo mentado en la Sección 2.5.1 e introduciendo ya el siguiente punto a tratar, la Tabla 2.4 incluye también los parámetros estadísticos y el número de muestras de los datos en función de la variable 'Desliz'.

Tabla 2.4: Parámetros estadísticos y número de muestras según el identificador del sujeto o la presencia de deslizamiento del sensor para la norma del ángulo de cadera y la norma del ángulo de rodilla en las pruebas iniciadas con la pierna derecha y en las pruebas iniciadas con la pierna izquierda.

	Ángulo Rodilla (Derecha)			**Ángulo Rodilla (Izquierda)**			**Ángulo Cadera (Derecha)**			**Ángulo Cadera (Izquierda)**		
Sujeto	Media	S,D,	N	Media	S,D,	N	Media	S,D,	N	Media	S,D,	N
0	4.86	12.53	5,360	3.76	12.70	5,207	0.63	5.43	5,360	0.85	5.75	5,207
1	6.51	13.30	5,306	6.60	12.72	5,105	6.91	7.86	5,306	9.47	9.01	5,105
2	10.26	13.24	6,097	11.04	12.24	6,299	6.92	8.32	6,097	8.89	9.01	6,299
3	13.49	16.61	5,028	13.47	15.94	4,877	12.61	16.02	5,028	15.39	15.10	4,877
4	9.71	10.90	4,759	9.05	11.09	4,677	-1.81	8.83	4,759	-0.80	9.24	4,677
5	6.19	14.91	4,426	3.67	14.03	4,347	1.67	8.30	4,426	3.05	9.16	4,347
6	6.61	12.83	4,997	7.42	12.78	4,800	4.31	7.29	4,997	5.42	8.38	4,800
7	9.44	13.13	4,850	9.40	11.90	4,833	6.48	10.83	4,850	6.91	10.38	4,833
Desliz	Media	S,D,	N	Media	S,D,	N	Media	S,D,	N	Media	S,D,	N
0	8.33	13.97	29,729	7.66	13.58	29,046	4.50	11.16	29,729	5.88	11.54	29,046
1	8.61	13.18	11,094	9.47	12.61	11,099	5.74	7.98	11,094	7.39	8.91	11,099

Análisis estadístico del efecto de la calibración

En la Sección 2.5.1, las pruebas de los sujetos S03 y S07 mostraban una diferencia mayor a 6° para θ_{ci}^{cf} potencialmente debido al deslizamiento de sensor durante las pruebas. Sin embargo, en lugar de eliminar directamente los registros afectados, se decidió contemplar si realmente había diferencias significativas entre las medias respecto al resto de sujetos debidas a la variable 'Desliz'. Para ello, en la función *check_inicio_fin* (Anexo 8.1), se llevó a cabo un análisis estadístico del efecto de dicho factor sobre los recién calculados ángulos ψ_c y ψ_r para cada una de las dos pruebas.

En una primera instancia, se comprobó si la distribución de los datos se podía asumir normal mediante varios gráficos y tests estadísticos. El *box plot* de los datos (Figura 2.16) muestra distribuciones con alto grado de sesgo y una kurtosis considerable.

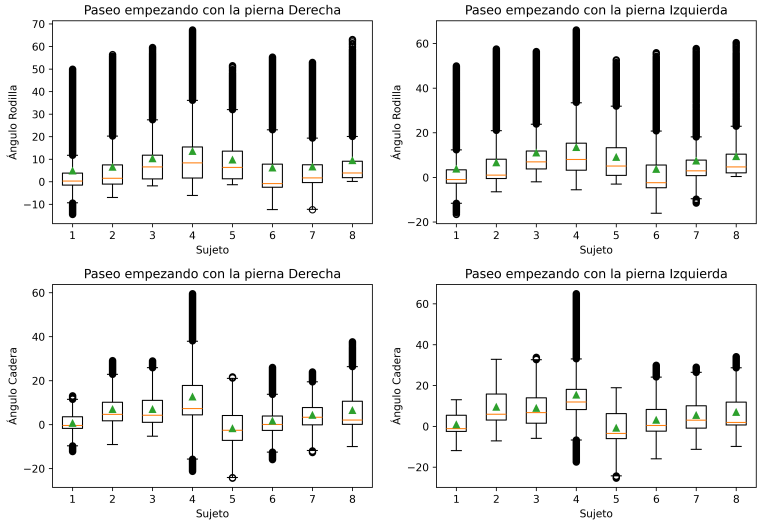


Figura 2.16: Box plot de los ángulos de cadera y rodilla según la prueba

Para asegurarnos de la no normalidad de los datos, se obtuvo también un gráfico QQ (cuantil-cuantil) (Figura 2.17) donde se observa la clara diferencia entre la distribución de los datos y una distribución normal (bisectriz del primer cuadrante).

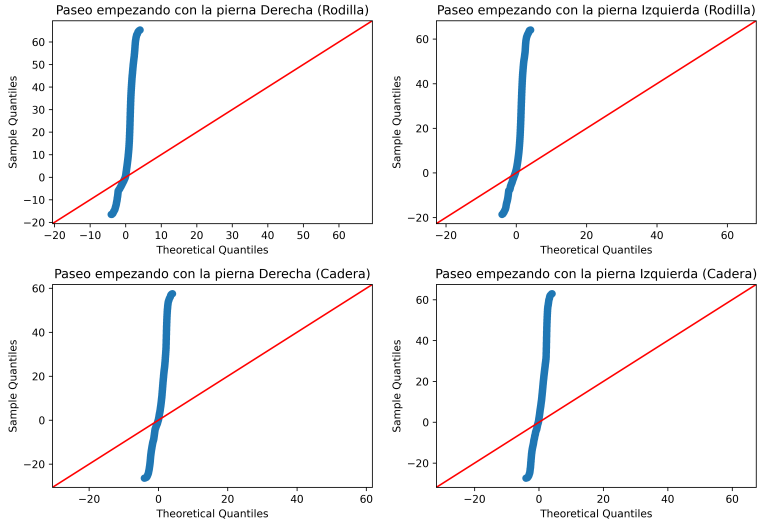


Figura 2.17: Gráfico cuantil-cuantil de los ángulo de cadera y rodilla según la prueba

Esto fue confirmado mediante los estadísticos estandarizados (Z-score) y los p-valores obtenidos para los tests de sesgo y kurtosis. Los valores recogidos en la Tabla 2.5 evidenciaron lo alejados que se hallaban los valores de sesgo y kurtosis de los parámetros de la distribución normal correspondiente ($p<0.05$).

Una vez confirmada la no normalidad de los datos, se trató de corregir por medio de la transformación Box-Cox (Osborne 2010) tras la cual el *box plot* pasó a ser el de la Figura 2.18 que ya se asemeja más a una normal.

De forma similar, el gráfico QQ (Figura 2.19) mostró una recta cuantil-cuantil mucho más semejante a la que correspondería para la distribución normal correspondiente.

Tabla 2.5: Tests de normalidad de los ángulos de rodilla y cadera según la prueba.

	Sesgo		Kurtosis	
	Z-Score	p-Value	Z-Score	p-Value
Ángulo Rodilla (Derecha)	7.25	**0.000**	-105.01	**0.000**
Ángulo Rodilla (Izquierda)	4.29	**0.000**	-56.81	**0.000**
Ángulo Cadera (Derecha)	0.98	**0.327**	-64.99	**0.000**
Ángulo Cadera (Izquierda)	-0.16	**0.871**	-56.85	**0.000**

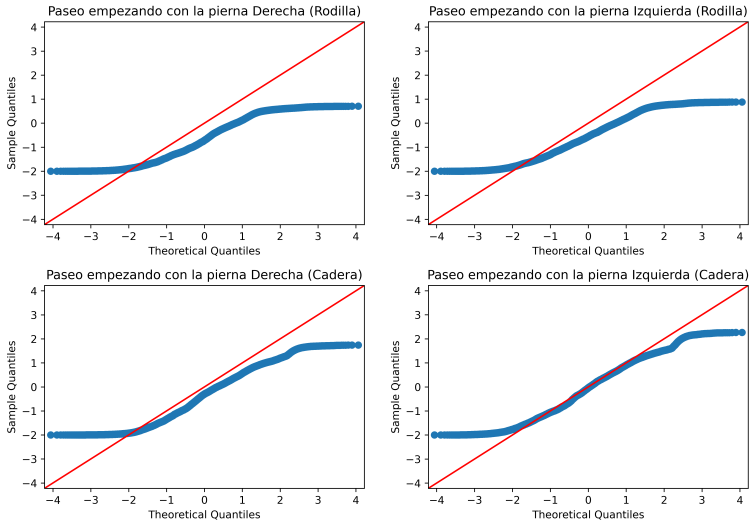


Figura 2.18: Box plot de los ángulos de cadera y rodilla según la prueba tras la transformación Box-Cox.

Tras evaluar de nuevo los tests de sesgo y kurtosis (Tabla 2.6), se reparó en que la diferencia del sesgo tras la transformación frente al sesgo de la distribución normal correspondiente ya no era estadísticamente significativa ($p>0.05$). Sin embargo, la kurtosis de algunos casos fue demasiado diferente a la kurtósis de una normal ($p<0.05$). Por lo tanto, quedaba en duda que realmente la transformación aplicada hubiera convertido la distribución de los datos a una normal.

Tabla 2.6: Tests de normalidad de los ángulos de rodilla y cadera según la prueba tras la transformación Box-Cox

	Sesgo		Kurtosis	
	Z-Score	p-Value	Z-Score	p-Value
Ángulo Rodilla (Derecha)	7.25	**0.000**	-105.01	**0.000**
Ángulo Rodilla (Izquierda)	4.29	**0.000**	-56.81	**0.000**
Ángulo Cadera (Derecha)	0.98	0.327	-64.99	**0.000**
Ángulo Cadera (Izquierda)	-0.16	0.871	-56.85	**0.000**

Por otro lado, se evaluó la hipótesis de homocedasticidad (i.e., igualdad de varianza entre las poblaciones, en este caso, sujetos) empleando el test de Levene. Los resultados indicaron que las diferencias entre las varianzas eran, en efecto, estadísticamente significativas para un error de primera especie (α) del 5 % ($p<0.05$).

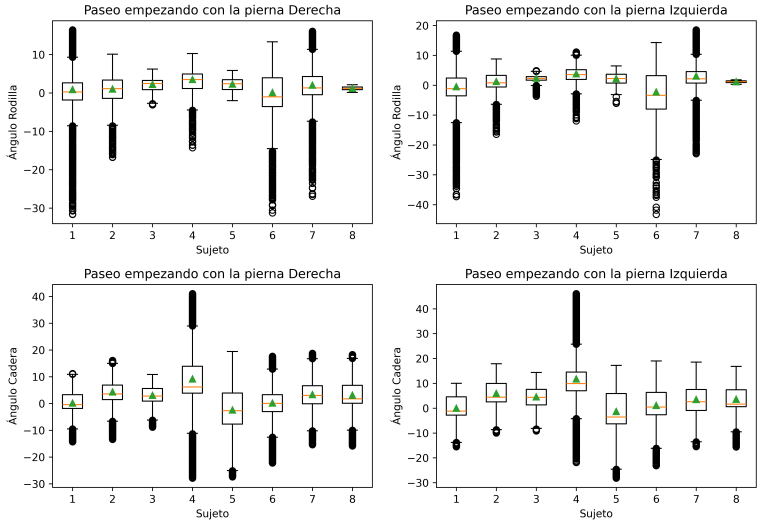


Figura 2.19: Gráfico cuantil-cuantil de los ángulos de cadera y rodilla según la prueba tras la transformación Box-Cox.

Teniendo en cuenta todo lo anterior, se decidió efectuar un Análisis de la Varianza (ANOVA) de 2 factores con términos de interacción, con errores estándar robustos a la heterocedasticidad presente en los datos mediante la corrección 'HC3' que permite ajustar la matriz de covarianza de los coeficientes. La Tabla 2.7 muestra los resultados del ANOVA, donde se observa que el factor sujeto es estadísticamente significativo ($p<0.05$) en todos los casos, mientras que el factor 'Desliz', que se pensaba que podría estar afectando a las medidas, no resultó estadísticamente significativo ($p>0.05$) en ningún caso. Además, se observa como en el caso de las pruebas realizadas empezando a caminar con la pierna (derecha) opuesta a donde se encontraban los sensores, el efecto de la interacción entre el factor 'Sujeto' y el factor 'Desliz' es estadísticamente significativo para los ángulos tanto de cadera como de rodilla ($p<0.05$). En cambio, para las pruebas realizadas empezando a caminar con la pierna (izquierda) donde se colocaron los sensores, el efecto de la interacción no es estadísticamente significativo ($p>0.05$).

Estos resultados indican un gran efecto del sujeto en la media y varianza de las medidas de los ángulo como se puede ver también en el *box plot* de la Figura 2.18. Por ende, puesto que el efecto del deslizamiento del sensor dista notablemente de ser significativo y tomando en consideración el bajo número de sujetos que participaron en el estudio, se concluyó que lo más apropiado dadas las circunstancias sería mantener los datos originales sin eliminar los registros de ningún sujeto.

2.5.3 Detección de los Periodos Dinámicos

En la dinámica de la marcha, la cadera es fundamental en la estabilización de la pelvis y la coordinación de ambos miembros inferiores. Esto hace que posea una amplia variedad y complejidad de movimientos (i.e., flexión, extensión, aducción, abducción, rotación externa, rotación interna) que le permiten adaptarse al terreno y amortiguar las fuerzas de impacto durante la marcha para facilitar el movimiento normal de los miembros inferiores (Ramakrishnan y Kadaba 1991). En comparación con la rodilla, ψ_c tiene menor amplitud que ψ_r y mayor variabilidad dado que el movimiento de la cadera se ejecuta en las tres direcciones, mientras que el de la rodilla se halla contenido principalmente en el plano sagital (Zhang, Vogler y Metaxas 2007). Bajo estas condiciones, se decidió tomar únicamente la componente en el eje Z medio-lateral de los ángulos de la rodilla y cadera en lugar de la norma, dado que el computar la norma nos habría hecho perder información del plano sagital, especialmente para la cadera. Incluso así, al ser más fácil detectar los cambios de ψ_r durante la marcha, este serán los datos empleados como señal principal para segmentar los periodos de marcha y las fases de los ciclos de la marcha. Una vez segmentado ψ_r, se replicará la clasificación a ψ_c para mantener la coherencia.

Con la función de *segmentación* de la señal mostrada en el Anexo 8.1 y aquí explicada, se pretendió abordar de forma sistemática la tarea de análisis y clasificación de los segmentos de la señal por medio de la aplicación de técnicas de transformación de la señal y umbralización dinámica con ventana deslizante. La función recibe tres parámetros: el vector de la señal 's' (que se corresponde con ψ_r), el vector de tiempo 't' y una constante

Tabla 2.7: ANOVA de dos factores con interacción y con coeficientes de matriz de covarianza corregidos para heterocedasticidad para determinar el efecto del deslizamiento de los sensores sobre los ángulos de rodilla y cadera según la prueba.

Ángulo Rodilla (Derecha)					
Fuente	df	sum_sq	mean_sq	F	PR(>F)
Sujeto	7	773.07	110.44	227.62	**0.000**
Desliz	1	0.53	0.53	1.10	0.294
Interacción	7	3.59	0.51	1.06	0.390
Resíduos	40,814	19,802.31	0.49	-	-
Ángulo Rodilla (Izquierda)					
Fuente	df	sum_sq	mean_sq	F	PR(>F)
Sujeto	7	1,485.84	212.26	478.42	**0.000**
Desliz	1	1.06	1.06	2.39	0.122
Interacción	7	3.88	0.55	1.25	0.271
Resíduos	40,135	17,806.88	0.44	-	-
Ángulo Cadera (Derecha)					
Fuente	df	sum_sq	mean_sq	F	PR(>F)
Sujeto	7	2,760.30	394.33	580.42	**0.000**
Desliz	1	1.45	1.45	2.14	0.144
Interacción	7	9.59	1.37	2.02	**0.049**
Resíduos	40,814	27,728.33	0.68	-	-
Ángulo Cadera (Izquierda)					
Fuente	df	sum_sq	mean_sq	F	PR(>F)
Sujeto	7	4,763.89	680.56	976.95	**0.000**
Desliz	1	0.42	0.42	0.61	0.436
Interacción	7	1.46	0.21	0.30	0.954
Resíduos	40,135	27,958.67	0.70	-	-

'k_e' para regular los parámetros internos. Tras cada ejecución, tres parámetros fueron devueltos: una figura para representar el proceso 'fig6', el tamaño de ventana 'k' y la clasificación de cada una de las muestras 'cc'.

Cálculo del tamaño de la ventana

En concreto, la clasificación de la señal se llevó a cabo empleando una ventana deslizante de tamaño constante 'k', computado al inicio de la ejecución por medio de la Ecuación 2.25. De este modo, se logró un tamaño de ventana proporcional al número de datos en la señal (N) y directamente proporcional a la k_e veces la desviación estándar de la señal σ_s. En base a los resultados heurísticos por prueba y error, se determinó el mejor valor de k_e para la clasificación de los periodos dinámicos como $k_e = 2.35$.

$$k = Ent\left(\frac{N}{k_e \cdot \sigma_s}\right) \tag{2.25}$$

Donde Ent indica la parte entera de la división. Es decir, se trata de una división entera (i.e., resto nulo).

Transformación de la señal

A continuación, la señal se transformó por la Ecuación 2.27 para maximizar los valores más altos durante los periodos dinámicos (especialmente los periodos de marcha) y minimizar los más bajos durante los periodos de pausa. Para ello, se calculó la exponencial del logaritmo de la señal y se descontó el sesgo o calculado para la muestra final de la primera ventana (Ecuación 2.26). Esto facilitó la umbralización y corrigió el sesgo de la medida adecuadamente.

$$o = e^{k_e \cdot ln(s[k]+min(s)} - min(s) \tag{2.26}$$

$$ss = e^{k_e \cdot ln(s+min(s)} - o \tag{2.27}$$

Dado que algunas señales podían presentar una cierta de deriva d tras la transformación que influenciaba la detección de los periodos dinámicos (Ecuación 2.28), se presentaban dos posibles opciones: la primera era modificar la Ecuación 2.27 para corregir la deriva sobre la propia señal y la segunda, obtener un umbral con un sesgo linealmente dependiente del tiempo. Tras varias pruebas, se determinó que la viabilidad de la segunda opción era mayor para la mayoría de los datos.

$$d = \frac{ss[-k+100] - ss[k]}{N - (2 \cdot k + 100)} \tag{2.28}$$

Ventana deslizante

Con todos los parámetros listos, se pasó a ejecutar el bucle principal de clasificación donde se dividió la señal en $N - k$ ventanas de tamaño k para clasificar cada una de ellas iterando. Para cada ventana n, un umbral dinámico 'th' fue calculado siguiendo la Ecuación 2.29 donde el valor del límite viene determinado por el sesgo, la deriva proporcional al número de ventanas clasificadas hasta ese punto y el 5 % del máximo valor de la señal ($max(ss)$).

$$th = 0.05 \cdot max(ss) + o + d \cdot n \tag{2.29}$$

Para cada ventana de la señal transformada ss, se guardaron todo los valores comprendidos en una lista ('$suma$') y se extrae el máximo valor. Inmediatamente después se evaluó el resultado para obtener la clasificación c de la n-ésima ventana indicando con un '0' o un '1' si el máximo valor superaba o no el límite th, es decir, si el sujeto se hallaba estático o no (Figura 2.20.a). El contra de este método es que, si bien sirve para discernir los periodos estáticos de los dinámicos, el umbral no diferencia entre los periodos de marcha y los periodos de giro. No obstante, en la Sección 2.5.4 a continuación se solventará pragmáticamente esta cuestión, reajustando c para contener únicamente los periodos dinámicos de marcha.

Si bien la clasificación de los periodos dinámicos era precisa, esta mostraba un desfase significativo respecto a la señal observada debido al enventanado de la señal (Figura 2.20.b). Dado que el bucle para clasificar las ventanas no cubría todas las muestras sino $N - k$, el resultado presentaba dicho desfase. Para corregirlo, se creó el vector cc para la clasificación final de todas las muestras donde se eliminaron los primeros $Ent(k/2) + Res(k/2)$ valores del vector de clasificación c y se añadió el último valor de c igual número de veces. De esta forma, se logró resolver el desfase y mejorar el resultado de la clasificación (Figura 2.20.c).

2.5.4 Segmentación y Etiquetado Manual de las Fases

A modo de referencia para los resultados del Capítulo 3 se etiquetaron manualmente de las dos fases principales de la marcha. La encargada de realizarlo fue la función *labeling* (Anexo 8.1), la cual recibe múltiples parámetros: la señal original ψ_r 'señal_sin_segmentar', el vector de tiempo de esta 'tiempo_sin_segmentar', la clasificación c obtenida en el paso anterior 'clas', el identificador del sujeto 'j', la articulación de la que provienen los datos 'joint' (que en este caso será siempre la rodilla) y el lado con el que se empieza a caminar en la prueba 'lado' (quitando las calibraciones, la primera prueba fue empezando con la pierna derecha y la segunda con la izquierda). Una vez ejecutada, esta función devolvió: dos figuras para representar el proceso 'fig8' y 'fig9', una lista con los índices de las muestras que marcan el inicio y el final de cada periodo de marcha 'indices_on_off' y otra lista con los índices que marcan el inicio y el final de cada una de las dos fases (apoyo y balanceo) de cada ciclo de marcha para cada uno de los 4 periodos de marcha realizados. No obstante, internamente, la función crea dos ficheros .csv, el primero para guardar la segmentación de los periodos de marcha (e.g., 'umbral_Rodilla_ciclos_Derecha_S01.csv') y el segundo para guardar la segmentación de las fases de cada uno de los ciclos de los 4 periodos de marcha (e.g., 'labels_Rodilla_fases_Derecha_S01.csv'). Realmente, estos son los resultados principales de esta función y por eso se exportaron.

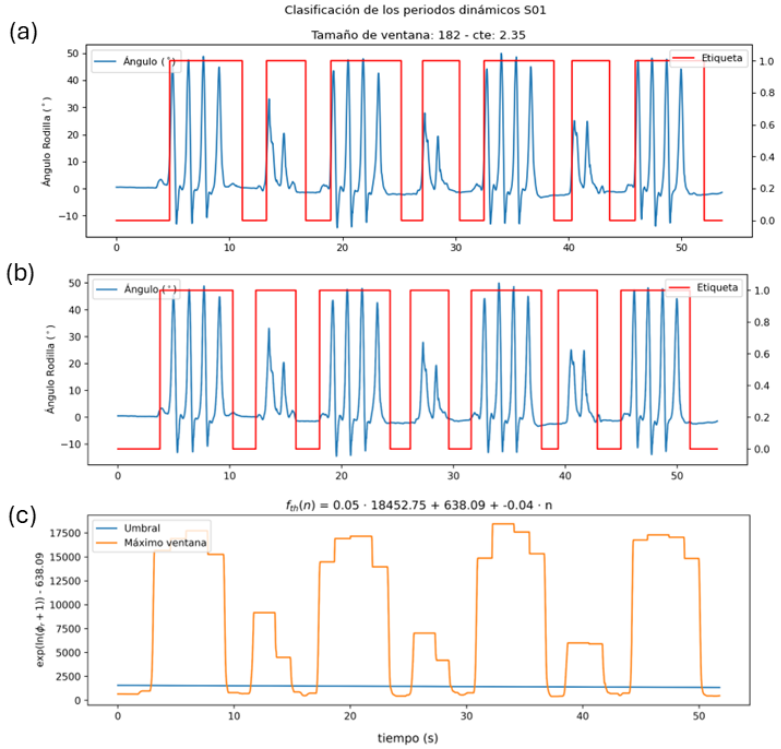


Figura 2.20: Resultado de (a) clasificación inicial de los periodos desfasada, (b) corrección del desfase de la clasificación y (c) umbralización adaptativa para el sujeto S01.

Segmentación de los periodos de marcha

La primera parte del proceso implicó segmentar la señal original ψ_r (señal_sin_segmentar) por los puntos de transición ('bp') donde el valor del vector de clasificación c obtenido en el paso anterior pasaba de 0 a 1 o viceversa. Excluyendo las transiciones que involucraran los giros de 180° realizados entre periodos, se obtuvieron los periodos de marcha de cada prueba segmentados ('señal_segmentada') y los instantes de tiempo correspondientes ('tiempo_segmentado'), a la par que se reajustó la clasificación c para obtener únicamente los periodos de marcha (imputando manualmente a '0' los periodos de giro). Tras esto, la segmentación correcta de sólo los periodos de marcha (Figura 2.21) se guardó en un DataFrame de la librería Pandas y se exportó en formato .csv con la función 'to_csv' perteneciente a dicha librería.

Etiquetado manual de las fases de la marcha

En sujetos sanos, cada ciclo de marcha de ψ_r se compone aproximadamente un 60 % por la fase de apoyo y un 40 % de la fase de balanceo (Tabla 1.1). El contacto inicial del talón (HS) define el inicio de la marcha y el inicio de la fase de apoyo, durante la cual ψ_r comienza a incrementarse lentamente con el final de la flexión de la cadera desde los 5-10° hasta los 15-20° antes de volver a descender durante la extensión de cadera, dando lugar a una primera parábola invertida. En los últimos 0.2-0.3 segundos de la fase de apoyo, a medida que se va despegando la planta del pie desde el talón, ψ_r comienza a aumentar cada vez más rápido hasta alcanzar unos 30°, momento en el cual el pie deja de estar en contacto con el suelo. El despegue del antepié del suelo (TO) define este instante entre el final de la fase de apoyo y el inicio de la fase de balanceo, momento durante el cual ψ_r alcanza su mayor velocidad de aumento durante todo el ciclo y llega hasta los 60-70° antes de volver a descender a los 5-10 ° para dar paso al HS del siguiente ciclo, dando lugar a una segunda parábola invertida con una pendiente mucho más abrupta (Ramakrishnan y Kadaba 1991). Teniendo estos criterios en cuenta, fue factible la identificación manual de los eventos clave.

Para ello, se representaron gráficamente cada uno de los periodos de marcha guardados en señal_segmentada y se llamó a la función 'ginput' del submodulo 'pyplot' de la librería Matplotlib para que nos permitiera interactuar con las figuras creadas y seleccionar manualmente los puntos de transición (eventos). La Figura 2.22 muestra, a modo de ejemplo, la selección de los puntos que definen ambas fases para un periodo de marcha de un sujeto concreto.

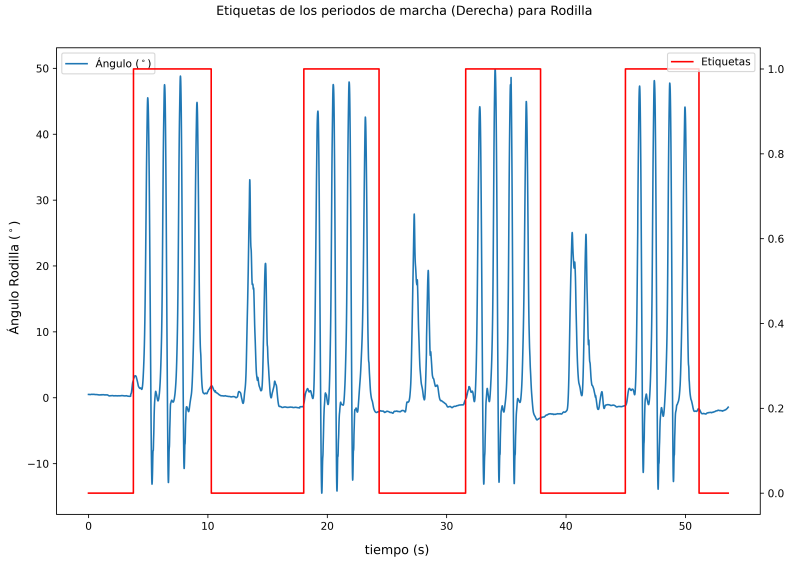


Figura 2.21: Segmentación de los periodos de marcha según el ángulo de flexo-extensión de la rodilla para la prueba empezando con la pierna derecha del sujeto S01.

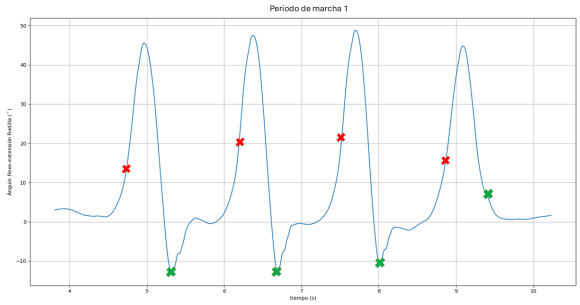


Figura 2.22: Selección manual de los dos eventos de la marcha (HS en verde y TO en rojo) para el primer periodo de marcha del sujeto S01.

En un proceso similar al de la Sección 2.5.4, una vez hallados los eventos se clasificó cada muestra de cada periodo de marcha en la señal_segmentada según perteneciera a la fase de apoyo o balanceo. Estas se representaron con ’1’ o ’0’, respectivamente, a modo de indicar el contacto con el suelo como ’Verdadero’ para la fase de apoyo y ’Falso’ para el balanceo (Figura 2.23). Para la exportación, los periodos de marcha segmentados, los vectores de tiempo asociados y las etiquetas imputadas manualmente (almacenadas en la variable ’labels_fases’) se guardaron en un DataFrame para cada uno de los periodos de marcha. Finalmente, se concatenaron los DataFrames creados en un único DataFrame ’df_ciclos’ que fue exportado a formato .csv mediante to_csv.

Detección, Segmentación y Etiquetado de Ángulo de Cadera

Como se mencionó al inicio de la Sección 2.5.3, los pasos de detección, segmentación y etiquetado hasta este punto se han llevado a cabo únicamente para ψ_r. Para mantener la coherencia, ψ_c se obtuvo a partir de los resultados de su contrapartida con la función *copy_label*. De este modo, los periodos de marcha de ψ_c se segmentaron en base a los periodos de marcha detectados para ψ_r (Figura 2.24), y lo mismo se hizo con la segmentación de los ciclos de marcha y el etiquetado de las fases (Figura 2.25).

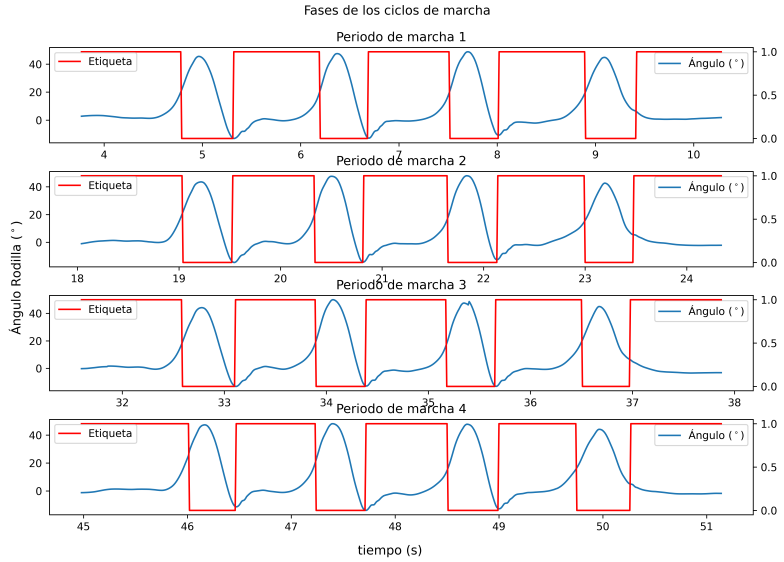


Figura 2.23: Clasificación de las fases de los periodos de marcha en función de el etiquetado manual según el ángulo de flexo-extensión de la rodilla para la prueba empezando con la pierna derecha del sujeto S01.

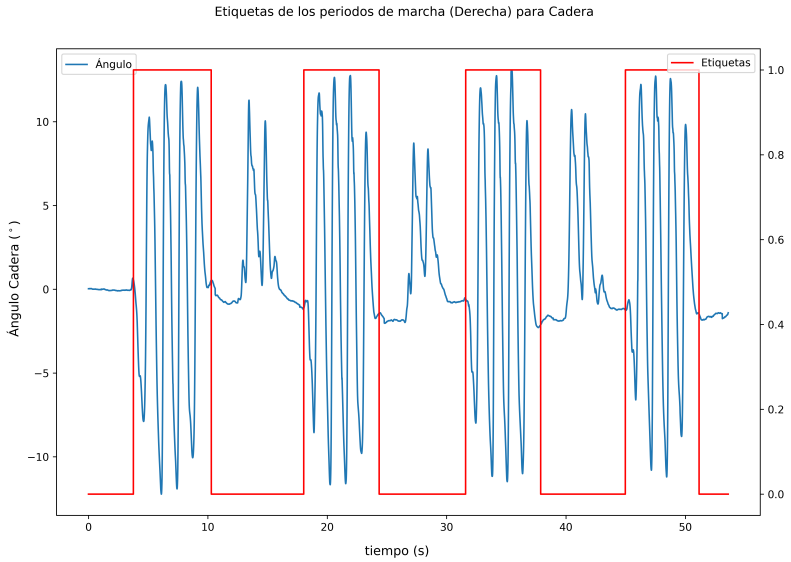


Figura 2.24: Segmentación de los periodos de marcha del ángulo de flexo-extensión de la cadera a partir de la segmentación del ángulo de flexo-extensión de la rodilla para la prueba empezando con la pierna derecha del sujeto S01.

Cálculo del porcentaje de ciclo y segmentación de los ciclos de marcha

Finalmente, el último paso que se llevó a cabo dentro del Preprocesado de los datos es el cálculo del porcentaje de ciclo (Θ) de marcha para cada ciclo y la segmentación de los ciclos (i.e., pasos). Esto se efectuó por medio de la función *phase_time_percentage* (Anexo 8.1), la cual se basta del identificador del sujeto 'j' para obtener el porcentaje de cada ciclo de marcha 'pct_ciclo', la derivada de este 'pct_der', los ciclos de marcha segmentados 'PASOS' y el número de muestras de cada ciclo 'muestras'. Esto se debe a que anteriormente guardamos los datos necesarios en un fichero .csv externo (cuyo nombre varía únicamente por el indicador del sujeto) el cual es leído internamente por la función 'read_csv'. Análogo a cómo se segmentaron los periodos de marcha en la Sección 2.5.4, se segmentó la señal original de los periodos de marcha segmentados 'señal_real' y los vectores de tiempo de cada ciclo 't_real' en función de las transiciones automáticamente identificadas en el vector de las etiquetas de las fases 'cc'. No obstante, en este caso sólo se segmentó por los puntos donde las etiquetas pasaban de 0 a 1 (i.e., fin del balanceo e inicio del apoyo) para cada uno de los periodos de marcha.

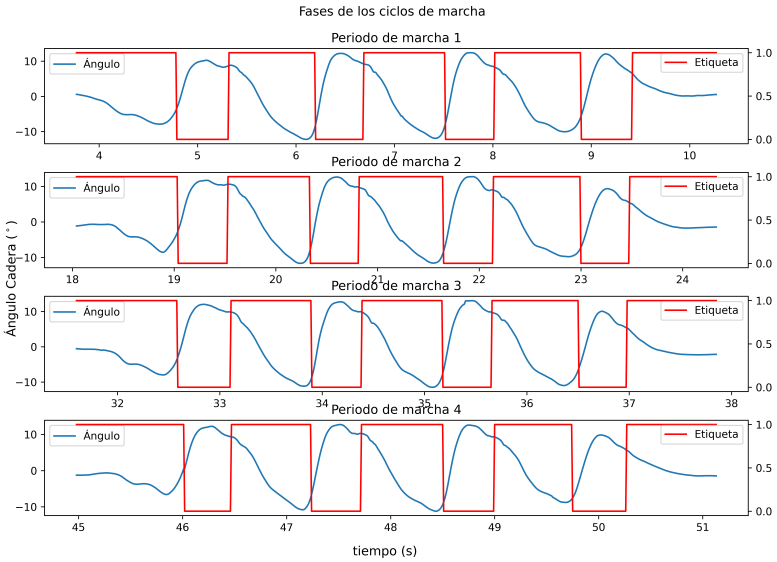


Figura 2.25: Clasificación de las fases de la marcha del ángulo de flexo-extensión de la cadera a partir de la segmentación del ángulo de flexo-extensión de la rodilla para la prueba empezando con la pierna derecha del sujeto S01.

Una vez segmentados los ciclos, se calculó el porcentaje de cada ciclo 'pct_ciclo' por medio de la función 'linspace' de la librería NumPy, creando un *array* formado por n números equiespaciados en el rango [0, 2π], donde n es el número de muestras del ciclo. La derivada del porcentaje se calculó a modo de visualizar más claramente las diferencias en las tasas de evolución de los ciclos. De forma simple, 'pct_der' se obtuvo por diferenciación numérica hacia adelante (Ecuación 2.30).

$$\Theta'(t) = \frac{\Theta(t + t_s) - \Theta(t)}{t_s} \tag{2.30}$$

Donde t_s es el tiempo de muestreo de la señal.

Por otro lado, los pasos (ciclos) segmentados y el número de muestras de cada paso fueron guardados en 'PASOS' y 'muestras', respectivamente. Además, se representaron los periodos de marcha junto al porcentaje Θ de cada ciclo del periodo y la derivada Θ' de este con la función 'plot' de la librería Matplotlib (Figuras 2.26 y 2.27).

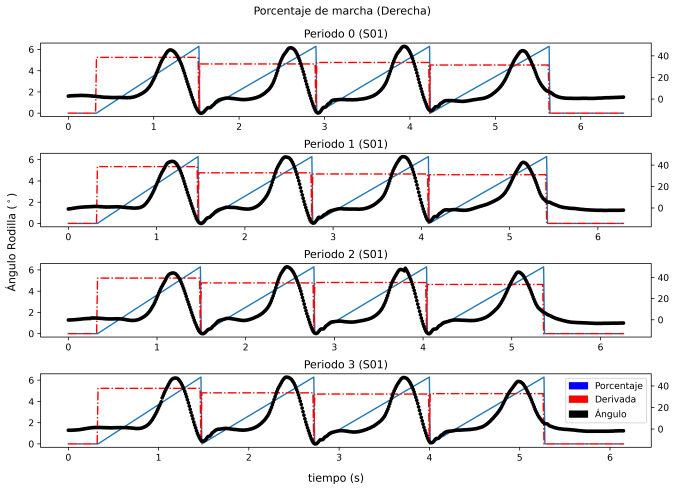


Figura 2.26: Porcentaje de los ciclos de la marcha (y su derivada) para el ángulo de flexo-extensión de la rodilla para la prueba empezando con la pierna derecha del sujeto S01.

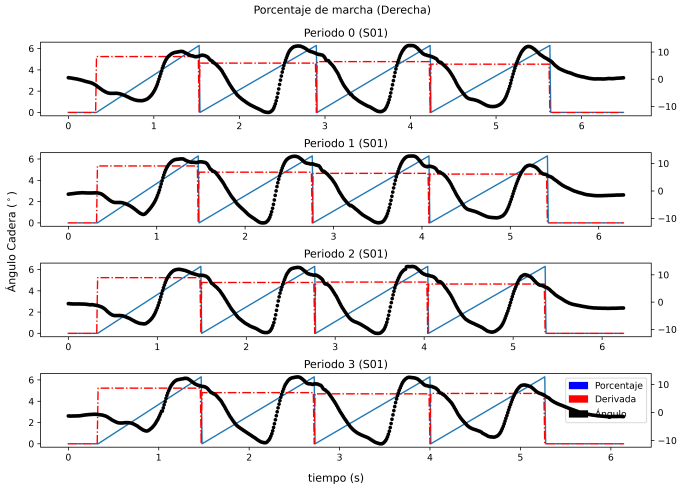


Figura 2.27: Porcentaje de los ciclos de la marcha (y su derivada) para el ángulo de flexo-extensión de la cadera para la prueba empezando con la pierna derecha del sujeto S01.

Por último, con el fin de visualizar la relación entre los ψ_c y ψ_r, se creó la función *Ciclograma* (Anexo 8.1) que recibe el indicador del sujeto 'j', los pasos segmentados 'PASOS' y el número de muestras de cada paso 'muestras'. De forma simple, para cada sujeto 'j' la función tomó los valores de ψ_c y ψ_r de todos los ciclos, los igualó en longitud repitiendo la muestra inicial y calculó el ciclo promedio para cadera ($\bar{\psi}_c$) y rodilla ($\bar{\psi}_r$). Seguidamente, se representó ($\bar{\psi}_c$) frente a ($\bar{\psi}_r$) en un ciclograma junto a los ciclos individuales en ψ_c y ψ_r distinguiéndolos por color según pertenecieran al registro de la prueba donde se comenzó a caminar con la pierna derecha o con la izquierda (Figura 2.28).

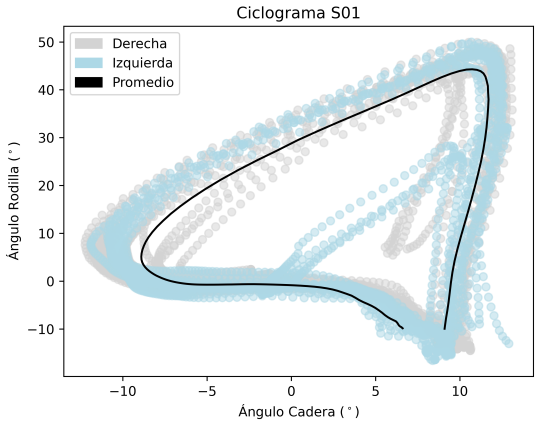


Figura 2.28: Ciclograma del ángulo de flexo-extensión de la rodilla frente al ángulo de flexo-extensión de la cadera para todos los pasos empezando con la pierna izquierda, para todos los pasos empezando con la pierna derecha y del promedio.

2.6 PREDICCIÓN DE LOS PORCENTAJES DE FASE CON *UNSCENTED KALMAN FILTER*

El filtro de Kalman (KF) es una técnica de estimación y predicción de los estados ocultos de sistemas lineales con cierto grado de estocasticidad (ruido) de naturaleza Gaussiana (i.e., sigue una distribución normal) (Sorenson 1966). Sin embargo, muchos de los sistemas del mundo real presentan no linealidad en mayor o menor medida, lo que merma la eficacia del KF tradicional. Para afrontar estas limitaciones, varias propuestas derivadas del KF tradicional fueron hechas. Una de estas, y probablemente la más utilizada debido a su alta precisión, es el '*Unscented Kalman Filter*' (UKF). Dada la índole no lineal de la marcha humana (West y Scafetta 2003), se escogió el UKF como método para la predicción de las fases de la marcha.

2.6.1 Fundamentos teóricos del Unscented Kalman Filter

Fundamentalmente, el UKF está definido entorno al concepto de la '*Unscented Transform*' (UT), un método (Julier y Uhlmann 1997) que permite calcular (propagar) los estadísticos (media y covarianza) de una variable aleatoria tras una transformación no lineal. Específicamente, la UT permite al UKF calcular la verdadera media y covarianza *a priori* de la distribución de cierta variable (normal o Gaussiana) y propagar los estadísticos *a posteriori* tras la transformación no lineal con precisión de tercer orden en la expansión por serie de Taylor para cualquier sistema no lineal. En comparación, otros KFs capaces de trabajar con sistemas no lineales como el '*Extended Kalman Filter*' (EKF) linearizan el sistema utilizando la expansión por serie de Taylor de primer orden, lo que lo hace considerablemente más errático ante sistemas altamente no lineales (Wan y Van Der Merwe 2000). Además de una mejor linearización del sistema, la complejidad computacional del UKF es comparable a la del EKF y no requieren de la computación de Jacobianos o derivadas que puede dar lugar a errores, lo que hace a los primeros un método mucho más eficiente, robusto y fácil de implementar y, en general, muy por encima del segundo en todos los aspectos.

En la práctica, la UT consiste en seleccionar un conjunto de puntos (conocidos como 'puntos sigma') a cierta distancia de la media de la distribución del estado inicial (Ecuación 2.31). En total, aparte de la media, son tomados 2 puntos por cada dimensión del vector de estados, es decir, 2N + 1 puntos sigma en total donde N es la dimensión del vector de estados. Además, cada uno de estos puntos lleva asociado un peso de una matriz de pesos, cuya suma es igual a la unidad, que determinan la contribución de cada uno a los estadísticos calculados *a posteriori*.

A continuación, estos puntos son transformados por la ecuación que define el sistema no lineal y los puntos transformados (que ya no se puede asumir que sigan la misma normal) son utilizados para calcular la nueva media y covarianza de la distribución del estado siguiente (transformado). Esta es precisamente la ventaja clave del UKF frente al EKF, ya que una simple linearización en torno a la media como en el último caso, puede dar lugar a errores altos en sistemas altamente no lineales, mientras que estos se minimizan significativamente si utilizamos la UT y recalculamos los estadísticos de la distribución de estados *a posteriori* como en el UKF (Figura 2.29).

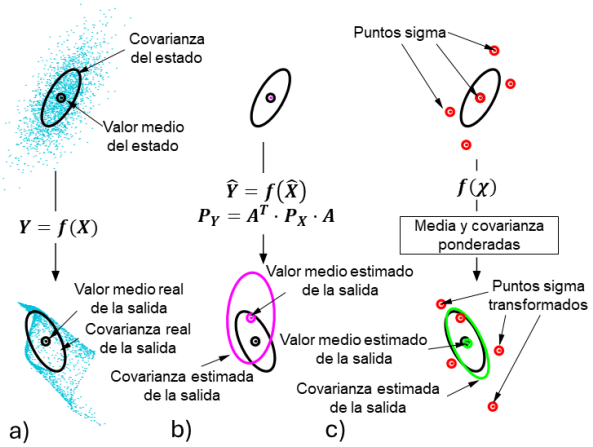


Figura 2.29: Comparativa entre (b) EKF y (c) UKF con las distribuciones predichas de las salidas frente a las reales (a) (Wan y Van Der Merwe 2000)

Cabe destacar que la UT no es tan precisa como otros métodos que emplean muestreo aleatorio repetitivo (método de Monte Carlo) y no es capaz de aproximar con la máxima precisión las distribuciones de estado que

no sigan una distribución normal. Sin embargo, empleando un volumen de datos varios ordenes de magnitud por debajo, la UT logra aproximaciones para sistemas no lineales precisas (al menos) hasta el momento de tercer orden (i.e., sesgo) en el caso de que los datos sigan una distribución Gaussiana. En caso de que la distribución de los datos no sea normal, la UT aún es capaz de lograr aproximaciones para sistemas no lineales precisas (al menos) hasta el momento de segundo orden (i.e., varianza). Asimismo, en ambos casos, la precisión de la aproximación de la UT puede alcanzar momentos de mayor orden en función de los parámetros α y β escogidos.

2.6.2 Implementación del Unscented Kalman Filter

La implementación del UKF se inicializa con una estimación inicial del (vector de) estado (X_0), una matriz de covarianza de la predicción o matriz de covarianza cruzada de los estados (P_0), la matriz de covarianza del ruido del sistema (Q) y la matriz de covarianza del ruido de la medida (R). Las dos primeras se actualizarán en los pasos siguientes, por lo que generalmente se asumen inicialmente de la forma $X_0 = [0, 0, ..., 0]_{(Nx1)}$ y $P_0 = I_{(NxN)}$. Las dos últimas deberán ser estimadas previamente a la inicialización del UKF y, si bien hay implementaciones donde el problema a tratar requiere que se vayan actualizando, en este caso, se asume que permanecerán como parámetros constantes del modelo. Una vez definidos los parámetros iniciales, el algoritmo se lleva a cabo recursivamente y se divide en tres principales pasos muy similares al KF tradicional, pero con pasos extra para la UT.

El primer paso es el 'muestreo de puntos sigma', donde se generan 2N + 1 puntos sigma (χ_t) a partir del estado actual (X_k) y la covarianza del estado actual (P_k) (Ecuación 2.31).

$$\begin{aligned} \chi_0 &= X_k \\ \chi_i &= X_k + \sqrt{(N+\lambda)\cdot P_k} \quad \forall i = 1, ..., N \\ \chi_i &= X_k + \sqrt{(N+\lambda)\cdot P_k} \quad \forall i = N+1, ..., 2N \end{aligned} \tag{2.31}$$

Donde N es la dimensión del vector de estados y λ un factor de escala que indica a cuánta distancia de la media se deben tomar los puntos sigma.

Además, en este paso se calculan también los pesos asociados a cada punto sigma. Evidenciado en la Ecuación 2.32, cuanto más se alejen los puntos sigma de la media (mayor λ), menor serán sus pesos asociados y mayor será el peso del punto de la media y viceversa. Además, los pesos variarán a cada paso puesto que son proporcionales a la desviación típica de los datos del estado actual en el tiempo k. Cabe notar que el peso del valor esperado para la covarianza de los estados ($W_0^{(c)}$) cuenta con un término extra dependiente de dos variables α y β que representan la varianza de los puntos sigma entorno a X_k y el conocimiento a priori de la distribución de X_k, respectivamente. Lo habitual es que sean valores positivos con α alcanzando valores próximos a cero y $\beta = 2$ para una distribución Gaussiana.

$$\begin{aligned} W_0^{(m)} &= W_0^{(c)} = \frac{\lambda}{N+\lambda} \\ W_i^{(m)} &= W_i^{(c)} = \frac{1}{2\cdot(N+\lambda)} \quad \forall i = 1, ..., 2N \end{aligned} \tag{2.32}$$

Seguidamente, en la 'predicción' se propagan los puntos sigma creados a través de la función no lineal que describe el sistema o proceso ($f(\chi)$) y, a partir de estos puntos transformados y sus pesos asociados ($W(\lambda, P_k)$) se calcula el valor esperado del estado en el siguiente instante de tiempo ($\hat{X}_{k+1}$) (Ecuación 2.33) y el valor esperado de la matriz de covarianza de los estados predichos para el siguiente instante de tiempo ($\hat{P}_{k+1}$) (Ecuación 2.34).

$$\hat{X}_{k+1} = \sum_{i=0}^{2L} W_i^{(m)} \cdot f(\chi_i) \tag{2.33}$$

$$\hat{P}_{k+1} = \sum_{i=0}^{2L} W_i^{(c)} \cdot \left[f(\chi_i) - \hat{X}_{k+1}\right] \cdot \left[f(\chi_i) - \hat{X}_{k+1}\right]^T + Q \tag{2.34}$$

Para el último paso, la 'actualización', se puede empezar generando 2N + 1 nuevos puntos sigma (χ') a partir del estado $\hat{X}_{k+1}$ y la $\hat{P}_{k+1}$ (Ecuación 2.31) y propagarlos a través de la función no lineal que describe las salidas medidas del sistema o proceso ($h(\chi')$). Alternativamente, se pueden propagar los puntos sigma transformados en lugar de volver a calcularlos ($h(f(\chi))$ y, junto a los pesos asociados ($W(\lambda, P_k)$), obtener el valor esperado de

la salida del sistema para el siguiente instante de tiempo ($\hat{Y}_{k+1}$) (Ecuación 2.35) y la matriz de covarianza de las salidas predichas (S_{k+1}) (Ecuación 2.36). Para este trabajo se seguirá esta segunda forma descrita.

$$\hat{Y}_{k+1} = \sum_{i=0}^{2L} W_i^{(m)} \cdot h(f(\chi_i)) \tag{2.35}$$

$$S_{k+1} = \sum_{i=0}^{2L} W_i^{(c)} \cdot \Big[h(f(\chi_i)) - \hat{Y}_{k+1}\Big] \cdot \Big[h(f(\chi_i)) - \hat{Y}_{k+1}\Big]^T + R \tag{2.36}$$

Tras esto, se computa la matriz de covarianza cruzada entre el estado del sistema y la medida en el siguiente instante de tiempo (C_{k+1}) (Ecuación 2.37) y se obtiene la constante (o ganancia) de Kalman para el siguiente instante de tiempo (K_{k+1}) (Ecuación 2.38).

$$C_{k+1} = \sum_{i=0}^{2L} W_i^{(c)} \cdot \Big[f(\chi_i) - \hat{X}_{k+1}\Big] \cdot \Big[h(f(\chi_i)) - \hat{Y}_{k+1}\Big]^T \tag{2.37}$$

$$K_{k+1} = C_{k+1} \cdot S_k^{-1} \tag{2.38}$$

Finalmente, se actualiza (corrige) $\hat{X}_{k+1}$ añadiendo un término proporcional al error en la salida predicha ($\hat{Y}_{k+1}$) frente a la real (Y_{k+1}) por la constante K_{k+1} (Ecuación 2.39) y se actualiza (corrige) $\hat{P}_{k+1}$ añadiendo un término proporcional a S_{k+1} por el cuadrado de la constante K_{k+1} (Ecuación 2.40).

$$X_{k+1} = \hat{X}_{k+1} + K_{k+1} \cdot (Y_{k+1} - \hat{Y}_{k+1}) \tag{2.39}$$

$$P_{k+1} = \hat{P}_{k+1} - K_{k+1} \cdot S_{k+1} \cdot K_{k+1}^T \tag{2.40}$$

De este modo, el algoritmo volvería a comenzar desde este punto calculando los puntos sigma, predicciones y actualizaciones para el siguiente instante de tiempo $k+2$ a partir del nuevo instante de tiempo actual $k+1$.

Complementando lo mencionado en esta sección, a continuación se describirá el proceso seguido para el cálculo de los parámetros de inicialización del filtro empezando por el vector del estado inicial X_0 y la matriz de correlación cruzada inicial P_0.

2.6.3 Inicialización del vector de estado

A menudo, ante la falta de información de los estados anteriores del sistema (como al inicializar el filtro) se toma un vector de N ceros, donde N es el número de variables de estado. Además de la simplicidad del resultado, este método permitió inicializar el vector de estado en un punto estable numéricamente y sin sesgo alguno, que fue actualizándose con las iteraciones del filtro, por lo que los valores del vector de estados sólo dependieron de los datos de entrada.

$$X_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.41}$$

Sin embargo, en este trabajo, las variables de estado consideradas son la velocidad angular del porcentaje de ciclo de marcha (Θ), el coseno de Θ y el seno de Θ. Inicializar el vector de estado de esta forma puede ser problemático al no representar un estado plausible del sistema. Por un lado, si $\cos\Theta = \sin\Theta = 0$, entonces el Θ inicial no está definido pues $\Theta = \pi/2$ o $\Theta = 3\pi/2$. Por otro lado, la idea de iniciar el ciclo de marcha con velocidad angular nula no se sostiene frente a la dinámica del movimiento. Por ende, se implementó una opción considerablemente más viable inicializando el vector de estados como:

$$X_0 = \begin{bmatrix} \cos\Theta_0 \\ \sin\Theta_0 \\ \omega_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2\pi\dfrac{C}{60} \end{bmatrix} \tag{2.42}$$

Donde se asumió que el ciclo empieza en $\Theta_0 = 0$,como cabría esperar, con una cierta $\omega_0 > 0$. Además, la ω se asumió aproximadamente constante, por lo que $\omega_0 = \omega = 2\pi \cdot C/60$ donde C es la cadencia de la marcha en pasos/minuto, equivalente a la frecuencia de la señal.

2.6.4 Inicialización de la matriz de covarianza cruzada

Al igual que en el caso anterior, es una práctica común que, ante la falta de información de los estados anteriores del sistema (como al inicializar el filtro), se tome la matriz de covarianza cruzada inicial P_0 como la $I_{(NxN)}$ donde N es el número de variables de estado. Los motivos, muy similares al anterior, son que esta inicialización es simple, numéricamente estable y representa una aproximación fiable para la incertidumbre del estado inicial.

$$P_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.43}$$

Alternativamente se puede inicializar la matriz de covarianza cruzada de los estados con los mismos valores de la matriz de covarianza del ruido de los estados (Q), descrita a continuación, dado que la incertidumbre del estado inicial depende únicamente del ruido del sistema. No obstante, para este trabajo, se empleó la inicialización con la matriz identidad para P dados todos los motivos previamente mentados. De esta forma, se tomó como cierta la asunción de que la incertidumbre inicial era la misma para todas las variables de estado (i.e. $\sigma^2_{cosTheta_0} = \sigma^2_{sinTheta_0} = \sigma^2_{omega_0}$ = 1). Además, al ser los elementos no diagonales cero se asumió que no existe correlación inicial entre las variables de estado.

2.6.5 Matriz de covarianza del ruido de las medidas

La matriz del ruido de las medidas (R) es un elemento clave en todo UKF que nos permite modelar la incertidumbre propia de las medidas con sensores. De algún modo, indica cuánto debemos desconfiar de los valores medidos en la actualización de los estados. Así, valores elevados en la matriz R indica que, para la estimación del siguiente estado, se pondrá más peso en el valor del estado anterior y la dinámica interna del sistema, y menos peso en el error entre la medida predicha y la medida real. La estrategia principal para su estimación pasa por el análisis de los residuos, aunque estos pueden obtenerse de varias fuentes.

Una de las formas consiste en estimar directamente a partir del sensor el ruido asociado a la medida. Para ello, se toman medidas con el sensor en estático o en condiciones completamente conocidas, se calculan los residuos y se estima R a partir de la covarianza de estos. De forma similar, también se puede hallar R de la covarianza los residuos entre la medida predicha por el modelo del proceso a estimar y la medida real del sensor. En concreto, en este trabajo se optó por esta segunda opción.

Para la obtención de la matriz R de la señal se definió la función '*matriz_covarianza_ruido_salidas*', mediante la cual se computó las series de Fourier para aproximar las señales promedio de los ángulos de cadera ($\overline{\psi}_c$) y de rodilla ($\overline{\psi}_r$) de cada sujeto con 15 componentes de Fourier, con lo cual obtuvimos la función de medidas ($H(x)$) para cada sujeto. La aproximación por series de Fourier es comúnmente empleada en las señales de naturaleza periódica como los ciclos de la marcha y nos proporcionó una función (no lineal) con la cual se pudo implementar un UKF siguiendo los pasos anteriormente descritos.

Calculando el error entre las curvas promedio y las series de Fourier, se obtuvieron los residuos para cadera y rodilla, de cuya covarianza se acabó obteniendo R. Además, para cumplir verdaderamente con los objetivos de este trabajo se decidió comparar el rendimiento del UKF según se emplease una matriz R global (obtenida a partir de los residuos de todos los sujetos) o la matriz R propia de cada sujeto. La idea de comparar los dos procedimientos es poder determinar cuál de los dos es más óptimo para el UKF.

Promediado de la señal

Para la aproximación por series de Fourier de $\overline{\psi}_c$ y $\overline{\psi}_r$, el primer paso fue obtener estos valores. Para poder manejar el conjunto de pasos de longitud no uniforme, lo primero que se hizo fue interpolar las señales de los pasos de todas las pruebas para cada sujeto con la función 'interp1d' de la librería NumPy para que todos tuvieran una longitud de N muestras. Seguidamente, se promediaron las curvas para cada sujeto, obteniendo las curvas promedio del ángulo de cadera ($\overline{\psi}_c$) (Figura 2.30) y del ángulo de rodilla ($\overline{\psi}_r$) (Figura 2.31). De este modo, se logró estandarizar los datos y asegurar una mayor eficiencia en el cómputo posterior de las series de Fourier.

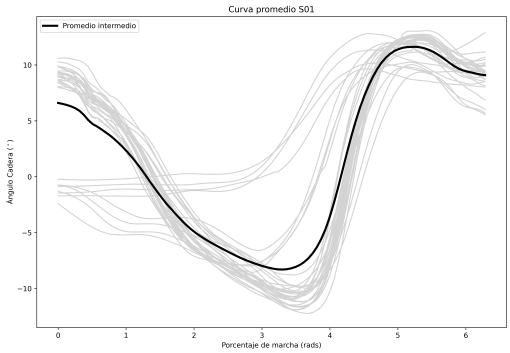


Figura 2.30: Promediado de ψ_c para los pasos iniciales, intermedios y finales para el sujeto S01.

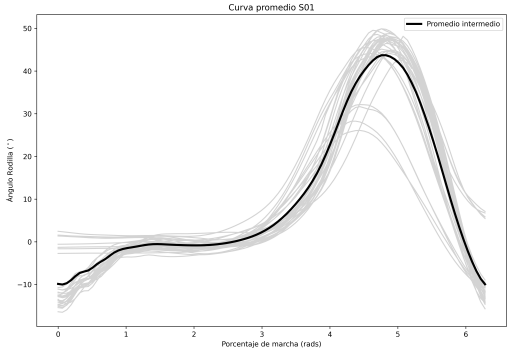


Figura 2.31: Promediado de ψ_r para los pasos iniciales, intermedios y finales para el sujeto S01.

Aproximación por Fourier

El siguiente paso, fue el paso clave, donde se aproximaron $\overline{\psi}_c$ y $\overline{\psi}_r$ por las series de Fourier Ψ_c y Ψ_r. Como se mencionó previamente, en el contexto del análisis de la marcha, esta aproximación consigue plasmar a la perfección la naturaleza periódica de los ciclos de la marcha a la par que filtra los componentes de ruido de las frecuencias más altas según el número de componentes y garantiza la periodicidad de la señal aproximada, lo cual nos beneficiará de cara a implementar el algoritmo recursivo de UKF.

El cálculo de los coeficientes de la serie se realizó resolviendo la Ecuación 2.44 por medio de la función 'lstsq' del submódulo 'linalg' para álgebra lineal de la librería NumPy que permite resolver ecuaciones matriciales lineales como esta por el método de mínimos cuadrados.

$$A \cdot x = b \tag{2.44}$$

Donde x es la matriz de coeficientes buscada, b son los valores de $\overline{\psi}_c$ o $\overline{\psi}_r$ y A es la matriz que contiene las funciones trigonométricas que definen los M componentes frecuenciales o armónicos de la señal (Matriz 2.45). El tamaño de la matriz A es de $2MxN$, por lo que el número de coeficientes de Fourier calculados (i.e., tamaño de la matriz x).

$$A = \begin{bmatrix} \cos(0 \cdot \Theta_N) & 0 \\ \cos(1 \cdot \Theta_N) & \sin(1 \cdot \Theta_N) \\ \cos(2 \cdot \Theta_N) & \sin(2 \cdot \Theta_N) \\ \vdots & \vdots \\ \cos((M-1) \cdot \Theta_N) & \sin((M-1) \cdot \Theta_N) \end{bmatrix} \quad x = \begin{bmatrix} a_0 & a_1 & a_2 \cdots a_M \\ b_0 & b_1 & b_2 \cdots b_m \end{bmatrix} \tag{2.45}$$

Donde Θ_N es el porcentaje de marcha Θ interpolado a N muestras.

Los valores Ψ_c y Ψ_r resultantes de la aproximación a una señal de N muestras para M coeficientes de Fourier se obtuvieron con la Ecuación 2.46.

$$\Psi(\Theta) = A \cdot x \tag{2.46}$$

Que equivale a la obtención de los valores de una serie de Fourier como la Ecuación 2.47 con los coeficientes de las Ecuaciones 2.48 y 2.49. Como se mencionó anteriormente, todo este proceso, se llevó a cabo en paralelo para los pasos iniciales, intermedios y finales de cada marcha independientemente. La Figura 2.32 muestra la reconstrucción de Ψ_c y Ψ_r frente a $\overline{\psi}_c$ y $\overline{\psi}_r$.

$$f(\Theta) = \sum_{i=0}^{M} [a_i \cos(i \cdot \Theta)] + \sum_{i=1}^{M} [b_i \sin(i \cdot \Theta)] \tag{2.47}$$

$$a_i = \frac{1}{\pi} \cdot \int_0^{2\pi} f(\Theta) \cdot \cos(i \cdot \Theta)\, d\Theta \tag{2.48}$$

$$b_i = \frac{1}{\pi} \cdot \int_0^{2\pi} f(\Theta) \cdot \sin(i \cdot \Theta)\, d\Theta \tag{2.49}$$

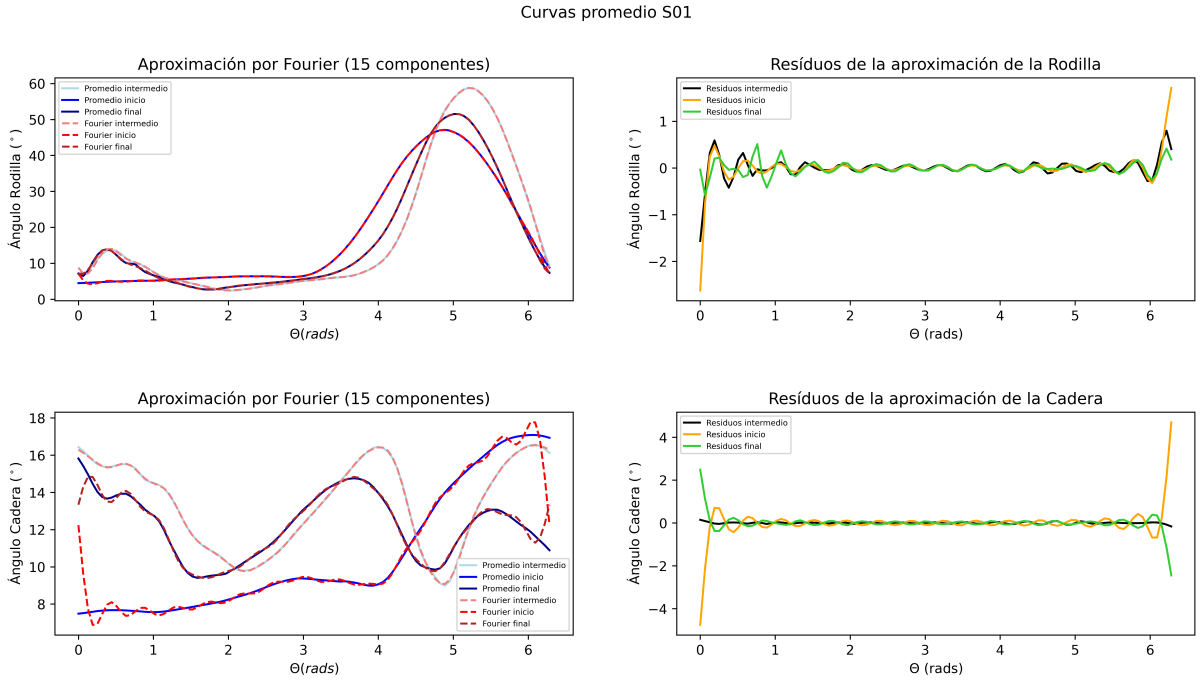


Figura 2.32: Aproximación por series de Fourier de 15 componentes frente a la señal promedio original interpolada para 100 muestras para el sujeto S01. A la derecha, el error de las aproximaciones en cada caso.

Cálculo de la matriz R

A partir de los valores originales de los periodos de marcha y los valores de la serie aproximada, mediante la función *matriz_2_covarianza_ruido_salidas* (Anexo 8.1) se calcularon los residuos de ambas variables (Ecuación 2.50) para los 8 sujetos. Calculando la matriz de covarianza de los distintos residuos se obtuvieron las matrices R_i para cada uno de los i sujetos (Ecuación 2.51).

$$\varepsilon_i = \begin{bmatrix} \overline{\psi}_{ci} - \Psi_{ci} \\ \overline{\psi}_{ri} - \Psi_{ri} \end{bmatrix} \tag{2.50}$$

$$R_i = Cov(\varepsilon_i) \tag{2.51}$$

Seguidamente, se puso en práctica el segundo método mencionado al inicio de esta Sección, concatenando los residuos (Ecuación 2.52) y computando la matriz de covarianza de los residuos concatenados, obteniendo efectivamente la matriz de covarianza del ruido de la medida del sistema (Ecuación 2.53) para todos los sujetos.

$$\epsilon = [\epsilon \quad \varepsilon_i] \quad i = 1, \ldots, 8 \tag{2.52}$$

$$R = Cov(\epsilon) \tag{2.53}$$

Finalmente, los resultado para las matrices particulares de cada sujeto fueron las siguientes matrices:

$$R_1 = \begin{bmatrix} 0.0031 & -0.0003 \\ -0.0003 & 0.0408 \end{bmatrix} \quad \mathsf{R}_2 = \begin{bmatrix} 0.0026 & 0.0057 \\ 0.0057 & 0.0537 \end{bmatrix} \quad \mathsf{R}_3 = \begin{bmatrix} 0.0080 & 0.0008 \\ 0.0008 & 0.0616 \end{bmatrix} \quad \mathsf{R}_4 = \begin{bmatrix} 0.0080 & -0.0166 \\ -0.0166 & 0.0537 \end{bmatrix}$$
$$\mathsf{R}_5 = \begin{bmatrix} 0.0425 & 0.0470 \\ 0.04702 & 0.0638 \end{bmatrix} \quad \mathsf{R}_6 = \begin{bmatrix} 0.0111 & 0.0186 \\ 0.0186 & 0.0462 \end{bmatrix} \quad \mathsf{R}_7 = \begin{bmatrix} 0.0053 & -0.0116 \\ -0.0116 & 0.0412 \end{bmatrix} \quad \mathsf{R}_8 = \begin{bmatrix} 0.0559 & 0.1173 \\ 0.1173 & 0.2638 \end{bmatrix} \tag{2.54}$$

Por otro lado, la matriz R global para todos los sujetos, tomó los valores siguientes:

$$R = \begin{bmatrix} 0.0169 & 0.0199 \\ 0.0199 & 0.0774 \end{bmatrix} \tag{2.55}$$

2.6.6 Matriz de covarianza del ruido de los estados

La matriz de la covarianza del ruido de los estados (Q) es uno de los parámetros claves si se desea lograr un UKF robusto y preciso ya que en ella se almacena la información relativa a la incertidumbre del sistema dinámico que no tiene en cuenta el modelo determinista del sistema. Gracias a esta, la evolución del vector de estados es capaz de adaptarse a las perturbaciones inherentes a los sistemas reales. Así, valores más altos en Q indican menor confianza en el modelo del sistema para la predicción del siguiente estado. Para su estimación se pueden distinguir dos principales métodos: simulación por Monte Carlo o deducción analítica.

El primer escenario es especialmente oportuno en casos donde la dinámica del sistema es compleja o no se conoce lo suficiente para poder estimar Q de forma analítica. El método de Monte Carlo consiste en generar múltiples muestras aleatorias del ruido del proceso y propagarlas a través de la función del sistema para obtener las predicciones de los estados. Calculando la covarianza de las predicciones de todos los estados, se obtiene una aproximación muy fiable de la matriz Q, si bien es computacionalmente muy costosa.

Por contra, el segundo método emplea significativamente menos recursos computacionales, pero necesita de un conocimiento profundo de las relaciones físicas y matemáticas de los estados. Si se dan estas condiciones, la incertidumbre del ruido de los estados se puede extraer directamente del análisis de las expresiones que definen el sistema. En concreto, en este trabajo se optó por probar ambos métodos para cumplir con los objetivos propuestos y determinar cuál de los dos procedimientos genera la matriz Q que más optimice el rendimiento del UKF.

Tras ver ambas matrices R y Q es sencillo percatarse que la ratio entre R y Q es un parámetro clave que ajusta el balance entre las predicciones del modelo y las actualizaciones de este. Un R/Q demasiado bajo, llevará al modelo a sobre-confiar en la dinámica interna del sistema, evitando la correcta actualización de los parámetros internos y dando lugar a grandes errores en la estimación. Análogamente, una ratio demasiado alta llevará al UKF a una actualización constante de los parámetros internos priorizando las medidas sobre la dinámica interna del

sistema, de nuevo resultando en predicciones poco precisas. Este es el motivo por el cual se hizo especial hincapié en probar varios procedimientos para estimar las matrices R y Q. Esto se vuelve clave en el caso de la última dado que su cálculo requiere de cierto grado de intuición y conocimiento de las ecuaciones del sistema.

Método de Monte Carlo

El método de Monte Carlo (MC) es una técnica numérica conocida que permite resolver problemas matemáticos por medio del muestreo (pseudo) aleatorio aprovechando la capacidad de los ordenadores de simular variables y/o procesos aleatorios. Si bien su estructura computacional es simple, la habilidad del MC de manejar sistemas complejos, afectados por factores aleatorios y/o multidimensionales es muy alta. Esto se debe a que, en este método, primero se extraen un gran número de simulaciones aleatorias independientes del sistema modelado, para luego promediarlas para extraer una estimación del valor esperado real del sistema (**sobol2018**).

En particular, el MC es muy útil en tareas que requieran una precisión moderada (e.g., 5-10 %), sistemas no lineales y/o sistemas complejos donde las soluciones analíticas tradicionales son desconocidas o inaccesibles. Sin embargo, la precisión de este algoritmo tiene un contra severo debido a la relación proporcionalmente directa entre la precisión de la estimación y el número de muestras N. Así, debido a la Ley de los grandes números, el error de estimación inversamente proporcional a $\sqrt{N}$

Debido a la alta no linealidad del proceso de la marcha humana, la deducción analítica precisa de la matriz Q puede complicarse, por lo que se consideró implementar el MC para obtener dicha matriz generando un gran número de simulaciones basadas en el ruido del proceso asumido *a priori* y calculando la covarianza de los residuos. Su puesta en práctica vino acompañada de un conjunto de suposiciones entre las que destacan un número de muestras lo suficientemente alto y una distribución del ruido Gaussiana.

En el caso de este trabajo, el vector de estado se define por el coseno y seno del porcentaje de marcha y la velocidad angular del ciclo. Aprovechando la relación entre estos, se pudo extraer empíricamente una matriz Q que capturase la variabilidad inherente al proceso de la marcha humana en base a los estados definidos del sistema. Este algoritmo se recoge en la función *matriz_1_covarianza_estados* (Anexo 8.1) donde se siguieron los siguientes pasos para determinar la matriz Q.

En primera instancia, considerando una cadencia de marcha típica entre 80 y 120 pasos por minuto, el error máximo esperado será de la mitad del rango, que expresado en velocidad angular del porcentaje de ciclo sería:

$$\varepsilon_\omega = 2\pi \cdot \frac{20}{60} = 2.094 rad/s \tag{2.56}$$

En segundo lugar, se calculó la desviación típica del error escalando este por ± 5 desviaciones típicas (i.e., el 99.7 % de los valores de error):

$$\varepsilon_\omega^{'} = \frac{\varepsilon_\omega}{5} = 0.418 rad/s \tag{2.57}$$

Hecho esto, se generaron N muestras aleatorias del error en la velocidad angular utilizando de una distribución Gaussiana con desviación típica $\varepsilon_\omega^{'}$. Específicamente, $N = 1e4$ muestras aleatorias se emplearon para este paso.

$$\{x\}_{i=1}^{N} \sim N(0, \varepsilon_\omega^{'}) \tag{2.58}$$

A partir de las muestras $\{x\}_i$ del error de la pulsación ω se computaron también los errores del coseno y del seno, dada la relación inherente entre estos. De esta forma, se obtuvieron los residuos:

$$\epsilon = \begin{bmatrix} \cos\{x\}_i \\ \sin\{x\}_i \\ \{x\}_i \end{bmatrix} \tag{2.59}$$

Finalmente, operando sobre los residuos concatenados calculados con la función 'cov' de la librería Numpy de Python, se obtuvo la estimación de la matriz Q a partir de los errores muestreados. El resultado de este procedimiento fue la Matriz 2.60.

$$\begin{bmatrix} 0.0162 & -0.0137 & -0.0162 \\ -0.0137 & 0.0162 & 0.0162 \\ -0.0162 & 0.0162 & 0.0176 \end{bmatrix} \tag{2.60}$$

Deducción analítica de la matriz Q

En el contexto de este trabajo, donde la velocidad angular del periodo de marcha ω es una variable Gaussiana aleatoria positiva con media μ y varianza σ_ω^2, se calculó la matriz de covarianza Q considerando las relaciones entre las variables de estado con ω:

$$Q = \begin{bmatrix} Var(\omega) & Cov(\omega, \cos\omega t) & Cov(\omega, \sin\omega t) \\ Cov(\omega, \cos\omega t) & Var(\cos\omega t) & Cov(\cos\omega t, \sin\omega t) \\ Cov(\omega, \sin\omega t) & Cov(\cos\omega t, \sin\omega t) & Var(\sin\omega t) \end{bmatrix} \tag{2.61}$$

- Cálculo de varianza de ω

La varianza de ω la calculamos considerando que la cadencia de pasos de las personas se encuentra normalmente entre 80 y 120 pasos por minuto (ppm). Esto nos dio un error máximo respecto a la media de $\pm 20ppm$, que expresado en velocidad angular fue:

$$\varepsilon_\omega = 2\pi \cdot \frac{20}{60} = 2.094 rad/s \tag{2.62}$$

Teniendo que en ± 3 desviaciones típicas de la media de toda Gaussiana se encuentran aproximadamente el 99.7 % de los datos, se tomaron ± 5 desviaciones típicas para escalar el error, considerando que este se encuentra en dicho rango.

$$\varepsilon_\omega^{'} = \frac{\varepsilon_\omega}{5} = 0.418 rad/s \tag{2.63}$$

Por lo que consideramos la varianza de ω como el cuadrado de $\varepsilon_\omega^{'}$.

$$Var(\omega) = (\varepsilon_\omega^{'})^2 = 0.175 (rad/s)^2 \tag{2.64}$$

- Cálculo de la covarianza entre ω y $\cos\omega t$

Por lo general, para dos variables aleatorias como ω y $\cos\omega t$ se cumple que el valor esperado del producto de ambas es igual al producto de sus valores esperados, siempre que no haya correlación entre estas. Aplicando la fórmula de la covarianza, obtuvimos:

$$Cov(\omega, \cos\omega t) = E[\omega \cdot \cos\omega t] - E[\omega] \cdot E[\cos\omega t] \tag{2.65}$$

Dado que ω se pude considerar positivo con ωt uniformemente distribuido entre $[0, 2\pi]$ y $\cos\omega t$ es una función periódica, a largo plazo asumimos que el valor esperado del coseno se aproxima a cero.

$$E[\cos\omega t] \approx 0 \tag{2.66}$$

Además, si ω es positivo y no hay correlación directa entre ω y $\cos\omega t$, asumimos:

$$E[\omega \cdot \cos\omega t] \approx 0 \tag{2.67}$$

Por lo tanto, de sustituir las Ecuaciones 2.66 y 2.67 en la Ecuación 2.65, se dedujo:

$$Cov(\omega, \cos\omega t) \approx 0 \tag{2.68}$$

- Cálculo de la covarianza entre ω y $\sin\omega t$

Análogo al caso anterior, la fórmula general para la covarianza entre ω y $\sin\omega t$ fue definida como:

$$Cov(\omega, \sin\omega t) = E[\omega \cdot \sin\omega t] - E[\omega] \cdot E[\sin\omega t] \tag{2.69}$$

Donde, bajo asunciones idénticas al caso anterior, se dedujo que:

$$Cov(\omega, \cos\omega t) \approx 0 \tag{2.70}$$

- Cálculo de la varianza de $\cos \omega t$

A continuación, para el cálculo de la varianza del coseno, nos basamos en:

$$Var(\cos \omega t) = E[\cos^2 \omega t] - (E[\cos \omega t])^2 \tag{2.71}$$

Donde a partir de la Ecuación 2.66 simplificamos a:

$$Var(\cos \omega t) = E[\cos^2 \omega t] \tag{2.72}$$

Cuyo resultado es $\frac{1}{2}$ dado que este es el valor de la media de $\cos^2$ y $\sin^2$ en un ciclo completo:

$$Var(\cos \omega t) \approx 0.5 \tag{2.73}$$

- Cálculo de la varianza de $\sin \omega t$

Análogamente al caso anterior y por los motivos mencionados, la varianza del seno se obtuvo como:

$$Var(\sin \omega t) = E[\sin^2 \omega t] - (E[\sin \omega t])^2 \approx 0.5 \tag{2.74}$$

- Cálculo de la covarianza entre $\cos \omega t$ y $\sin \omega t$

El último elemento restante por calcular fue la covarianza entre el coseno y el seno, la cual, aplicando la fórmula estándar de la covarianza quedó como:

$$Cov(\cos \omega t, \sin \omega t) = E[\cos \omega t \cdot \sin \omega t] - E[\cos \omega t] \cdot E[\sin \omega t] \tag{2.75}$$

Que simplificamos con la Ecuación 2.66 y su análoga para el seno, junto a la aplicación de la identidad trigonométrica del producto a suma (Ecuación 2.76:

$$E[\cos \omega t \cdot \sin \omega t] = \frac{1}{2} \cdot E[\sin 2\omega t] \approx 0 \tag{2.76}$$

De donde se dedujo:

$$Cov(\cos \omega t, \sin \omega t) \approx 0 \tag{2.77}$$

- Cálculo de la matriz Q

Finalmente, sustituyendo los valores calculados en la Matriz 2.61 de referencia obtuvimos:

$$Q = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.175 \end{bmatrix} \tag{2.78}$$

Esta matriz refleja la varianza del ruido inherente a las variables de estado obtenida de forma analítica bajo las suposiciones de independencia y periodicidad mencionadas. Con esta ya pudimos implementar el UKF declarando una clase '*UnscentedKalmanFilter*' dentro de la cual definimos los métodos '*reset_state*' para cambiar el estado interno al asignado, '*predict*' para predecir el siguiente estado y '*update*' para corregir el estado en base al error entre la salida predicha y la salida real y la constante K_k. De este modo, se implementaron uno a uno los pasos de la Sección 2.6.2 y se obtuvieron los resultados a continuación.

2.6.7 Función de Transición de los Estados

La función de transición de los estados f debe describir la dinámica de los estados del sistema a lo largo del tiempo. Dado el vector de estado empleado para el UKF de este trabajo (Ecuacuón 2.79) y una frecuencia de muestreo f_s conocida (en este caso 100 Hz), podemos definir la transición del estado en el instante de tiempo t_k al t_{k+1} según la Ecuación 2.80. Para determinarla, se consideró la fórmula del ángulo suma y una velocidad angular constante del porcentaje de ciclo.

$$X_k = \begin{bmatrix} \cos \omega t_k \\ \sin \omega t_k \\ \omega \end{bmatrix} \tag{2.79}$$

$$\begin{aligned} X_{k+1} = \begin{bmatrix} \cos \omega t_{k+1} \\ \sin \omega t_{k+1} \\ \omega \end{bmatrix} = \begin{bmatrix} \cos \omega (t_k + \frac{1}{fs}) \\ \sin \omega (t_k + \frac{1}{fs}) \\ \omega \end{bmatrix} = \begin{bmatrix} \cos \omega t_k + \omega \frac{1}{fs} \\ \sin \omega t_k + \omega \frac{1}{fs} \\ \omega \end{bmatrix} \\ = \begin{bmatrix} \cos \omega t_k \cdot \cos \omega \frac{1}{fs} - \sin \omega t_k \cdot \sin \omega \frac{1}{fs} \\ \sin \omega t_k \cdot \cos \omega \frac{1}{fs} + \cos \omega t_k \cdot \sin \omega \frac{1}{fs} \\ \omega \end{bmatrix} = \begin{bmatrix} \cos \omega t_k \cdot \cos \omega \frac{1}{fs} - \sin \omega t_k \cdot \sin \omega \frac{1}{fs} \\ \sin \omega t_k \cdot \cos \omega \frac{1}{fs} + \cos \omega t_k \cdot \sin \omega \frac{1}{fs} \\ \omega \end{bmatrix} \end{aligned} \tag{2.80}$$

2.6.8 Función de las Salidas Medidas

La función de las salidas medidas (h) traduce el vector de estados a cierto valor en el espacio de medidas. En el UKF diseñado en este trabajo, el vector de salidas Y se compone de las series de Fourier de los ángulos de flexo-extensión de la cadera y de la rodilla, los cuales son función del porcentaje del ciclo de marcha (Ecuación 2.47). Por otro lado, el porcentaje de ciclo se puede extraer a partir de las variables de estado empleando la función 'arctan2' de la librería Numpy, según la Ecuación 2.81, lo cual nos permite indexar directamente el valor de la señal de salida a partir de las variables de estado ($Y(\Theta)$).

$$\Theta_k = arctan2 \left(\frac{\sin \omega t_k}{\cos \omega t_k} \right) \tag{2.81}$$

Por último, para este trabajo, se empleó como función de las salidas medidas del UKF una única serie Fourier promedio de todos los sujetos para los ángulos de flexo-extensión de la cadera y de la rodilla (Figura 2.33).

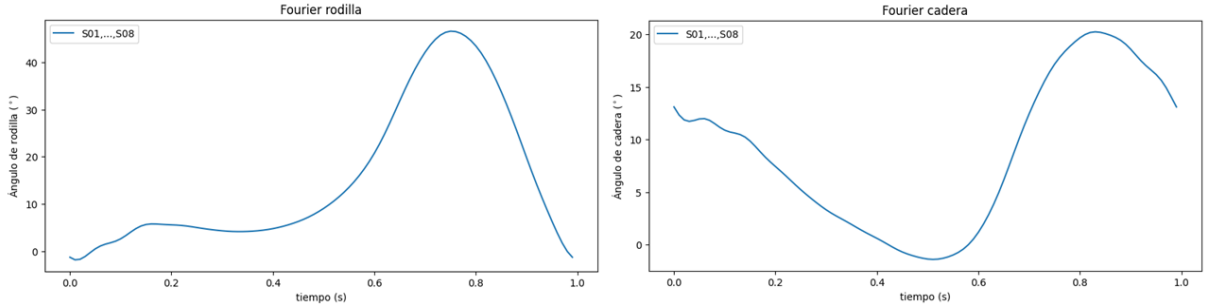


Figura 2.33: Series promedio de Fourier de los ángulos de flexo-extensión de la cadera y de la rodilla para todos los sujetos.

Capítulo 3

RESULTADOS

En el capítulo de Resultados, se presentarán primeramente los resultados de la detección (del inicio y fin) de los periodos de marcha empleando la umbralización adaptativa. Como se explicó en el capítulo de Metodología, tras segmentar manualmente las normas de los ángulos de cadera y rodilla para todos los pasos de cada ciclo, sus valores se promediaron para cada sujeto y se obtuvo una aproximación por serie de Fourier de ambas. Esta aproximación facilitó los cálculos de los valores de alguno de los parámetros fundamentales del *Unscented Kalman Filter* (UKF) del cual aquí se muestran los resultados de la predicción, junto a métricas de error y varianza que permitieron comprender mejor la viabilidad de los resultados.

Para representar el rendimiento del UKF de la forma adecuada, se planteó un escenario ideal donde se trató de predecir el porcentaje de ciclo de marcha estimado ($\hat{\Theta}$) para el paso promedio de cada sujeto y los ángulos promedios $\hat{\psi}_c$ y $\hat{\psi}_r$ asociados por la función de salidas. Tras esto, en un escenario más realista, se trató de predecir $\hat{\Theta}$ durante los periodos de marcha y los ángulos ψ_c y ψ_r asociados por la función de salidas.

3.1 DETECCIÓN DE LOS PERIODOS DE MARCHA

Los resultados parciales de la detección de los periodos de marcha fueron presentados para el caso del sujeto S01 en el capítulo de Metodología al explicar la umbralización adaptativa realizada. En esta primera parte de los resultados se mostrarán el resto de detecciones de marcha llevadas a cabo en todos los sujetos. En las Figuras, se muestran los ángulos de flexo-extensión de rodilla (arriba) y cadera (abajo), según la prueba: a la derecha los ángulos de la primera prueba en la que se empezó a caminar con la pierna izquierda) y a la izquierda, los ángulos de la segunda prueba en la que se empezó a caminar con la pierna derecha. En rojo, el resultado de la detección que indica 1 para la detección de los acontecimientos de marcha y 0 para el estado estático y los giros. Comparando las imágenes, se aprecia claramente que el método de umbralización adaptativa funcionó correctamente para todos los sujetos aun cuando las curvas no fueran las más estables o poseyeran algo de sesgo o deriva en el tiempo.

3.2 EVALUACIÓN Y SELECCIÓN DE LOS PARÁMETROS

Hacia el final del Capítulo de Metodología, se introdujeron varias alternativas para las matrices de covarianza del ruido Q y R con la idea de determinar la combinación de parámetros que optimizase el rendimiento del UKF. A continuación, se presentan como métricas el error cuadrático medio del porcentaje de fase de marcha, del ángulo de rodilla y del ángulo de cadera. Para simplificar los resultados y ahorrar espacio, en las Tablas 3.1 y3.2 únicamente se muestran las métricas resultantes de la predicción con el UKF implementado con los datos del ciclo promedio de cada sujeto (datos ideales, no realistas). No obstante, esto no invalida las conclusiones que se puedan extraer de estos valores ya que se puede comprobar que las métricas de la predicción con las medidas de los periodos de marcha reales, mantienen la misma tendencia (Anexo 8.2).

Figura 3.1: Detección del inicio y fin de los periodos de marcha de todos los sujetos en función de los ángulos de rodilla o cadera y del lado con el que se empezase a caminar.

De las Tablas 3.1 y3.2 a continuación, podemos ver claramente que los menores RMSE se obtienen empleando la matriz Q por el método MC, si bien en combinación con la R_i individual, la Q obtenida analíticamente puede rivalizar con la primera en el porcentaje de fase de marcha y en el ángulo de cadera. En un primer vistazo, llama la atención que la combinación de la Q por el método MC con la R global, sea la clara opción a escoger en todos los casos, salvo para el porcentaje, donde la media del RMSE promedio indica una diferencia de 7 centenas con la combinación de la Q por MC con la R individual. No obstante, si nos fijamos en los valores del RMSE promedio por sujeto, se observa que el sujeto 7 constituye un valor atípico que aumenta la media. Por contra, si tomamos las medianas para estos casos, obtenemos un valor de 0.0985 para la combinación de Q por el método MC con la R global y 0.0995 para la opción de la Q por MC con la R individual. Esto refleja el motivo por el cual se acabó

eligiendo la combinación de Q por el método MC con la R global, que serán las matrices que se emplearás para las siguientes secciones de los Resultados.

Tabla 3.1: Promedio del error cuadrático medio (RMSE) para la predicción del porcentaje de fase de marcha, ángulo de rodilla y ángulo de cadera mediante el UKF implementado en el caso ideal con las posibles combinaciones de las matrices R y Q planteadas en el Capítulo de Metodología. Notar que para simplificar esta implementación inicial, las medidas utilizadas para actualizar los estados fueron las curvas promedio de los sujetos.

	RMSE promedio (Porcentaje)			
	R global		R individual	
Sujeto	Q analítica	Q monte carlo	Q analítica	Q monte carlo
1	0.63	0.06	0.63	0.06
2	0.51	0.10	0.14	0.10
3	0.65	0.10	0.11	0.10
4	0.66	0.15	0.15	0.14
5	0.64	0.10	0.10	0.10
6	0.99	0.12	0.13	0.13
7	0.14	0.66	0.14	0.13
8	0.08	0.08	0.09	0.08
Media	**0.54**	**0.17**	**0.19**	**0.10**
	RMSE promedio (Rodilla)			
	R global		R individual	
Sujeto	Q analítica	Q monte carlo	Q analítica	Q monte carlo
1	0.46	0.30	0.45	0.30
2	2.09	0.81	1.45	0.83
3	0.93	0.47	0.60	0.47
4	0.44	0.39	0.57	0.39
5	0.61	0.18	0.26	0.18
6	1.60	0.55	0.58	0.55
7	0.82	0.54	0.69	0.54
8	0.10	0.16	0.25	0.18
Media	**0.88**	**0.43**	**0.61**	**0.43**
	RMSE promedio (Cadera)			
	R global		R individual	
Sujeto	Q analítica	Q monte carlo	Q analítica	Q monte carlo
1	0.14	0.18	0.19	0.18
2	0.41	0.24	0.50	0.25
3	0.13	0.18	0.11	0.18
4	0.55	0.52	0.65	0.52
5	0.17	0.21	0.16	0.15
6	0.70	0.17	0.21	0.17
7	0.26	0.19	0.13	0.19
8	0.27	0.49	0.50	0.60
Media	**0.33**	**0.27**	**0.31**	**0.28**

3.3 PREDICCIÓN DE LOS PORCENTAJES DE FASE Y LOS ÁNGULOS ARTICULARES ASOCIADOS

Una vez esclarecidas definitivamente las matrices R y Q a emplear, en esta tercera Sección de los Resultados se muestran los resultados de la implementación de un *Unscented Kalman Filter* (UKF) para la predicción del porcentaje de fase y de los ángulos articulares asociados. Para analizar el comportamiento del UKF en condiciones varias, se plantearon dos escenarios: un escenario ideal donde se emplearon los ángulos promedio de rodilla y cadera y otro más realista donde periodos de marcha individuales fueron empleados.

Tabla 3.2: Coeficientes de correlación (CC) para el coseno y el seno del porcentaje de fase de marcha, para el ángulo de rodilla y para el ángulo de cadera mediante el UKF implementado en el caso ideal con las posibles combinaciones de las matrices R y Q planteadas en el Capítulo de Metodología. Nuevamente, notar que para simplificar esta implementación inicial, las medidas utilizadas para actualizar los estados fueron las curvas promedio de los sujetos.

	Coeficiente Correlación (Coseno)			
	R global		R individual	
Sujeto	Q analítica	Q monte carlo	Q analítica	Q monte carlo
1	0.93	0.99	0.95	1.00
2	0.92	0.99	0.91	1.00
3	0.92	0.99	0.94	0.99
4	0.91	0.99	0.94	0.99
5	0.93	1.00	0.92	1.00
6	0.92	0.98	0.91	0.99
7	0.94	0.99	0.93	0.98
8	0.92	0.99	0.94	1.00
Media	**0.92**	**0.99**	**0.93**	**0.99**
	Coeficiente Correlación (Seno)			
	R global		R individual	
Sujeto	Q analítica	Q monte carlo	Q analítica	Q monte carlo
1	0.98	1.00	0.99	1.00
2	0.97	0.99	0.94	0.99
3	0.98	1.00	0.98	1.00
4	0.96	0.99	0.97	0.99
5	0.98	0.99	0.96	0.99
6	0.95	0.99	0.96	0.99
7	0.98	0.99	0.96	0.99
8	0.98	0.99	0.98	1.00
Media	**0.97**	**0.99**	**0.97**	**0.99**
	Coeficiente Correlación (Rodilla)			
	R global		R individual	
Sujeto	Q analítica	Q monte carlo	Q analítica	Q monte carlo
1	1.00	1.00	1.00	1.00
2	0.99	1.00	1.00	1.00
3	1.00	1.00	1.00	1.00
4	1.00	1.00	1.00	1.00
5	1.00	1.00	1.00	1.00
6	1.00	1.00	1.00	1.00
7	1.00	1.00	1.00	1.00
8	1.00	1.00	1.00	1.00
Media	**1.00**	**1.00**	**1.00**	**1.00**
	Coeficiente Correlación (Cadera)			
	R global		R individual	
Sujeto	Q analítica	Q monte carlo	Q analítica	Q monte carlo
1	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	1.00
3	1.00	1.00	1.00	1.00
4	1.00	1.00	1.00	1.00
5	1.00	1.00	1.00	1.00
6	1.00	1.00	1.00	1.00
7	1.00	1.00	1.00	1.00
8	1.00	1.00	1.00	1.00
Media	**1.00**	**1.00**	**1.00**	**1.00**

3.3.1 Escenario Ideal para el UKF

En el caso ideal, el UKF se utilizó para predecir el porcentaje de ciclo de marcha Θ en cada instante de tiempo para cada sujeto utilizando como medidas de referencia $\hat{\psi}_c$ y $\hat{\psi}_r$.

Resultados de la predicción en el caso ideal

Los resultados mostrados a continuación, muestran el excelente rendimiento tanto en la comparación directa con los datos reales (Figuras 3.2, 3.3 y 3.4) como en todas las métricas medidas.

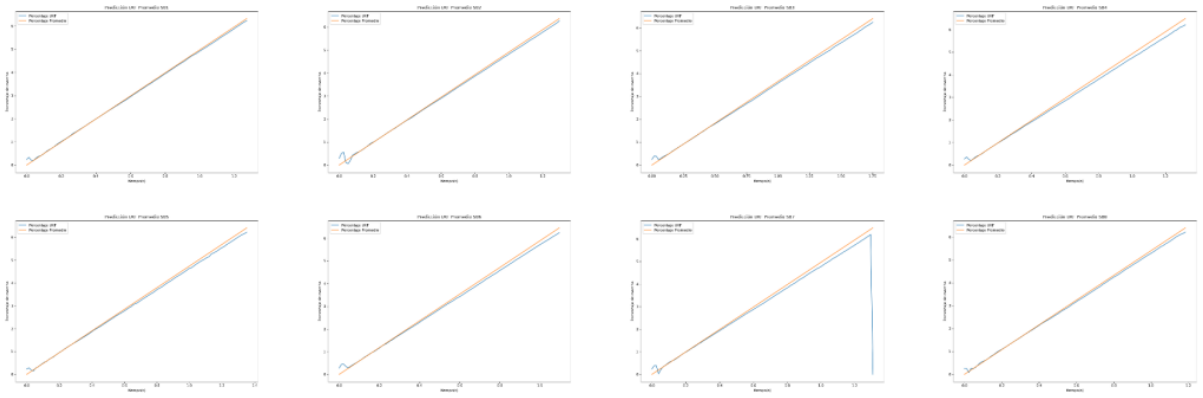


Figura 3.2: Resultados de la predicción del porcentaje de fase (linea azul) durante el paso promedio (linea naranja) de cada uno de los sujetos.

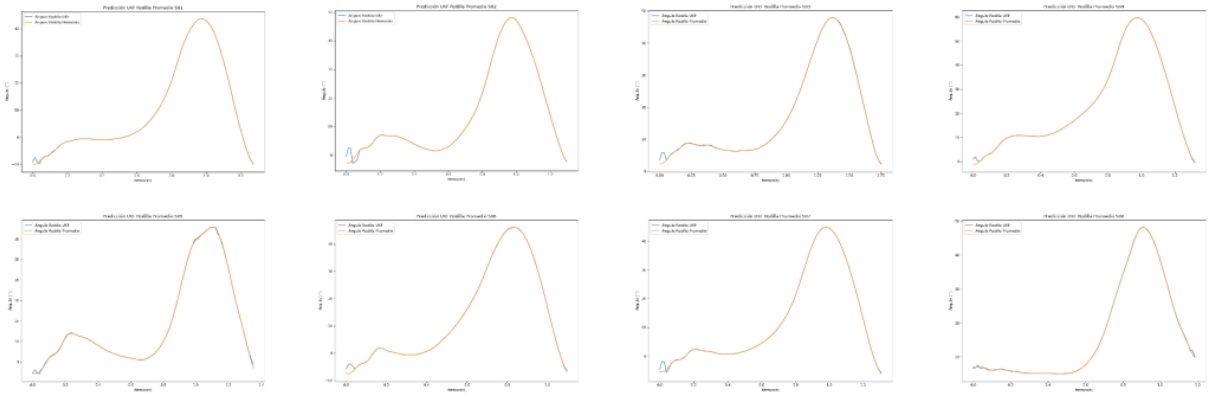


Figura 3.3: Resultados de la predicción del ángulo de flexo-extensión de rodilla (linea azul) durante el paso promedio (linea naranja) de cada uno de los sujetos.

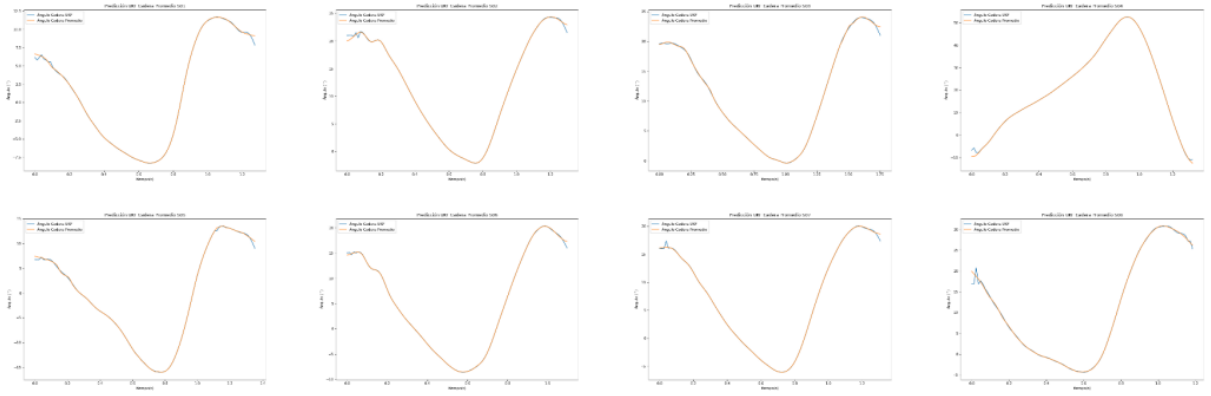


Figura 3.4: Resultados de la predicción del ángulo de flexo-extensión de cadera (linea azul) durante el paso promedio (linea naranja) de cada uno de los sujetos.

Observando las anteriores figuras la conclusión inevitable es que el rendimiento del UKF en el caso ideal es prácticamente perfecto salvo por el instante inicial y el instante final. En el ambos, el hecho de haber aproximado

la función de las medidas del sistema por series de Fourier hizo que el UKF tendiera a predecir curvas completamente periódicas, si bien, en realidad, las medidas de las curvas de la marcha (e incluso el promedio de estas) no resultaron ser periódicas exactamente, bien por el error del sensor, bien por aspectos inherentes del modelo de la marcha. Sumado a esto, en el instante inicial los parámetros predefinidos del filtro aún se hallan convergiendo a los verdaderos parámetros de la curva a estimar, lo que refleja el motivo por el cual el UKF es propenso a errores al inicio de la predicción.

Residuos de los estados predichos en el caso ideal

Los residuos en este caso ideal (Figura 3.5) oscilan entorno al valor medio 0, acercándose significativamente durante la porción intermedia del paso, lo que parece indicar que se alcanza una alta precisión una vez el UKF logra estabilizarse. Sin embargo, al inicio y al final del ciclo, el error alcanza los 1-1.5°, lo cual muestra nuevamente las dificultades del UKF implementado por predecir Θ en las zonas de transición. En estas zonas, el UKF espera una curva periódica debido a su función de salidas basada en series de Fourier, pero en su lugar recibe datos (ideales) reales, no necesariamente periódicos.

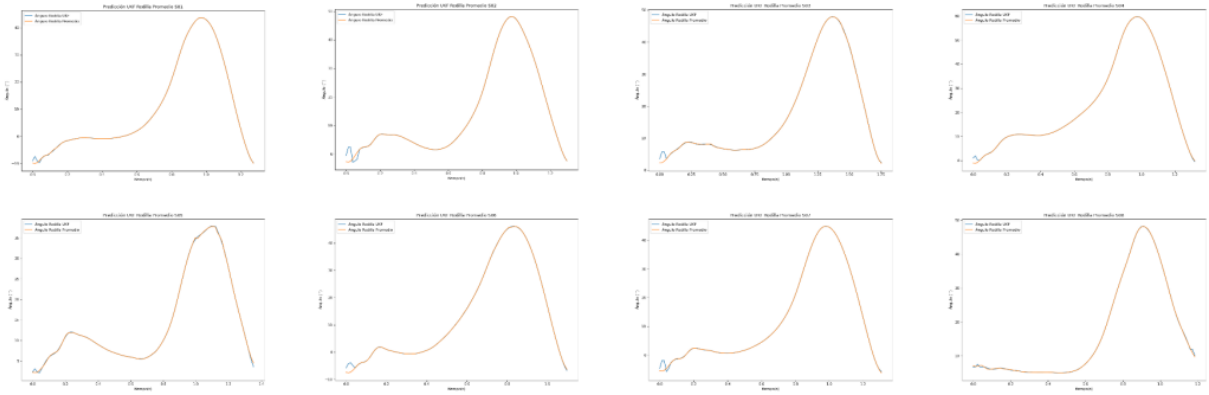


Figura 3.5: Residuos de la predicción del ángulo de rodilla (linea azul) y del ángulo de cadera (linea naranja) durante el paso promedio de cada uno de los sujetos.

Error cuadrático medio en el caso ideal

Asimismo, el error cuadrático medio (RMSE) de las variables de estado del UKF en este caso ideal mostró un claro comportamiento exponencial decreciente, estabilizándose cerca de 0.1 (Figura 3.6). Esto sugiere que, con el tiempo, el UKF efectivamente reduce la incertidumbre, mejorando las predicciones de los estados estimados.

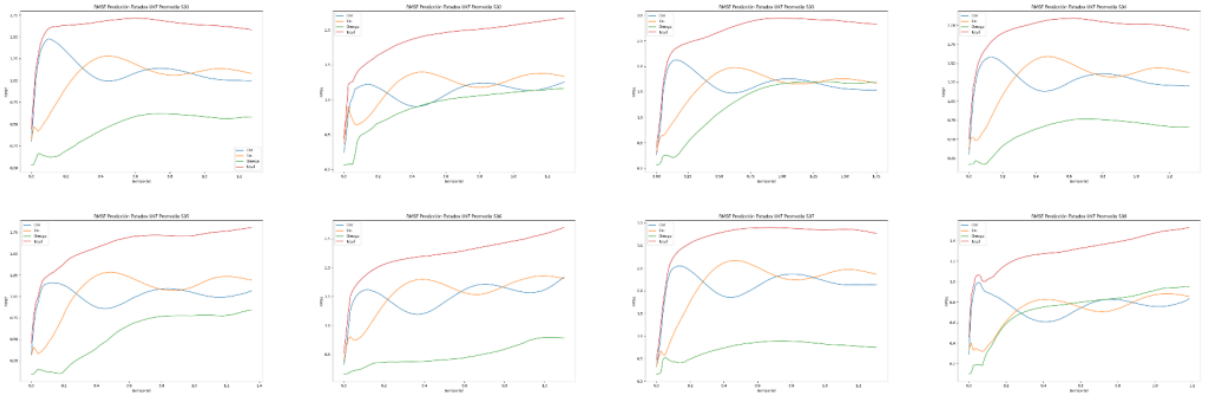


Figura 3.6: Error cuadrático medio (RMSE) de la predicción del coseno del porcentaje de fase (linea azul), del seno del porcentaje de fase (linea naranja) y de la velocidad angular del ciclo (linea verde) durante el paso promedio de cada uno de los sujetos.

Otras métricas de rendimiento en el caso ideal

La traza de la matriz de covarianza cruzada P expresa la varianza total de los estados predichos. Se puede comprobar que la tendencia de la traza durante la ejecución del UKF en el caso ideal es a descender exponencialmente hasta un valor de 0.1 aproximadamente (Anexo 8.2). Esta dinámica de la varianza total indica que, en el caso ideal, el UKF reduce la incertidumbre de la predicción con el paso del tiempo.

Por otro lado, también se comprueba que el mínimo de los valores singulares de la matriz P converge rápidamente a un valor de cercano a 0.05 (Anexo 8.2). El hecho de que el mínimo valor singular sea mayor que 0, indica que dicha matriz permanece definida positiva durante todo el proceso, garantizando la estabilidad del UKF como algoritmo.

Por último, se evaluó la precisión de las estimaciones de las medidas mediante la verosimilitud estadística de las salidas predichas (Anexo 8.2). La tendencia en este caso, fue a converger rápidamente a un estado estacionario entorno a 0. Esto indica que la verosimilitud de las medidas predichas por el UKF mejora rápidamente con el tiempo por medio de la actualización correcta del vector de estados.

3.3.2 Escenario Realista para el UKF

En el segundo escenario descrito, se empleó el UKF para predecir el porcentaje de fase de marcha durante los periodos de marcha segmentados individualmente. Si bien el primer escenario es técnicamente válido, para cumplir con los objetivos del trabajo el método debe poder acabarse implementando en un exoesqueleto donde no tendrá otra cosa sino datos reales que no podrán promediarse si se desea una predicción en tiempo real. El objetivo de este segundo escenario es, pues, probar el UKF diseñado en condiciones más variables y realistas a partir de los valores calculados en crudo (Figuras 3.7, 3.8, 3.9).

Resultados de la predicción en el caso realista

A continuación se muestran los resultados para las pruebas realizadas empezando a caminar con la pierna izquierda (que es la que llevaba las IMUs). Las figuras de la prueba empezando con la pierna derecha y sus métricas se encuentran en el Anexo 8.2. Como se puede observar, los resultados de este escenario fueron ligeramente inferiores al caso ideal dado el incremento de variabilidad, pero demostraron la robustez y adaptabilidad del modelo a cambios severos de la morfología de las curvas de los ángulos de cadera y rodilla durante la marcha.

Residuos de los estados predichos en el caso realista

En esta instancia, los residuos calculados mantuvieron la tendencia a decrecer hacia los pasos intermedios y crecer hacia los pasos inicial y final en las transiciones entre el estado estático y dinámico. Sin embargo, en este caso, la oscilación del valor entorno a 0 (varianza) es varias veces mayor al caso ideal. Tal era la diferencia entre las varianzas del caso ideal con este, que se decidió aplicar un factor de escalado a la matriz R para ajustarse a esta mayor variabilidad de los datos reales. Este factor se determinó por el método heurístico de prueba y error. En concreto se redefinió como $R' = R \cdot 10^{3.5}$. Tras esto, la varianza de los residuos siguió siendo mayor que en el caso ideal, pero ya se estabilizaba más entorno a 0 en las regiones intermedias de las curvas (Figura 3.10).

Error cuadrático medio en el caso realista

El gráfico del RMSE para este complejo escenario muestra una dinámica bastante común donde, en los primeros 2 segundos, el RMSE del conjunto de todas las variables de estado alcanza los 5°, tras lo cual comienza a descender exponencialmente hasta los 1-2° donde pasa a ser asintótico (Figura 3.11). Además, si nos fijamos en las curvas, el RMSE de la variable de estado de la velocidad angular ω es la componente principal del RMSE total. Esto sugiere que el UKF diseñado cuenta con dificultades para la predicción de cambios bruscos en la velocidad angular.

Complementando al gráfico, dado que el escenario realista es crucial para determinar la viabilidad de esta aplicación del UKF, se obtuvo la Tabla 3.3 donde se muestran los valores del RMSE promedio para cada uno de los periodos de marcha, para cada uno de los sujetos y para cada una de las pruebas, además del valor medio del RMSE según la prueba. Así, se pueden comprobar las conclusiones extraídas del gráfico con valores numéricos donde vemos que el RMSE para la ω es el componente que más contribuye al RMSE total. De la misma forma, se observa que el RMSE de las pruebas que empezaron con la pierna izquierda es entorno a 1° menor al de aquellas que empezaron con la derecha. La explicación más plausible de este hecho es que, al estar los sensores en la pierna izquierda y no en la derecha, en aquellas pruebas donde se empezó a caminar con la pierna derecha, el primer paso registrado por las IMUs no fue sino el segundo paso de la prueba. Teniendo esto en cuenta, si consideramos

que los parámetros iniciales del UKF asumen la partida desde el reposo, es natural que las predicciones de una señal que no parte del reposo presenten mayor RMSE que las que sí lo hacen.

Tabla 3.3: RMSE de los estados predichos frente a los reales para la implementación del UKF en el caso realista.

Sujeto	Periodo de marcha	Derecha				Izquierda			
		RMSE (cos)	RMSE (sin)	RMSE (omega)	RMSE (total)	RMSE (cos)	RMSE (sin)	RMSE (omega)	RMSE (total)
1	1	1.45	0.80	9.88	10.05	2.05	1.72	6.95	7.67
	2	1.62	1.20	7.58	7.90	1.52	1.23	6.39	6.75
	3	1.62	1.10	8.45	8.72	1.51	1.24	6.34	6.71
	4	1.61	1.18	8.09	8.39	1.69	1.36	6.04	6.49
2	1	2.09	1.29	5.68	6.21	1.89	0.92	5.58	5.99
	2	1.66	1.09	6.82	7.13	1.77	0.96	5.49	5.87
	3	1.51	1.03	7.92	8.17	1.66	0.88	5.28	5.62
	4	1.74	1.20	6.78	7.14	1.74	0.91	5.59	5.94
3	1	1.81	1.30	6.24	6.68	1.80	1.35	5.66	6.18
	2	1.51	0.85	8.09	8.30	1.62	1.08	5.54	5.92
	3	1.75	1.25	7.85	8.20	1.70	0.86	4.87	5.25
	4	1.64	1.25	6.82	7.17	1.64	0.88	4.67	5.05
4	1	1.44	1.20	4.94	5.31	1.80	1.41	5.04	5.55
	2	1.73	1.23	5.39	5.82	1.82	1.41	5.05	5.58
	3	1.50	1.07	6.37	6.67	1.77	1.17	5.39	5.81
	4	1.53	1.08	6.47	6.77	2.42	2.20	4.74	5.83
5	1	2.22	1.72	5.66	6.41	1.98	1.20	5.10	5.64
	2	1.75	1.02	8.01	8.30	2.05	0.95	5.24	5.73
	3	2.02	1.11	8.07	8.41	2.01	1.19	4.72	5.30
	4	1.21	0.57	11.00	11.08	2.12	1.13	5.20	5.77
6	1	1.73	1.24	6.70	7.07	1.34	1.12	5.45	5.74
	2	1.63	1.15	6.80	7.11	1.39	1.18	5.25	5.58
	3	1.66	1.25	7.31	7.65	1.36	1.00	5.42	5.69
	4	1.88	1.49	7.36	7.83	1.22	0.98	5.11	5.35
7	1	1.74	1.28	6.76	7.19	1.98	1.72	6.41	7.11
	2	1.72	1.24	7.76	8.11	1.50	1.14	7.08	7.38
	3	1.64	1.21	8.33	8.62	1.58	1.27	6.53	6.89
	4	1.53	1.02	8.29	8.52	1.61	1.24	6.76	7.11
8	1	1.83	1.16	6.28	6.67	1.91	1.04	5.46	5.89
	2	1.75	1.22	6.09	6.49	2.04	1.17	5.08	5.63
	3	2.05	1.50	6.71	7.24	1.92	1.02	5.72	6.13
	4	1.89	1.33	6.20	6.66	1.81	0.92	5.51	5.89
Promedio	-	**1.70**	**1.18**	**7.21**	**7.56**	**1.76**	**1.18**	**5.58**	**6.03**

Coeficiente de correlación en el caso realista

Al igual que en la métrica anterior, para evaluar la implementación del UKF en el escenario realista se computó también el coeficiente de correlación (CC) para cada uno de los periodos de marcha, para cada uno de los sujetos y para cada una de las pruebas, además del valor medio del CC según la prueba (Tabla 3.4). En este caso, se hallaron los CCs de los estados *cos* y *sin* y de las dos salidas. Al contrario que en el RMSE, el CC de la predicción del UKF con los valores reales para los estados del filtro y los datos de aquellas pruebas empezando a caminar con la pierna derecha es mayor. Esta aparente contradicción con el RMSE en realidad nos revela que en estas pruebas empezando con la derecha, la predicción sigue más fielmente la forma de la curva real de los datos o estados, pero cuenta con un desfase respecto a los valores reales que incrementa el RMSE. La conclusión para las pruebas empezando con la pierna izquierda es simplemente la inversa, aunque, en realidad, las diferencias son mínimas, por lo que bien se podrían haber obviado.

Otras métricas de rendimiento en el caso realista

Análogo al caso ideal, se computaron otras tres métricas. La traza de la matriz P para este caso no ideal, exhibe oscilaciones mucho mayores (mayor varianza) y, en algunos casos, no converge a un valor fijo (Anexo 8.2). Este comportamiento indica que la incertidumbre sobre la predicción de los estados, lejos de ser estática, fluctuó con el tiempo.

Por otro lado, el mínimo valor singular de la matriz P no descendió del 0.05, por lo que conocemos que P es definida positiva durante toda la prueba (Anexo 8.2), garantizando la estabilidad y viabilidad del UKF.

La verosimilitud de las salidas siguió una evolución donde trató de converger a 0, pero no alcanzó como en el primer caso, sino que únicamente logró aumentar cerca de 2° (Anexo 8.2). Esto indica que, aún cuando la predicción del estado mejora, existe un cierto grado de desacuerdo no resuelto entre las salidas predichas y la salidas reales en las fases dinámicas de la marcha.

Tabla 3.4: Coeficientes de correlación de los estados predichos con el estado real y de las salidas predicas con las salidas reales para la implementación del UKF en el caso realista.

Sujeto	Periodo de marcha	Derecha				Izquierda			
		CC (coseno)	CC (seno)	CC (rodilla)	CC (cadera)	CC (coseno)	CC (seno)	CC (rodilla)	CC (cadera)
1	1	0.79	0.76	0.99	0.87	0.76	0.70	0.98	0.93
	2	0.87	0.87	0.99	0.85	0.80	0.72	0.99	0.94
	3	0.84	0.83	0.99	0.83	0.73	0.82	0.98	0.93
	4	0.83	0.83	0.99	0.83	0.76	0.76	0.99	0.94
2	1	0.76	0.91	0.99	0.76	0.63	0.69	0.99	0.90
	2	0.83	0.90	0.99	0.75	0.76	0.79	0.98	0.91
	3	0.87	0.90	0.99	0.77	0.50	0.83	0.98	0.91
	4	0.84	0.90	0.99	0.72	0.60	0.68	0.98	0.92
3	1	0.69	0.87	0.99	0.90	0.70	0.85	0.98	0.91
	2	0.74	0.82	0.97	0.93	0.59	0.81	0.97	0.88
	3	0.57	0.79	0.97	0.89	0.57	0.72	0.97	0.87
	4	0.65	0.77	0.97	0.92	0.33	0.62	0.97	0.63
4	1	0.74	0.83	0.98	0.96	0.67	0.85	0.98	0.96
	2	0.67	0.83	0.98	0.97	0.68	0.84	0.98	0.96
	3	0.72	0.75	0.97	0.96	0.70	0.85	0.98	0.96
	4	0.79	0.83	0.98	0.97	0.62	0.79	0.97	0.93
5	1	0.63	0.80	0.94	0.86	0.53	0.77	0.92	0.75
	2	0.62	0.73	0.92	0.51	0.61	0.74	0.92	0.83
	3	0.50	0.71	0.93	0.83	0.57	0.25	0.91	0.71
	4	0.69	0.66	0.92	0.74	0.61	0.79	0.92	0.89
6	1	0.84	0.88	0.99	0.73	0.67	0.83	0.98	0.90
	2	0.86	0.90	0.99	0.79	0.65	0.84	0.98	0.88
	3	0.88	0.90	0.99	0.71	0.63	0.79	0.98	0.91
	4	0.75	0.87	0.99	0.93	0.18	0.63	0.97	0.65
7	1	0.58	0.76	0.98	0.90	0.72	0.80	0.98	0.94
	2	0.86	0.84	0.99	0.77	0.77	0.87	0.98	0.94
	3	0.87	0.87	0.99	0.72	0.70	0.83	0.99	0.93
	4	0.81	0.83	0.98	0.72	0.68	0.83	0.98	0.92
8	1	0.71	0.87	0.99	0.96	0.62	0.78	0.98	0.96
	2	0.63	0.80	0.98	0.96	0.48	0.74	0.98	0.94
	3	0.71	0.87	0.98	0.96	0.58	0.80	0.98	0.96
	4	0.68	0.85	0.98	0.96	0.47	0.74	0.96	0.94
Promedio	-	**0.74**	**0.83**	**0.98**	**0.84**	**0.62**	**0.76**	**0.97**	**0.89**

Por último, a modo de contar con más información sobre el rendimiento del modelo en el caso realista, se calculó una métrica más del error: el error medio absoluto (MAE), cuyas gráficas muestran curvas consistentes con las del RMSE (Anexo 8.2). Esto nos indica que no hay valores especialmente atípicos en los errores, por lo que podemos tomar los promedios con relativa confianza de que no van a poseer sesgo.

Figura 3.7: Resultados de la predicción del porcentaje de fase (linea azul) durante los periodos de marcha reales (linea naranja) de cada uno de los sujetos durante las pruebas comenzando a caminar con la pierna izquierda.

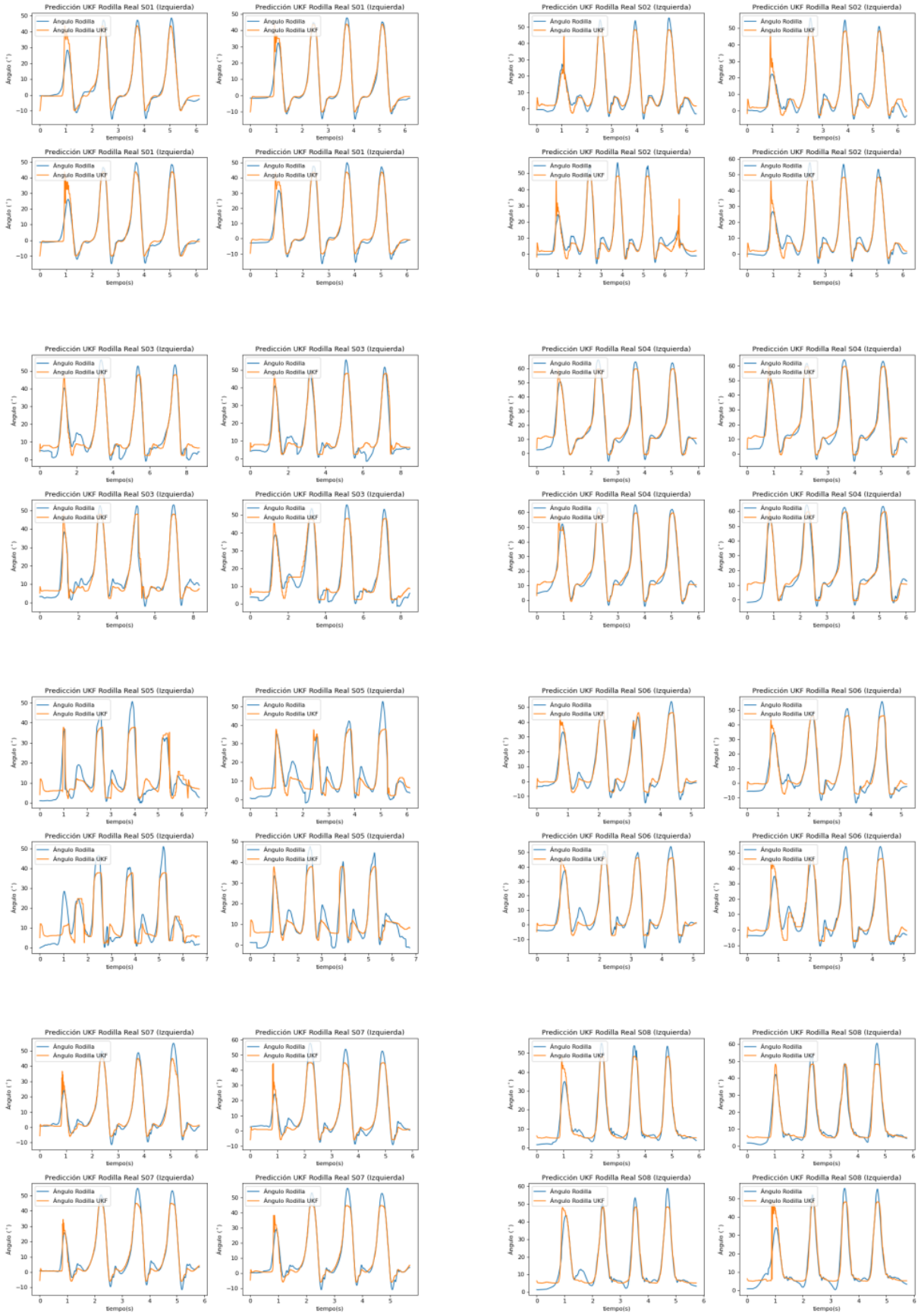


Figura 3.8: Resultados de los ángulos de flexo-extensión de la rodilla (linea azul) durante los periodos de marcha reales (linea naranja) de cada uno de los sujetos durante las pruebas comenzando a caminar con la pierna izquierda.

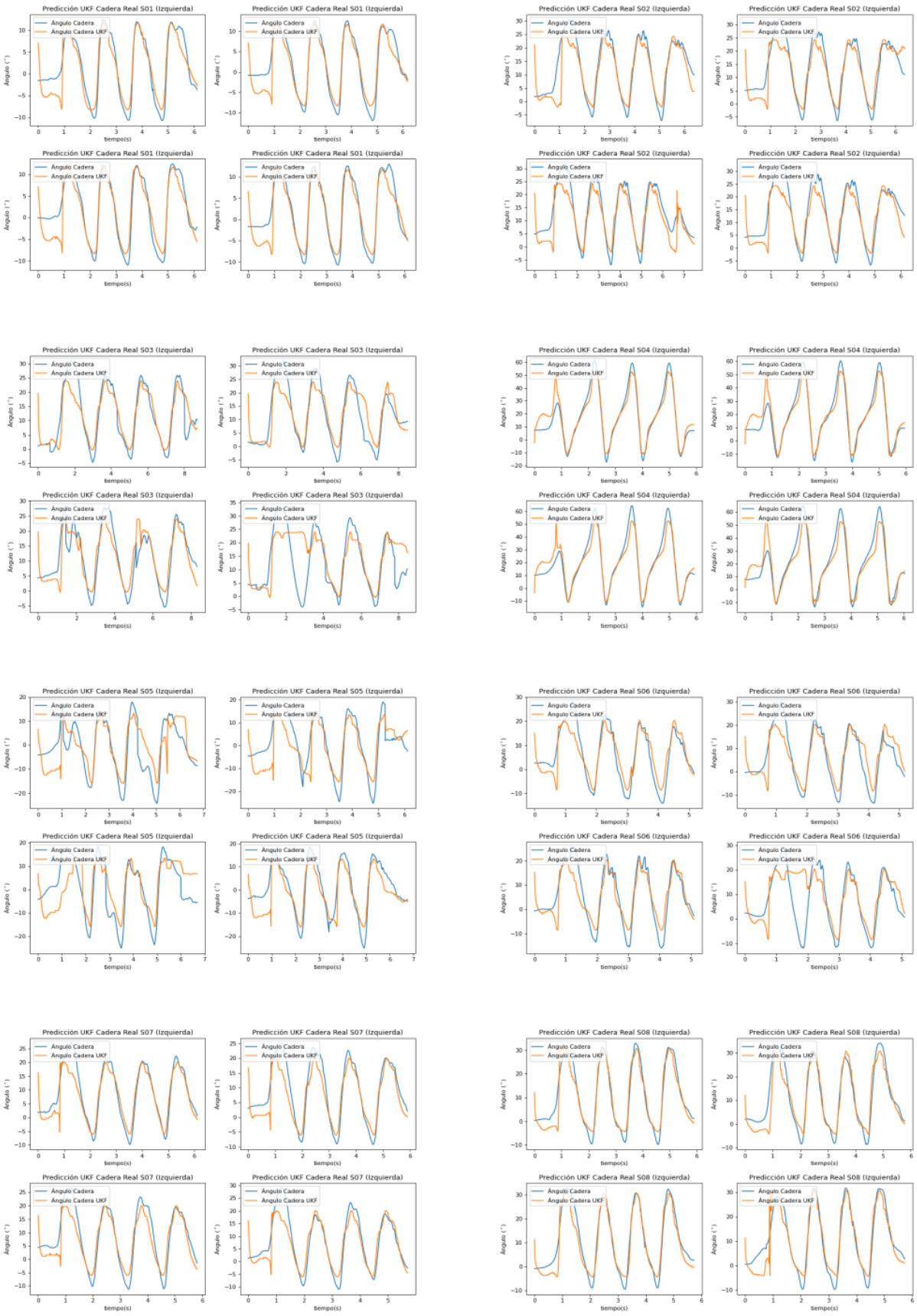


Figura 3.9: Resultados de los ángulos de flexo-extensión de la cadera (linea azul) durante los periodos de marcha reales (linea naranja) de cada uno de los sujetos durante las pruebas comenzando a caminar con la pierna izquierda.

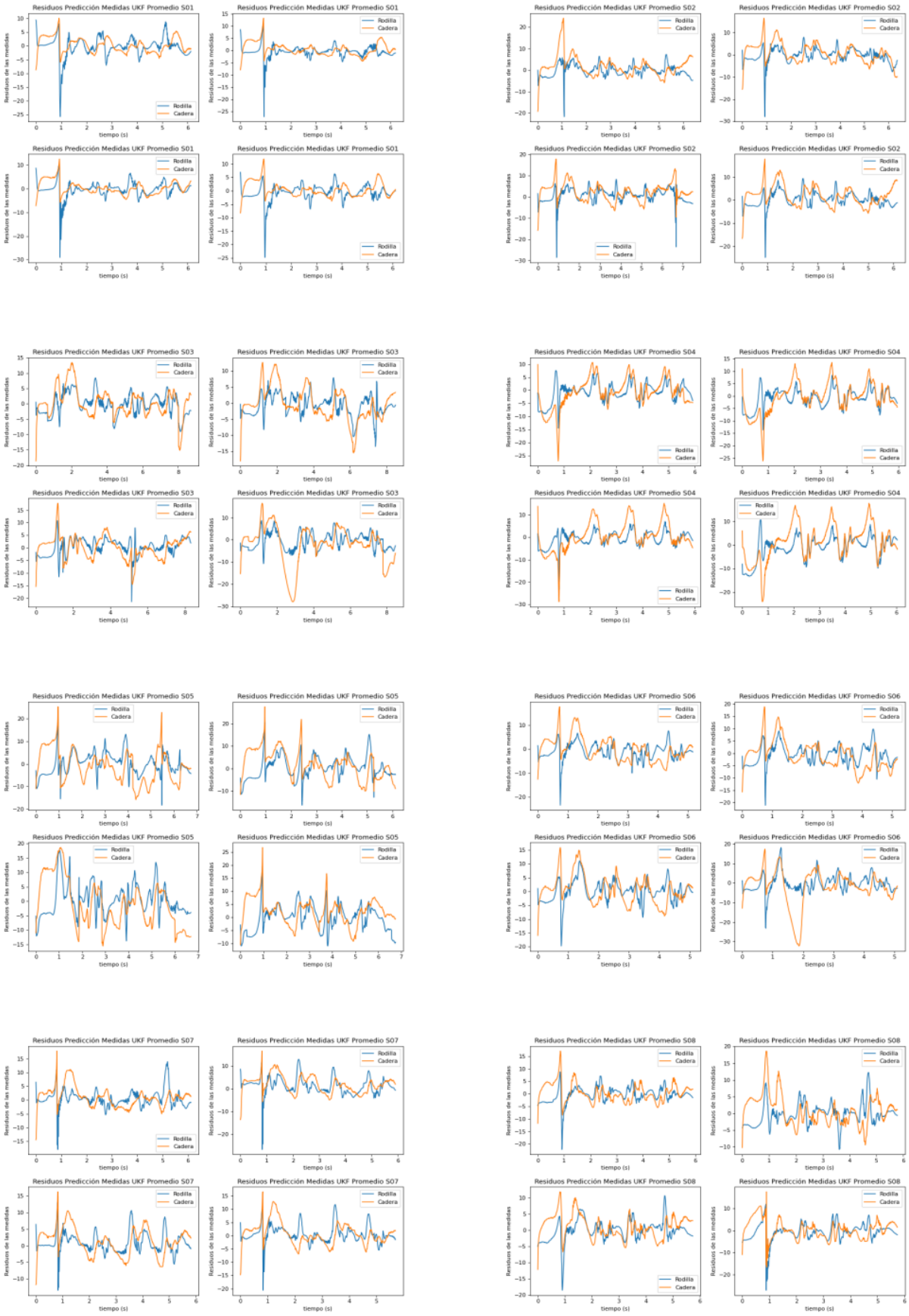


Figura 3.10: Residuos de la predicción del ángulo de rodilla (linea azul) y del ángulo de cadera (linea naranja) durante los periodos de marcha reales de cada uno de los sujetos durante las pruebas comenzando a caminar con la pierna izquierda.

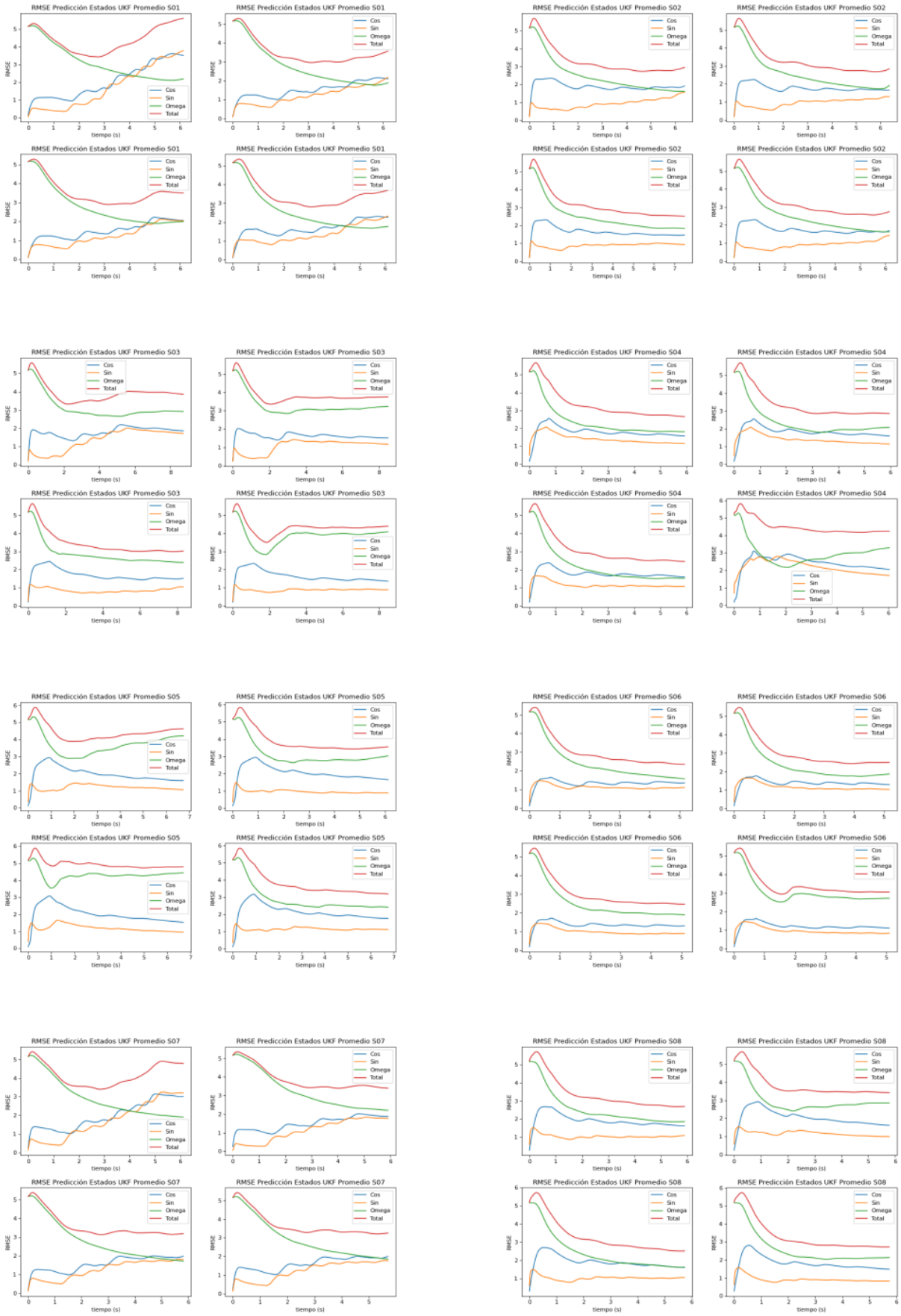


Figura 3.11: Error cuadrático medio (RMSE) de la predicción del coseno del porcentaje de fase (linea azul), del seno del porcentaje de fase (linea naranja) y de la velocidad angular del ciclo (linea verde) durante los periodos de marcha reales de cada uno de los sujetos durante las pruebas comenzando a caminar con la pierna izquierda.

Capítulo 4

DISCUSIÓN

Los resultados obtenidos al implementar el *Unscented Kalman Filter* (UKF) para la predicción del porcentaje de la fase de la marcha y los ángulos de flexo-extensión de cadera y rodilla demuestran tanto el potencial como las limitaciones del enfoque empleado en diferentes escenarios. En este Capítulo se analizan las implicaciones de los hallazgos, se compara el desempeño en los dos escenarios y se exploran los potenciales factores subyacentes que contribuyen a los resultados observados.

4.1 Análisis de la Viabilidad del Modelo

En primera instancia, en el escenario ideal, donde a la UKF se le asignó la tarea de predecir el porcentaje promedio del ciclo de la marcha utilizando los ángulos promedio de rodilla y cadera como mediciones, el desempeño fue cuanto menos ejemplar. Los residuos, RMSE, la traza de la matriz de covarianza, el mínimo valor propio y las métricas de verosimilitud indicaron que el UKF fue muy eficaz en este entorno controlado.

Los residuos casi nulos durante los pasos intermedios y la rápida estabilización del RMSE resaltan la capacidad del UKF para realizar un seguimiento preciso del ciclo de la marcha cuando se le proporcionan datos promediados y consistentes. Asimismo, la caída exponencial de la traza de la matriz de covarianza y la estabilización del mínimo valor propio subrayan aún más la eficacia del filtro para reducir la incertidumbre y mantener la estabilidad en sus predicciones.

El éxito en el escenario ideal se puede atribuir a la precisión de la función de medidas derivada de la aproximación de la serie de Fourier. Este modelado preciso permitió al UKF hacer predicciones muy precisas, minimizando el error incluso en las fases de transición más desafiantes (i.e., menos periódicas) al principio y al final del ciclo. Por un lado, emplear la serie de Fourier como aproximación para la señal conlleva residuos mayores en los extremos de los ciclos. No obstante, esto garantiza una salida periódica que permite al UKF obtener predicciones consistentes y robustas frente al ruido de la señal de la marcha ya que al aproximar a cualquier número de componentes de Fourier menor al 50 % de las muestras, estaremos efectivamente filtrando las altas frecuencias donde predomina el ruido.

4.2 Implicaciones para aplicaciones prácticas

Si bien el escenario ideal proporciona un punto de referencia para el rendimiento potencial del UKF, es importante tener en cuenta que tales condiciones rara vez se encuentran en aplicaciones del mundo real. Sin embargo, estos resultados sí validan el modelo subyacente y la capacidad del UKF cuando se aplican a datos consistentes y estables, proporcionando una base sólida para aplicaciones más complejas.

En el escenario realista, la UKF enfrentó el desafío de predecir el porcentaje de la fase de la marcha durante series de caminatas individuales, introduciendo variabilidad y cambios dinámicos que no estaban presentes en el escenario ideal. El desempeño, aunque ligeramente peor, siguió siendo satisfactorio, lo que demuestra la robustez del UKF implementado en condiciones más prácticas.

Una de las observaciones clave en este escenario fue el aumento de los residuos y errores durante los pasos iniciales y finales del ciclo de la marcha. Estas fases implican aceleración y desaceleración, lo que introduce dinámicas de estado no estacionario que son más difíciles de predecir con precisión para el UKF. Los artefactos observados en las predicciones durante estas fases resaltan las limitaciones de la aproximación de la serie de Fourier para capturar estos rápidos cambios.

Aún a pesar de los desafíos iniciales, el UKF mostró un mejor rendimiento durante los pasos intermedios del ciclo de la marcha, donde la marcha se vuelve más estable. En estos periodos de la señal, los residuos disminuyeron considerablemente y el RMSE alcanzó un valor asintótico estable, lo que indica que el UKF se adaptó efectivamente a las condiciones del estado estacionario incluso dentro de la variabilidad de las series de caminatas individuales.

Por otro lado, las oscilaciones y aumentos ocasionales de la traza de la matriz de covarianza cruza y el mínimo valor propio exhibieron la incertidumbre fluctuante en las estimaciones de estado. Esto sugiere que, si bien el UKF puede adaptarse a condiciones dinámicas, hay margen de mejora en el manejo de la variabilidad y la reducción de las oscilaciones de la incertidumbre.

En cuanto a la estimación de las mediciones, se conoce que los valores de verosimilitud (logarítmica) por debajo de cero y mantenidos en el tiempo indican un nivel persistente de error significativo entre las predicciones y las mediciones reales. Esto puede atribuirse a las limitaciones de la función de medición derivada de la aproximación de la serie de Fourier, que parece no capturar completamente la complejidad de los patrones de marcha individuales en los casos donde no se cumple estrictamente la condición de periodicidad. Por lo tanto, mejorar la función de medición mediante la incorporación de modelos más detallados y precisos podría mejorar el desempeño del UKF.

4.3 Reevaluación de las Matrices de Covarianza

Al utilizar el método de Monte Carlo para estimar la matriz de covarianza del ruido del proceso Q, se trató de capturar la variabilidad en el sistema de forma robusta y adaptable, aún a costa de introducir cierta aleatoriedad y variabilidad inherentes. Si bien no se consideró que el efecto de emplear esta Q pudiera ser significativo, debemos considerar todas las posibilidades incluyendo la de que Q contribuya a las oscilaciones observadas en el escenario realista.

Por el contrario, una derivación analítica de Q implicaría un enfoque más determinista que podría reducir parte de la variabilidad, pero requeriría un conocimiento preciso de la dinámica del sistema y las características del ruido. Las ventajas del enfoque de MC son su flexibilidad y adaptabilidad, particularmente en condiciones complejas y variables, mientras que para el enfoque analítico son su estabilidad y consistencia cuando se comprende bien la dinámica del sistema. En este trabajo se decidió emplear la opción de MC, dado el rendimiento que esta presentó, pero es posible que con las modificaciones adecuadas, la derivación analítica sobrepase a MC, por lo que sería interesante probarlo en futuros trabajos.

4.4 Implicaciones prácticas y trabajo futuro

Si bien el trabajo realizado ha sido relativamente exhaustivo, realizando una valoración completa hay, por lo menos, 4 puntos que se podrían incluir en futuras investigaciones y/o deberían ser mejorados tales como se listan a continuación:

1. El trabajo futuro debería centrarse en desarrollar modelos más detallados y precisos de la función de medición, incorporando potencialmente técnicas de aprendizaje automático para capturar la complejidad de los patrones de marcha individuales.
2. Dado el desempeño del UKF en el escenario realista, se podrían explorar técnicas de filtrado adaptativo o modelos híbridos que combinen UKF con otros métodos de filtrado para mejorar la solidez y la precisión ante las condiciones de estado no estacionario.
3. Si bien la validación del UKF en los escenarios presentados proporciona una base sólida para prever el rendimiento que tendría en su aplicación en el análisis de la marcha del mundo real, serían necesarias más pruebas y validaciones en condiciones diversas y prácticas para garantizar la fiabilidad y la generalización del modelo.
4. Optimizar la implementación del filtro para lograr eficiencia computacional, sin comprometer la precisión, es otra área importante para futuras investigaciones.

Capítulo 5

CONCLUSIÓN

Habiendo concluido este Trabajo Fin de Máster, se abre la veda a la extracción y presentación en este apartado final de las conclusiones derivadas de éste. En primer lugar, se ha podido ratificar la hipótesis planteada en el primer capítulo al lograr predecir satisfactoriamente el inicio, fin y porcentaje de fase de los periodos de marcha y los resultados obtenidos no dejan nada que desear con respecto a algunos métodos presentes en la literatura, incluyendo métodos más avanzados con aprendizaje automático.

Además, se puede afirmar que se ha logrado cumplir con los objetivos secundarios establecidos. De estos, cabe destacar la implementación de un diseño experimental propio para la extracción de los datos. Bajo otras circunstancias no habría sido posible tomar directamente los datos, quedando a la merced del rigor experimental del investigador que los tomase. En cambio, poder tomar los datos directamente y dirigir a los participantes directamente es muy probable que haya tenido bastante que ver en el éxito de este trabajo.

Recapitulando lo visto, la implementación del *Unscented Kalman Filter* para la detección y predicción del porcentaje de la fase de la marcha y la predicción del ángulo de las articulaciones demostró un rendimiento robusto tanto en escenarios ideales como realistas. El filtro redujo efectivamente el error de predicción y la incertidumbre, particularmente en condiciones de estado estacionario durante la marcha, a la par que destacó los desafíos de las fases de estado no estacionario y la variabilidad en los periodos de marcha individuales. Entre las posibles mejoras de cara al trabajo futuro, debería centrarse en mejorar la precisión del modelo, manejar la variabilidad y optimizar la eficiencia computacional para mejorar la aplicabilidad práctica del UKF en el análisis de la marcha y campos relacionados. Asimismo, queda pendiente para futuros trabajo explorar la combinación del UKF con otras metodologías.

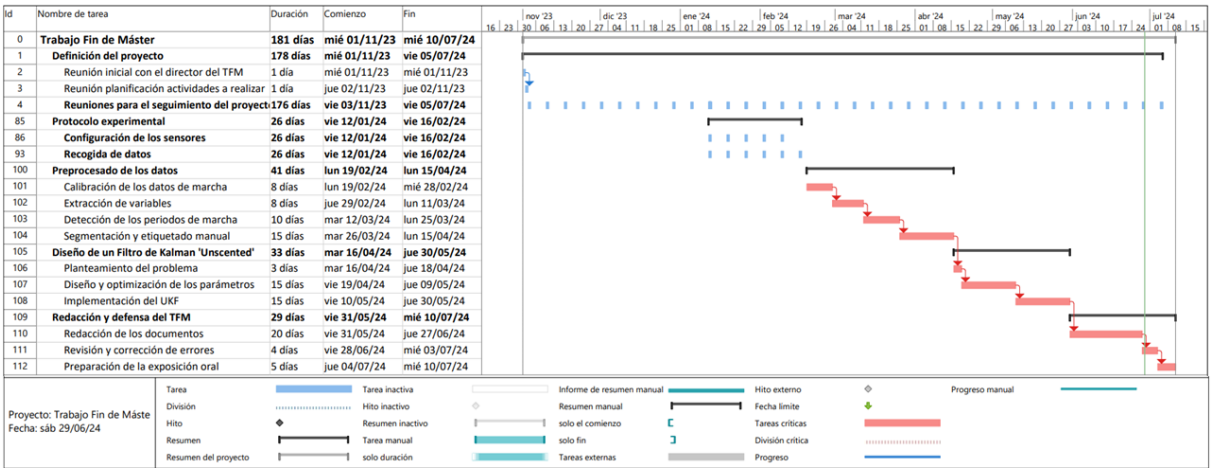
Parte II

Presupuesto

Capítulo 6

DIAGRAMA DE GANTT

En el sexto capítulo de este trabajo, se presenta un diagrama de Gantt que proporciona una visualización detallada del cronograma seguido durante el desarrollo de este documento. Así, la Figura 6.1 ofrece una perspectiva general del proyecto, mostrando las tareas a realizar y los tiempos estimados para cada una, lo que optimiza la organización global del proyecto.



Id	Nombre de tarea	Duración	Comienzo	Fin
0	**Trabajo Fin de Máster**	**181 días**	**mié 01/11/23**	**mié 10/07/24**
1	**Definición del proyecto**	**178 días**	**mié 01/11/23**	**vie 05/07/24**
2	Reunión inicial con el director del TFM	1 día	mié 01/11/23	mié 01/11/23
3	Reunión planificación actividades a realizar	1 día	jue 02/11/23	jue 02/11/23
4	**Reuniones para el seguimiento del proyecto**	**176 días**	**vie 03/11/23**	**vie 05/07/24**
85	**Protocolo experimental**	**26 días**	**vie 12/01/24**	**vie 16/02/24**
86	**Configuración de los sensores**	**26 días**	**vie 12/01/24**	**vie 16/02/24**
93	**Recogida de datos**	**26 días**	**vie 12/01/24**	**vie 16/02/24**
100	**Preprocesado de los datos**	**41 días**	**lun 19/02/24**	**lun 15/04/24**
101	Calibración de los datos de marcha	8 días	lun 19/02/24	mié 28/02/24
102	Extracción de variables	8 días	jue 29/02/24	lun 11/03/24
103	Detección de los periodos de marcha	10 días	mar 12/03/24	lun 25/03/24
104	Segmentación y etiquetado manual	15 días	mar 26/03/24	lun 15/04/24
105	**Diseño de un Filtro de Kalman 'Unscented'**	**33 días**	**mar 16/04/24**	**jue 30/05/24**
106	Planteamiento del problema	3 días	mar 16/04/24	jue 18/04/24
107	Diseño y optimización de los parámetros	15 días	vie 19/04/24	jue 09/05/24
108	Implementación del UKF	15 días	vie 10/05/24	jue 30/05/24
109	**Redacción y defensa del TFM**	**29 días**	**vie 31/05/24**	**mié 10/07/24**
110	Redacción de los documentos	20 días	vie 31/05/24	jue 27/06/24
111	Revisión y corrección de errores	4 días	vie 28/06/24	mié 03/07/24
112	Preparación de la exposición oral	5 días	jue 04/07/24	mié 10/07/24

Figura 6.1: Diagrama de Gantt del Trabajo Fin de Máster realizado

Capítulo 7

PRESUPUESTO

En relación con el capítulo anterior, este penúltimo capítulo se dedica a la presentación del estudio económico que acompaña al Trabajo Fin de Máster. Para su realización, se utilizó el software Arquímedes, desarrollado por CYPE.

El objetivo de este presupuesto es simular los costos del proyecto bajo condiciones realistas, considerando al estudiante como un ingeniero biomédico junior, con un sueldo por hora asignado según convenio. Se contabilizan un total de 500 horas efectivas de trabajo, correspondientes a los 20 créditos universitarios asignados a la realización del Trabajo Fin de Máster.

Dado que el trabajo incluye una parte experimental, el cuadro de maquinaria (Sección 7.3) no solo considera los costos amortizados del portátil del estudiante, sino también los de los sensores empleados para el registro de las mediciones. De igual forma, en el cuadro de materiales (Sección 7.2), además del costo de las licencias de los programas utilizados, en caso de ser de pago, se contabiliza el costo de las suscripciones a las revistas empleadas para la revisión de la literatura en el Capítulo1. Finalmente, en la hoja resumen (Sección 7.8) se refleja la tasa de interés general del 16 %, junto al beneficio industrial del 7 % y el impuesto sobre el valor añadido (IVA) del 21 %.

7.1 CUADRO DE PRECIOS DE MANO DE OBRA

Cuadro de mano de obra — Página 1

Núm.	Código	Denominación de la mano de obra	Precio	Horas	Total
1	MO.IBS	Ingeniero Biomédico Senior	31,00	107,00 h	3.317,00
2	MO.IBJ	Ingeniero Biomédico Junior	15,00	500,00 h	7.500,00
3	MO.SUJ	Ocho sujetos que se presentaron voluntarios para que se les tomaran datos de marcha	1,00	12,00 h	12,00
				Total mano de obra:	10.829,00

7.2 CUADRO DE PRECIOS MATERIALES

Cuadro de materiales Página 1

Núm.	Código	Denominación del material	Precio	Cantidad	Total
1	MAT.LOL	Licencia para la redacción del texto en el editor de latex online Overleaf	1,00	80,00 h	80,00
2	MAT.LO365	Licencia Office 365	0,90	32,50 h	29,70
3	MAT.LW11	Licencia Windows 11	0,90	374,00 h	336,70
4	MAT.VSC	Editor de código fuente Visual Studio Code	0,10	115,00 h	11,50
5	MAT.MTM	Interfaz gráfica para el manejo de los dispositivos de captura de movimiento desarrollado por Xsens para la fácil calibración y visualización de los datos.	0,10	12,00 h	1,20
6	MAT.BIB	Suscripciones a las bases de datos científicas consultadas (e.g., Science Direct, Web of Science, IEEE Xplore)	0,01	149,00 h	1,80
				Total materiales:	460,90

7.3 CUADRO DE PRECIOS DE MAQUINARIA

Cuadro de maquinaria Página 1

Núm.	Código	Denominación de la maquinaria	Precio	Cantidad	Total
1	MAT.IMU	Kit de invetigación con unidades de medida inercial modelo MTw Awinda de la marca Xsens y los accesorios necesarios.	6,97	24,00 h	167,28
2	MAQ.PCp	Portátil MSI	2,34	449,00 h	1.050,66
				Total maquinaria:	1.217,94

7.4 CUADRO DE PRECIOS UNITARIOS

Cuadro de precios nº 1

Nº	Designación	Importe	
		En cifra (Euros)	En letra (Euros)
	1 Definición del proyecto		
1.1	h Reunión inicial con el director del TFM	46,00	CUARENTA Y SEIS EUROS
1.2	h Reunión planificación actividades a realizar	46,00	CUARENTA Y SEIS EUROS
1.3	h Reuniones para el seguimiento del proyecto	46,00	CUARENTA Y SEIS EUROS
	2 Investigación del estado del arte		
2.1	h Investigación del estado del arte	18,25	DIECIOCHO EUROS CON VEINTICINCO CÉNTIMOS
	3 Protocolo experimental		
3.1	h Configuración de los sensores	25,31	VEINTICINCO EUROS CON TREINTA Y UN CÉNTIMOS
3.2	h Recogida de datos	26,21	VEINTISEIS EUROS CON VEINTIUN CÉNTIMOS
	4 Preprocesado de los datos		
4.1	h Calibración de los datos de marcha	18,34	DIECIOCHO EUROS CON TREINTA Y CUATRO CÉNTIMOS
4.2	h Instalación de programas necesarios	18,57	DIECIOCHO EUROS CON CINCUENTA Y SIETE CÉNTIMOS
4.3	h Detección de los periodos de marcha	18,57	DIECIOCHO EUROS CON CINCUENTA Y SIETE CÉNTIMOS
4.4	h Segmentación y etiquetado manual	18,57	DIECIOCHO EUROS CON CINCUENTA Y SIETE CÉNTIMOS
	5 Diseño de un Filtro de Kalman 'Unscented'		
5.1	h Planteamiento del problema	20,67	VEINTE EUROS CON SESENTA Y SIETE CÉNTIMOS
5.2	h Diseño y optimización de los parámetros	18,24	DIECIOCHO EUROS CON VEINTICUATRO CÉNTIMOS
5.3	h Implementación del UKF	48,34	CUARENTA Y OCHO EUROS CON TREINTA Y CUATRO CÉNTIMOS
	6 Redacción y defensa del TFM		
6.1	h Redacción de los documentos	19,25	DIECINUEVE EUROS CON VEINTICINCO CÉNTIMOS
6.2	h Revisión de los documentos y corrección de errores	22,75	VEINTIDOS EUROS CON SETENTA Y CINCO CÉNTIMOS
6.3	h Preparación de la exposición oral	19,14	DIECINUEVE EUROS CON CATORCE CÉNTIMOS

Valencia, Septiemre 2022
Ingeniería Biomédica

Javier Martín Molina

Presupuesto TFM Página 1

7.5 CUADRO DE PRECIOS DESCOMPUESTOS

Anejo de justificación de precios

Nº	Código	Ud	Descripción				Total
			1 Definición del proyecto				
1.1	01.01	**h**	**Reunión inicial con el director del TFM**				
	MO.IBS			1,00 h	Ingeniero Biomédico Senior	31,00	31,00
	MO.IBJ			1,00 h	Ingeniero Biomédico Junior	15,00	15,00
					Precio total por h		**46,00**
1.2	01.02	**h**	**Reunión planificación actividades a realizar**				
	MO.IBS			1,00 h	Ingeniero Biomédico Senior	31,00	31,00
	MO.IBJ			1,00 h	Ingeniero Biomédico Junior	15,00	15,00
					Precio total por h		**46,00**
1.3	01.03	**h**	**Reuniones para el seguimiento del proyecto**				
	MO.IBS			1,00 h	Ingeniero Biomédico Senior	31,00	31,00
	MO.IBJ			1,00 h	Ingeniero Biomédico Junior	15,00	15,00
					Precio total por h		**46,00**
			2 Investigación del estado del arte				
2.1	02.01	**h**	**Investigación del estado del arte**				
	MO.IBJ			1,00 h	Ingeniero Biomédico Junior	15,00	15,00
	MAQ.PCp			1,00 h	Portátil MSI	2,34	2,34
	MAT.BIB			1,00 h	Suscripciones a las bases de datos consultadas	0,01	0,01
	MAT.LW11			1,00 h	Licencia Windows 11	0,90	0,90
					Precio total por h		**18,25**
			3 Protocolo experimental				
3.1	03.01	**h**	**Configuración de los sensores**				
	MO.IBJ			1,00 h	Ingeniero Biomédico Junior	15,00	15,00
	MAQ.PCp			1,00 h	Portátil MSI	2,34	2,34
	MAT.IMU			1,00 h	Xsens MTw Awinda Kit	6,97	6,97
	MAT.LW11			1,00 h	Licencia Windows 11	0,90	0,90
	MAT.MTM			1,00 h	MT Manager 2022.0	0,10	0,10
					Precio total por h		**25,31**
3.2	03.02	**h**	**Recogida de datos**				
	MO.IBJ			1,00 h	Ingeniero Biomédico Junior	15,00	15,00
	MO.SUJ			1,00 h	Sujetos voluntarios	1,00	1,00
	MAQ.PCp			1,00 h	Portátil MSI	2,34	2,34
	MAT.IMU			1,00 h	Xsens MTw Awinda Kit	6,97	6,97
	MAT.LW11			1,00 h	Licencia Windows 11	0,90	0,90
					Precio total por h		**26,21**
			4 Preprocesado de los datos				
4.1	04.01	**h**	**Calibración de los datos de marcha**				
	MO.IBJ			1,00 h	Ingeniero Biomédico Junior	15,00	15,00
	MAQ.PCp			1,00 h	Portátil MSI	2,34	2,34
	MAT.LW11			1,00 h	Licencia Windows 11	0,90	0,90
	MAT.VSC			1,00 h	Visual Studio Code	0,10	0,10
					Precio total por h		**18,34**

4.2 04.02	**h**	**Instalación de programas necesarios**			
MO.IBJ		1,00 h	Ingeniero Biomédico Junior	15,00	15,00
MAQ.PCp		1,00 h	Portátil MSI	2,34	2,34
MAT.LW11		1,00 h	Licencia Windows 11	0,90	0,90
MAT.LO365		0,25 h	Licencia Office 365	0,90	0,23
MAT.VSC		1,00 h	Visual Studio Code	0,10	0,10
			Precio total por h		**18,57**
4.3 04.03	**h**	**Detección de los periodos de marcha**			
MO.IBJ		1,00 h	Ingeniero Biomédico Junior	15,00	15,00
MAQ.PCp		1,00 h	Portátil MSI	2,34	2,34
MAT.LW11		1,00 h	Licencia Windows 11	0,90	0,90
MAT.LO365		0,25 h	Licencia Office 365	0,90	0,23
MAT.VSC		1,00 h	Visual Studio Code	0,10	0,10
			Precio total por h		**18,57**
4.4 04.04	**h**	**Segmentación y etiquetado manual**			
MO.IBJ		1,00 h	Ingeniero Biomédico Junior	15,00	15,00
MAQ.PCp		1,00 h	Portátil MSI	2,34	2,34
MAT.LW11		1,00 h	Licencia Windows 11	0,90	0,90
MAT.LO365		0,25 h	Licencia Office 365	0,90	0,23
MAT.VSC		1,00 h	Visual Studio Code	0,10	0,10
			Precio total por h		**18,57**

5 Diseño de un Filtro de Kalman 'Unscented'

5.1 05.01	**h**	**Planteamiento del problema**			
MO.IBS		0,10 h	Ingeniero Biomédico Senior	31,00	3,10
MO.IBJ		1,00 h	Ingeniero Biomédico Junior	15,00	15,00
MAQ.PCp		1,00 h	Portátil MSI	2,34	2,34
MAT.LW11		0,25 h	Licencia Windows 11	0,90	0,23
			Precio total por h		**20,67**
5.2 05.02	**h**	**Diseño y optimización de los parámetros**			
MO.IBJ		1,00 h	Ingeniero Biomédico Junior	15,00	15,00
MAQ.PCp		1,00 h	Portátil MSI	2,34	2,34
MAT.LW11		1,00 h	Licencia Windows 11	0,90	0,90
			Precio total por h		**18,24**
5.3 05.03	**h**	**Implementación del UKF**			
MO.IBS		1,00 h	Ingeniero Biomédico Senior	31,00	31,00
MO.IBJ		1,00 h	Ingeniero Biomédico Junior	15,00	15,00
MAQ.PCp		1,00 h	Portátil MSI	2,34	2,34
			Precio total por h		**48,34**

6 Redacción y defensa del TFM

6.1 06.01	**h**	**Redacción de los documentos**			
MO.IBJ		1,00 h	Ingeniero Biomédico Junior	15,00	15,00
MAQ.PCp		1,00 h	Portátil MSI	2,34	2,34
MAT.LW11		1,00 h	Licencia Windows 11	0,90	0,90
MAT.BIB		0,50 h	Suscripciones a las bases de datos consultadas	0,01	0,01
MAT.LOL		1,00 h	Licencia Overleaf	1,00	1,00
			Precio total por h		**19,25**
6.2 06.02	**h**	**Revisión de los documentos y corrección de errores**			
MO.IBS		0,25 h	Ingeniero Biomédico Senior	31,00	7,75
MO.IBJ		1,00 h	Ingeniero Biomédico Junior	15,00	15,00
			Precio total por h		**22,75**

6.3 06.04	**h**	**Preparación de la exposición oral**				
MO.IBJ		1,00	h	Ingeniero Biomédico Junior	15,00	15,00
MAQ.PCp		1,00	h	Portátil MSI	2,34	2,34
MAT.LW11		1,00	h	Licencia Windows 11	0,90	0,90
MAT.LO365		1,00	h	Licencia Office 365	0,90	0,90
				Precio total por h		**19,14**

7.6 CUADRO DE MEDICIONES

01 Definición del proyecto

Nº	Ud	Descripción		Medición
01.01	H	Reunión inicial con el director del TFM		
			Total h :	1,00
01.02	H	Reunión planificación actividades a realizar		
			Total h :	2,00
01.03	H	Reuniones para el seguimiento del proyecto		
			Total h :	40,00

02 Investigación del estado del arte

Nº	Ud	Descripción		Medición
02.01	H	Investigación del estado del arte		
			Total h :	100,00

03 Protocolo experimental

Nº	Ud	Descripción		Medición
03.01	H	Configuración de los sensores		
			Total h :	12,00
03.02	H	Recogida de datos		
			Total h :	12,00

04 Preprocesado de los datos

Nº	Ud	Descripción		Medición
04.01	H	Calibración de los datos de marcha		
			Total h :	25,00
04.02	H	Instalación de programas necesarios		
			Total h :	20,00
04.03	H	Detección de los periodos de marcha		
			Total h :	30,00
04.04	H	Segmentación y etiquetado manual		
			Total h :	40,00

05 Diseño de un Filtro de Kalman 'Unscented'

Nº	Ud	Descripción		Medición
05.01	H	Planteamiento del problema		
			Total h :	20,00
05.02	H	Diseño y optimización de los parámetros		
			Total h :	40,00
05.03	H	Implementación del UKF		
			Total h :	60,00

06 Redacción y defensa del TFM

Nº	Ud	Descripción		Medición
06.01	H	Redacción de los documentos		
			Total h :	80,00
06.02	H	Revisión de los documentos y corrección de errores		
			Total h :	8,00
06.04	H	Preparación de la exposición oral		
			Total h :	10,00

7.7 PRESUPUESTOS PARCIALES

Presupuesto parcial nº 01 Definición del proyecto

Nº	Ud	Descripción		Medición	Precio	Importe
01.01	H	Reunión inicial con el director del TFM				
			Total h :	1,00	46,00	**46,00**
01.02	H	Reunión planificación actividades a realizar				
			Total h :	2,00	46,00	**92,00**
01.03	H	Reuniones para el seguimiento del proyecto				
			Total h :	40,00	46,00	**1.840,00**
		Total Presupuesto parcial nº 01 Definición del proyecto :				**1.978,00**

Presupuesto parcial nº 02 Investigación del estado del arte

Nº	Ud	Descripción		Medición	Precio	Importe
02.01	H	Investigación del estado del arte				
			Total h :	100,00	18,25	**1.825,00**
		Total Presupuesto parcial nº 02 Investigación del estado del arte :				**1.825,00**

Presupuesto parcial nº 03 Protocolo experimental

Nº	Ud	Descripción		Medición	Precio	Importe
03.01	H	Configuración de los sensores				
			Total h :	12,00	25,31	**303,72**
03.02	H	Recogida de datos				
			Total h :	12,00	26,21	**314,52**
		Total Presupuesto parcial nº 03 Protocolo experimental :				**618,24**

Presupuesto parcial nº 04 Preprocesado de los datos

Nº	Ud	Descripción		Medición	Precio	Importe
04.01	H	Calibración de los datos de marcha				
			Total h :	25,00	18,34	**458,50**
04.02	H	Instalación de programas necesarios				
			Total h :	20,00	18,57	**371,40**
04.03	H	Detección de los periodos de marcha				
			Total h :	30,00	18,57	**557,10**
04.04	H	Segmentación y etiquetado manual				
			Total h :	40,00	18,57	**742,80**
		Total Presupuesto parcial nº 04 Preprocesado de los datos :				**2.129,80**

Presupuesto parcial nº 05 Diseño de un Filtro de Kalman 'Unscented'

Nº	Ud	Descripción		Medición	Precio	Importe
05.01	H	Planteamiento del problema				
			Total h :	20,00	20,67	**413,40**
05.02	H	Diseño y optimización de los parámetros				
			Total h :	40,00	18,24	**729,60**
05.03	H	Implementación del UKF				
			Total h :	60,00	48,34	**2.900,40**
		Total Presupuesto parcial nº 05 Diseño de un Filtro de Kalman 'Unscented' :				**4.043,40**

Presupuesto parcial nº 06 Redacción y defensa del TFM

Nº	Ud	Descripción		Medición	Precio	Importe
06.01	H	Redacción de los documentos				
			Total h :	80,00	19,25	**1.540,00**
06.02	H	Revisión de los documentos y corrección de errores				
			Total h :	8,00	22,75	**182,00**
06.04	H	Preparación de la exposición oral				
			Total h :	10,00	19,14	**191,40**
		Total Presupuesto parcial nº 06 Redacción y defensa del TFM :				**1.913,40**

7.8 HOJA RESUMEN DEL PRESUPUESTO

Proyecto: Presupuesto TFM

Capítulo	Importe
1 Definición del proyecto	**1.978,00**
2 Investigación del estado del arte	**1.825,00**
3 Protocolo experimental	**618,24**
4 Preprocesado de los datos	**2.129,80**
5 Diseño de un Filtro de Kalman 'Unscented'	**4.043,40**
6 Redacción y defensa del TFM	**1.913,40**
Presupuesto de ejecución material	**12.507,84**
16% de gastos generales	2.001,25
7% de beneficio industrial	875,55
Suma	**15.384,64**
21% IVA	3.230,77
Presupuesto de ejecución por contrata	**18.615,41**

Asciende el presupuesto de ejecución por contrata a la expresada cantidad de DIECIOCHO MIL SEISCIENTOS QUINCE EUROS CON CUARENTA Y UN CÉNTIMOS.

Valencia, Diciemre 2023
Ingeniería Biomédica

Javier Martín Molina

Parte III

Anexos

Capítulo 8

ANEXOS

8.1 ANEXO I: Implementación del código

En esta primera sección del Anexo se adjunta el código de Python empleado para este trabajo. Para tratar de no expandir demasiado la longitud de un sólo *script*, se empleó un *script* principal `main.py` donde se fueron llamando el resto de *scrips* complementarios. La estructura seguida dentro del `main.py` fue la misma que la descrita a lo largo de la memoria.

Para la importación, calibración y comprobación de la calibración de los datos de las IMUs, se empleó el *script* `calibracion.py` donde se definieron las funciones `importar_datos`, `alpha_beta_theta`, `inicio_fin`, `euler_zxy` y `check_inicio_fin`. Además de estas, se definieron dos funciones auxiliares *loadquat* y *arctan2_2pi* para importar debidamente los cuaterniones y corregir la salida de arctan2 de NumpPy a un rango $[0, 2\pi]$. A lo largo de `calibracion.py` se emplean objetos y métodos de la clase *quaternion* definida en otro *script* `quaternion.py` y funciones como la `q2zxy` del *script* `rotations.py`.

Para la representación gráfica en el tiempo de los datos importados y las variables calculadas para cada sujeto, se utilizó el *script* `plot_var.py` donde se definieron las funciones `plot_flex`, `plt_acel` y `plot_gyro`.

Para la detección y segmentación de los periodos de marcha y el etiquetado manual de las fases de apoyo y balanceo en cada ciclo, se hizo uso del *script* `segmentation.py` donde se definieron las funciones `segmentation`, `labeling` y `copy_label`.

Para el cálculo y representación del porcentaje de fase de los ciclos de marcha, se empleó el *script* `Porcentaje.py` donde se definió la función `phase_time_percentage`. En este *script*, además, también se definió la función `Ciclograma` para representar los ángulos de flexo-extensión de la cadera frente a los ángulos de flexo-extensión de la rodilla.

Para el cálculo de la matriz de covarianza del ruido sobre los estados, se hizo uso del *script* `MatrizCovarianzaEstados.py` donde se definió la función `matriz_1_covarianza_ruido_estados` que devolvió la matriz Q por el método de Monte Carlo.

Para el cálculo de la matriz de covarianza del ruido sobre las salidas, se hizo uso del *script* `MatrizCovarianzaSalida.py` donde se definió la función `matriz_2_covarianza_ruido_salidas` que devolvió las matrices R_i para cada sujeto y la R del conjunto.

Para la implementación final del *Unscented Kalman Filter*, se decidió también implementarlo en un *sript* separado del `main.py` llamado `UnscentedKalmanFilter.py`, donde se define la función `UKF_ideal` para implementar el UKF en el caso ideal y la función `UKF_realista` para el caso realista. En ambos casos, emplean objetos de la clase `UnscentedKalmanFilter` definida en otro *script* bajo el nombre `UKF.py`. Además, en `UnscentedKalmanFilter.py` se definen, al inicio, varias funciones para facilitar el cálculo de las métricas evaluadas.

8.1.1 *main.py*

```
# main.py
"""
Created on Thu May 2 11:39:16 2024

@author: Javier Martín Molina
"""
import matplotlib.pyplot as plt
from data_processing.calibracion import importar_datos, alfa_beta_theta, inicio_fin,
    euler_zxy, check_inicio_fin
from graphics.plot_var import plot_flex, plot_acel, plot_gyro
from segmentation.segmentation import segmentation, labeling, copy_label
from Ciclograma.Porcentaje import phases_time_percentage, Ciclograma
from UnscentedKalmanFilter.UnscentedKalmanFiltet import UKF_ideal, UKF_realista

def main():
#_____________________________________PREPROCESADO DE LOS
    DATOS_____________________________________
   # Inicializamos algunas variables con los nombres de las IMUs y varias listas vacías
       para los datos.
   imu = ['00B41DCE', '00B41DD3', '00B41DC9']
   datos_ci, datos_d, datos_i, datos_cf = [], [], [], []
   t_ci, t_d, t_i, t_cf = [], [], [], []

   for j in range(8):

     # Importamos los datos de los ficheros .mtb
     datos, t, oQi = importar_datos(imu, j)

     # Guardamos las variables
     datos_ci.append(datos[0])
     datos_d.append(datos[1])
     datos_i.append(datos[2])
     datos_cf.append(datos[3])
     t_ci.append(t[0])
     t_d.append(t[1])
     t_i.append(t[2])
     t_cf.append(t[3])

     # Extraemos los cuaterniones de calibración y los quaterniones caminando calibrados
         (quat_T)
     # tanto empezando por la izquierda como por la derecha
     _, quat_T = alfa_beta_theta(oQi[0], oQi[1]) # oQi[1] = oQi_d
     _, [Angulo_cadera_d, Angulo_rodilla_d] = euler_zxy(quat_T, j)

     _, quat_T = alfa_beta_theta(oQi[0], oQi[2]) # oQi[2] = oQi_i
     _, [Angulo_cadera_i, Angulo_rodilla_i] = euler_zxy(quat_T, j)

     # Ploteamos los gráficos para los ángulos de FE de rodilla y cadera
     fig1 = plot_flex([Angulo_rodilla_d, Angulo_rodilla_i, Angulo_cadera_d,
         Angulo_cadera_i], j)

     # Ploteamos los gráficos de aceleración y su norma y guardamos las variables
     fig2, fig3, _, _ = plot_acel([datos_d[j], datos_i[j]], [t_d[j], t_i[j]], j)

     # # Ploteamos los gráficos de velocidad angular y su norma y guardamos las variables
     fig4, fig5, _, _ = plot_gyro([datos_d[j], datos_i[j]], [t_d[j], t_i[j]], j)
#_____________________________________SEGMENTACIÓN DE LA
    SEÑAL_____________________________________
```

```
  # Segmentación RODILLA
  ke = 2.35
  fig6d, kd, clas_d = segmentation(Angulo_rodilla_d, t[1], ke, j)
  fig6i, ki, clas_i = segmentation(Angulo_rodilla_i, t[2], ke, j)

  # Guardamos las figuras creadas para cada sujeto.
  fig1.savefig(f'Angulo_FE_S0{str(j + 1)}.png', dpi = 600)
  fig2.savefig(f'Acelerometro_D_S0{str(j + 1)}.png', dpi=600)
  fig3.savefig(f'Acelerometro_I_S0{str(j + 1)}.png', dpi=600)
  fig4.savefig(f'Giroscopio_D_S0{str(j + 1)}.png', dpi=600)
  fig5.savefig(f'Giroscopio_I_S0{str(j + 1)}.png', dpi=600)
  fig6d.savefig(f'Umbralizacion_Rodilla_D_S0{str(j + 1)}.png', dpi=600)
  fig6i.savefig(f'Umbralizacion_Rodilla_I_S0{str(j + 1)}.png', dpi=600)
  plt.close('all')

#  Etiquetamos manualmente las fases de los ángulos de rodilla:
  fig8, fig9, _, _ = labeling(Angulo_rodilla_d, t[1], clas_d, kd, j, 'Rodilla',
      'Derecha')
  fig10, fig11, _, _ = labeling(Angulo_rodilla_i, t[2], clas_i, ki, j, 'Rodilla',
      'Izquierda')

  # Segmentación los periodos de marcha y etiqutamos las fases de los ángulos de cadera
      en función
  # de los ángulos de rodilla
  fig12, fig13 = copy_label(Angulo_cadera_d, 'Rodilla', 'Cadera', 'Derecha', j)
  fig14, fig15 = copy_label(Angulo_cadera_i, 'Rodilla', 'Cadera', 'Izquierda', j)

  fig16, fig17, fig18, fig19, _, _, _, _, PASOS, muestras, _, _, _ =
      phases_time_percentage(j)
  fig20, _, _ = Ciclograma(PASOS, muestras, j)

  # Guardamos en ficheros las figuras creadas
  fig8.savefig(f'Rodilla_ON_OFF_D_S0{str(j + 1)}.png', dpi=600)
  fig9.savefig(f'Rodilla_FASES_D_S0{str(j + 1)}.png', dpi=600)
  fig10.savefig(f'Rodilla_ON_OFF_I_S0{str(j + 1)}.png', dpi=600)
  fig11.savefig(f'Rodilla_FASES_I_S0{str(j + 1)}.png', dpi=600)
  fig12.savefig(f'Cadera_ON_OFF_D_S0{str(j + 1)}.png', dpi=600)
  fig13.savefig(f'Cadera_FASES_D_S0{str(j + 1)}.png', dpi=600)
  fig14.savefig(f'Cadera_ON_OFF_I_S0{str(j + 1)}.png', dpi=600)
  fig15.savefig(f'Cadera_FASES_I_S0{str(j + 1)}.png', dpi=600)
  fig16.savefig(f'Porcentaje_Rodilla_D_S0{str(j + 1)}.png', dpi=600)
  fig17.savefig(f'Porcentaje_Rodilla_I_S0{str(j + 1)}.png', dpi=600)
  fig18.savefig(f'Porcentaje_Cadera_D_S0{str(j + 1)}.png', dpi=600)
  fig19.savefig(f'Porcentaje_Cadera_I_S0{str(j + 1)}.png', dpi=600)
  fig20.savefig(f'Ciclograma_S0{str(j + 1)}.png', dpi=600)
  plt.close('all')

# Comprobamos la diferencia entre el ángulo de calibración inicial vs final. Idealmente
    < 3º;
# como mucho < 6º
inicio_fin()

# Evaluamos si debemos retirar los ensayos aparentemente defectuosos de los sujetos S03
    (j=2) y S07 (j=6)
boxplotpre, qqplotpre, boxplotpost, qqplotpost, _ = check_inicio_fin([2, 6])

# Comprobamos que las gráficas estén bien y las guardamos
plt.show(block = True)
boxplotpre.savefig(f'Box_Plot_Pre_Calibración.png', dpi=600)
qqplotpre.savefig(f'QQ_Plot_Pre_Calibración.png', dpi=600)
boxplotpost.savefig(f'Box_Plot_Post_Calibración.png', dpi=600)
```

```
        qqplotpost.savefig(f'QQ_Plot_Post_Calibración.png', dpi=600)
        plt.close('all')

    # ______________________________________UNSCENTED KALMAN
        FILTER_____________________________________________

    # Llamamos a las funciones creadas
    UKF_ideal()
    UKF_realista()
```

8.1.2 *Calibracion.py*

```
# Calibración.py
import os
import numpy as np
from numpy import hstack, zeros, ones, array, arctan2, cos, sin, arange, percentile, shape,
    asarray, pi
from numpy.linalg import norm
import pandas as pd
from pandas import read_csv as read
from quaternions_dev.quaternions.quaternions import quaternion as Q
from quaternions_dev.quaternions.rotations import q2zxy
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.formula.api import ols
from scipy.stats import levene, skewtest, kurtosistest, yeojohnson

def loadquat(datos):
    # Sacamos los quaterniones de las útlimas 4 columnas por orden
    q0 = datos[:,-4]
    q1 = datos[:,-3]
    q2 = datos[:,-2]
    q3 = datos[:,-1]
    # q = array([q0, q1, q2, q3]).T
    return Q(q0, q1, q2, q3)

def arctan2_2pi(y, x):
    angle = arctan2(y, x)
    if type(angle) is np.float64:
        if angle < 0:
            angle += 2 * pi
    else:
        for i in range(len(angle)):
            if angle[i] < 0:
                angle[i] += 2 * pi
    return angle


#
    ------------------------------------------------------------------------------------------------------
def importar_datos(imu: list, sujeto: int):
    j = sujeto
    datos_ci = []
    datos_i = []
    datos_d = []
    datos_cf = []
    t_ci = []
    t_cf = []
    t_i = []
```

```
t_d = []
oQi_ci = []
oQi_d = []
oQi_i = []
oQi_cf = []
os.chdir(r'C:\Users\marti\OneDrive\Documentos\23-24\TFM\kk2\Export 2')

for n in range(0, len(imu)):
    # Leemos tomando seperación irregular y saltamos las primeras filas (texto)
    datos = read(r'.\S0' + str(j + 1) + '_Calibracion_Inicial_' + imu[n] + '.txt',
        sep='\s+', skiprows=12)

    # Datos de la calibración inicial
    if n > 0:
        dif = len(datos_ci[-1])- len(datos)

        if dif > 0:
            datos_ci = list(asarray(datos_ci)[:, :-abs(dif), :])
            datos = asarray(datos)

        elif dif < 0:
            datos = asarray(datos)[:-abs(dif), :]

        else:
            datos = asarray(datos)

    else:
        datos = asarray(datos)

    datos_ci.append(datos)

    # Datos de paseo empezando con la pierna derecha
    datos = read(r'.\S0' + str(j + 1) + '_Pie_Derecho_' + imu[n] + '.txt', sep='\s+',
        skiprows=12)

    if n > 0:
        dif = len(datos_d[-1])- len(datos)

        if dif > 0:
            datos_d = list(asarray(datos_d)[:, :-abs(dif), :])
            datos = asarray(datos)

        elif dif < 0:
            datos = asarray(datos)[:-abs(dif), :]

        else:
            datos = asarray(datos)

    else:
        datos = asarray(datos)

    datos_d.append(datos)

    # Datos del paseo empezando con la pierna izquierda
    datos = read(r'.\S0' + str(j + 1) + '_Pie_Izquierdo_' + imu[n] + '.txt', sep='\s+',
        skiprows=12)

    if n > 0:
        dif = len(datos_i[-1])- len(datos)

        if dif > 0:
```

```
                datos_i = list(asarray(datos_i)[:, :-abs(dif), :])
                datos = asarray(datos)

            elif dif < 0:
                datos = asarray(datos)[:-abs(dif), :]

            else:
                datos = asarray(datos)

        else:
            datos = asarray(datos)

        datos_i.append(datos)

        # Datos de la calibración final
        datos = read(r'.\S0' + str(j + 1) + '_Calibracion_Final_' + imu[n] + '.txt', sep='\s+',
            skiprows=12)
        if n > 0:
            dif = len(datos_cf[-1])- len(datos)

            if dif > 0:
                datos_cf = list(asarray(datos_cf)[:, :-abs(dif), :])
                datos = asarray(datos)

            elif dif < 0:
                datos = asarray(datos)[:-abs(dif), :]

            else:
                datos = asarray(datos)

        else:
            datos = asarray(datos)

        datos_cf.append(datos)

        # Creamos objetos de clase cuaternión de los datos y los anexionamos a las listas
        nuevo_ci = loadquat(datos_ci[n])
        oQi_ci.append(Q(*nuevo_ci/norm(nuevo_ci, axis = 0)))
        nuevo_i = loadquat(datos_i[n])
        oQi_i.append(Q(*nuevo_i/norm(nuevo_i, axis = 0)))
        nuevo_d = loadquat(datos_d[n])
        oQi_d.append(Q(*nuevo_d/norm(nuevo_d, axis = 0)))
        nuevo_cf = loadquat(datos_cf[n])
        oQi_cf.append(Q(*nuevo_cf/norm(nuevo_cf, axis = 0)))

    # Asumiendo 100 Hz de frecuencia de muestreo en base a los datos de los sensores y el
        manual de uso
    fs = 100 # Hz
    t_ci = arange(0, shape(datos_ci[0])[0]/fs, 1/fs)
    t_d = arange(0, shape(datos_d[0])[0]/fs, 1/fs)
    t_i = arange(0, shape(datos_i[0])[0]/fs, 1/fs)
    t_cf = arange(0, shape(datos_cf[0])[0]/fs, 1/fs)

    datos = [datos_ci, datos_d, datos_i, datos_cf]
    t = [t_ci, t_d, t_i, t_cf]
    oQi = [oQi_ci, oQi_d, oQi_i, oQi_cf]

    return datos, t, oQi

#
    ------------------------------------------------------------------------------------------------------
```

```
def alfa_beta_theta(oQi_calibracion: Q, oQi_medidas: Q):
    # Separamos los cuaterniones de las medidas caminando para cada articulación: pelvis, fémur
        o tibia
    shape_min = min(np.shape(oQi_medidas[0]), np.shape(oQi_medidas[1]),
        np.shape(oQi_medidas[2]))
    lmin = shape_min[1]
    oQip = Q(*np.array(oQi_medidas[0])[:, : lmin])
    oQif = Q(*np.array(oQi_medidas[1])[:, : lmin])
    oQit = Q(*np.array(oQi_medidas[2])[:, : lmin])

    # Separamos los cuaterniones de las medidas estáticas para cada articulación: pelvis, fémur
        o tibia
    shape_min = min(np.shape(oQi_calibracion[0]), np.shape(oQi_calibracion[1]),
        np.shape(oQi_calibracion[2]))
    lmin = shape_min[1]
    oQip_c = Q(*np.array(oQi_calibracion[0])[:, : lmin])
    oQif_c = Q(*np.array(oQi_calibracion[1])[:, : lmin])
    oQit_c = Q(*np.array(oQi_calibracion[2])[:, : lmin])

    # Creamos los cuaterniones de los giros elementales de pi/2 en los ejes x, y, z.
    Qx = Q(*ones(shape_min) * array([cos(pi/4), sin(pi/4), 0, 0], ndmin = 2).T)
    Qy = Q(*ones(shape_min) * array([cos(pi/4), 0, sin(pi/4), 0], ndmin = 2).T)
    Qz = Q(*ones(shape_min) * array([cos(pi/4), 0, 0, sin(pi/4)], ndmin = 2).T)

    # Convertimos a secuencia de Euler ZXY para sacar Theta y calculamos el cuaternión del giro
        puro´
    # en el caso estático
    Q_p = oQip_c * (Qx * Qx * Qy).c
    Euler = q2zxy(Q_p)
    Theta = Euler[0]
    Q_prima = Q(cos(Theta / 2), zeros(shape(Theta)), zeros(shape(Theta)), sin(Theta / 2))
    Q_epsilon = (Q_prima * Qx).c * oQip_c * (Qx * Qy).c

    # Sacamos los giros de los sensores a los segmentos asociados en cada caso.
    pQi = Q_epsilon * (Qx * Qy)

    oQf = Q_prima * Qx
    fQi = oQf.c * oQif_c

    oQt = Q_prima * Qx
    tQi = oQt.c * oQit_c

    # Concatenamos para igualar el número de muestras entre las medidas de calibración y
    # las medidas caminando. También se podría interpolar las medidas de calibración
    pQi = asarray(pQi)
    fQi = asarray(fQi)
    tQi = asarray(tQi)

    l = shape(oQip)[1]
    ll = shape(pQi)[1]
    while ll < l:
        pQi = hstack([pQi, pQi])
        ll = shape(pQi)[1]
    pQi = Q(*pQi[:, :-abs(ll-l)])

    l = shape(oQif)[1]
    ll = shape(fQi)[1]
    while ll < l:
        fQi = hstack([fQi, fQi])
        ll = shape(fQi)[1]
```

```
    fQi = Q(*fQi[:, :-abs(ll-l)])

    l = shape(oQit)[1]
    ll = shape(tQi)[1]
    while ll < l:
        tQi = hstack([tQi, tQi])
        ll = shape(tQi)[1]
    tQi = Q(*tQi[:, :-abs(ll-l)])

    # Finalmente, calculamos el cuaternión del giro Theta en los casos dinámicos
    oQp = oQip * pQi.c
    oQf = oQif * fQi.c
    oQt = oQit * tQi.c

    quat_AB = [pQi, fQi, tQi]
    quat_T = [oQp, oQf, oQt]

    return quat_AB, quat_T


#
    ------------------------------------------------------------------------------------------------------
def inicio_fin():
    imu = ['00B41DCE', '00B41DD3', '00B41DC9']
    angulos_inicio_fin = pd.DataFrame([])

    for j in range(0, 8):
        # Separamos los cuaterniones según el segmento al que correspondan
        _, _, oQi = importar_datos(imu, j)
        quat_AB, _ = alfa_beta_theta(oQi[0], oQi[1])
        pQi_ci = quat_AB[0]
        fQi_ci = quat_AB[1]
        tQi_ci = quat_AB[2]

        quat_AB, _ = alfa_beta_theta(oQi[3], oQi[1])
        pQi_cf = quat_AB[0]
        fQi_cf = quat_AB[1]
        tQi_cf = quat_AB[2]

        # Calculamos el cuaternión que expresa el giro del SdR del sensor en la calibración
            inicial al
        # SdR del sensor en la calibración final y extraemos el ángulo de giro con arctan2
            corregida.
        lmin = min(shape(asarray(pQi_cf))[1], shape(asarray(pQi_ci))[1])
        qcal_pelvis = Q(*asarray(pQi_cf)[:, :lmin]).c * Q(*asarray(pQi_ci)[:, :lmin])
        pelvis_inicio_fin = [percentile(2 * (arctan2_2pi(norm(qcal_pelvis.v), qcal_pelvis.e)),
            5) *180/pi - 180,
                             percentile(2 * (arctan2_2pi(norm(qcal_pelvis.v), qcal_pelvis.e)), 50)
                                 *180/pi - 180,
                             percentile(2 * (arctan2_2pi(norm(qcal_pelvis.v), qcal_pelvis.e)), 95)
                                 *180/pi - 180]
        lmin = min(shape(asarray(fQi_cf))[1], shape(asarray(fQi_ci))[1])
        qcal_femur = Q(*asarray(fQi_cf)[:,:lmin]).c * Q(*asarray(fQi_ci)[:, :lmin])
        femur_inicio_fin = [percentile(2 * (arctan2_2pi(norm(qcal_femur.v), qcal_femur.e)), 5)
            *180/pi - 180,
                             percentile(2 * (arctan2_2pi(norm(qcal_femur.v), qcal_femur.e)), 50)
                                 *180/pi - 180,
                             percentile(2 * (arctan2_2pi(norm(qcal_femur.v), qcal_femur.e)), 95)
                                 *180/pi - 180]
        lmin = min(shape(asarray(tQi_cf))[1], shape(asarray(tQi_ci))[1])
        qcal_tibia = Q(*asarray(tQi_cf)[:, :lmin]).c * Q(*asarray(tQi_ci)[:, :lmin])
```

```
        tibia_inicio_fin = [percentile(2 * (arctan2_2pi(norm(qcal_tibia.v), qcal_tibia.e)), 5)
            *180/pi - 180,
                        percentile(2 * (arctan2_2pi(norm(qcal_tibia.v), qcal_tibia.e)), 50)
                            *180/pi - 180,
                        percentile(2 * (arctan2_2pi(norm(qcal_tibia.v), qcal_tibia.e)), 95)
                            *180/pi - 180]

        # Guardamos los datos en un DataFrame para exportarlo a un .csv
        angulos = pd.DataFrame({'Sujeto': [j + 1] * 3,
                            'Cuantil': ['Q1', 'Q2', 'Q3'],
                            'Pelvis': pelvis_inicio_fin,
                            'Fémur': femur_inicio_fin,
                            'Tibia': tibia_inicio_fin})
        angulos_inicio_fin = pd.concat([angulos_inicio_fin, angulos])

    angulos_inicio_fin.to_csv('Ángulos_calibracion_final_inicial.csv', index = False)


#
     ------------------------------------------------------------------------------------------------------
def euler_zxy(quat_T, j):
    # Separamos los cuaterniones de los datos según el segmento
    lmin = min(shape(quat_T[0])[1], shape(quat_T[1])[1], shape(quat_T[2])[1])
    oQp = Q(*asarray(quat_T[0])[:, :lmin])
    oQf = Q(*asarray(quat_T[1])[:, :lmin])
    oQt = Q(*asarray(quat_T[2])[:, :lmin])

    # Calculamos los ángulos de Euler de los giros según la secuencia ZXY a partir del
        cuaternión que
    # pasa del SdR del fémur a la cadera
    pQf = oQp.c * oQf
    Euler_cadera = q2zxy(pQf.c)
    norma_Euler_cadera = np.linalg.norm(Euler_cadera, axis = 0)

    # Calculamos los ángulos de Euler de los giros según la secuencia ZXY a partir del
        cuaternión que
    # pasa del SdR de la tibia al fémur
    fQt = oQf.c * oQt
    Euler_rodilla = q2zxy(fQt)
    norma_Euler_rodilla = np.linalg.norm(Euler_rodilla, axis = 0)

    # Extraemos el ángulo del eje Z (Flexo-extensión) corrigiendo algunos cambios de ejes
    eje = 0
    if j in [3,4,5]: eje = 1

    i = 0
    if j in [0,4,5,7]: i = 1

    Angulo_rodilla = (-1)**(i) * np.array(Euler_rodilla[eje])

    eje = 0
    if j in [3,4,5]: eje = 1

    i = 0
    if j in [0,4,5,7]: i = 1

    Angulo_cadera = (-1)**(i) * np.array(Euler_cadera[eje])

    return [norma_Euler_cadera, norma_Euler_rodilla], [Angulo_cadera, Angulo_rodilla]

#_____________________________________________________________________________________________________
```

```
def check_inicio_fin(defectuosos):
    # Cambiamos al directorio con los datos
    os.chdir(r'C:\Users\marti\OneDrive\Documentos\23-24\TFM\kk2\Export 2\CSV')

    #Creamos las variables que necesitamos a priori
    fig, ax = plt.subplots(2, 2, figsize=(12, 8), dpi = 80)
    fig1, ax1 = plt.subplots(2, 2, figsize=(12, 8), dpi = 80)
    fig2, ax2 = plt.subplots(2, 2, figsize=(12, 8), dpi = 80)
    fig3, ax3 = plt.subplots(2, 2, figsize=(12, 8), dpi = 80)
    tabla1 = pd.DataFrame([])
    tabla2 = pd.DataFrame([])
    tabla3 = pd.DataFrame([])
    tablaNormalPre = pd.DataFrame([])
    tablaNormalPost = pd.DataFrame([])
    anova_table = pd.DataFrame([])
    p_Levene = []

    for u1, Joint in enumerate(['Rodilla', 'Cadera']):
        for u2, Lado in enumerate(['Derecha', 'Izquierda']):
            normaJ = pd.DataFrame([])
            normas_box = []

            # Leemos los datos de cada sujeto y creamos las variables para contenerlos
            for j in range(8):
                datos = pd.read_csv(r'.\umbral_' + f'{Joint}_ciclos_{Lado}_S0{j + 1}.csv')
                jota = pd.DataFrame({'J': np.ones_like(datos.iloc[:,1]) * [j + 1],
                                     'Slip': np.zeros_like(datos.iloc[:,1]) + 1 if j in defectuosos
                                         else np.zeros_like(datos.iloc[:,1]),
                                     'Ángulo': (datos.iloc[:,1])})
                normas_box.append(list(datos.iloc[:,1]))
                normaJ = pd.concat([normaJ, jota], axis = 0)

            # Convertimos el número del sujeto y la presencia o no de desliz del sensor
            # a variables categóricas
            normaJ['J'] = normaJ['J'].astype('category')
            normaJ['J'] = normaJ['J'].cat.codes
            normaJ['Slip'] = normaJ['Slip'].astype('category')
            normaJ['Slip'] = normaJ['Slip'].cat.codes

            # Número de muestras por sujeto
            tabla1 = pd.concat([tabla1, normaJ.groupby('J')['Slip'].value_counts()], axis = 1)

            # Media y desviación típica por grupo
            tabla2 = pd.concat([tabla2, normaJ.groupby('J').agg(['mean', 'std'])], axis = 1)

            # Media y desviación típica por desliz o no
            tabla3 = pd.concat([tabla3, normaJ.groupby('Slip').agg(['mean', 'std'])], axis = 1)

            # Evaluamos la normalidad de los datos original (Sesgo, Kurtosis, Box Plot y Gráfico
                QQ)
            skew = skewtest(normaJ.Ángulo)
            kurt = kurtosistest(normaJ.Ángulo)
            KurtoSkePre = pd.DataFrame({('', ''): [f'{Joint} {Lado}'],
                                        ('Std. Skewness','Statistic'): [skew.statistic],
                                        ('Std. Skewness','p-Value'): [skew.pvalue],
                                        ('Std. Kurtosis','Statistic'): [kurt.statistic],
                                        ('Std. Kurtosis','p-Value'): [kurt.pvalue]})
            tablaNormalPre = pd.concat([tablaNormalPre, KurtoSkePre])

            # Box Plot de los datos originales por sujeto
            ax[u1, u2].boxplot(normas_box, showmeans=True)
```

```
        ax[u1, u2].set_title(f'Paseo empezando con la pierna {Lado}')
        ax[u1, u2].set_xlabel('Sujeto')
        ax[u1, u2].set_ylabel(f'Ángulo {Joint}')

        # QQ Plot de los datos originales por sujeto
        sm.qqplot(normaJ.Ángulo -2, line = '45', ax = ax1[u1, u2])
        ax1[u1, u2].set_title(f'Paseo empezando con la pierna {Lado} ({Joint})')

        # Aplicamos la transformación Box-Cox (una variante)
        normabox = [yeojohnson(element)[0] for element in normas_box]
        normaJ.Ángulo = yeojohnson(np.array(abs(normaJ.Ángulo)))[0]

        # Volvemos a evaluar la normalidad tras transformar (Sesgo, Kurtosis, Box Plot y
            Gráfico QQ)
        skew = skewtest(normaJ.Ángulo)
        kurt = kurtosistest(normaJ.Ángulo)
        KurtoSkePost = pd.DataFrame({('', ''): [f'{Joint} {Lado}'],
                                ('Std. Skewness','Statistic'): [skew.statistic],
                                ('Std. Skewness','p-Value'): [skew.pvalue],
                                ('Std. Kurtosis','Statistic'): [kurt.statistic],
                                ('Std. Kurtosis','p-Value'): [kurt.pvalue]})
        tablaNormalPost = pd.concat([tablaNormalPost, KurtoSkePost])

        # QQ Plot de los datos transformados por sujeto
        sm.qqplot(normaJ.Ángulo -2, line = '45', ax = ax2[u1, u2])
        ax2[u1, u2].set_title(f'Paseo empezando con la pierna {Lado} ({Joint})')

        # Box Plot de los datos trasnformados por sujeto
        ax3[u1, u2].boxplot(normabox, showmeans=True)
        ax3[u1, u2].set_title(f'Paseo empezando con la pierna {Lado}')
        ax3[u1, u2].set_xlabel('Sujeto')
        ax3[u1, u2].set_ylabel(f'Ángulo {Joint}')

        # Evaluamos homocedasticidad tras transformar los datos con el test de Levene
        p_Levene += [levene(*normabox, center='mean').pvalue]

        # ANOVA de 2 factores más interacción con coeficientes de matriz de covarianza
        # corregidos para heterocedasticidad (no se cumple homocedasticidad)
        model = ols('Ángulo ~ C(J) + C(Slip) + C(J):C(Slip)', data =
            normaJ).fit(cov_type='HC3')
        anova_table = pd.concat([anova_table, sm.stats.anova_lm(model, typ = 1,
            robust='HC3')], axis = 1)

# Guardamos .csv
tabla1 = tabla1.reset_index()
tabla1.columns = ['Sujeto', 'Desliz', 'Rodilla Derecha', 'Rodilla Izquierda', 'Cadera
    Derecha', 'Cadera Izquierda']
tabla1.to_csv(f'Número_muestras_por_sujeto_prueba.csv', index = False)

tabla2 = tabla2.reset_index()
tabla2 = tabla2.drop(( 'Slip', 'std'), axis = 1)
tabla2 = tabla2.rename(columns = {'J':'Sujeto', 'Slip': 'Desliz', 'mean': 'Media',
    'std':'S.D.'})
tabla2.to_csv(f'Estadísticos_por_sujeto_prueba.csv', index = False)

tabla3 = tabla3.reset_index()
tabla3 = tabla3.drop('J', axis =1)
tabla3 = tabla3.rename(columns = {'Slip': 'Desliz', 'mean': 'Media', 'std':'S.D.'})
tabla3.to_csv(f'Estadísticos_por_desliz_prueba.csv', index = False)

tablaNormalPre.to_csv('Check_Pre_Normal_por_prueba.csv', index = False)
```

```
    tablaNormalPost.to_csv('Check_Post_Normal_por_prueba.csv', index = False)
    anova_table.to_csv('ANOVA_por_prueba.csv', index = False)

    fig.subplots_adjust(hspace = 0.3)
    fig1.subplots_adjust(hspace = 0.3)
    fig2.subplots_adjust(hspace = 0.3)
    fig3.subplots_adjust(hspace = 0.3)

    print(f'Levene (P-valor): {p_Levene}')

    return fig, fig1, fig2, fig3, anova_table
```

8.1.3 plot_var.py

```
# plot_var.py

import numpy as np
from numpy.linalg import norm as norm
import matplotlib.pyplot as plt
from matplotlib.pyplot import subplots, subplots_adjust

def plot_flex(datos:list, j: int):
    # Separamos las variables de ángulos de flexo-extensión según sean de la cadera o la
        rodilla y
    # según sea el paseo empezando con la pierna derecha o con la izquierda
    Angulo_rodilla_d, Angulo_rodilla_i, Angulo_cadera_d, Angulo_cadera_i = datos
    ts =1/100
    fig = plt.figure(figsize=(12, 8), dpi = 80)
    plt.subplot(2,2,1)
    plt.plot(np.arange(0, len(Angulo_rodilla_d)*ts, ts), Angulo_rodilla_d)
    plt.title('Ángulo Rodilla (Derecha)')
    plt.legend(['Z', 'X', 'Y'])
    plt.xlabel('tiempo (s)')
    plt.ylabel('Ángulo ($^\circ$)')
    plt.subplot(2,2,2)
    plt.plot(np.arange(0, len(Angulo_rodilla_i)*ts, ts), Angulo_rodilla_i)
    plt.title('Ángulo Rodilla (Izquierda)')
    plt.legend(['Z', 'X', 'Y'])
    plt.xlabel('tiempo (s)')
    plt.ylabel('Ángulo ($^\circ$)')
    plt.subplot(2,2,3)
    plt.plot(np.arange(0, len(Angulo_cadera_d)*ts, ts), Angulo_cadera_d)
    plt.title('Ángulo Cadera (Derecha)')
    plt.legend(['Z', 'X', 'Y'])
    plt.xlabel('tiempo (s)')
    plt.ylabel('Ángulo ($^\circ$)')
    plt.subplot(2,2,4)
    plt.plot(np.arange(0, len(Angulo_cadera_i)*ts, ts), Angulo_cadera_i)
    plt.title('Ángulo Cadera (Izquierda)')
    plt.legend(['Z', 'X', 'Y'])
    plt.xlabel('tiempo (s)')
    plt.ylabel('Ángulo ($^\circ$)')
    plt.subplots_adjust(hspace = 0.3)
    fig.suptitle(f'Sujeto SO{j+1}')

    return fig

def plot_acel(datos: list, tiempo: list, j: int):
    # Separamos las variables de aceleración lineal según sean de la cadera o la rodilla y
```

```
    # según sea el paseo empezando con la pierna derecha o con la izquierda
    etq = ['Pelvis', 'Fémur', 'Tibia']
    lado = [' (Derecha)', ' (Izquierda)']
    Acel_d, Acel_i, norma_Acel_d, norma_Acel_i = [], [], [], []

    fig2, axs2 = subplots(3, 2)
    for row in range(0, 3):
        # nn = 3 * j + row
        Acel_d.append(datos[0][row][:, 1 : 4]) # datos del paseo empezando por la pierna derecha
        Acel_i.append(datos[1][row][:, 1 : 4]) # datos del paseo empezando por la pierna
            izquierda

        for col in range(0, 2):
            acel = [Acel_d[-1], Acel_i[-1]]
            axs2[row, col].plot(tiempo[col], acel[col][:, 0], label = 'X')
            axs2[row, col].plot(tiempo[col], acel[col][:, 1], label = 'Y')
            axs2[row, col].plot(tiempo[col], acel[col][:, 2], label = 'Z')
            axs2[row, col].legend(loc = 'best', fontsize = 10)
            axs2[row, col].set(ylabel = 'Aceleración ($m/s^{2}$)')
            axs2[row, col].set_title(etq[row] + ' - SO' + str(j + 1) + lado[col])

    subplots_adjust(hspace = 0.5)

    # Representamos también la norma de las aceleraciones representadas antes
    fig3, axs3 = subplots(3, 2)
    for row in range(0, 3):
        norma_Acel_d.append(norm(Acel_d[row], axis = 1))
        norma_Acel_i.append(norm(Acel_i[row], axis = 1))

        for col in range(0, 2):
            norma = [norma_Acel_d[-1], norma_Acel_i[-1]]
            axs3[row, col].plot(tiempo[col], norma[col])
            axs3[row, col].set(xlabel = 'tiempo (s)', ylabel = 'Norma aceleración ($m/s^{2}$)')
            axs3[row, col].set_title(etq[row] + ' - SO' + str(j + 1) + lado[col])

    subplots_adjust(hspace = 0.5)

    Acel = [Acel_d, Acel_i]
    Acel_norm = [norma_Acel_d, norma_Acel_i]

    return fig2, fig3, Acel, Acel_norm


def plot_gyro(datos: list, tiempo: list, j: int):
    # Separamos las variables de velocidad angular según sean de la cadera o la rodilla y
    # según sea el paseo empezando con la pierna derecha o con la izquierda
    etq = ['Pelvis', 'Fémur', 'Tibia']
    lado = [' (Derecha)', ' (Izquierda)']
    Gyro_d, Gyro_i, norma_Gyro_d, norma_Gyro_i = [], [], [], []

    fig4, axs4 = subplots(3, 2)
    for row in range(0, 3):
        nn = 3 * (j - 2) + row
        Gyro_d.append(datos[0][row][:, 4 : 7]) # datos del paseo empezando por la pierna derecha
        Gyro_i.append(datos[1][row][:, 4 : 7]) # datos del paseo empezando por la pierna
            izquierda

        for col in range(0, 2):
            gyro = [Gyro_d[-1], Gyro_i[-1]]
            axs4[row, col].plot(tiempo[col], gyro[col][:, 0], label = 'X')
            axs4[row, col].plot(tiempo[col], gyro[col][:, 1], label = 'Y')
```

```
            axs4[row, col].plot(tiempo[col], gyro[col][:, 2], label = 'Z')
            axs4[row, col].legend(loc = 'best', fontsize = 10)
            axs4[row, col].set(xlabel = 'tiempo (s)', ylabel = 'Velocidad angular (rads/s)')
            axs4[row, col].set_title(etq[row] + ' - S0' + str(j + 1) + lado[col])

    subplots_adjust(hspace = 0.5)

    # Representamos también la norma de las velocidades angulares representadas antes
    fig5, axs5 = subplots(3, 2)
    for row in range(0, 3):
        norma_Gyro_d.append(norm(Gyro_d[row], axis = 1))
        norma_Gyro_i.append(norm(Gyro_i[row], axis = 1))

        for col in range(0, 2):
            norma = [norma_Gyro_d[-1], norma_Gyro_i[-1]]
            axs5[row, col].plot(tiempo[col], norma[col])
            axs5[row, col].set(xlabel = 'tiempo (s)', ylabel = 'Norma aceleración ($m/s^{2}$)')
            axs5[row, col].set_title(etq[row] + ' - S0' + str(j + 1) + lado[col])

    subplots_adjust(hspace = 0.5)

    Gyro = [Gyro_d, Gyro_i]
    Gyro_norm = [norma_Gyro_d, norma_Gyro_i]

    return fig4, fig5, Gyro, Gyro_norm
```

8.1.4 *segmentation.py*

```
# segmentation.py

import os
import numpy as np
from numpy import std, log, exp
import matplotlib.pyplot as plt
from matplotlib.pyplot import plot, subplots, setp, subplots_adjust
import pandas as pd

def segmentation(s: list, tiempo: list, ke, j):
    fig6, axs6 = plt.subplots(2, figsize=(12, 8), dpi = 80)
    # Tamaño de ventana
    k = int(len(s) // (ke * np.std(s)))
    # Mínimo valor de la señal
    minimo = abs(min(s)) + 1
    # Sesgo cte. de la señal transformada (Offset inicial)
    o = np.exp(ke * np.log(s[k] + minimo)) - minimo
    # Señal maximizando valores altos y minimizando bajos
    ss = np.exp(ke * (np.log(abs(np.array(s) + minimo)))) - o
    # Deriva de la señal transformada (Tendencia inicial)
    d = (ss[- k - 100] - ss[k]) / (len(ss) - 2 * k- 100)
    # Inicializamos otras variables.
    c = [] # Clasificación.
    thh = []
    susumma = []
    susu = []
    # Para cada ventana de la señal sumamos y comparamos con el valor del umbral.
    for n in range(0, len(ss) - k):
        # Corregimos el threshold adaptativamente asumiendo linealidad
        th = 0.05 * max(ss) + o + d * n
        summa = []
```

```
            # Tomamos cada muestra de la ventana
            for u in range(0, k):
                summa.append((ss[n + u]))

                # c siempre empieza en 0 (quieto)
                if n == 0:
                    c.append(0)

            if max(summa) > th and n > 0:
                c.append(1)

            elif max(summa) <= th and n > 0:
                c.append(0)

            # Guardamos las demás variables que queramos plotear
            thh.append(th)
            susumma.append(max(summa))
            susu.append(std(summa))

        # Repetimos el útlimo valor al final la mitad de la ventana veces (redondeado al alza)
        cc = c.extend([c[-1]] * (k // 2 + k % 2))
        # Luego quitamos la misma cantidad del inicio para quitar el desfase.
        cc = c[(k // 2 + k % 2) :]

        # Gráficos
        axs7 = axs6[0].twinx()
        axs6[0].plot(tiempo[0 : len(c)], s[0 : len(c)], label = 'Ángulo ($^\circ$)')
        axs7.plot(tiempo[0 : len(cc)], cc, 'r', label = 'Etiqueta')
        axs6[0].set_title(f'Tamaño de ventana: {k} - cte: {ke}')
        axs6[0].set_ylabel(f'Ángulo Rodilla ($^\circ$)')
        axs6[0].legend(loc = 'upper left', fontsize = 10)
        axs7.legend(loc = 'upper right', fontsize = 10)
        plt.setp(axs7, ylim = [-0.1, 1.1])
        axs6[1].plot(tiempo[0 : len(thh)], thh, label = 'Umbral')
        axs6[1].plot(tiempo[0 : len(susumma)], susumma, label = f'Máximo ventana')
        axs6[1].set_title('$f_{th}(n)$' + f' = 0.05 $\cdot$ {round(max(ss), 2)} + {round(o, 2)} +
            {round(d, 2)} $\cdot$ n')
        axs6[1].set_ylabel('exp(ln($\phi_r + 1$)) - '+ str(round(o, 2)))
        axs6[1].legend(loc = 'upper left', fontsize = 10)
        fig6.suptitle(f'Clasificación de los periodos dinámicos S0{j + 1}', y = 0.95, x = 0.508)
        fig6.supxlabel('tiempo (s)', y = +0.04)
        plt.subplots_adjust(hspace = 0.3)

        return fig6, k, cc

#_____________________________________________________________________________________________________
def labeling(señal_sin_segmentar: list, tiempo_sin_segmentar: list, clas: list,
             k: list, j: int, joint: str, lado: str):

    señal_segmentada = []
    tiempo_segmentado = []
    indices_on_off = []
    labels_on_off = [0] * len(señal_sin_segmentar)
    indices_fases = []
    indices_f = []
    df_ciclos = pd.DataFrame([])
    fases = 2
    pasos = 4

    # Sacamos los indices de los puntos de cambio de la señal de periodos de marcha detectados
```

```
bp = [0] + [idx for idx in range(1, len(clas) - k) if clas[idx] != clas[idx - 1]]+
    [len(señal_sin_segmentar)] # breakpoints (localización)

# Para cada una de las muestras de de los 4 pasos
for i in range(0, len(bp)//4):
    # Evitamos los índices de giros y periodos de pausa, para eliminarlos
    idx = 4 * i + 1
    idx1 = bp[idx]
    idx2 = bp[idx + 1]

    # Guardamos los índices para más adelante
    indices_on_off.extend([(idx1, idx2)])
    señal_segmentada.append(señal_sin_segmentar[idx1 : idx2])
    tiempo_segmentado.append(tiempo_sin_segmentar[idx1 : idx2])
    labels_on_off[idx1 : idx2] = [1] * (idx2 - idx1)

# Guardamos las labels en un fichero aparte:
df_on_off = pd.DataFrame({'Tiempo (s)': tiempo_sin_segmentar,
                    'Ángulo ($^\circ$)': señal_sin_segmentar,
                    'On/Off': labels_on_off})
df_on_off.to_csv(f'umbral_{joint}_ciclos_{lado}_S0{j + 1}.csv', index=False)

# Etiquetamos la señal manualmente para separar la fase de apoyo y balanceo
groundtruth = plt.figure(figsize=(12, 8), dpi = 80)

for i, (slot, ciclo) in enumerate(zip(tiempo_segmentado, señal_segmentada)):
    labels_fases = [0] * len(ciclo)
    plot(slot, ciclo)
    plt.title(f'Ciclo de marcha {i + 1}')
    plt.xlabel('tiempo (s)')
    plt.ylabel(f'Ángulo flexo-extensión {joint} ($^\circ$)')
    plt.grid(True)
    plt.show(block = False)
    eventos = plt.ginput(n = fases * pasos, timeout = 0)
    eventos = [(slot[0], ciclo[0])] + eventos + [(slot[-1], ciclo[-1])]
    eventos = sorted(eventos, key = lambda t: t[0])
    groundtruth.clear(True)

    # Extraemos los índices de transición de una fase a otra y creamos una señal booleana
        que lo represente,
    # indicando con 0 la fase de balanceo y 1 la fase de apoyo.
    for idx in range(0, len(eventos) - 1):
        # Índices de inicio y fin de ciclo
        idx1 = int(((eventos[idx][0] - slot[0])/(slot[-1] - slot[0])) * len(ciclo))
        idx2 = int(((eventos[idx + 1][0] - slot[0])/(slot[-1] - slot[0])) * len(ciclo))

        # Guardamos los índices para más adelante
        indices_f.extend([(idx1, idx2)])

        # Si es apoyo (par), ponemos 1
        if idx % 2 == 0:
            labels_fases[idx1: idx2] = [1] * (idx2 - idx1)

    indices_fases.append(indices_f)
    ciclos_new = pd.DataFrame({'Tiempo (s)': slot,
                           'Ángulo ($^\circ$)': ciclo,
                           'Labels': labels_fases})
    df_ciclos = pd.concat([df_ciclos, ciclos_new], axis = 1) # Concatenamos en horizontal

df_ciclos.to_csv(f'labels_{joint}_fases_{lado}_S0{j + 1}.csv', index = False) # No
    guardamos los indices del df
```

```
    plt.close(groundtruth)

    # Visualizamos las etiquetas guardadas
    fig8, axs = plt.subplots(figsize=(12, 8), dpi = 80)
    axs2 = axs.twinx()
    axs.plot(tiempo_sin_segmentar, señal_sin_segmentar, label = 'Ángulo ($^\circ$)')
    axs2.plot(tiempo_sin_segmentar, labels_on_off, 'r', label = 'Etiquetas')
    axs.legend(loc = 'upper left', fontsize = 10)
    axs2.legend(loc = 'upper right', fontsize = 10)
    subplots_adjust(hspace = 0.3)
    fig8.supxlabel('tiempo (s)', y = + 0.04)
    fig8.supylabel('Ángulo Rodilla ($^\circ$)', x = + 0.07)
    fig8.suptitle(f'Etiquetas de los periodos de marcha ({lado}) para {joint}', y = 0.95, x =
        0.508)

    fig9, axs = plt.subplots(4, figsize=(12, 8), dpi = 80)
    for i, (slot, ciclo) in enumerate(zip(tiempo_segmentado, señal_segmentada)):
        slot = df_ciclos.iloc[:, 3 * i]
        ciclo = df_ciclos.iloc[:, 3 * i + 1]
        labels = df_ciclos.iloc[:, 3 * i + 2]
        axs2 = axs[i].twinx()
        axs[i].plot(slot, ciclo, label = 'Ángulo ($^\circ$)')
        axs2.plot(slot, labels, 'r', label = 'Etiqueta')
        axs[i].set_title(f'Periodo de marcha {i + 1}')
        axs[i].legend(loc = 'upper right', fontsize = 10)
        axs2.legend(loc = 'upper left', fontsize = 10)
    subplots_adjust(hspace = 0.3)
    fig9.supxlabel('tiempo (s)', y = +0.04)
    fig9.supylabel('Ángulo Rodilla ($^\circ$)', x = + 0.08)
    fig9.suptitle('Fases de los ciclos de marcha', y = 0.95, x = 0.508)

    return fig8, fig9, indices_on_off, indices_fases

#_____________________________________________________________________________________________________
def copy_label(señal_sin_segmentar: list, jointfrom: str, lado:str, j: int):
    # Nos movemos al directorio con los datos
    os.chdir(r'C:\Users\marti\OneDrive\Documentos\23-24\TFM\kk2\Export 2\CSV')

    # Leemos los datos de referencia
    datos_ciclo = pd.read_csv(r'.\umbral_' + jointfrom + '_ciclos_' + lado + '_S0' + str(j + 1)
        + '.csv')
    datos = pd.read_csv(r'.\labels_'+ jointfrom +'_fases_' + lado + '_S0' + str(j + 1) + '.csv')

    # Copiamos los vectores de tiempo
    tiempo_ciclo = datos_ciclo.iloc[:, 0]
    tiempo_segmentado = [list(datos.iloc[datos.iloc[:,i].first_valid_index() :
        datos.iloc[:,i].last_valid_index(), i]) for i in range(0, 12, 3)]

    # Breakpoints de los ciclos de marcha de la rodilla
    t_ciclos = [(int(datos.iloc[datos.iloc[:, 3 * i].first_valid_index(), 3 * i] * 100),
              int(datos.iloc[datos.iloc[:, 3 * i].last_valid_index(), 3 * i] * 100)) for i in
                  range(0, 4)]
    t_ciclos = [element for innerList in t_ciclos for element in innerList] # Aplanamos la lista

    # Segmentamos la señal de la cadera por los mismos puntos
    señal_segmentada = [señal_sin_segmentar[t_ciclos[i] : t_ciclos[i + 1]] for i in range(0,
        len(t_ciclos) - 1, 2)]

    # Copiar clasificación de las fases de la rodilla a la cadera
    clas_ciclo = datos_ciclo.iloc[:, 2]
```

```
    clas = [list(datos.iloc[datos.iloc[:,i].first_valid_index() :
        datos.iloc[:,i].last_valid_index(), i]) for i in range(2, 12, 3)]

    # Guardamos el periodo de marcha etiquetado de la cadera en .csv
    lmin = min(len(tiempo_ciclo), len(señal_sin_segmentar), len(clas_ciclo))
    copia_ciclo = pd.DataFrame({'Tiempo (s)': tiempo_ciclo[:lmin],
                            'Ángulo ($^\circ$)': señal_sin_segmentar[:lmin],
                            'On/Off': clas_ciclo[:lmin]})
    copia_ciclo.to_csv(f'umbral_Cadera_ciclos_{lado}_SO{j + 1}.csv', index = False)

    fig, axs = subplots(1, figsize=(12, 8), dpi = 80)
    axs2 = axs.twinx()
    axs.plot(tiempo_ciclo[:lmin], señal_sin_segmentar[:lmin], label = 'Ángulo')
    axs2.plot(tiempo_ciclo[:lmin], clas_ciclo[:lmin], 'r', label = 'Etiquetas')
    axs.legend(loc = 'upper left', fontsize = 10)
    axs2.legend(loc = 'upper right', fontsize = 10)
    subplots_adjust(hspace = 0.3)
    fig.supxlabel('tiempo (s)', y = +0.04)
    fig.supylabel('Ángulo Cadera ($^\circ$)', x = + 0.07)
    fig.suptitle(f'Etiquetas de los periodos de marcha ({lado}) para Cadera', y = 0.95, x =
        0.508)

    # Creamos un DF con estos valores para guardarlo en .csv
    copia_cadera = pd.DataFrame([])
    fig2, ax = subplots(4, figsize=(12, 8), dpi = 80)

    for i in range(0,4):
        lmin = min(len(tiempo_segmentado[i]), len(señal_segmentada[i]), len(clas[i]))
        new_copia = pd.DataFrame({'Tiempo (s)': tiempo_segmentado[i][:lmin],
                                'Ángulo ($^\circ$)': señal_segmentada[i][:lmin],
                                'Labels': clas[i][:lmin]})
        copia_cadera = pd.concat([copia_cadera, new_copia], axis = 1)
        ax2 = ax[i].twinx()
        ax[i].plot(tiempo_segmentado[i][:lmin], señal_segmentada[i][:lmin], label = 'Ángulo')
        ax2.plot(tiempo_segmentado[i][:lmin], clas[i][:lmin], 'r', label = 'Etiqueta')
        ax[i].set_title(f'Periodo de marcha {i + 1}')
        ax2.legend(loc = 'upper right', fontsize = 10)
        ax[i].legend(loc = 'upper left', fontsize = 10)

    subplots_adjust(hspace = 0.3)
    fig2.supxlabel('tiempo (s)', y = +0.04)
    fig2.supylabel('Ángulo Cadera ($^\circ$)', x = + 0.08)
    fig2.suptitle('Fases de los ciclos de marcha', y = 0.95, x = 0.508)

    copia_cadera.to_csv(f'labels_Cadera_fases_{lado}_SO{j + 1}.csv', index = False)

    return fig, fig2
```

8.1.5 Porcentaje.py

```
# Porcentaje.py

import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
from matplotlib.pyplot import subplots
```

```
def phases_time_percentage(j: int, plots = True):
    # Nos movemos a la carpeta con los datos
    os.chdir(r'C:\Users\marti\OneDrive\Documentos\23-24\TFM\kk2\Export 2\CSV')

    # Inicializamos las variable a emplear
    Nmarchas = 4
    Y, X = [], []
    pct_ciclo, pct_der = [], []
    Lado = ['Derecha', 'Izquierda']
    Joint = ['Rodilla', 'Cadera']
    PASOS = []
    pct_PASOS = []
    pct_PASOS_der = []
    muestras = []

    # Para rodilla y cadera
    for f in range(len(Joint)):
        joint = Joint[f]
        PAS = []
        pct_paso = []
        pct_paso_der = []

        # Para las dos pruebas caminando
        for lado in Lado:
            # Leemos los datos y los guardamos en sus respectivas variables
            datos = pd.read_csv(r'.\labels_'+ joint +'_fases_' + lado + '_S0' + str(j + 1) +
                '.csv')
            señal_real, t_datos, t_real = [], [], []
            señal_real = [list(datos.iloc[datos.iloc[:, 3 * i + 1].first_valid_index():
                datos.iloc[:, 3 * i + 1].last_valid_index(), 3 * i + 1])
                      for i in range(0, Nmarchas)]
            t_datos = [list(datos.iloc[datos.iloc[:, 3 * i].first_valid_index(): datos.iloc[:, 3
                * i].last_valid_index(), 3 * i])
                  for i in range(0, Nmarchas)]
            t_real = [list(np.array(t_datos[i]) - t_datos[i][0]) for i in range(0, Nmarchas)]
            cc = [list(datos.iloc[datos.iloc[:, 3 * i + 2].first_valid_index(): datos.iloc[:, 3 *
                i + 2].last_valid_index(), 3 * i + 2])
                  for i in range(0, Nmarchas)]
            X.append(t_real)
            Y.append(señal_real)

            # Para cada uno de los periodos de marcha
            for n in range(0, Nmarchas):
                # Calculamos las transiciones de las que sacaremos los porcentajes de fase
                transitions = [int(len(cc[n]) / (3 * t_real[n][-1]))] + [i for i in
                    range(len(cc[n]) - 1) if cc[n][i] == 0 and cc[n][i + 1] == 1]
                pct = np.zeros_like(cc[n], dtype = float)

                # Para cada una de las transiciones detectadas (pasos)
                for idx in range(1, len(transitions)):
                    # Índices de inicio y fin de cada paso
                    start = transitions[idx - 1]
                    end = transitions[idx]
                    longitud_ciclo = end - start

                    # Porcentaje de cada paso y su derivada
                    ts = 1/100
                    pct_paso.append(np.linspace(0, 2 * np.pi, longitud_ciclo))
                    pct_paso_der.append(np.diff(pct_paso[-1])/ts)
```

```
                # Porcentajes de los pasos concatenados
                pct[start : end] = pct_paso[-1]

                # Valores de cada paso
                PAS.append(señal_real[n][start : end])

                # Número de muestras de cada paso
                muestras += [longitud_ciclo]

            pct_ciclo.append(list(pct))

            # Derivada del porcentaje de todos los pasos concatenados
            dt = t_real[0][-1]/len(pct)
            der = list(np.diff(np.unwrap(pct))/dt)
            der += [der[-1]]
            pct_der.append(der)

    # Guardamos los valores
    PASOS.append(PAS)
    pct_PASOS.append(pct_paso)
    pct_PASOS_der.append(pct_paso_der)

# Gráficas
fig0, axs = subplots(4, figsize=(12, 8), dpi = 80)
handle1 = mpatches.Patch(color = 'blue', label = 'Porcentaje')
handle2 = mpatches.Patch(color = 'red', label = 'Derivada')
handle3 = mpatches.Patch(color = 'black', label = 'Ángulo')
plt.legend(loc = 'upper right', fontsize = 10, handles = [handle1, handle2, handle3])

for n in range(0, 4):
    axs2 = axs[n].twinx()
    axs[n].plot(X[0][n], pct_ciclo[n][:len(X[0][n])])
    axs[n].plot(X[0][n], pct_der[n][:len(X[0][n])], 'r-.')
    axs2.plot(X[0][n], Y[0][n], 'k.')
    axs[n].set_title(f'Periodo {n} (SO{j + 1})')

plt.subplots_adjust(hspace = 0.5)
fig0.suptitle(f'Porcentaje de marcha (Derecha)', y = 0.95, x = 0.508)
fig0.supylabel(f'Ángulo Rodilla ($^\circ$)', x = + 0.08)
fig0.supxlabel('tiempo (s)', y = +0.04)

fig1, axs = subplots(4, figsize=(12, 8), dpi = 80)
handle1 = mpatches.Patch(color = 'blue', label = 'Porcentaje')
handle2 = mpatches.Patch(color = 'red', label = 'Derivada')
handle3 = mpatches.Patch(color = 'black', label = 'Ángulo')
plt.legend(loc = 'upper right', fontsize = 10, handles = [handle1, handle2, handle3])

for n in range(0, 4):
    axs2 = axs[n].twinx()
    axs[n].plot(X[1][n], pct_ciclo[n + 4][:len(X[1][n])])
    axs[n].plot(X[1][n], pct_der[n + 4][:len(X[1][n])], 'r-.')
    axs2.plot(X[1][n], Y[1][n], 'k.')
    axs[n].set_title(f'Periodo {n} (SO{j + 1})')

plt.subplots_adjust(hspace = 0.5)
fig1.suptitle(f'Porcentaje de marcha (Izquierda)', y = 0.95, x = 0.508)
fig1.supylabel(f'Ángulo Rodilla ($^\circ$)', x = + 0.08)
fig1.supxlabel('tiempo (s)', y = +0.04)


fig2, axs = subplots(4, figsize=(12, 8), dpi = 80)
```

```
    handle1 = mpatches.Patch(color = 'blue', label = 'Porcentaje')
    handle2 = mpatches.Patch(color = 'red', label = 'Derivada')
    handle3 = mpatches.Patch(color = 'black', label = 'Ángulo')
    plt.legend(loc = 'upper right', fontsize = 10, handles = [handle1, handle2, handle3])

    for n in range(0, 4):
        axs2 = axs[n].twinx()
        axs[n].plot(X[2][n], pct_ciclo[n][:len(X[2][n])])
        axs[n].plot(X[2][n], pct_der[n][:len(X[2][n])], 'r-.')
        axs2.plot(X[2][n], Y[2][n], 'k.')
        axs[n].set_title(f'Periodo {n} (SO{j + 1})')

    plt.subplots_adjust(hspace = 0.5)
    fig2.suptitle(f'Porcentaje de marcha (Derecha)', y = 0.95, x = 0.508)
    fig2.supylabel(f'Ángulo Cadera ($^\circ$)', x = + 0.08)
    fig2.supxlabel('tiempo (s)', y = +0.04)

    fig3, axs = subplots(4, figsize=(12, 8), dpi = 80)
    handle1 = mpatches.Patch(color = 'blue', label = 'Porcentaje')
    handle2 = mpatches.Patch(color = 'red', label = 'Derivada')
    handle3 = mpatches.Patch(color = 'black', label = 'Ángulo')
    plt.legend(loc = 'upper right', fontsize = 10, handles = [handle1, handle2, handle3])

    for n in range(0, 4):
        axs2 = axs[n].twinx()
        axs[n].plot(X[3][n], pct_ciclo[n + 4][:len(X[3][n])])
        axs[n].plot(X[3][n], pct_der[n + 4][:len(X[3][n])], 'r-.')
        axs2.plot(X[3][n], Y[3][n], 'k.')
        axs[n].set_title(f'Periodo {n} (SO{j + 1})')

    plt.subplots_adjust(hspace = 0.5)
    fig3.suptitle(f'Porcentaje de marcha (Izquierda)', y = 0.95, x = 0.508)
    fig3.supylabel(f'Ángulo Cadera ($^\circ$)', x = + 0.08)
    fig3.supxlabel('tiempo (s)', y = +0.04)

    if plots is not True:
        plt.close(fig0)
        plt.close(fig1)
        plt.close(fig2)
        plt.close(fig3)

    # pct_ciclo es una lista donde [0-4] son los % de la rodilla derecha y [4-8] de la
        izquierda. De [8-16] lo mismo para cadera.
    # pct_der parecido pero para la derivada
    # fig0 es la derecha, fig1 la izquierda
    # PASOS tiene 2 listas [0] con todos los pasos de la rodilla y [1] la cadera. El primer 50%
        es derecha y el otro 50% izquierda
    # muestras es 1 lista con la longitud de todos los pasos. Los primeros 32 son de la rodilla
        y los 16 primeros de estos de la derecha.

    return fig0, fig1, fig2, fig3, pct_ciclo, pct_der, pct_PASOS, pct_PASOS_der, PASOS, muestras

def Ciclograma(PASOS, muestras, j: int):
    # Representamos el ángulo de rodilla en el tiempo frente al ángulo de
    # la cadera en el tiempo de todos los pasos y del promedio
    rodilla = PASOS[0]
    cadera = PASOS[1]
    fig1 = plt.figure()

    for _, (paso_r, paso_c) in enumerate(zip(rodilla[:len(rodilla)//2],
        cadera[:len(cadera)//2])):
```

```
        add = abs(max(muestras) - len(paso_r))
        paso_r[:0] = [paso_r[0]] * add
        add = abs(max(muestras) - len(paso_c))
        paso_c[:0] = [paso_c[0]] * add
        plt.scatter(paso_c, paso_r, color = 'lightgray', alpha = 0.5)

    for _, (paso_r, paso_c) in enumerate(zip(rodilla[len(rodilla)//2:],
         cadera[len(cadera)//2:])):
        add = abs(max(muestras) - len(paso_r))
        paso_r[:0] = [paso_r[0]] * add
        add = abs(max(muestras) - len(paso_c))
        paso_c[:0] = [paso_c[0]] * add
        plt.scatter(paso_c, paso_r, color = 'lightblue', alpha = 0.5)

    avg_rodilla = np.mean(rodilla, axis = 0)
    avg_cadera = np.mean(cadera, axis = 0)
    plt.plot(avg_cadera, avg_rodilla, 'k-')
    plt.xlabel('Ángulo Cadera ($^\circ$)')
    plt.ylabel('Ángulo Rodilla ($^\circ$)')
    handle1 = mpatches.Patch(color = 'lightgray', label = 'Derecha')
    handle2 = mpatches.Patch(color = 'lightblue', label = 'Izquierda')
    handle3 = mpatches.Patch(color = 'black', label = 'Promedio')
    plt.legend(loc = 'best', fontsize = 10, handles = [handle1, handle2, handle3])
    plt.title(f'Ciclograma SO{j + 1}')

    return fig1, avg_rodilla, avg_cadera
```

8.1.6 MatrizCovarianzaSalida.py

```
# MatrizCovarianzaSalida.py

from Ciclograma.Porcentaje import phases_time_percentage
from numpy import mean, cos, sin, pi
import numpy as np
from matplotlib.pyplot import subplots
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d

def matriz_2_covarianza_ruido_salidas (N: int, M:int, show = True):
    # Inicializamos las variables que necesitaremos
    ResAll = np.array([[],[]])
    Alpha_Fourier = [[], []]
    alpha_fourier = [[], []]
    Alpha_mean = []
    Joint = ['Rodilla', 'Cadera']
    R_i = []

    for j in range(8):
        _, _, _, _, _, _, pct_PASOS, _, PASOS, _, _, _, _ = phases_time_percentage(j, False)
        Alpha_real = []
        Theta_p = np.linspace(0, 2*pi, N)

        for t in range(len(Joint)):
            Alpha_p = []
            fig0 = plt.figure(figsize=(12, 8), dpi=100)

            # Interpolamos los valores de los ángulos de cadera y rodilla (señal) de cada paso
            # a un mismo número de muestras y hallamos el paso promedio
            for i, (Alpha, Theta) in enumerate(zip(PASOS[t], pct_PASOS[t])):
```

```
            Alpha_p.append(interp1d(Theta, Alpha, fill_value="extrapolate")(Theta_p))
            plt.plot(Theta_p, Alpha_p[i], color = 'lightgray')

        Alpha_real.append(mean(Alpha_p, axis = 0))

        if show:
            plt.plot(Theta_p, Alpha_real[t], 'k', linewidth = 3.0, label = 'Promedio
                intermedio')
            plt.xlabel('Porcentaje de marcha (rads)')
            plt.ylabel(f'Ángulo {Joint[t]} ($^\circ$)')
            plt.title(f'Curva promedio S0{j + 1}')
            plt.legend(fontsize = 10, loc = 'best')
            fig0.savefig(f'Promediado2_Señal_{Joint[t]}_S0{str(j + 1)}.png', dpi=600)
        else:
            plt.close(fig0)

    # Nos definimos una matriz con los armónicos sgún el número M deteminado
    A = np.vstack([[cos(i*Theta_p) for i in range(M)], [sin(i*Theta_p) for i in range(1,
        M)]]).T
    Alpha_aprox = []
    Fourier = []
    Res = []
    fig1, ax = subplots(2, 2, figsize=(12, 8), dpi=100)
    for t in range(len(Joint)):
        # Extraemos los coeficientes de Fourier resolviendo por mínimos cuadrados
        Fourier.append(np.linalg.lstsq(A, Alpha_real[t], rcond = None)[0])

        # Ángulos promedio de rodilla y cadera aproximados por Fourier
        Alpha_aprox.append(A @ Fourier[t])

        # Residuos de la señal real vs Fourier
        Res.append(Alpha_real[t] - Alpha_aprox[t])

        #Gráficos
        ax[t, 0].plot(Theta_p, Alpha_real[t], color = 'lightblue', label = 'Promedio
            intermedio')
        ax[t, 0].plot(Theta_p, Alpha_aprox[t], color = 'lightcoral', linestyle = 'dashed',
            label = 'Fourier intermedio')
        ax[t, 0].legend(fontsize = 6, loc = 'best')
        ax[t, 0].set_xlabel('$\Theta (rads)$')
        ax[t, 0].set_ylabel(f'Ángulo {Joint[t]} ($^\circ$)')
        ax[t, 0].set_title(f'Aproximación por Fourier ({M} componentes)')
        ax[t, 1].plot(Theta_p, Res[t], color = 'black', label = 'Residuos intermedio')
        ax[t, 1].set_xlabel('$\Theta$ (rads)')
        ax[t, 1].set_ylabel(f'Ángulo {Joint[t]} ($^\circ$)')
        ax[t, 1].set_title(f'Resíduos de la aproximación de la {Joint[t]}')
        ax[t, 1].legend(fontsize = 6, loc = 'best')
    fig1.suptitle(f'Curvas promedio S0{j + 1}')
    plt.subplots_adjust(hspace = 0.3)

    if show:
        fig1.savefig(f'Aproximación2_Fourier_S0{str(j + 1)}.png', dpi=600)

    else:
        plt.close(fig1)

    # Matriz de covarianza de salidas para todos los pasos de cada uno de los sujetos
    R_i.append(np.cov(Res))

    # Series de Fourier para todos los pasos de cada uno de los sujetos
    alpha_fourier[0].append(Alpha_aprox[0])
```

```
        alpha_fourier[1].append(Alpha_aprox[1])

        # Señal real promedio para todos los pasos de cada uno de los sujetos
        Alpha_mean.append(Alpha_real)

        # Matriz de residuos de todos los sujetos
        ResAll = np.hstack([ResAll, Res])

    # Matriz R para todos los pasos de todos los sujetos
    R = np.cov(ResAll)

    # Serie de Fourier promedio de todos los sujetos
    Alpha_Fourier[0] = np.mean(alpha_fourier[0], axis = 0)
    Alpha_Fourier[1] = np.mean(alpha_fourier[1], axis = 0)

    return R, R_i, Alpha_Fourier, alpha_fourier, Alpha_mean
```

8.1.7 MatrizCovarianzaEstados.py

```
#MatrizCovarianzaEstados.py

import numpy as np
import matplotlib.pyplot as plt

def matriz_1_covarianza_estados(show = True):
    '''
    1. El rango de la cadencia son aproximadamente 40 pasos por minuto (entre 80ppm y 120ppm)

    2. Consideramos que el error máximo es la mitad del rango.

    3. Lo estimamos en términos de pulsación (W)

    4. Como es el error máximo, estimamos la desviación típica como 1/5 del error máximo

    5. La varianza es el cuadrado de la desviación típica

    6. Calculamos para un tiempo de muestreo dado, cuanto es ese error en términos de la
        variable de
    estado X y la variable de estado Y

    7. Obtenemos la matriz de covarianzas
    '''
    # Calculamos el error máximo respecto al valor esperado (media) en pulsación, considerando
        una
    # cadencia entre de 80 y 120 pasos por minuto.
    Emaximo = 2 * np.pi * 20/60
    Dtipica = Emaximo / 5.

    # Aplicamos Montecarlo para estimar la matriz de covarianzas
    SEW = np.random.randn(10000)*Dtipica # Samples errores de W

    errores_estados = np.vstack([np.cos(SEW), np.sin(SEW), SEW])

    Q = np.cov(errores_estados)

    if show:
        # Opcionalmente, visualizamos los errores
        plt.figure()
        plt.plot(SEW)
```

```
        plt.title('Monte Carlo Samples of W Errors')
        plt.xlabel('Sample Index')
        plt.ylabel('W Error')
        plt.show()

    return Q
```

8.1.8 UnscentedKalmanFilter.py

```
# UnscentedKalmanFilter.py

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
from data_processing.calibracion import arctan2_2pi
from Ciclograma.Porcentaje import phases_time_percentage
from UnscentedKalmanFilter.MatrizCovarianzaSalida import matriz_2_covarianza_ruido_salidas
from UnscentedKalmanFilter.MatrizCovarianzaEstados import matriz_1_covarianza_estados
from UnscentedKalmanFilter.UKF import UnscentedKalmanFilter

def rmse(error):
    """Calcular el error cuadrático medio entre predicciones y objetivos"""
    return np.sqrt(np.mean(error ** 2))

def mae(error):
    """Calcular el error absoluto medio entre predicciones y objetivos."""
    return np.mean(abs(error))

def nrmse(error, rango):
    """Calcular el error absoluto medio entre predicciones y objetivos."""
    return np.sqrt(np.mean(error ** 2))/(rango)

def nees(Error, P):
    """Calcular el error de estimación normalizado al cuadrado entre predicciones y
        objetivos."""
    return np.mean(Error.T @ np.linalg.inv(P) @ Error)

def corrcoef(predictions, targets):
    """Calcular el coeficiente de correlación entre predicciones y objetivos."""
    return np.corrcoef(np.array(predictions), np.array(targets))

def logL(error_salida, covmat):
    """Calcular la probabilidad logarítmica entre predicciones y objetivos"""
    O = 2 # Dimension salidas
    Res = np.array(error_salida).reshape(O, 1)
    log_det_R = np.log(np.linalg.det(2 * np.pi * covmat))
    mahalanobis_distance = error_salida.T @ np.linalg.inv(covmat) @ error_salida
    return -0.5 * (log_det_R + mahalanobis_distance)

def inicio(sujeto: int):
    global ts
    global j
    global muestras
    global pct_ciclo
    global pct_PASOS
    global AlphaReal_mean
    global AlphaFourier_mean
    global R_global
```

```
    global R_individual
    global Q_analitica
    global Q_monte
    j = sujeto
    ts =1/100
    _, _, _, _, pct_ciclo, _, pct_PASOS, _, _, muestras = phases_time_percentage(j, False)

    # Sacamos las matrices de covarianza del ruido de las salidas para todos los sujetos y para
        cada sujeto,
    # así como las series de Fourier de cada sujeto y la serie de Fourier promedio
    R_global, R_individual, _, alpha_mean, AlphaReal_mean =
        matriz_2_covarianza_ruido_salidas(100, 15, False)
    AlphaFourier_mean = np.array(alpha_mean)[:, j, :]
    # Matriz de covarianza del ruido de los estados por deducción analítica
    Q_analitica = np.array([[0.5, 0., 0.],
                            [0., 0.5, 0.],
                            [0., 0., 0.175]])
    # Matriz de covarianza del ruido de los estados por monte carlo
    Q_monte = matriz_1_covarianza_estados(False)
    # Q_monte = np.array([[0.016165930207671042, -0.013669827513671637, -0.016166402807885526],
    #                 [-0.013669827513671637, 0.016164300574682412, 0.016173260941919454],
    #                 [-0.016166402807885526, 0.016173260941919454, 0.017604789407225364]])

def funcion_estados(estado):
    X = estado[0, 0]
    Y = estado[1, 0]
    W = estado[2, 0]

    Cos = np.cos(W * ts)
    Sen = np.sin(W * ts)

    X_hat = X * Cos - Y * Sen
    Y_hat = X * Sen + Y * Cos
    W_hat = W

    return np.array([X_hat, Y_hat, W_hat]).reshape(3,1)

def funcion_medidas(estado):
    if estado is not np.array:
        estado = np.array(estado)
        estado = estado.reshape(3,1)

    X = estado[0, 0]
    Y = estado[1, 0]
    W = estado[2, 0]

    Theta = arctan2_2pi(Y, X)

    # Calculamos la salida real en base al valor en la serie de Fourier promedio al que
        correspondería
    # dicho Theta
    Z1_hat = AlphaFourier_mean[0][int((Theta / (2 * np.pi)) * len(AlphaFourier_mean[0]))] #
        Rodilla
    Z2_hat = AlphaFourier_mean[1][int((Theta / (2 * np.pi)) * len(AlphaFourier_mean[1]))] #
        Cadera

    return np.array([Z1_hat, Z2_hat]).reshape(2,1)

#_____________________________________________________________________________________________
def UKF_ideal(show = True):
    # Para todos los sujetos
```

```
for j in range(8):
    # Inicializamos las variables necesarias
    RMSE_porcentaje = []
    RMSE_Rodilla = []
    RMSE_Cadera = []
    inicio(j)
    Nmean = np.linspace(0, np.mean(muestras) * ts, 100)
    pct_inter = []

    # Calculamos el porcentaje de fase del paso promedio para comparar después con la
        predicción
    for pct in pct_PASOS[0]:
        Npct = np.linspace(0, len(pct) * ts, len(pct))
        pct_inter.append(interp1d(Npct, pct, fill_value='extrapolate')(Nmean))

    pct_mean = np.mean(pct_inter, axis = 0)
    # Inicializamos los parámetros para el filtro
    N = 3 # estados
    O = 2 # salidas
    C = 100 # Cadencia esperada (pasos/min)
    P = np.eye(N)
    Q = Q_monte
    R = R_global
    X_0 = [1., 0., 2*np.pi*C/60]

    # Obtenemos los valores de la señal real promediada que le pasaremos al filtro como
        medidas
    Alpha_real = AlphaReal_mean[j]

    # Creamos el objeto UKF
    UKF = UnscentedKalmanFilter(N, O, funcion_estados, funcion_medidas, P, Q, R)

    # Definimos el estado inicial
    UKF.reset_state(X_0)

    # Definimos e inicializamos variables que necesitaremos
    pct_UKF = []
    angulos_UKF = [[], []]
    ajuste = []
    eigenmin = []
    residuos = [[], []]
    res_cos, res_sin, res_omega, res_estados = [], [], [], []
    rmse_cos, rmse_sin, rmse_omega, rmse_estados = [], [], [], []
    mae_cos, mae_sin, mae_omega, mae_estados = [], [], [], []
    nrmse_cos, nrmse_sin, nrmse_omega, nrmse_estados = [], [], [], []
    nees_cos, nees_sin, nees_omega, nees_estados = [], [], [], []
    cc_estados = []
    mu_real = [np.cos(pct_mean),
               np.sin(pct_mean),
               (max(pct_mean) / max(Nmean)) * np.ones_like(pct_mean)]
    rango_cos = max(mu_real[0]) - min(mu_real[0])
    rango_sin = max(mu_real[1]) - min(mu_real[1])
    rango_omega = max(mu_real[2]) - min(mu_real[2])
    rango_estado = np.max(mu_real) - np.min(mu_real)
    MU = []
    LoL = []

    # Para cada muestra de los datos medidos
    for i, (rodilla, cadera) in enumerate(zip(Alpha_real[0], Alpha_real[1])):
        # Utilizamos la función __call__ de la clase UKF para predecir el siguiente estado
            (mu)
```

```
        # y actualizar en función de las medidas que le damos (Z1)
        Z1 = [rodilla, cadera]
        mu, P = UKF(Z1)

        # Calculamos el porcentaje de fase del ciclo de marcha en función del coseno y el seno
        pct_UKF += [arctan2_2pi(mu[1], mu[0])]
        angulos_UKF[0] += [funcion_medidas(mu)[0,0]]
        angulos_UKF[1] += [funcion_medidas(mu)[1,0]]
        MU.append(mu)

        # MÉTRICAS ESTADOS
        # Error de los estados predichos
        error_cos = mu_real[0][i] - mu[0]
        error_sin = mu_real[1][i] - mu[1]
        error_omega = mu_real[2][i] - mu[2]
        error_estados = np.sqrt(error_cos**2 + error_sin**2 + error_omega**2) # Norma

        # Residuos de los estados predichos
        res_cos += [error_cos]
        res_sin += [error_sin]
        res_omega += [error_omega]
        res_estados += [error_estados]

        # RMSE
        rmse_cos += [rmse(np.array(res_cos))]
        rmse_sin += [rmse(np.array(res_sin))]
        rmse_omega += [rmse(np.array(res_omega))]
        rmse_estados += [rmse(np.array(res_estados))]

        # MAE
        mae_cos += [mae(np.array(res_cos))]
        mae_sin += [mae(np.array(res_sin))]
        mae_omega += [mae(np.array(res_omega))]
        mae_estados += [mae(np.array(res_estados))]

        # NRMSE
        nrmse_cos += [rmse(np.array(res_cos))/rango_cos]
        nrmse_sin += [rmse(np.array(res_sin))/rango_sin]
        nrmse_omega += [rmse(np.array(res_omega))/rango_omega]
        nrmse_estados += [rmse(np.array(res_estados))/rango_estado]

        # NEES
        nees_cos += [nees(np.array([[error_cos]]), np.array([[P[0,0]]]))]
        nees_sin += [nees(np.array([[error_sin]]), np.array([[P[1,1]]]))]
        nees_omega += [nees(np.array([[error_omega]]), np.array([[P[2,2]]]))]
        nees_estados += [nees(np.array([[error_cos, error_sin, error_omega]]).T, P)]

        # MÉTRICAS SALIDAS
        # Error de las salidas predichas
        residuos[0] += [rodilla - funcion_medidas(mu)[0,0]]
        residuos[1] += [cadera - funcion_medidas(mu)[1,0]]

        # Varianza del ajuste
        ajuste += [np.trace(P)/3]
        # Mínimo valor singular
        eigenmin += [np.min(np.linalg.eigvals(P))]

        # (Log) Verosimilitud de la medida
        LoL += list(logL(funcion_medidas(mu) - np.array([[rodilla], [cadera]]), UKF._S))

    # Coeficiente de correlación de los estados y las medidas
```

```
cc_estados = [np.corrcoef(np.array(MU)[:,n], np.array(mu_real).T[:,n])[0,1] for n in
    range(len(mu_real))]
cc_salidas = [np.corrcoef(np.array(angulos_UKF).T[:,n],
    np.array(Alpha_real).T[:,n])[0,1] for n in range(len(Alpha_real))]

RMSE_porcentaje += [rmse(np.array(pct_UKF) - np.array(pct_mean))]
RMSE_Rodilla += [rmse(np.array(angulos_UKF[0]) - np.array(Alpha_real[0]))]
RMSE_Cadera += [rmse(np.array(angulos_UKF[1]) - np.array(Alpha_real[1]))]

if show:

    fig1 = plt.figure(figsize=(12, 8), dpi=80)
    plt.plot(Nmean, angulos_UKF[0], label = 'Ángulo Rodilla UKF')
    plt.plot(Nmean, Alpha_real[0], label = 'Ángulo Rodilla Promedio')
    plt.ylabel('Ángulo ($^\circ$)')
    plt.xlabel('tiempo(s)')
    plt.legend(loc = 'upper left', fontsize = 10)
    plt.title(f'Predicción UKF Rodilla Promedio S0{j + 1}')

    fig2 = plt.figure(figsize=(12, 8), dpi=80)
    plt.plot(Nmean, angulos_UKF[1], label = 'Ángulo Cadera UKF')
    plt.plot(Nmean, Alpha_real[1], label = 'Ángulo Cadera Promedio')
    plt.ylabel('Ángulo ($^\circ$)')
    plt.xlabel('tiempo(s)')
    plt.legend(loc = 'upper left', fontsize = 10)
    plt.title(f'Predicción UKF Cadera Promedio S0{j + 1}')

    fig3 = plt.figure(figsize=(12, 8), dpi=80)
    plt.plot(Nmean, pct_UKF, label = 'Porcentaje UKF')
    plt.plot(Nmean, pct_mean, label = 'Porcentaje Promedio')
    plt.xlabel('tiempo(s)')
    plt.ylabel('Porcentaje de marcha')
    plt.legend(loc = 'upper left', fontsize = 10)
    plt.title(f'Predicción UKF Promedio S0{j + 1}')

    fig4 = plt.figure(figsize=(12, 8), dpi=100)
    plt.plot(Nmean, ajuste)
    plt.ylabel('Traza de P')
    plt.xlabel('tiempo (s)')
    plt.title(f'Predicción UKF Promedio S0{j + 1}')

    fig5 = plt.figure(figsize=(12, 8), dpi=100)
    plt.plot(Nmean, eigenmin)
    plt.ylabel('Mínimo valor singular')
    plt.xlabel('tiempo (s)')
    plt.title(f'Mínimo Valor Singular UKF Promedio S0{j + 1}')

    fig6 = plt.figure(figsize=(12, 8), dpi=100)
    plt.plot(Nmean, residuos[0])
    plt.plot(Nmean, residuos[1])
    plt.legend(['Rodilla', 'Cadera'])
    plt.ylabel('Residuos de las medidas')
    plt.xlabel('tiempo (s)')
    plt.title(f'Residuos Predicción Medidas UKF Promedio S0{j + 1}')

    fig7 = plt.figure(figsize=(12, 8), dpi=100)
    plt.plot(Nmean, res_cos)
    plt.plot(Nmean, res_sin)
    plt.plot(Nmean, res_omega)
    plt.plot(Nmean, res_estados)
    plt.legend(['Cos', 'Sin', 'Omega', 'Total'])
```

```
plt.ylabel('Real - Predicho')
plt.xlabel('tiempo (s)')
plt.title(f'Residuos Predicción Estados UKF Promedio SO{j + 1}')

fig8 = plt.figure(figsize=(12, 8), dpi=100)
plt.plot(Nmean, rmse_cos)
plt.plot(Nmean, rmse_sin)
plt.plot(Nmean, rmse_omega)
plt.plot(Nmean, rmse_estados)
plt.legend(['Cos', 'Sin', 'Omega', 'Total'])
plt.ylabel('RMSE')
plt.xlabel('tiempo (s)')
plt.title(f'RMSE Predicción Estados UKF Promedio SO{j + 1}')

fig9 = plt.figure(figsize=(12, 8), dpi=100)
plt.plot(Nmean, mae_cos)
plt.plot(Nmean, mae_sin)
plt.plot(Nmean, mae_omega)
plt.plot(Nmean, mae_estados)
plt.legend(['Cos', 'Sin', 'Omega', 'Total'])
plt.ylabel('MAE')
plt.xlabel('tiempo (s)')
plt.title(f'MAE Predicción Estados UKF Promedio SO{j + 1}')

fig10 = plt.figure(figsize=(12, 8), dpi=100)
plt.plot(Nmean, nrmse_cos)
plt.plot(Nmean, nrmse_sin)
plt.plot(Nmean, nrmse_omega)
plt.plot(Nmean, nrmse_estados)
plt.legend(['Cos', 'Sin', 'Omega', 'Total'])
plt.ylabel('NRMSE')
plt.xlabel('tiempo (s)')
plt.title(f'NRMSE Predicción Estados UKF Promedio SO{j + 1}')

fig11 = plt.figure(figsize=(12, 8), dpi=100)
plt.plot(Nmean, nees_cos)
plt.plot(Nmean, nees_sin)
plt.plot(Nmean, nees_omega)
plt.plot(Nmean, nees_estados)
plt.legend(['Cos', 'Sin', 'Omega', 'Total'])
plt.ylabel('NEES')
plt.xlabel('tiempo (s)')
plt.title(f'NEES Predicción Estados UKF Promedio SO{j + 1}')

fig12 = plt.figure(figsize=(12, 8), dpi=100)
plt.plot(Nmean, LoL)
plt.ylabel('Probabilidad Log')
plt.xlabel('tiempo (s)')
plt.title(f'Log likelihood Predicción Medidas UKF Promedio SO{j + 1}')

fig1.savefig(f'(1) UKF - SO{j + 1}.png')
fig2.savefig(f'(2) UKF - SO{j + 1}.png')
fig3.savefig(f'(3) UKF - SO{j + 1}.png')
fig4.savefig(f'(4) UKF - SO{j + 1}.png')
fig5.savefig(f'(5) UKF - SO{j + 1}.png')
fig6.savefig(f'(6) UKF - SO{j + 1}.png')
fig7.savefig(f'(7) UKF - SO{j + 1}.png')
fig8.savefig(f'(8) UKF - SO{j + 1}.png')
fig9.savefig(f'(9) UKF - SO{j + 1}.png')
fig10.savefig(f'(10) UKF - SO{j + 1}.png')
fig11.savefig(f'(11) UKF - SO{j + 1}.png')
```

```
            fig12.savefig(f'(12) UKF - S0{j + 1}.png')

        print(f'CC estados:\n{cc_estados}')
        print(f'CC salidas:\n{cc_salidas}')

        plt.close('all')


        print('_____________ RMSE MEDIA (UKF PROMEDIO) ____________')
        print(f'Porcentaje = {np.mean(RMSE_porcentaje)}')
        print(f'Rodilla = {np.mean(RMSE_Rodilla)}')
        print(f'Cadera = {np.mean(RMSE_Cadera)}')

        print('____________ RMSE MEDIANA (UKF PROMEDIO) ___________')
        print(f'Porcentaje = {np.median(RMSE_porcentaje)}')
        print(f'Rodilla = {np.median(RMSE_Rodilla)}')
        print(f'Cadera = {np.median(RMSE_Cadera)}')
    print(f'Matriz R:\n{R}')
    print(f'Matriz Q:\n{Q}')

#_____________________________________________________________________________________________________________
def UKF_realista(show = True):
    coeficientes_correlacion = pd.DataFrame([])

    # Para todos los sujetos
    for j in range(8):

        #Inicializamos las variables que necesitamos
        cc_sujeto = pd.DataFrame([])
        print(f'UKF real S0{j+1}')
        RMSE_porcentaje = []
        RMSE_Rodilla = []
        RMSE_Cadera = []
        Lado = ['Derecha', 'Izquierda']

        # Para las dos pruebas: empezando con la pierna derecha y empezando con la pierna
            izquierda
        for i in range(2):
            cc_lado = pd.DataFrame([])
            lado = Lado[i]
            inicio(j)
            print(f'Lado: {lado}')

            # Extraemos los periodos segmentados reales
            datos_rod = pd.read_csv(r'.\labels_'+ 'Rodilla' +'_fases_' + lado + '_S0' + str(j +
                1) + '.csv')
            datos_cad = pd.read_csv(r'.\labels_'+ 'Cadera' +'_fases_' + lado + '_S0' + str(j + 1)
                + '.csv')
            rodilla_real = [list(datos_rod.iloc[datos_rod.iloc[:, 3 * i + 1].first_valid_index():
                datos_rod.iloc[:, 3 * i + 1].last_valid_index(), 3 * i + 1])
                                for i in range(0, 4)]
            cadera_real = [list(datos_cad.iloc[datos_cad.iloc[:, 3 * i + 1].first_valid_index():
                datos_cad.iloc[:, 3 * i + 1].last_valid_index(), 3 * i + 1])
                                for i in range(0, 4)]

            fig13, ax13 = plt.subplots(2, 2, figsize=(12, 8), dpi=80)
            fig14, ax14 = plt.subplots(2, 2, figsize=(12, 8), dpi=80)
            fig15, ax15 = plt.subplots(2, 2, figsize=(12, 8), dpi=80)
            fig16, ax16 = plt.subplots(2, 2, figsize=(12, 8), dpi=80)
            fig17, ax17 = plt.subplots(2, 2, figsize=(12, 8), dpi=80)
            fig18, ax18 = plt.subplots(2, 2, figsize=(12, 8), dpi=80)
```

```
fig19, ax19 = plt.subplots(2, 2, figsize=(12, 8), dpi=80)
fig20, ax20 = plt.subplots(2, 2, figsize=(12, 8), dpi=80)
fig21, ax21 = plt.subplots(2, 2, figsize=(12, 8), dpi=80)
fig22, ax22 = plt.subplots(2, 2, figsize=(12, 8), dpi=80)
fig23, ax23 = plt.subplots(2, 2, figsize=(12, 8), dpi=80)
fig24, ax24 = plt.subplots(2, 2, figsize=(12, 8), dpi=80)

for paseo in range(4):
    print(f'Periodo: {paseo}')
    lmin = min(len(rodilla_real[paseo]), len(cadera_real[paseo]))
    N = 3
    O = 2
    C = 100 # Cadencia esperada (pasos/min)
    k = 3.5
    P = np.eye(N)
    Q = Q_monte
    R = R_global*10**(k)
    X_0 = [1., 0., 2 * np.pi * C/60]

    # Obtenemos los valores de la señal real de los periodos de marcha enteros, sin
        segmentar los pasos.
    Alpha = [rodilla_real[paseo][:lmin], cadera_real[paseo][:lmin]]

    # Creamos el objeto UKF
    UKF = UnscentedKalmanFilter(N, O, funcion_estados, funcion_medidas, P, Q, R)

    # Definimos el estado inicial
    UKF.reset_state(X_0)

    # Definimos e inicializamos variables que necesitaremos
    pct_UKF = []
    angulos_UKF = [[], []]
    Nmin = np.linspace(0, lmin * ts, lmin)
    ajuste = []
    eigenmin = []
    residuos = [[], []]
    res_cos, res_sin, res_omega, res_estados = [], [], [], []
    rmse_cos, rmse_sin, rmse_omega, rmse_estados = [], [], [], []
    mae_cos, mae_sin, mae_omega, mae_estados = [], [], [], []
    nrmse_cos, nrmse_sin, nrmse_omega, nrmse_estados = [], [], [], []
    nees_cos, nees_sin, nees_omega, nees_estados = [], [], [], []
    cc_estados = [], [], [], []
    mu_real = [np.cos(pct_ciclo[paseo + 4 * i][:lmin]),
           np.sin(pct_ciclo[paseo + 4 * i][:lmin]),
           (max(pct_ciclo[paseo + 4 * i][:lmin]) / max(Nmin)) *
               np.ones_like(pct_ciclo[paseo + 4 * i][:lmin])]
    rango_cos = max(mu_real[0]) - min(mu_real[0])
    rango_sin = max(mu_real[1]) - min(mu_real[1])
    rango_omega = max(mu_real[2]) - min(mu_real[2])
    rango_estado = np.max(mu_real) - np.min(mu_real)
    MU = []
    LoL = []

    for _, (rodilla, cadera) in enumerate(zip(Alpha[0], Alpha[1])):
        # Utilizamos la función __call__ de la clase UKF para predecir el siguiente
            estado (mu)
        # y actualizar en función de las medidas que le damos (Zl)
        Zl = [rodilla, cadera]
        mu, P = UKF(Zl)
```

```
            # Calculamos el porcentaje de fase del ciclo de marcha en función del coseno y
                el seno
            pct_UKF += [arctan2_2pi(mu[1], mu[0])]
            angulos_UKF[0] += [funcion_medidas(mu)[0,0]] # Rodilla
            angulos_UKF[1] += [funcion_medidas(mu)[1,0]] # Cadera
            MU.append(mu)

            # MÉTRICAS ESTADOS
            # Error de los estados predichos
            error_cos = mu_real[0][i] - mu[0]
            error_sin = mu_real[1][i] - mu[1]
            error_omega = mu_real[2][i] - mu[2]
            error_estados = np.sqrt(error_cos**2 + error_sin**2 + error_omega**2) # Norma

            # Residuos de los estados predichos
            res_cos += [error_cos]
            res_sin += [error_sin]
            res_omega += [error_omega]
            res_estados += [error_estados]

            # RMSE
            rmse_cos += [rmse(np.array(res_cos))]
            rmse_sin += [rmse(np.array(res_sin))]
            rmse_omega += [rmse(np.array(res_omega))]
            rmse_estados += [rmse(np.array(res_estados))]

            # MAE
            mae_cos += [mae(np.array(res_cos))]
            mae_sin += [mae(np.array(res_sin))]
            mae_omega += [mae(np.array(res_omega))]
            mae_estados += [mae(np.array(res_estados))]

            # NRMSE
            nrmse_cos += [rmse(np.array(res_cos))/rango_cos]
            nrmse_sin += [rmse(np.array(res_sin))/rango_sin]
            nrmse_omega += [rmse(np.array(res_omega))/rango_omega]
            nrmse_estados += [rmse(np.array(res_estados))/rango_estado]

            # NEES
            nees_cos += [nees(np.array([[error_cos]]), np.array([[P[0,0]]]))]
            nees_sin += [nees(np.array([[error_sin]]), np.array([[P[1,1]]]))]
            nees_omega += [nees(np.array([[error_omega]]), np.array([[P[2,2]]]))]
            nees_estados += [nees(np.array([[error_cos, error_sin, error_omega]]).T, P)]

            # MÉTRICAS SALIDAS
            # Error de las salidas predichas
            residuos[0] += [rodilla - funcion_medidas(mu)[0,0]]
            residuos[1] += [cadera - funcion_medidas(mu)[1,0]]

            # Varianza del ajuste
            ajuste += [np.trace(P)/3]

            # Mínimo valor singular
            eigenmin += [np.min(np.linalg.eigvals(P))]

            # (Log) Verosimilitud de la medida
            LoL += list(logL(funcion_medidas(mu) - np.array([[rodilla], [cadera]]),
                UKF._S))

        # Coeficiente de correlación de los estados y las medidas
```

```
cc_estados = [np.corrcoef(np.array(MU)[:,n], np.array(mu_real).T[:,n])[0,1] for n
    in range(len(mu_real))]
cc_salidas = [np.corrcoef(np.array(angulos_UKF).T[:,n],
    np.array(Alpha).T[:,n])[0,1] for n in range(len(Alpha))]

cc_lado = pd.concat([cc_lado, pd.DataFrame([cc_estados[:2] + cc_salidas])], axis
    = 0)

print(f'CC estados :\n{cc_estados}')
print(f'CC salidas:\n{cc_salidas}')



RMSE_porcentaje += [rmse(np.array(pct_UKF) - np.array(pct_ciclo[paseo + 4 *
    i][:lmin]))]
RMSE_Rodilla += [rmse(np.array(angulos_UKF[0]) - np.array(Alpha[0]))]
RMSE_Cadera += [rmse(np.array(angulos_UKF[1]) - np.array(Alpha[1]))]



[row, col] = [paseo//2, paseo%2]
ax13[row, col].plot(Nmin, Alpha[0], label = 'Ángulo Rodilla')
ax13[row, col].plot(Nmin, angulos_UKF[0], label = 'Ángulo Rodilla UKF')
ax13[row, col].legend(loc = 'upper left', fontsize = 10)
ax13[row, col].set_ylabel('Ángulo ($^\circ$)')
ax13[row, col].set_xlabel('tiempo(s)')
ax13[row, col].set_title(f'Predicción UKF Rodilla Real S0{j + 1} ({lado})')
fig13.subplots_adjust(hspace = 0.3)

ax14[row, col].plot(Nmin, Alpha[1], label = 'Ángulo Cadera')
ax14[row, col].plot(Nmin, angulos_UKF[1], label = 'Ángulo Cadera UKF')
ax14[row, col].legend(loc = 'upper left', fontsize = 10)
ax14[row, col].set_ylabel('Ángulo ($^\circ$)')
ax14[row, col].set_xlabel('tiempo(s)')
ax14[row, col].set_title(f'Predicción UKF Cadera Real S0{j + 1} ({lado})')
fig14.subplots_adjust(hspace = 0.3)

ax15[row, col].plot(Nmin, pct_ciclo[paseo + 4 * i][:lmin], label = 'Porcentaje')
ax15[row, col].plot(Nmin, pct_UKF, 'k', label = 'Porcentaje UKF')
ax15[row, col].legend(loc = 'upper left', fontsize = 10)
ax15[row, col].set_xlabel('tiempo(s)')
ax15[row, col].set_ylabel('Porcentaje de marcha')
ax15[row, col].set_title(f'Predicción UKF Real S0{j + 1} ({lado})')
fig15.subplots_adjust(hspace = 0.3)

ax16[row, col].plot(Nmin, ajuste)
ax16[row, col].set_ylabel('Traza de P')
ax16[row, col].set_xlabel('tiempo (s)')
ax16[row, col].set_title(f'Predicción UKF Real S0{j + 1} ({lado})')
fig16.subplots_adjust(hspace = 0.3)

ax17[row,col].plot(Nmin, eigenmin)
ax17[row,col].set_ylabel('Mínimo valor singular')
ax17[row,col].set_xlabel('tiempo (s)')
ax17[row,col].set_title(f'Mínimo Valor Singular UKF Promedio S0{j + 1}')
fig17.subplots_adjust(hspace = 0.3)

ax18[row,col].plot(Nmin, residuos[0])
ax18[row,col].plot(Nmin, residuos[1])
ax18[row,col].legend(['Rodilla', 'Cadera'])
ax18[row,col].set_ylabel('Residuos de las medidas')
```

```
ax18[row,col].set_xlabel('tiempo (s)')
ax18[row,col].set_title(f'Residuos Predicción Medidas UKF Promedio S0{j + 1}')
fig18.subplots_adjust(hspace = 0.3)

ax19[row,col].plot(Nmin, res_cos)
ax19[row,col].plot(Nmin, res_sin)
ax19[row,col].plot(Nmin, res_omega)
ax19[row,col].plot(Nmin, res_estados)
ax19[row,col].legend(['Cos', 'Sin', 'Omega', 'Total'])
ax19[row,col].set_ylabel('Real - Predicho')
ax19[row,col].set_xlabel('tiempo (s)')
ax19[row,col].set_title(f'Residuos Predicción Estados UKF Promedio S0{j + 1}')
fig19.subplots_adjust(hspace = 0.3)

ax20[row,col].plot(Nmin, rmse_cos)
ax20[row,col].plot(Nmin, rmse_sin)
ax20[row,col].plot(Nmin, rmse_omega)
ax20[row,col].plot(Nmin, rmse_estados)
ax20[row,col].legend(['Cos', 'Sin', 'Omega', 'Total'])
ax20[row,col].set_ylabel('RMSE')
ax20[row,col].set_xlabel('tiempo (s)')
ax20[row,col].set_title(f'RMSE Predicción Estados UKF Promedio S0{j + 1}')
fig20.subplots_adjust(hspace = 0.3)

ax21[row,col].plot(Nmin, mae_cos)
ax21[row,col].plot(Nmin, mae_sin)
ax21[row,col].plot(Nmin, mae_omega)
ax21[row,col].plot(Nmin, mae_estados)
ax21[row,col].legend(['Cos', 'Sin', 'Omega', 'Total'])
ax21[row,col].set_ylabel('MAE')
ax21[row,col].set_xlabel('tiempo (s)')
ax21[row,col].set_title(f'MAE Predicción Estados UKF Promedio S0{j + 1}')
fig21.subplots_adjust(hspace = 0.3)


ax22[row,col].plot(Nmin, nrmse_cos)
ax22[row,col].plot(Nmin, nrmse_sin)
ax22[row,col].plot(Nmin, nrmse_omega)
ax22[row,col].plot(Nmin, nrmse_estados)
ax22[row,col].legend(['Cos', 'Sin', 'Omega', 'Total'])
ax22[row,col].set_ylabel('NRMSE')
ax22[row,col].set_xlabel('tiempo (s)')
ax22[row,col].set_title(f'NRMSE Predicción Estados UKF Promedio S0{j + 1}')
fig22.subplots_adjust(hspace = 0.3)

ax23[row,col].plot(Nmin, nees_cos)
ax23[row,col].plot(Nmin, nees_sin)
ax23[row,col].plot(Nmin, nees_omega)
ax23[row,col].plot(Nmin, nees_estados)
ax23[row,col].legend(['Cos', 'Sin', 'Omega', 'Total'])
ax23[row,col].set_ylabel('NEES')
ax23[row,col].set_xlabel('tiempo (s)')
ax23[row,col].set_title(f'NEES Predicción Estados UKF Promedio S0{j + 1}')
fig23.subplots_adjust(hspace = 0.3)

ax24[row,col].plot(Nmin, LoL)
ax24[row,col].set_ylabel('Probabilidad Log')
ax24[row,col].set_xlabel('tiempo (s)')
ax24[row,col].set_title(f'Log likelihood Predicción Medidas UKF Promedio S0{j +
    1}')
fig24.subplots_adjust(hspace = 0.3)
```

```
            if show:
                fig13.savefig(f'(13) UKF - SO{j + 1} ({lado}).png')
                fig14.savefig(f'(14) UKF - SO{j + 1} ({lado}).png')
                fig15.savefig(f'(15) UKF - SO{j + 1} ({lado}).png')
                fig16.savefig(f'(16) UKF - SO{j + 1} ({lado}).png')
                fig17.savefig(f'(17) UKF - SO{j + 1} ({lado}).png')
                fig18.savefig(f'(18) UKF - SO{j + 1} ({lado}).png')
                fig19.savefig(f'(19) UKF - SO{j + 1} ({lado}).png')
                fig20.savefig(f'(20) UKF - SO{j + 1} ({lado}).png')
                fig21.savefig(f'(21) UKF - SO{j + 1} ({lado}).png')
                fig22.savefig(f'(22) UKF - SO{j + 1} ({lado}).png')
                fig23.savefig(f'(23) UKF - SO{j + 1} ({lado}).png')
                fig24.savefig(f'(24) UKF - SO{j + 1} ({lado}).png')

            else:
                plt.close('all')

            cc_sujeto = pd.concat([cc_sujeto, cc_lado], axis = 1)


        print('_____________ RMSE MEDIA (UKF REAL) ____________')
        print(f'Porcentaje = {np.mean(RMSE_porcentaje)}')
        print(f'Rodilla = {np.mean(RMSE_Rodilla)}')
        print(f'Cadera = {np.mean(RMSE_Cadera)}')

        print('____________ RMSE MEDIANA (UKF REAL) ___________')
        print(f'Porcentaje = {np.median(RMSE_porcentaje)}')
        print(f'Rodilla = {np.median(RMSE_Rodilla)}')
        print(f'Cadera = {np.median(RMSE_Cadera)}')
        coeficientes_correlacion = pd.concat([coeficientes_correlacion, cc_sujeto], axis = 0)

    print(f'Matriz R:\n{R}')
    print(f'Matriz Q:\n{Q}')
    coeficientes_correlacion.to_csv('CoeficientesCorrelacion_realista.csv')
```

8.1.9 Clase UKF.py

```
'''
Implementación del filtro Unscented Kalman Filter

A partir de:
-
    https://towardsdatascience.com/the-unscented-kalman-filter-anything-ekf-can-do-i-can-do-it-better-ce7c773cf
- https://en.wikipedia.org/wiki/Kalman_filter#Unscented_Kalman_filter

Juanma Belda Julio de 2024

'''

# Librerías
import numpy as np
from scipy.linalg import sqrtm, pinv

# Funciones para estimar las matrices de covarianza en cálculos basados en puntos Sigma
def SigmaCov(W, Error):
    '''
    Matriz de covarianzas asociada a una matriz de sigmapoints.
```

```
    # Inputs

    - W : numpy array (1xN)
        Matriz de pesos asociada a los sigma points

    - Error : numpy array(Nx1)
        Matriz con los errores de la predicción ya sea estados o salidas

    # Return
        numpy array (NxN)
            Matriz de covarianzas
    '''

    N, S = Error.shape

    TheCorrs = [W[i]*(Error[:,i].reshape(N,1) @ Error[:,i].reshape(1,N)) for i in range(S)]

    return sum(TheCorrs)

def SigmaCrossCov(W, Error_A, Error_B):
    '''
    Matriz de covarianzas cruzadas de dos matrices

    # Inputs

    - W : numpy array (N)
        Matriz de pesos asociada a los sigma points

    - Error_A : numpy array(N1xN)
        Matriz con los errores de la predicción de los estados

    - Error_B : numpy array(N2xN)
        Matriz con los errores de la predicción de las salidas

    # Return
        numpy array (N1xN2)
            Matriz de covarianzas cruzadas
    '''

    S  = W.shape[0]
    N1 = Error_A.shape[0]
    N2 = Error_B.shape[0]

    TheCorrs = [W[i]*(Error_A[:,i].reshape(N1,1) @ Error_B[:,i].reshape(1,N2)) for i in
        range(S)]

    return sum(TheCorrs)


# La definición de la clase
class UnscentedKalmanFilter():

    def __init__(self, N, O, predict_state, predict_measurement, P, Q, R, slambda=1):

        self._N = N # Número de estados
        self._O = O # Dimensión de la salida

        self._f = predict_state #Función de los estados
        self._h = predict_measurement # Función de las medidas

        # TODO: Comprobar que las dimensiones son consistentes
```

```
        self._P = P # Matriz de covarianzas de la predicción
        self._Q = Q # Matriz de covarianzas del estado
        self._R = R # Matriz de covarianzas de la medida

        self._slambda = slambda

        # Nos creamos la matriz de pesos
        W = np.zeros(2*N+1)
        W[0] = slambda/(N + slambda)
        W[1:] = 1/(2*(N + slambda))

        self._W = W

        # La matriz de estados por defecto
        self.reset_state()

    def reset_state(self, X = None):

        if X is None:
            self._mu = np.zeros([self._N, 1])
        else:
            self._mu = np.array(X).reshape(self._N, 1)

        # Nos creamos los SigmaPoints
        Chi = np.hstack([self._mu,
                         self._mu + sqrtm((self._N + self._slambda)*self._P),
                         self._mu - sqrtm((self._N + self._slambda)*self._P)])

        self._Chi = Chi



    def predict(self):

        # Los sigma points trasladados en el tiempo
        gChi = np.array([self._f(X.reshape(self._N,1))[:, 0] for X in self._Chi.T]).T

        mu_p = np.sum(self._W * gChi, axis=1).reshape(self._N,1)

        # sigma_p = (W * ((gChi - mu_p))) @ (gChi - mu_p).T + Q
        P_p = SigmaCov(self._W, gChi - mu_p) + self._Q


        return mu_p, gChi, P_p


    def update(self, Zl):

        # Ponemos Zl como array columna
        Zl = np.array(Zl).reshape(self._O, 1)

        # Propagamos el estado
        mu_p, gChi, P_p = self.predict()

        Z = np.array([self._h(X.reshape(self._N,1))[:, 0] for X in gChi.T]).T
        Zg = np.sum(self._W*Z, axis=1).reshape(self._O,1) # Z gorrito

        # S = (W * ((Z - Zg))) @ (Z - Zg).T + R
        S = SigmaCov(self._W, (Z - Zg)) + self._R
```

```
        C = SigmaCrossCov(self._W, gChi - mu_p, Z - Zg)

        K = C @ pinv(S)

        # Los estados actualizados
        mu_n = mu_p + K @ (Zl - Zg)

        # las covarianzas de la estimación del estado actualizadas
        P_n = P_p - K @ S @ K.T

        self._S = S

        return mu_n, P_n

    def __call__(self, Zl):

        mu, P = self.update(Zl)

        self.reset_state(mu)
        self._P = P

        return list(mu[:,0]), P
```

8.1.10 Clase: quaternion.py

```
# quaternions.py
"""
Created on Tue Sep 8 15:59:17 2015

@author: juanma
"""

from numpy import *
from math import acos

def dot(v1, v2):
    '''Returns the dot product of two 3D vectors

    example
    =======

    >>> dot(vector3D(rand(3)),vector3D(rand(3)))



    :author: Juanma Belda (2015)'''

    val = 0.

    for a, b in zip(list(v1), list(v2)):
        val += a*b

    return val

class vector3D(object):
    '''A 3D vector management

    examples
    ========
```

```
>>> # Create a couple of vectors
>>> a = vector3D([1.,0.,0.])
>>> b = vector3D(0.,1.,0.) # Both declarations are permited
>>> # Vectorial product
>>> a * b
>>> # Dot product c = []
    for a, b in zip(self, value):
        print a + b
        c.append(a +
>>> dot(a,b)
>>> # Sum
>>> a + b
>>> # module
>>> mod(a)
>>> # Product per an scalar
>>> 3. * a

:author: Juanma Belda (2015)'''

_index = 0

def __init__(self, *args):
    if len(args) == 3:
        self._v = list(args)
    elif (len(args) == 1) & (len(args[0])==3):
        self._v = list(args[0])
    else:
        raise Exception("Incorrect length")


def __mul__(self, v2):
    x = self[1]*v2[2] - self[2]*v2[1]
    y = self[2]*v2[0] - self[0]*v2[2]
    z = self[0]*v2[1] - self[1]*v2[0]

    return vector3D(x,y,z)

def __rmul__(self, valor):
    return vector3D(valor*self[0], valor*self[1], valor*self[2])

def __add__(self, v2):
    val = [self[0]+v2[0], self[1]+v2[1], self[2]+v2[2]]

    return vector3D(val)

def __len__(self):
    return 3

def __getitem__(self, index):
    if (index >= 0) & (index <= 2):
        return self._v[index]
    else:
        raise Exception("Index out of bounds")

def __setitem__(self, index, value):
    if (index >= 0) & (index <= 2):
        self._v[index] = value
    else:
        raise Exception("Index out of bounds")
```

```
    def __repr__(self):
        #a = "(%f, %f, %f)" % tuple(self._v)

        return repr(tuple(self._v))

    def __iter__(self):
        return self

    def __next__(self):
        try:
            a = self[self._index]
        except Exception:
            self._index = 0
            raise StopIteration

        self._index += 1
        return a

class quaternion(object):
    '''Algegra of quaternions:

    :author: Juanma Belda (2015)
    :email: juanma.belda@ibv.upv.es'''

    _index = 0

    def __init__(self, *args):

        if len(args) == 4:
            self._q = list(args)
        elif (len(args) == 2) & (len(args[1])==3):
            self._q = [args[0], args[1][0], args[1][1], args[1][2]]
        elif (len(args) == 1) & (len(args[0])==4):
            self._q = list(args[0])
        else:
            raise Exception("Incorrect length")

    def _get_escalar(self):
        return self._q[0]

    def _get_vector(self):
        return vector3D(self._q[1:4])

    e = property(_get_escalar) # The escalar
    v = property(_get_vector) # The vector

    def _conj(self):
        return quaternion(self[0], -self[1], -self[2], -self[3])

    c = property(_conj) # Conjugated quaternion

    def _norm(self):
        '''Returns the norm of the quaternion'''
        return sqrt(dot(self._q, self._q))

    n = property(_norm)

    def normalize(self):
        '''Returns a normalized version of the quaternion'''
        return quaternion(*tuple(self._q)) * (1./self._norm())
```

```
def __len__(self):
    return 4

def __getitem__(self, index):
    if (index >= 0) & (index <= 3):
        return self._q[index]
    else:
        raise Exception("Index out of bounds")

def __setitem__(self, index, value):
    if (index >= 0) & (index <= 3):
        self._q[index] = value
    else:
        raise Exception("Index out of bounds")

def __add__(self, value):
    c = [self[c]+value[c] for c in range(4)]

    return quaternion(*tuple(c))



def __mul__(self, value):
    if isinstance(value, vector3D):
        value = quaternion(0,value)

    if isinstance(value, float) or isinstance(value, int):
        value = quaternion(value,0,0,0)

    e1 = self.e
    e2 = value.e

    v1 = self.v
    v2 = value.v

    escalar = e1*e2 - dot(v1,v2)
    vector = e1*v2 + e2*v1 + v1*v2

    return quaternion(escalar, vector)

def __rmul__(self, valor):
    return quaternion(valor*self[0], valor*self[1], valor*self[2], valor*self[3])


def __repr__(self):
    #a = "(%f, %f, %f)" % tuple(self._v)

    return repr(self._q[0]) + "+" + repr(tuple(self._q[1:4]))

def __iter__(self):
    return self

def __next__(self):
    try:
        a = self[self._index]
    except Exception:
        self._index = 0
        raise StopIteration
```

```
        self._index += 1
        return a
```

8.1.11 rotations.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed May 24 08:35:45 2023

Transformations from quaternions to euler angles

@author: jmbelda
"""

from numpy import arctan2, arccos, arcsin, pi, array, sqrt, zeros


rad2deg = 180/pi

def q2xyx(q, deg=True):

    if deg:
        mult = rad2deg
    else:
        mult = 1

    euler = [arctan2(2*q[1]*q[2]-2*q[0]*q[3], 2*q[1]*q[3]+2*q[0]*q[2]),
            arccos(q[1]**2 + q[0]**2 - q[3]**2 - q[2]**2),
            arctan2(2*q[1]*q[2]+2*q[0]*q[3], -2*q[1]*q[3]+2*q[0]*q[2])]

    return list(array(euler)*mult)

def q2xyz(q, deg=True):

    if deg:
        mult = rad2deg
    else:
        mult = 1

    euler = [arctan2(2*q[2]*q[3]+2*q[0]*q[1], q[3]**2-q[2]**2-q[1]**2+q[0]**2),
            -arcsin(2*q[1]*q[3]-2*q[0]*q[2]),
            arctan2(2*q[1]*q[2]+2*q[0]*q[3], q[1]**2+q[0]**2-q[3]**2-q[2]**2)]

    return list(array(euler)*mult)



def q2xzx(q, deg=True):

    if deg:
        mult = rad2deg
    else:
        mult = 1

    euler = [arctan2(2*q[1]*q[3]+2*q[0]*q[2], -2*q[1]*q[2]+2*q[0]*q[3]),
            arccos(q[1]**2 + q[0]**2 - q[3]**2 - q[2]**2),
            arctan2(2*q[1]*q[3]-2*q[0]*q[2], 2*q[1]*q[2]+2*q[0]*q[3])]

    return list(array(euler)*mult)
```

```
def q2xzy(q, deg=True):

    if deg:
        mult = rad2deg
    else:
        mult = 1

    euler = [arctan2(-2*q[2]*q[3]+2*q[0]*q[1], q[2]**2-q[3]**2-q[1]**2+q[0]**2),
            arcsin(2*q[1]*q[2]+2*q[0]*q[3]),
            arctan2(-2*q[1]*q[3]+2*q[0]*q[2], q[1]**2+q[0]**2-q[3]**2-q[2]**2)]

    return list(array(euler)*mult)

def q2yxy(q, deg=True):

    if deg:
        mult = rad2deg
    else:
        mult = 1

    euler = [arctan2(2*q[1]*q[2]+2*q[0]*q[3], -2*q[2]*q[3]+2*q[0]*q[2]),
            arccos(q[0]**2 - q[1]**2 + q[2]**2 - q[3]**2),
            arctan2(2*q[1]*q[2]-2*q[0]*q[3], 2*q[2]*q[3]+2*q[0]*q[1])]

    return list(array(euler)*mult)

def q2yxz(q, deg=True):

    if deg:
        mult = rad2deg
    else:
        mult = 1


    euler = [arctan2(-2*q[1]*q[3]+2*q[0]*q[2], q[0]**2-q[1]**2-q[2]**2+q[3]**2),
            arcsin(2*q[2]*q[3]+2*q[0]*q[1]),
            arctan2(-2*q[1]*q[2]+2*q[0]*q[3], q[0]**2-q[1]**2+q[2]**2-q[3]**2)]

    return list(array(euler)*mult)

def q2yzx(q, deg=True):

    if deg:
        mult = rad2deg
    else:
        mult = 1


    euler = [arctan2(2*q[1]*q[3]+2*q[0]*q[2], q[0]**2+q[1]**2-q[2]**2-q[3]**2),
            -arcsin(2*q[1]*q[2]-2*q[0]*q[3]),
            arctan2(2*q[2]*q[3]+2*q[0]*q[1], q[0]**2-q[1]**2+q[2]**2-q[3]**2)]

    return list(array(euler)*mult)

def q2yzy(q, deg=True):

    if deg:
        mult = rad2deg
    else:
        mult = 1
```

```
    euler = [arctan2(2*q[2]*q[3]-2*q[0]*q[1], 2*q[1]*q[2]+2*q[0]*q[3]),
            arccos(q[0]**2 - q[1]**2 + q[2]**2 - q[3]**2),
            arctan2(2*q[2]*q[3]+2*q[0]*q[1], -2*q[1]*q[2]+2*q[0]*q[3])]

    return list(array(euler)*mult)

def q2zxy(q, deg=True):

    if deg:
        mult = rad2deg
    else:
        mult = 1


    euler = [arctan2(2*q[1]*q[2]+2*q[0]*q[3], q[0]**2-q[1]**2+q[2]**2-q[3]**2),
            -arcsin(2*q[2]*q[3]-2*q[0]*q[1]),
            arctan2(2*q[1]*q[3]+2*q[0]*q[2], q[0]**2-q[1]**2-q[2]**2+q[3]**2)]

    return list(array(euler)*mult)

def q2zxz(q, deg=True):

    if deg:
        mult = rad2deg
    else:
        mult = 1


    euler = [arctan2(2*q[1]*q[3]-2*q[0]*q[2], 2*q[2]*q[3]+2*q[0]*q[1]),
            arccos(q[0]**2 - q[1]**2 - q[2]**2 + q[3]**2),
            arctan2(2*q[1]*q[3]+2*q[0]*q[2], -2*q[2]*q[3]+2*q[0]*q[1])]

    return list(array(euler)*mult)

def q2zyx(q, deg=True):

    if deg:
        mult = rad2deg
    else:
        mult = 1


    euler = [arctan2(-2*q[1]*q[2]+2*q[0]*q[3], q[0]**2+q[1]**2-q[2]**2-q[3]**2),
            arcsin(2*q[1]*q[3]+2*q[0]*q[2]),
            arctan2(-2*q[2]*q[3]+2*q[0]*q[1], q[0]**2-q[1]**2-q[2]**2+q[3]**2)]

    return list(array(euler)*mult)

def q2zyz(q, deg=True):

    if deg:
        mult = rad2deg
    else:
        mult = 1


    euler = [arctan2(2*q[2]*q[3]+2*q[0]*q[1], -2*q[1]*q[3]+2*q[0]*q[2]),
            arccos(q[0]**2 - q[1]**2 - q[2]**2 + q[3]**2),
            arctan2(2*q[2]*q[3]-2*q[0]*q[1], 2*q[1]*q[3]+2*q[0]*q[2])]
```

```
    return list(array(euler)*mult)

#%% Matrix to quaternion
# https://www.euclideanspace.com/maths/geometry/rotations/conversions/matrixToQuaternion/
def mat2quat(R):
    '''
    Convierte una matriz de rotación en un cuaternion

    Parameters
    ----------
    R : numpy array
        Matriz de rotación.

    Returns
    -------
    numpy array
        Cuaternión.

    '''

    tr = R[0,0] + R[1,1] + R[2,2]

    if tr > 0:
        S = sqrt(tr+1.0) * 2
        qw = 0.25*S
        qx = (R[2,1] - R[1,2])/S
        qy = (R[0,2] - R[2,0])/S
        qz = (R[1,0] - R[0,1])/S
    elif (R[0,0] > R[1,1]) & (R[0,0] > R[2,2]):
        S = sqrt(1.0+ R[0,0] - R[1,1] - R[2,2]) * 2
        qw = (R[2,1] - R[1,2])/S
        qx = 0.25*S
        qy = (R[0,1] + R[1,0])/S
        qz = (R[0,2] + R[2,0])/S
    elif R[1,1] > R[2,2]:
        S = sqrt(1.0 + R[1,1] - R[0,0] - R[2,2]) * 2
        qw = (R[0,2] - R[2,0])/S
        qx = (R[0,1] + R[1,0])/S
        qy = 0.25*S
        qz = (R[1,2] + R[2,1])/S
    else:
        S = sqrt(1.0 + R[2,2] - R[0,0] - R[1,1]) * 2
        qw = (R[1,0] - R[0,1])/S
        qx = (R[0,2] + R[2,0])/S
        qy = (R[1,2] + R[2,1])/S
        qz = 0.25*S

    return array([qw, qx, qy, qz])



#%% Cuaternion to matrix
def quat2mat(q):
    '''
    Convierte un cuaternión en una matriz de rotación

    (Verificar si la matriz es la invertida)

    Parameters
    ----------
```

```
    q : numpy array
        Cuaternión.

    Returns
    -------
    numpy array
        Matriz de rotación.

    '''

    R = zeros([3,3])

    R[0,0] = q[0]**2 + q[1]**2 - q[2]**2 - q[3]**2
    R[0,1] = 2*q[1]*q[2] + 2*q[3]*q[0]
    R[0,2] = 2*q[1]*q[3] - 2*q[2]*q[0]

    R[1,0] = 2*q[1]*q[2] - 2*q[3]*q[0]
    R[1,1] = q[0]**2 - q[1]**2 + q[2]**2 - q[3]**2
    R[1,2] = 2*q[2]*q[3] + 2*q[1]*q[0]

    R[2,0] = 2*q[1]*q[3] + 2*q[2]*q[0]
    R[2,1] = 2*q[2]*q[3] - 2*q[1]*q[0]
    R[2,2] = q[0]**2 - q[1]**2 - q[2]**2 + q[3]**2

    return R
```

8.2 ANEXO II: Figuras de las métricas del rendimiento del UKF realista

En esta segunda Sección de los Anexos, se muestran las figuras de los resultados de la predicción del porcentaje de fase para todos los sujetos durante las pruebas realizadas empezando a caminar con el pie derecho. Si bien son similares a los presentados en la memoria.

Además, se muestran las gráficas de las métricas del rendimiento del UFK en el contexto realista para ambas pruebas. Entre estas, el error absoluto medio, el error cuadrático medio normalizado, la (log) verosimilitud de las salidas, la traza de la matriz de correlación cruzada y el mínimo valor singular de dicha matriz.

8.2.1 Otra métricas de rendimiento de la predicción del UKF en el caso ideal

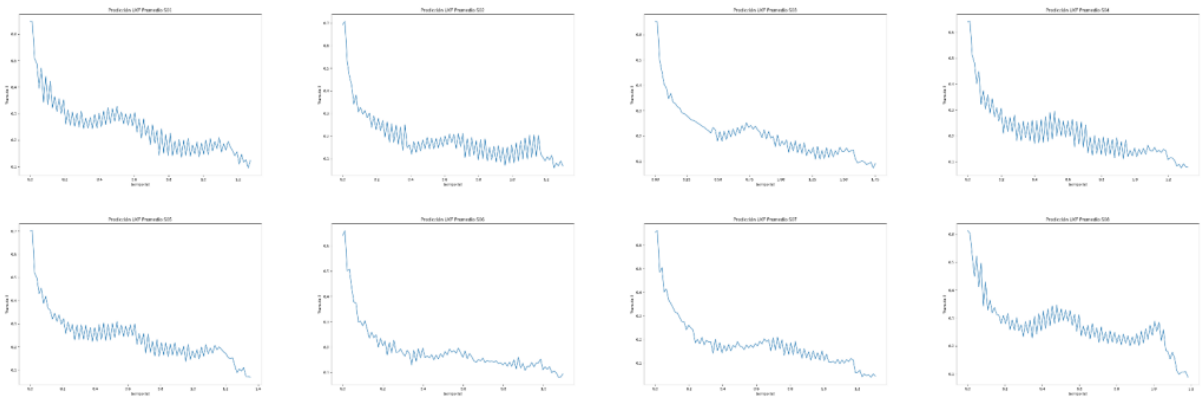


Figura 8.1: Traza de la matriz de covarianza cruzada del UKF en el escenario ideal

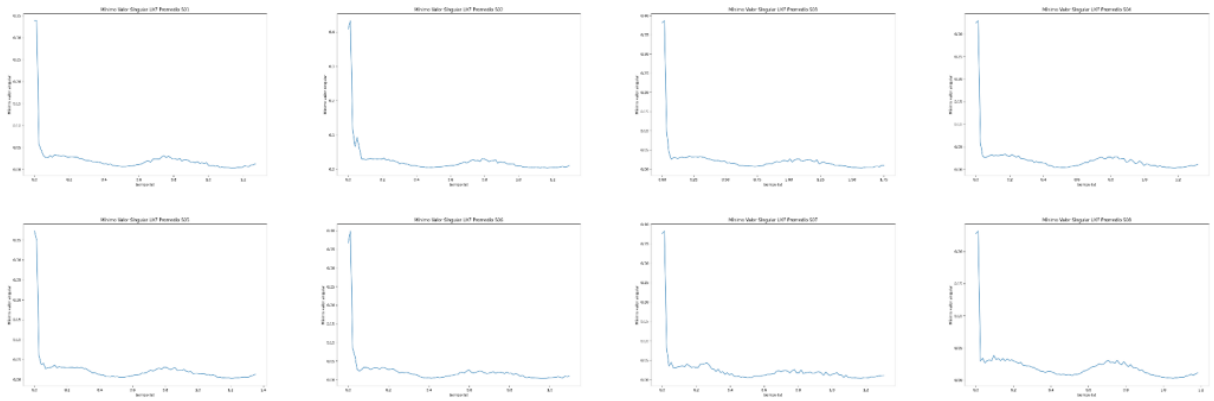


Figura 8.2: Mínimo valor singular de la matriz de covarianza cruzada del UKF en el escenario ideal

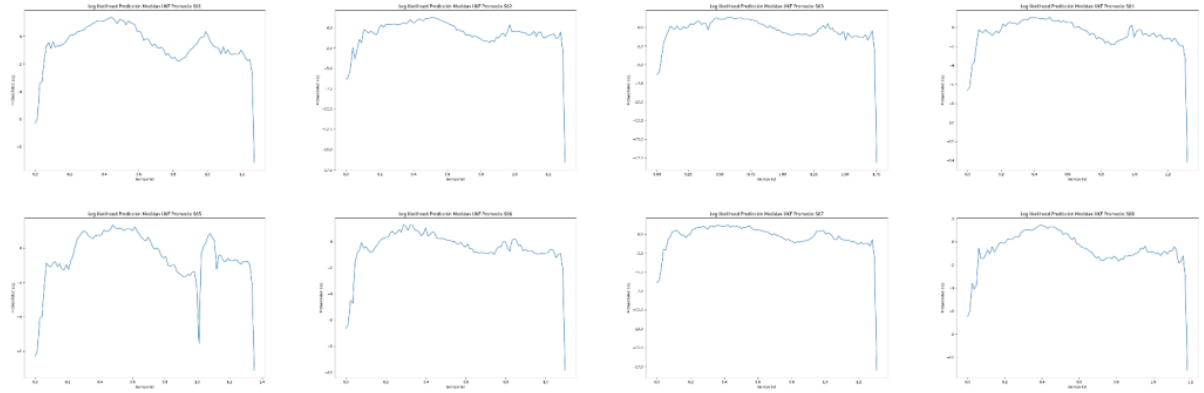


Figura 8.3: (Log) verosimilitud de las salidas obtenidas de las predicciones del UKF en el escenario ideal

8.2.2 Otras métricas de rendimiento de la predicción del UKF en el caso realista para las pruebas empezando con la pierna izquierda

8.2.3 Resultados y métricas de rendimiento completos de la predicción del UKF en el caso realista para las pruebas empezando con la pierna derecha

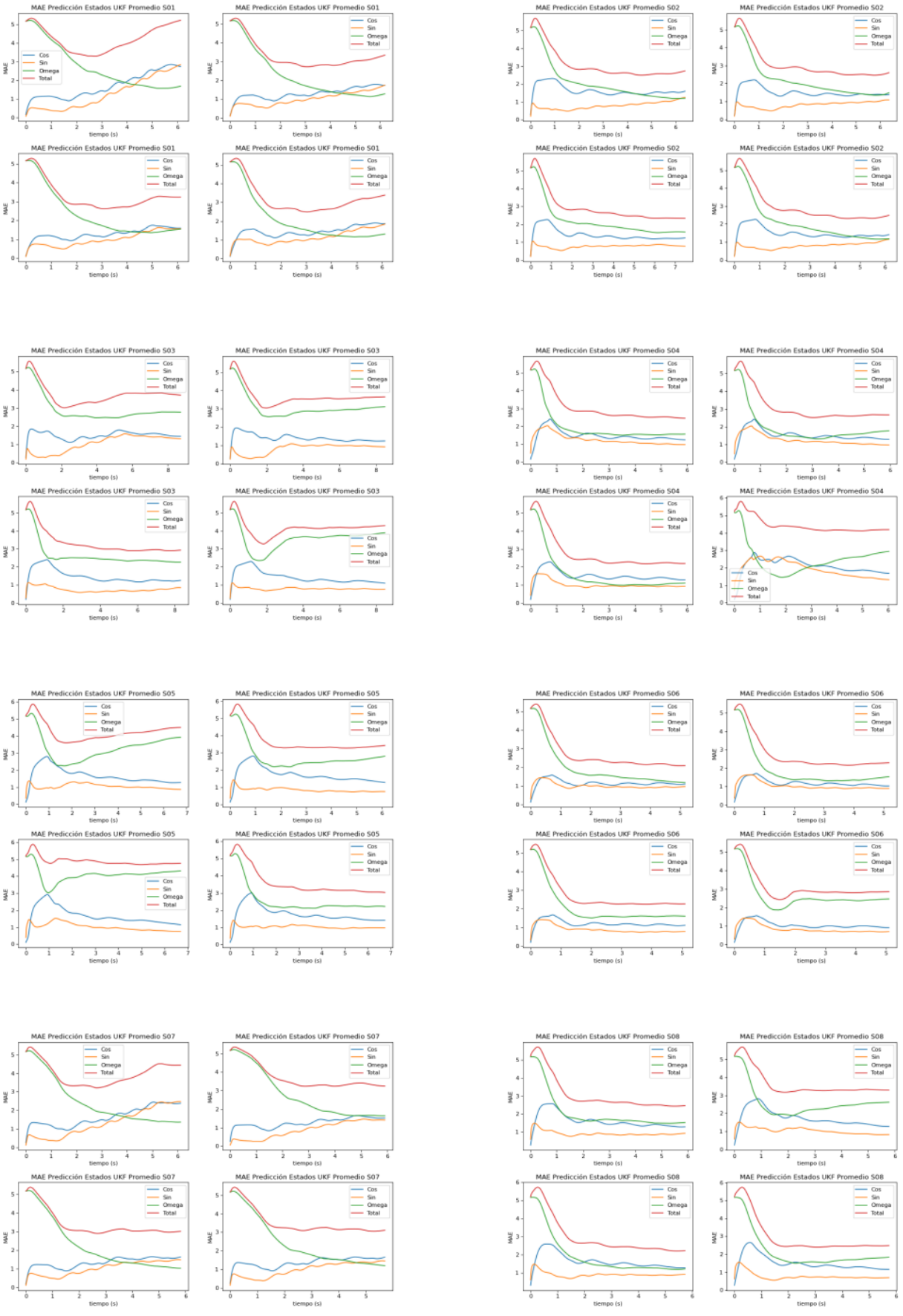


Figura 8.4: Error absoluto medio (MAE) de las predicciones del UKF en el escenario realista para las pruebas empezando con la pierna izquierda

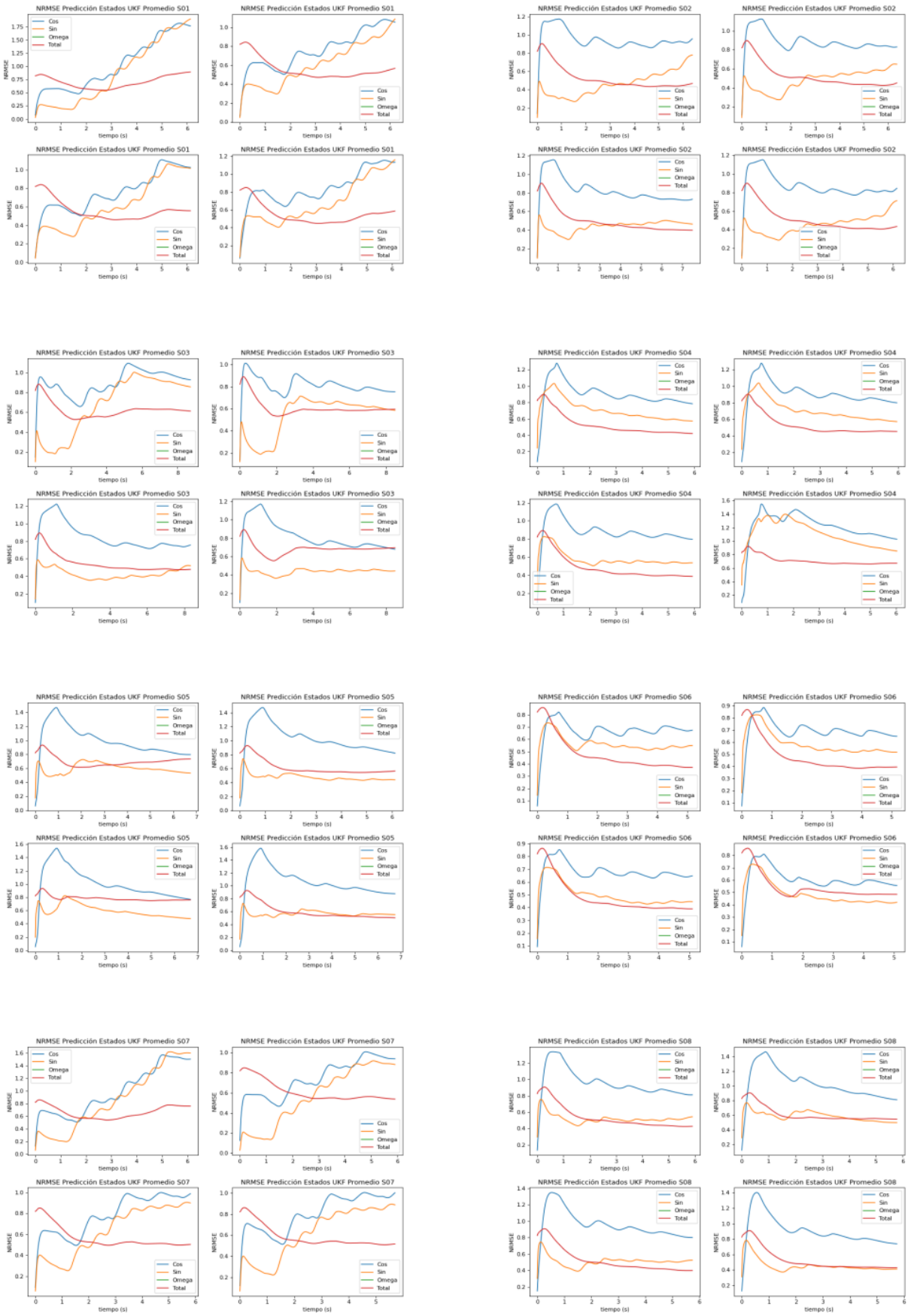


Figura 8.5: Error cuadrático medio normalizado (NRMSE) de las predicciones del UKF en el escenario realista para las pruebas empezando con la pierna izquierda

Figura 8.6: Traza de la matriz de covarianza cruzada de las predicciones del UKF en el escenario realista para las pruebas empezando con la pierna izquierda

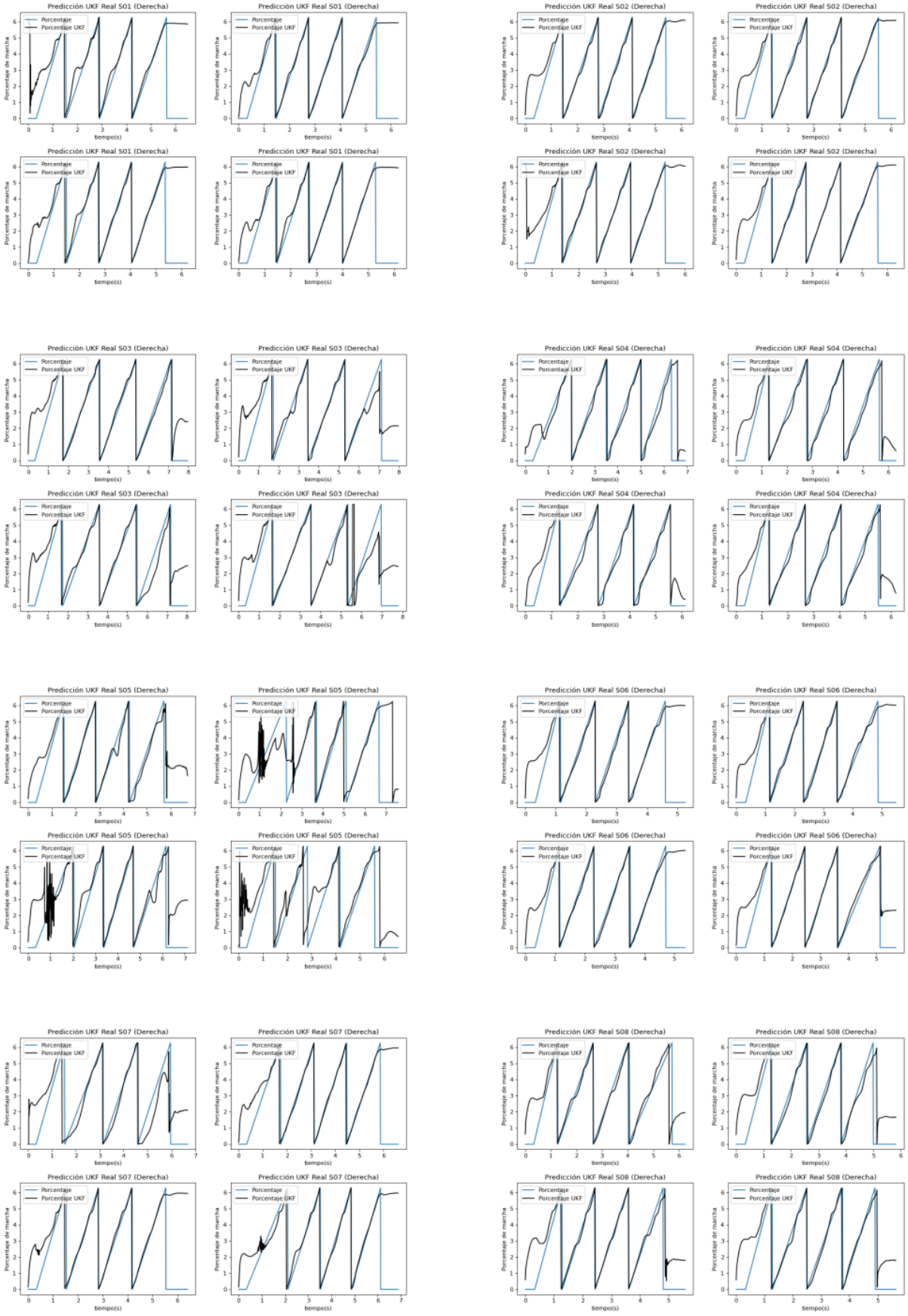


Figura 8.7: Resultados de la predicción del porcentaje de fase (linea azul) durante los periodos de marcha reales (linea naranja) de cada uno de los sujetos durante las pruebas comenzando a caminar con la pierna derecha.

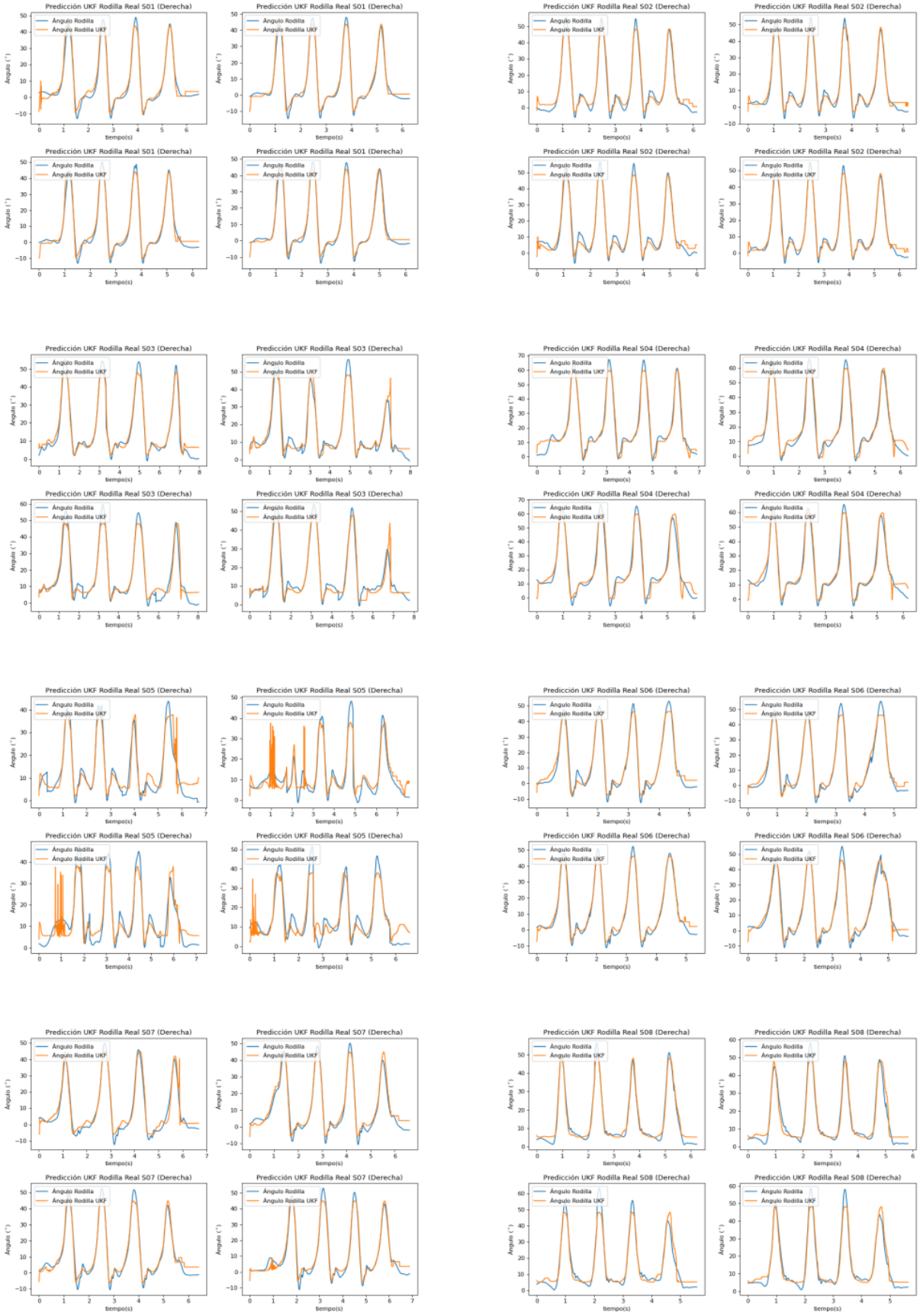


Figura 8.8: Resultados de los ángulos de flexo-extensión de la rodilla (linea azul) durante los periodos de marcha reales (linea naranja) de cada uno de los sujetos durante las pruebas comenzando a caminar con la pierna derecha.

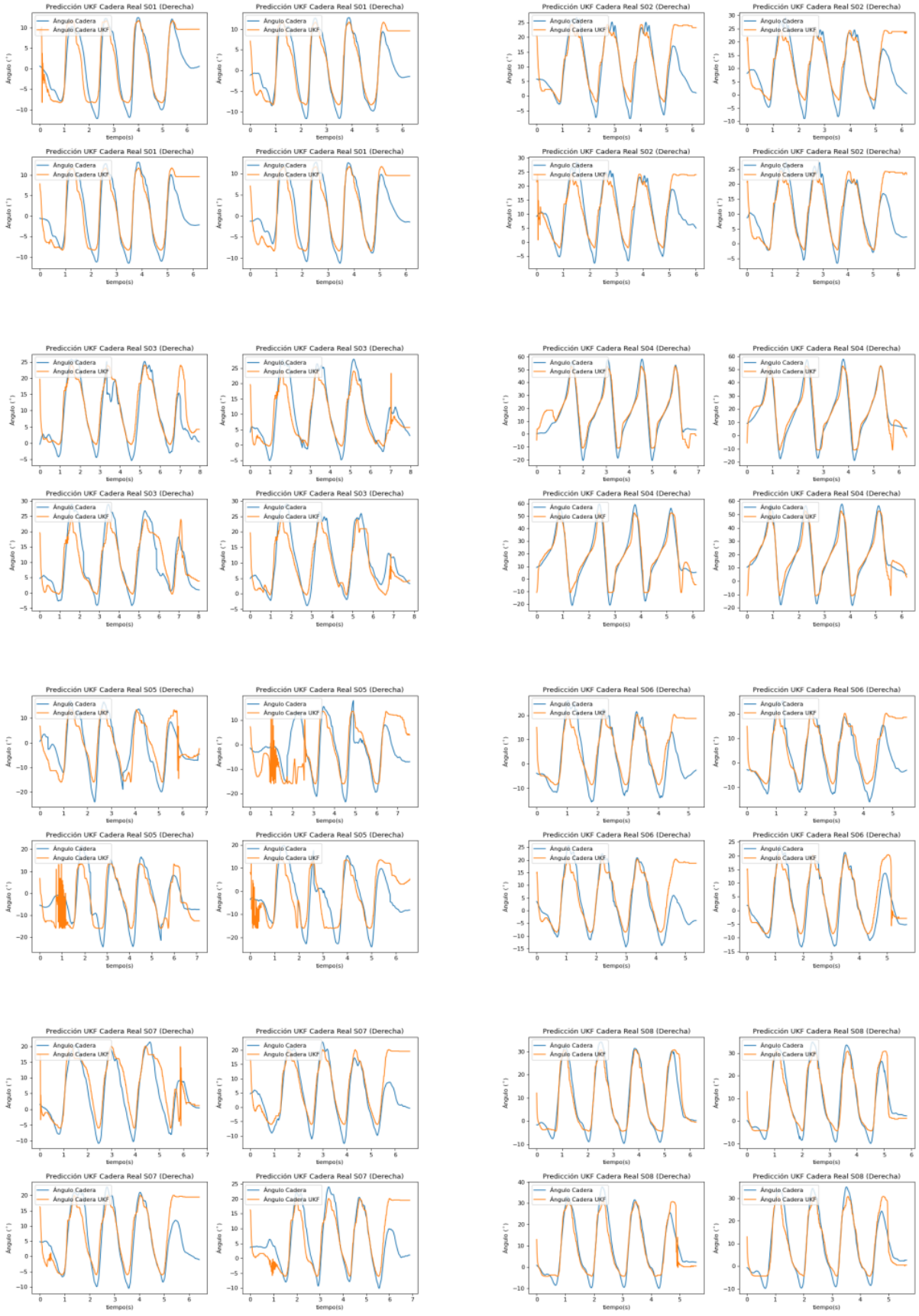


Figura 8.9: Resultados de los ángulos de flexo-extensión de la cadera (linea azul) durante los periodos de marcha reales (linea naranja) de cada uno de los sujetos durante las pruebas comenzando a caminar con la pierna derecha.

Figura 8.10: Residuos de la predicción del ángulo de rodilla (linea azul) y del ángulo de cadera (linea naranja) durante los periodos de marcha reales de cada uno de los sujetos durante las pruebas comenzando a caminar con la pierna derecha

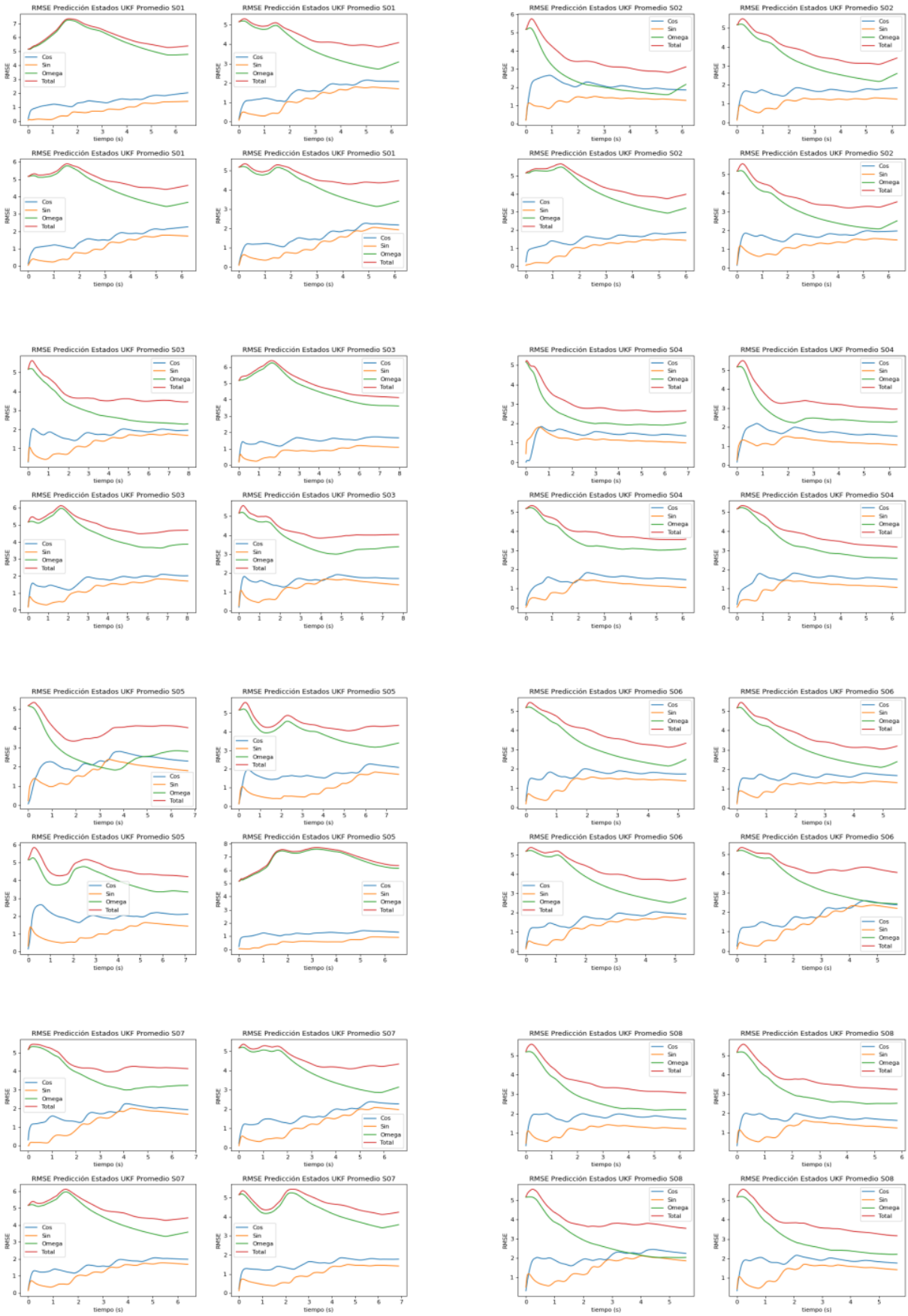


Figura 8.11: Error cuadrático medio (RMSE) de la predicción del coseno del porcentaje de fase (linea azul), del seno del porcentaje de fase (linea naranja) y de la velocidad angular del ciclo (linea verde) durante los periodos de marcha reales de cada uno de los sujetos durante las pruebas comenzando a caminar con la pierna derecha

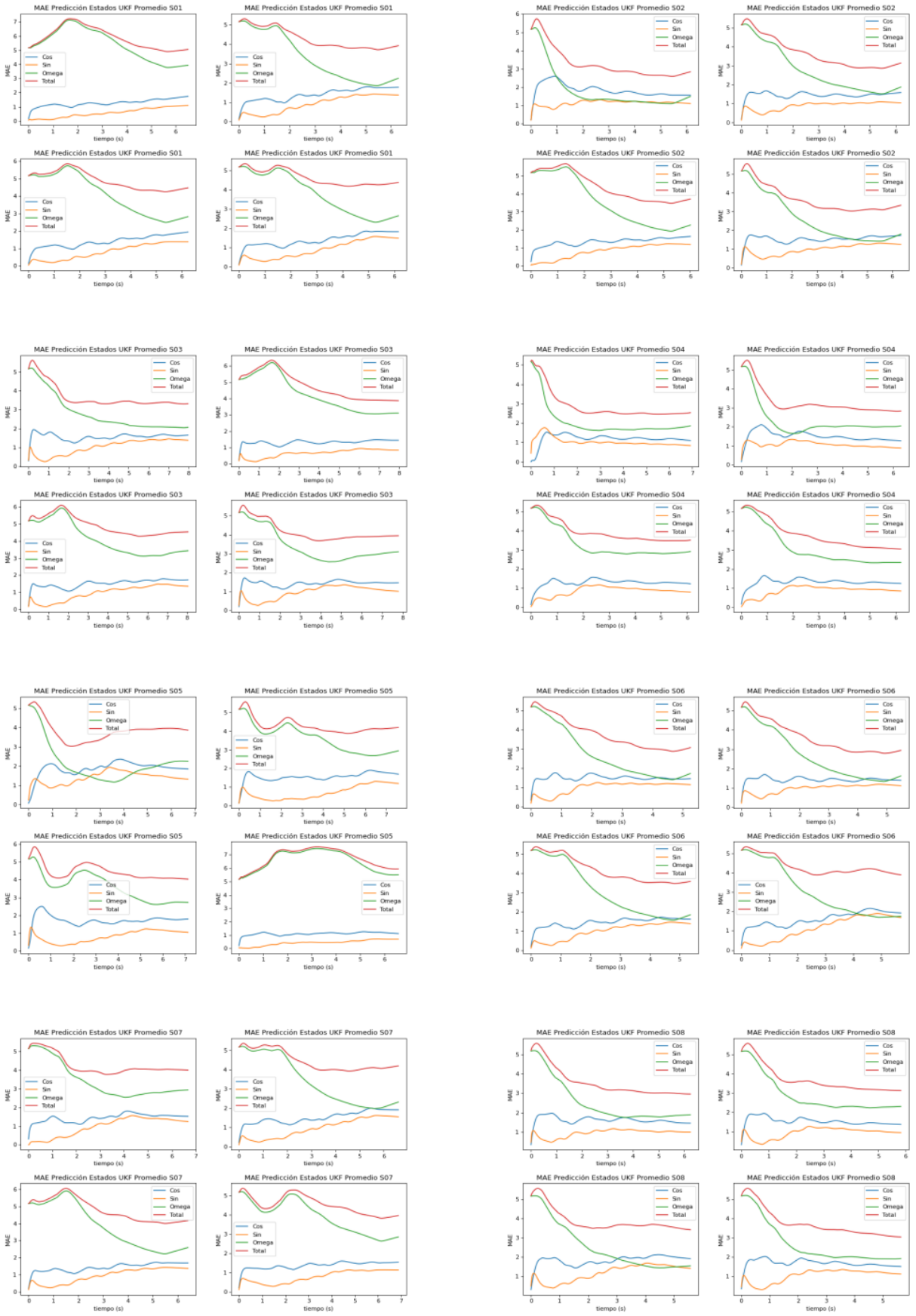


Figura 8.12: Error absoluto medio (MAE) de las predicciones del UKF en el escenario realista para las pruebas empezando con la pierna derecha

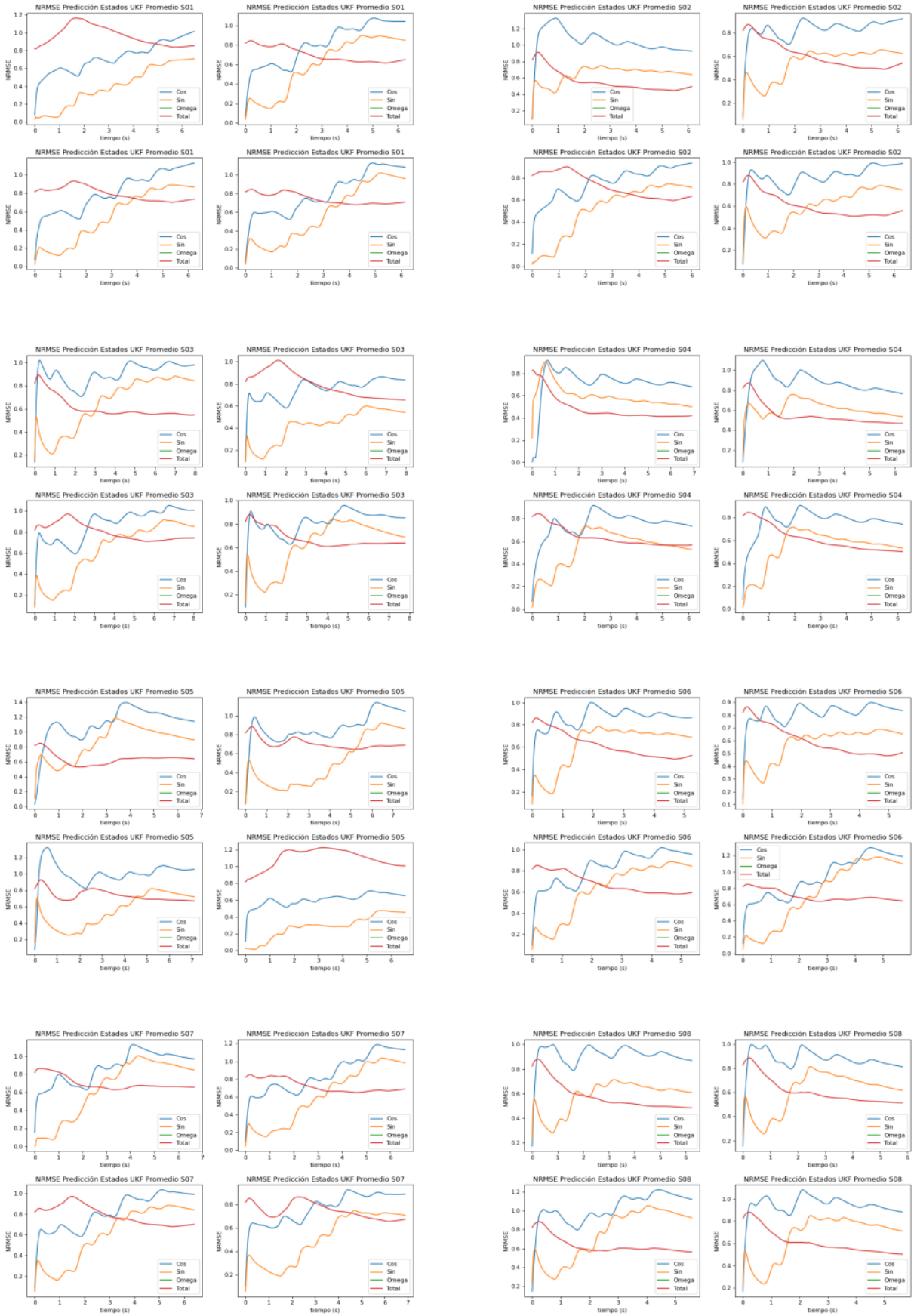


Figura 8.13: Error cuadrático medio normalizado (NRMSE) de las predicciones del UKF en el escenario realista para las pruebas empezando con la pierna derecha

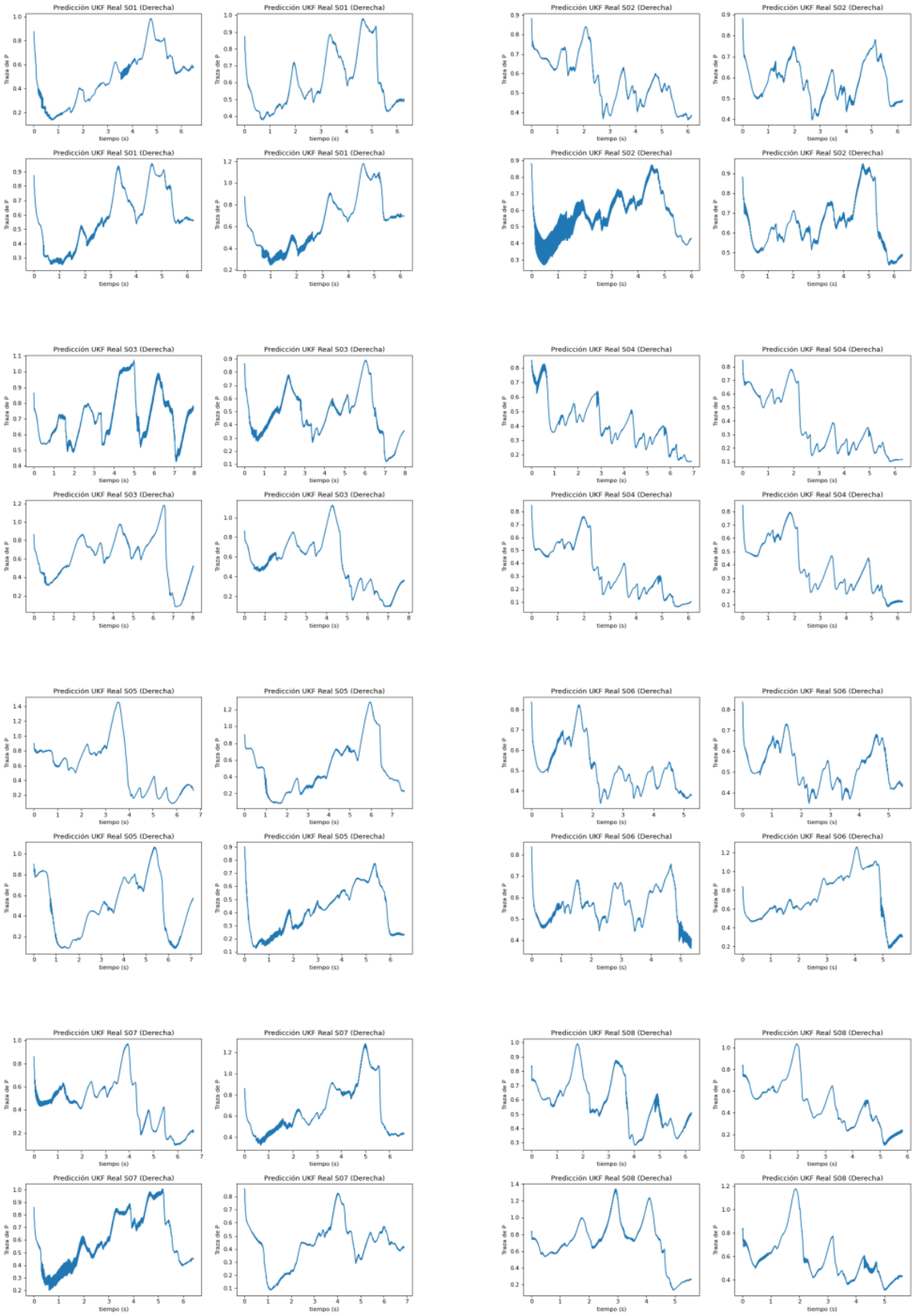


Figura 8.14: Traza de la matriz de covarianza cruzada de las predicciones del UKF en el escenario realista para las pruebas empezando con la pierna derecha

Figura 8.15: Mínimo valor singular de la matriz de covarianza cruzada del UKF en el escenario realista para las pruebas empezando con la pierna derecha

Bibliografía

Agarwal, Devraj (2016). *A Review of the Math used in training a Neural Network — levelup.gitconnected.com*. Accessed 30-05-2024 (vid. pág. 32).

Aggarwal, Priyanka (2010). *MEMS-based integrated navigation*. Artech House (vid. pág. 10).

Ahmad, Norhafizan et al. (2013). «Reviews on various inertial measurement unit (IMU) sensor applications». En: *International Journal of Signal Processing Systems* 1.2, págs. 256-262 (vid. pág. 12).

Albawi, Saad, Tareq Abed Mohammed y Saad Al-Zawi (2017). «Understanding of a convolutional neural network». En: págs. 1-6. doi: 10.1109/ICEngTechnol.2017.8308186 (vid. págs. 33, 34).

Amjadi, Morteza, Min Seong Kim e Inkyu Park (2014). «Flexible and sensitive foot pad for sole distributed force detection». En: págs. 764-767 (vid. págs. 15, 16).

André, João et al. (2020). «Markerless gait analysis vision system for real-time gait monitoring». En: págs. 269-274 (vid. págs. 9, 21).

Andrés Diaz, Christian et al. (2009). «Detección, rastreo Y reconstrucción tridimensional de marcadores pasivos para análisis de movimiento humano. CINEMED II». En: *Revista Ingeniería Biomédica* 3.6, págs. 56-67 (vid. pág. 7).

Arellano-González, Juan Carlos et al. (2021). «A practical review of the biomechanical parameters commonly used in the assessment of human gait». En: *Revista mexicana de ingeniería biomédica* 42.3 (vid. pág. 3).

Arshad, Muhammad Zeeshan et al. (nov. de 2022). «Gait Events Prediction Using Hybrid CNN-RNN-Based Deep Learning Models through a Single Waist-Worn Wearable Sensor». English. En: *SENSORS* 22.21. doi: 10.3390/s22218226 (vid. págs. 21, 38, 40).

Bai, Lu et al. (2022). «Upper Limb Position Tracking with a Single Inertial Sensor Using Dead Reckoning Method with Drift Correction Techniques». En: *Sensors* 23.1, pág. 360 (vid. pág. 11).

Baker, Jessica M (2018). «Gait disorders». En: *The American journal of medicine* 131.6, págs. 602-607 (vid. págs. 3, 5-7).

Bandos, Tatyana V, Lorenzo Bruzzone y Gustavo Camps-Valls (2009). «Classification of hyperspectral images with regularized linear discriminant analysis». En: *IEEE Transactions on Geoscience and Remote Sensing* 47.3, págs. 862-873 (vid. págs. 27, 28).

Barkalov, Alexander et al. (2024). «Basic Approaches for Reducing Power Consumption in Finite State Machine Circuits—A Review». En: *Applied Sciences* 14.7, pág. 2693 (vid. pág. 26).

Bartlett, Harrison L. y Michael Goldfarb (jun. de 2018). «A Phase Variable Approach for IMU-Based Locomotion Activity Recognition». English. En: *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING* 65.6, págs. 1330-1338. issn: 0018-9294. doi: 10.1109/TBME.2017.2750139 (vid. págs. 21, 25, 40).

Begon, Mickaël et al. (2009). «Computation of the 3D kinematics in a global frame over a 40 m-long pathway using a rolling motion analysis system». En: *Journal of biomechanics* 42.16, págs. 2649-2653 (vid. pág. 8).

Bennett, James S et al. (2021). «Precision magnetometers for aerospace applications: A review». En: *Sensors* 21.16, pág. 5568 (vid. pág. 12).

Bourlard, Hervé y Samy Bengio (2001). «Hidden Markov models and other finite state automata for sequence processing». En: (vid. págs. 26, 27).

Brard, Raphael et al. (mayo de 2022). «A Novel Walking Activity Recognition Model for Rotation Time Series Collected by a Wearable Sensor in a Free-Living Environment». English. En: *SENSORS* 22.9. doi: 10.3390/s22093555 (vid. págs. 21, 40).

Chattopadhyay, Sourav y Anup Nandy (2018). «Human Gait Modelling Using Hidden Markov Model For Abnormality Detection». English. En: TENCON IEEE Region 10 Conference Proceedings. IEEE-Region-10 Conference (IEEE TENCON), IEEE Reg 10, SOUTH KOREA, OCT 28-31, 2018, págs. 0623-0628. issn: 2159-3442 (vid. págs. 21, 40).

Chen, C.-F. et al. (2021a). «A Novel Gait Pattern Recognition Method Based on LSTM-CNN for Lower Limb Exoskeleton». English. En: *Journal of Bionic Engineering* 18.5, págs. 1059-1072. issn: 16726529 (ISSN). doi: 10.1007/s42235-021-00083-y (vid. págs. 21, 34, 40).

Chen, Tianyu et al. (2021b). «A novel calibration method for gyro-accelerometer asynchronous time in foot-mounted pedestrian navigation system». En: *Sensors* 22.1, pág. 209 (vid. págs. 4, 25).

Cheng, Shihao, Edgar Bolívar-Nieto y Robert D. Gregg (2021). «Real-Time Activity Recognition With Instantaneous Characteristic Features of Thigh Kinematics». En: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 29, págs. 1827-1837. issn: 1558-0210. doi: 10.1109/TNSRE.2021.3107780 (vid. págs. 21, 25, 40).

Chinimilli, Prudhvi Tej et al. (2019). «Human Locomotion Assistance using Two-Dimensional Features Based Adaptive Oscillator». English. En: Wearable-Robotics-Association Conference (WearRAcon), Scottsdale, AZ, MAR 26-28, 2019, págs. 92-98. doi: 10.1109/wearracon.2019.8719628 (vid. págs. 13, 21, 25, 29, 40).

Chou, Jui-Sheng, Dinh-Nhat Truong y Chih-Fong Tsai (2021). «Solving regression problems with intelligent machine learner for engineering informatics». En: *Mathematics* 9.6, pág. 686 (vid. pág. 29).

Chung, Junyoung et al. (2014). «Empirical evaluation of gated recurrent neural networks on sequence modeling». En: *arXiv preprint arXiv:1412.3555* (vid. pág. 38).

Cortes, Corinna y Vladimir Vapnik (1995). «Support-vector networks». En: *Machine learning* 20, págs. 273-297 (vid. pág. 28).

Cui, Rubiao et al. (2024). «A Temperature Compensation Approach for Micro-Electro-Mechanical Systems Accelerometer Based on Gated Recurrent Unit–Attention and Robust Local Mean Decomposition–Sample Entropy–Time-Frequency Peak Filtering». En: *Micromachines* 15.4, pág. 483 (vid. pág. 11).

David Li, Y. y E. T. Hsiao-Wecksler (2013). «Gait mode recognition and control for a portable-powered ankle-foot orthosis». English. En: *IEEE International Conference on Rehabilitation Robotics* 6650373. issn: 19457901 (ISSN); 978-146736024-1 (ISBN). doi: 10.1109/ICORR.2013.6650373. PMID: 24187192 (vid. págs. 21, 25, 40).

Delashmit, Walter H, Michael T Manry et al. (2005). «Recent developments in multilayer perceptron neural networks». En: 7, pág. 33 (vid. pág. 32).

Diao, Zhanlin et al. (2013). «Analysis and compensation of MEMS gyroscope drift». En: págs. 592-596 (vid. pág. 12).

Diebel, James et al. (2006). «Representing attitude: Euler angles, unit quaternions, and rotation vectors». En: *Matrix* 58.15-16, págs. 1-35 (vid. págs. 51, 54).

Dietterich, Thomas G (2000). «Ensemble methods in machine learning». En: págs. 1-15 (vid. págs. 29-31).

Ding, Shuo et al. (dic. de 2018a). «Gait Event Detection of a Lower Extremity Exoskeleton Robot by an Intelligent IMU». English. En: *IEEE SENSORS JOURNAL* 18.23, págs. 9728-9735. issn: 1530-437X. doi: 10.1109/JSEN.2018.2871328 (vid. págs. 21, 25, 40).

Ding, Z. et al. (2018b). «The real time gait phase detection based on long short-term memory». English. En: *Proceedings - 2018 IEEE 3rd International Conference on Data Science in Cyberspace, DSC 2018* 8411835, págs. 33-38. issn: 978-153864210-8 (ISBN). doi: 10.1109/DSC.2018.00014 (vid. págs. 21, 40).

Dinovitzer, Hannah, Mohammad Shushtari y Arash Arami (ago. de 2023). «Accurate Real-Time Joint Torque Estimation for Dynamic Prediction of Human Locomotion». English. En: *IEEE TRANSACTIONS ON BIOMEDICAL ENGINEERING* 70.8, págs. 2289-2297. issn: 0018-9294. doi: 10.1109/TBME.2023.3240879 (vid. págs. 8, 21, 40).

Djurić-Jovičić, Milica D, Nenad S Jovičić y Dejan B Popović (2011). «Kinematics of gait: new method for angle estimation based on accelerometers». En: *Sensors* 11.11, págs. 10571-10585 (vid. pág. 11).

D'Lima, Darryl D et al. (2000). «Standard for hip joint coordinate system recommendations from the ISB standardization committee». En: *Jul* 17, págs. 1-8 (vid. págs. 46, 50).

Drouot, C. y N. Jarrassé (2022). «Improved real-time gait phase detection using simple force sensors, sequencing conditions and smart fault management». English. En: *2022 IEEE International Conference on Robotics and Biomimetics, ROBIO 2022*, págs. 2328-2335. issn: 978-166548109-0 (ISBN). doi: 10.1109/ROBIO55434.2022.10011720 (vid. págs. 16, 21, 25, 26, 40).

Duan, Pu et al. (2017). «Bio-inspired real-time prediction of human locomotion for exoskeletal robot control». En: *Applied Sciences* 7.11, pág. 1130 (vid. pág. 3).

D'Antonio, Erika et al. (2020). «A markerless system for gait analysis based on OpenPose library». En: págs. 1-6 (vid. pág. 40).

D'Antonio, Erika et al. (2021). «Validation of a 3D markerless system for gait analysis based on OpenPose and two RGB webcams». En: *IEEE Sensors Journal* 21.15, págs. 17064-17075 (vid. pág. 40).

Eddy, Sean R (1996). «Hidden markov models». En: *Current opinion in structural biology* 6.3, págs. 361-365 (vid. pág. 26).

Edelstein, AS et al. (2008). «Advances in magnetometry through miniaturization». En: *Journal of Vacuum Science & Technology A* 26.4, págs. 757-762 (vid. pág. 12).

Elbes, Mohammed, Ala Al-Fuqaha y Ammar Rayes (2012). «Gyroscope drift correction based on TDoA technology in support of pedestrian dead reckoning». En: págs. 314-319 (vid. pág. 10).

Emmerich, Harald y Martin Schofthaler (2000). «Magnetic field measurements with a novel surface micromachined magnetic-field sensor». En: *IEEE Transactions on Electron Devices* 47.5, págs. 972-977 (vid. pág. 12).

Ershadi, Ghazal et al. (2022). «GAIToe: Gait Analysis Utilizing an IMU for Toe Walking Detection and Intervention». English. En: Lecture Notes of the Institute for Computer Sciences Social Informatics and Telecommunications Engineering 432. Ed. por S Spinsante, B Silva y R Goleva. 8th European-Alliance-for-Innovation (EAI) International Conference on IoT Technologies for Health Care (HealthyIoT), ELECTR NETWORK, NOV 24-26, 2021, págs. 180-195. issn: 1867-8211. doi: 10.1007/978-3-030-99197-5_15 (vid. págs. 21, 40).

Fabiyi, Samson Damilola et al. (2021). «Folded LDA: extending the linear discriminant analysis algorithm for feature extraction and data reduction in hyperspectral remote sensing». En: *IEEE Journal of selected topics in applied earth observations and remote sensing* 14, págs. 12312-12331 (vid. pág. 28).

Ferraris, Franco, Ugo Grimaldi y Marco Parvis (1995). «Three-Axis Rate Gyros and Accelerometers». En: *Sensors and Materials* 7.5, págs. 311-330 (vid. pág. 10).

Gao, Jing et al. (2019). «Abnormal Gait Recognition Algorithm Based on LSTM-CNN Fusion Network». English. En: *IEEE ACCESS* 7, págs. 163180-163190. issn: 2169-3536. doi: 10.1109/ACCESS.2019.2950254 (vid. págs. 21, 40).

Gonçalves, H. R. et al. (2018). «Real-time tool for human gait detection from lower trunk acceleration». English. En: *Advances in Intelligent Systems and Computing* 747, págs. 9-18. issn: 21945357 (ISSN); 978-331977699-6 (ISBN). doi: 10.1007/978-3-319-77700-9_2 (vid. págs. 21, 40).

Graves, Alex y Jürgen Schmidhuber (2005). «Framewise phoneme classification with bidirectional LSTM and other neural network architectures». En: *Neural networks* 18.5-6, págs. 602-610 (vid. pág. 38).

Grimmer, Martin et al. (jul. de 2019). «Stance and Swing Detection Based on the Angular Velocity of Lower Limb Segments During Walking». English. En: *FRONTIERS IN NEUROROBOTICS* 13. issn: 1662-5218. doi: 10.3389/fnbot.2019.00057 (vid. págs. 21, 40).

Gu, Xiao et al. (2018). «Markerless gait analysis based on a single RGB camera». En: págs. 42-45 (vid. págs. 9, 21, 40).

Guo, Yanming et al. (2016). «Deep learning for visual understanding: A review». En: *Neurocomputing* 187, págs. 27-48 (vid. pág. 34).

Haddaway, Neal Robert et al. (2015). «The role of Google Scholar in evidence reviews and its applicability to grey literature searching». En: *PloS one* 10.9, e0138237 (vid. pág. 42).

Hellmann, Martin (2001). «Fuzzy logic introduction». En: *Université de Rennes* 1.1 (vid. pág. 24).

Hinton, Geoffrey E et al. (2012). «Improving neural networks by preventing co-adaptation of feature detectors». En: *arXiv preprint arXiv:1207.0580* (vid. pág. 36).

Hirasawa, Masaki, Hidetaka Okada y Makoto Shimojo (2008). «The development of the plantar pressure sensor shoes for gait analysis». En: *Journal of Robotics and Mechatronics* 20.2, pág. 289 (vid. pág. 15).

Hoang, Minh Long y Antonio Pietrosanto (2021). «Yaw/Heading optimization by drift elimination on MEMS gyroscope». En: *Sensors and Actuators A: Physical* 325, pág. 112691 (vid. pág. 12).

Hoffmann, Karl (1974). *Applying the wheatstone bridge circuit*. HBM Darmstadt, Germany (vid. pág. 16).

Howard, H Ge, Amir H Behbahani y Robert T M'Closkey (2018). «MEMS gyro drift compensation using multiple rate measurements derived from a single resonator». En: págs. 288-293 (vid. pág. 12).

Hu, Fo et al. (mayo de 2021). «A novel fusion strategy for locomotion activity recognition based on multimodal signals». English. En: *BIOMEDICAL SIGNAL PROCESSING AND CONTROL* 67. issn: 1746-8094. doi: 10.1016/j.bspc.2021.102524 (vid. págs. 17, 21, 23, 40).

Huo, Weiguang et al. (oct. de 2018). «Adaptive FES Assistance Using a Novel Gait Phase Detection Approach». En: págs. 1-9. issn: 2153-0866. doi: 10.1109/IROS.2018.8594051 (vid. págs. 21, 40).

Ibe, Oliver (2013). *Markov processes for stochastic modeling*. Newnes (vid. pág. 27).

«IEEE standard specification format guide and test procedure for single-axis interferometric fiber optic gyros» (1998). En: *IEEE Aerospace and Electronic Systems Society Gyro and Accelerometer Panel*. Institute of Electrical y Electronics Engineers (vid. pág. 11).

Ioffe, Sergey y Christian Szegedy (2015). «Batch normalization: Accelerating deep network training by reducing internal covariate shift». En: págs. 448-456 (vid. págs. 35, 36).

Iosa, Marco et al. (2013). «The golden ratio of gait harmony: repetitive proportions of repetitive gait phases». En: *BioMed research international* 2013 (vid. págs. 3, 5).

Irby, Ethan (2013). *Gradient Descent vs Stochastic Gradient Descent vs Mini-batch SGD — medium.com*. Accessed 30-05-2024 (vid. pág. 35).

Ivanenko, Yuri P, Richard E Poppele y Francesco Lacquaniti (2004). «Five basic muscle activation patterns account for muscle activity during human locomotion». En: *The Journal of physiology* 556.1, págs. 267-282 (vid. pág. 17).

Iyer, Brijesh, AM Rajurkar y Venkat Gudivada (2020). *Applied computer vision and image processing*. Springer (vid. pág. 25).

Jagdale, Devang (2021). «Finite state machine in game development». En: *International Journal of Advanced Research in Science, Communication and Technology* 10.1 (vid. pág. 26).

Jani, Darshan et al. (sep. de 2022). «An Efficient Gait Abnormality Detection Method Based on Classification». English. En: *JOURNAL OF SENSOR AND ACTUATOR NETWORKS* 11.3. doi: 10.3390/jsan11030031 (vid. págs. 15, 21, 22, 31, 40).

Jiang, X. et al. (2018). «Exploration of Gait Parameters Affecting the Accuracy of Force Myography-Based Gait Phase Detection». English. En: Proceedings of the IEEE RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics. 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (BIOROB), Enschede, NETHERLANDS, AUG 26-29, 2018, págs. 1205-1210. issn: 2155-1782 (vid. págs. 16, 21, 28, 40).

JKO (s.f.). *MTw Awinda User Manual*. English. Ver. MW0502P.L. Xsens Technologies B.V. 88 págs. May 3, 2018 (vid. págs. 46, 52).

Joshi, Chetas D, Uttama Lahiri y Nitish V Thakor (2013). «Classification of gait phases from lower limb EMG: Application to exoskeleton orthosis». En: págs. 228-231 (vid. pág. 17).

Julier, Simon J y Jeffrey K Uhlmann (1997). «New extension of the Kalman filter to nonlinear systems». En: *Signal processing, sensor fusion, and target recognition VI*. Vol. 3068. Spie, págs. 182-193 (vid. pág. 66).

Jung, Pyeong-gook et al. (2021). «A Novel Gait Phase Detection Algorithm for Foot Drop Correction through Optimal Hybrid FES-Orthosis Assistance». English. En: IEEE International Conference on Robotics and Automation ICRA. IEEE International Conference on Robotics and Automation (ICRA), Xian, PEOPLES R CHINA, MAY 30-JUN 05, 2021, págs. 10391-10397. issn: 1050-4729. doi: 10.1109/ICRA48506.2021.9561497 (vid. págs. 21, 40).

Karakish, Mohamed, Moustafa A. Fouz y Ahmed ELsawaf (nov. de 2022). «Gait Trajectory Prediction on an Embedded Microcontroller Using Deep Learning». English. En: *SENSORS* 22.21. doi: 10.3390/s22218441 (vid. págs. 21, 34, 40).

Kovač, Ida, Vladimir Medved y Ljerka Ostojić (2010). «Spatial, temporal and kinematic characteristics of traumatic transtibial amputees' gait». En: *Collegium antropologicum* 34.1, págs. 205-213 (vid. pág. 10).

Krausz, Nili E. y Levi J. Hargrove (ago. de 2021). «Sensor Fusion of Vision, Kinetics, and Kinematics for Forward Prediction During Walking With a Transfemoral Prosthesis». En: *IEEE Transactions on Medical Robotics and Bionics* 3.3, págs. 813-824. issn: 2576-3202. doi: 10.1109/TMRB.2021.3082206 (vid. págs. 17, 21, 23, 28, 40).

Kruk, Eline Van der y Marco M Reijne (2018). «Accuracy of human motion capture systems for sport applications; state-of-the-art review». En: *European journal of sport science* 18.6, págs. 806-819 (vid. págs. 8-10).

Lahat, Dana, Tülay Adali y Christian Jutten (2015). «Multimodal data fusion: an overview of methods, challenges, and prospects». En: *Proceedings of the IEEE* 103.9, págs. 1449-1477 (vid. pág. 23).

Lam, Winnie WT, Yuk Ming Tang y Kenneth NK Fong (2023). «A systematic review of the applications of markerless motion capture (MMC) technology for clinical measurement in rehabilitation». En: *Journal of NeuroEngineering and Rehabilitation* 20.1, pág. 57 (vid. pág. 10).

Latash, Mark L (2012). *Fundamentals of motor control*. Academic Press (vid. pág. 14).

Lee, Minhyung et al. (2009). «Gait analysis to classify external load conditions using linear discriminant analysis». En: *Human movement science* 28.2, págs. 226-235 (vid. pág. 27).

Lee, Yeonkyung y Hoon Yoo (2017). «Low-cost 3D motion capture system using passive optical markers and monocular vision». En: *Optik* 130, págs. 1397-1407 (vid. pág. 8).

Linnell, Jeffrey (2015). *Programming of a robotic arm using a motion capture system*. US Patent 9,056,396 (vid. pág. 8).

Lipomi, Darren J et al. (2011). «Skin-like pressure and strain sensors based on transparent elastic films of carbon nanotubes». En: *Nature nanotechnology* 6.12, págs. 788-792 (vid. pág. 16).

Liu, K. et al. (2023a). «A Novel Gait Phase Recognition Method Based on DPF-LSTM-CNN Using Wearable Inertial Sensors». English. En: *Sensors* 23.5905. issn: 14248220 (ISSN). doi: 10.3390/s23135905. PMID: 37447755 (vid. págs. 21, 34, 40).

Liu, Kai et al. (2009). «The development of micro-gyroscope technology». En: *Journal of Micromechanics and Microengineering* 19.11, pág. 113001 (vid. pág. 11).

Liu, Yancheng et al. (2014). «Gait phase varies over velocities». En: *Gait & posture* 39.2, págs. 756-760 (vid. pág. 5).

Liu, Yong et al. (dic. de 2023b). «Gait Phase Recognition for Soft Exoskeleton Assistance Based on Inertial Sensors». En: págs. 1-6. doi: 10.1109/ROBIO58561.2023.10354811 (vid. págs. 13, 21, 29, 40).

Lu, Y. et al. (2023). «An IMU-Based Real-Time Gait Detection Method for Intelligent Control of Knee Assistive Devices». English. En: *IEEE Transactions on Instrumentation and Measurement* 72.2532309, págs. 1-9. issn: 00189456 (ISSN). doi: 10.1109/TIM.2023.3329222 (vid. págs. 21, 40).

Luo, Ren C y Michael G Kay (1988). «Multisensor integration and fusion: issues and approaches». En: 931, págs. 42-49 (vid. págs. 23, 27).

MacKay, David JC (1997). *Ensemble learning for hidden Markov models*. Inf. téc. Citeseer (vid. pág. 27).

Maletsky, Lorin P, Junyi Sun y Nicholas A Morton (2007). «Accuracy of an optical active-marker system to track the relative motion of rigid bodies». En: *Journal of biomechanics* 40.3, págs. 682-685 (vid. pág. 8).

Mansoor, Shoaib et al. (2019). «Improved attitude determination by compensation of gyroscopic drift by use of accelerometers and magnetometers». En: *Measurement* 131, págs. 582-589 (vid. pág. 12).

Maqbool, H. F. et al. (2016). «Real-time gait event detection for lower limb amputees using a single wearable sensor». English. En: IEEE Engineering in Medicine and Biology Society Conference Proceedings. Ed. por J Patton et al. 38th Annual International Conference of the IEEE-Engineering-in-Medicine-and-Biology-Society (EMBC), Orlando, FL, AUG 16-20, 2016, págs. 5067-5070. issn: 1557-170X (vid. págs. 21, 25, 40).

Martinez-Hernandez, Uriel, Adrian Rubio-Solis y Abbas A. Dehghani-Sanij (2018). «Recognition of walking activity and prediction of gait periods with a CNN and first-order MC strategy». English. En: Proceedings of the IEEE RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics. 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (BIOROB), Enschede, NETHERLANDS, AUG 26-29, 2018, págs. 897-902. issn: 2155-1782 (vid. págs. 21, 27, 40).

Martini, Elena et al. (2020). «Pressure-sensitive insoles for real-time gait-related applications». En: *Sensors* 20.5, pág. 1448 (vid. pág. 16).

Mason, R et al. (2023). «Wearables for running gait analysis: a systematic...» En: (vid. pág. 15).

Matthew, Robert Peter, Sarah Seko y Ruzena Bajcsy (2017). «Fusing motion-capture and inertial measurements for improved joint state recovery: An application for sit-to-stand actions». En: págs. 1893-1896 (vid. pág. 8).

Meng, Tong et al. (2020). «A survey on machine learning for data fusion». En: *Information Fusion* 57, págs. 115-129 (vid. pág. 23).

Menolotto, Matteo et al. (2020). «Motion capture technology in industrial applications: A systematic review». En: *Sensors* 20.19, pág. 5687 (vid. págs. 8, 9).

Michał, B. et al. (2017). «Gait phase recognition for exoskeleton control using adaptive neuro fuzzy inference system». English. En: *Annals of DAAAM and Proceedings of the International DAAAM Symposium*, págs. 845-853. issn: 17269679 (ISSN); 978-390273411-2 (ISBN). doi: 10.2507/28th.daaam.proceedings.118 (vid. págs. 16, 21, 24, 40).

Mienye, Ibomoiye Domor, Yanxia Sun y Zenghui Wang (2019). «Prediction performance of improved decision tree-based algorithms: a review». En: *Procedia Manufacturing* 35, págs. 698-703 (vid. pág. 25).

Miles, David M (2017). «Advances in fluxgate magnetometry for space physics». En: (vid. pág. 11).

Mor, Bhavya, Sunita Garhwal y Ajay Kumar (2021). «A systematic review of hidden Markov models and their applications». En: *Archives of computational methods in engineering* 28, págs. 1429-1448 (vid. págs. 26, 27).

Moro, Matteo et al. (2022). «Markerless vs. marker-based gait analysis: A proof of concept study». En: *Sensors* 22.5, pág. 2011 (vid. págs. 9, 40).

Mundt, Marion et al. (ene. de 2020). «Prediction of lower limb joint angles and moments during gait using artificial neural networks». English. En: *MEDICAL & BIOLOGICAL ENGINEERING & COMPUTING* 58.1, págs. 211-225. issn: 0140-0118. doi: 10.1007/s11517-019-02061-3 (vid. págs. 21, 22, 34, 40).

Nadvi, Gaviraj S et al. (2012). «Micromachined force sensors using thin film nickel–chromium piezoresistors». En: *Journal of Micromechanics and Microengineering* 22.6, pág. 065002 (vid. pág. 15).

Nakano, Nobuyasu et al. (2020). «Evaluation of 3D markerless motion capture accuracy using OpenPose with multiple video cameras». En: *Frontiers in sports and active living* 2, pág. 50 (vid. págs. 8, 10).

NATO (2017). *NATO Science and Technology Organization — sto.nato.int*. Accessed 29-05-2024 (vid. pág. 13).

Ng, Andrew Y (2004). «Feature selection, L 1 vs. L 2 regularization, and rotational invariance». En: pág. 78 (vid. págs. 35, 36).

Nguyen, L. V., H. M. La y T. H. Duong (2015). «Dynamic Human Gait Phase Detection Algorithm». English. En: *Proceedings - ISSAT International Conference on Modeling of Complex Systems and Environments 2015*, págs. 91-95 (vid. págs. 21, 40).

Ning, Zhiwen et al. (2021). «Improved MEMS magnetometer adaptive filter noise reduction and compensation method». En: *IEEE Sensors Journal* 22.2, págs. 1252-1264 (vid. pág. 12).

Novak, D. et al. (2013). «Automated detection of gait initiation and termination using wearable sensors». English. En: *Medical Engineering and Physics* 35.12, págs. 1713-1720. issn: 18734030 (ISSN). doi: 10.1016/j.medengphy.2013.07.003. PMID: 23938085 (vid. págs. 21, 40).

Nozaki, Yoshitaka y Takashi Watanabe (2019). «A Basic Study on Detection of Movement State in Stride by Artificial Neural Network for Estimating Stride Length of Hemiplegic Gait Using IMU». English. En: IEEE Engineering in Medicine and Biology Society Conference Proceedings. 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, GERMANY, JUL 23-27, 2019, págs. 3151-3154. issn: 1557-170X. doi: 10.1109/embc.2019.8856352 (vid. págs. 21, 34, 40).

Oh, Hye-Won y Young-Dae Hong (abr. de 2023). «Divergent Component of Motion-Based Gait Intention Detection Method Using Motion Information From Single Leg». English. En: *JOURNAL OF INTELLIGENT & ROBOTIC SYSTEMS* 107.4. issn: 0921-0296. doi: 10.1007/s10846-023-01843-0 (vid. págs. 21, 25, 40).

Osborne, Jason (2010). «Improving your data transformations: Applying the Box-Cox transformation». En: *Practical Assessment, Research, and Evaluation* 15.1 (vid. pág. 56).

O'shea, Keiron y Ryan Nash (2015). «An introduction to convolutional neural networks». En: *arXiv preprint arXiv:1511.08458* (vid. págs. 32-34).

Park, Tae-Geun y Jung-Yup Kim (sep. de 2022). «Real-time prediction of walking state and percent of gait cycle for robotic prosthetic leg using artificial neural network». English. En: *INTELLIGENT SERVICE ROBOTICS* 15.4, SI, págs. 527-536. issn: 1861-2776. doi: 10.1007/s11370-022-00434-6 (vid. págs. 17, 21, 40).

Parmar, Aakash, Rakesh Katariya y Vatsal Patel (2019). «A review on random forest: An ensemble classifier». En: págs. 758-763 (vid. pág. 31).

Pasciuto, Ilaria et al. (2015). «How angular velocity features and different gyroscope noise types interact and determine orientation estimation accuracy». En: *Sensors* 15.9, págs. 23983-24001 (vid. págs. 10, 11).

Pazar, A. et al. (2022). «Gait Phase Recognition using Textile-based Sensor». English. En: *Proceedings - 7th International Conference on Computer Science and Engineering, UBMK 2022*, págs. 338-343. issn: 978-166547010-0 (ISBN). doi: 10.1109/UBMK55850.2022.9919491 (vid. págs. 38, 40).

Pfister, Alexandra et al. (2014). «Comparative abilities of Microsoft Kinect and Vicon 3D motion capture for gait analysis». En: *Journal of medical engineering & technology* 38.5, págs. 274-280 (vid. pág. 9).

Pirker, Walter y Regina Katzenschlager (2017). «Gait disorders in adults and the elderly: A clinical guide». En: *Wiener Klinische Wochenschrift* 129.3-4, págs. 81-95 (vid. págs. 6, 7).

Popescu, Marius-Constantin et al. (2009). «Multilayer perceptron and neural networks». En: *WSEAS Transactions on Circuits and Systems* 8.7, págs. 579-588 (vid. págs. 32, 35).

Potluri, Sasanka et al. (2019). «Deep Learning based Gait Abnormality Detection using Wearable Sensor System». English. En: IEEE Engineering in Medicine and Biology Society Conference Proceedings. 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, GERMANY, JUL 23-27, 2019, págs. 3613-3619. issn: 1557-170X. doi: 10.1109/embc.2019.8856454 (vid. págs. 21, 38, 40).

Prasanth, Hari et al. (2021). «Wearable sensor-based real-time gait detection: A systematic review». En: *Sensors* 21.8, pág. 2727 (vid. pág. 10).

Prigent, Gaëlle et al. (sep. de 2023). «A robust walking detection algorithm using a single foot-worn inertial sensor: validation in real-life settings». English. En: *MEDICAL & BIOLOGICAL ENGINEERING & COMPUTING* 61.9, págs. 2341-2352. issn: 0140-0118. doi: 10.1007/s11517-023-02826-x (vid. págs. 21, 40).

Prikhodko, Igor P, Alexander A Trusov y Andrei M Shkel (2013). «Compensation of drifts in high-Q MEMS gyroscopes using temperature self-sensing». En: *Sensors and Actuators A: Physical* 201, págs. 517-524 (vid. pág. 12).

Qi, Bing et al. (2022). «A novel temperature drift error precise estimation model for MEMS accelerometers using microstructure thermal analysis». En: *Micromachines* 13.6, pág. 835 (vid. págs. 10, 11).

Rahul, M (2018). «Review on motion capture technology». En: *Global Journal of Computer Science and Technology* 18.F1, págs. 23-26 (vid. pág. 10).

Ramakrishnan, HK y MP Kadaba (1991). «On the estimation of joint kinematics during gait». En: *Journal of biomechanics* 24.10, págs. 969-977 (vid. págs. 58, 61).

Ramchoun, Hassan et al. (2016). «Multilayer perceptron: Architecture optimization and training». En: (vid. págs. 32, 35).

Rana, NK (2009). «Application of force sensing resistor (FSR) in design of pressure scanning system for plantar pressure measurement». En: 2, págs. 678-685 (vid. pág. 15).

Renani, Mohsen Sharifi et al. (sep. de 2021). «The Use of Synthetic IMU Signals in the Training of Deep Learning Models Significantly Improves the Accuracy of Joint Kinematic Predictions». English. En: *SENSORS* 21.17. doi: 10.3390/s21175876 (vid. págs. 21, 40).

Rezaei, A. et al. (2018). «Preliminary Investigation of Textile-Based Strain Sensors for the Detection of Human Gait Phases Using Machine Learning». English. En: Proceedings of the IEEE RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics. 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (BIOROB), Enschede, NETHERLANDS, AUG 26-29, 2018, págs. 563-568. issn: 2155-1782 (vid. pág. 40).

Romero-Hernandez, Pedro, Javier de Lope Asiain y Manuel Grana (2019). «Deep Learning Prediction of Gait Based on Inertial Measurements». English. En: Lecture Notes in Computer Science 11486. Ed. por JMF Vicente et al. 8th International Work-Conference on the Interplay Between Natural and Artificial Computation (IWINAC), Almeria, SPAIN, JUN 03-07, 2019, págs. 284-290. issn: 0302-9743. doi: 10.1007/978-3-030-19591-5_29 (vid. págs. 21, 40).

Romijnders, Robbin et al. (mayo de 2022). «A Deep Learning Approach for Gait Event Detection from a Single Shank-Worn IMU: Validation in Healthy and Neurological Cohorts». English. En: *SENSORS* 22.10. doi: 10.3390/s22103859 (vid. págs. 8, 21, 40).

Rosenblatt, Frank (1958). «Perceptron a theory of statistical separability in congnitive system». En: *(No Title)* (vid. pág. 32).

Ross, Timothy J (2009). *Fuzzy logic with engineering applications*. John Wiley & Sons (vid. pág. 24).

Sabri, Naseer et al. (2013). «Fuzzy inference system: Short review and design». En: *Int. Rev. Autom. Control* 6.4, págs. 441-449 (vid. pág. 24).

Sagi, Omer y Lior Rokach (2018). «Ensemble learning: A survey». En: *Wiley interdisciplinary reviews: data mining and knowledge discovery* 8.4, e1249 (vid. págs. 30, 31).

Sahoo, Saikat et al. (abr. de 2020). «A Geometry Recognition-Based Strategy for Locomotion Transitions Early Prediction of Prosthetic Devices». En: *IEEE Transactions on Instrumentation and Measurement* 69.4, págs. 1259-1267. issn: 1557-9662. doi: 10.1109/TIM.2019.2909246 (vid. págs. 17, 21, 40).

Salcedo-Sanz, Sancho et al. (2014). «Support vector machines in engineering: an overview». En: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 4.3, págs. 234-267 (vid. págs. 27-29).

Sanchez Manchola, Miguel D. et al. (jul. de 2019). «Gait Phase Detection for Lower-Limb Exoskeletons using Foot Motion Data from a Single Inertial Measurement Unit in Hemiparetic Individuals». English. En: *SENSORS* 19.13. issn: 1424-8220. doi: 10.3390/s19132988 (vid. págs. 21, 40).

Schuster, Mike y Kuldip K Paliwal (1997). «Bidirectional recurrent neural networks». En: *IEEE transactions on Signal Processing* 45.11, págs. 2673-2681 (vid. págs. 36-38).

Schuy, J. et al. (2015). «Design & Evaluation of a Sensor Minimal Gait Phase and Situation Detection Algorithm of Human Walking». English. En: IEEE-RAS International Conference on Humanoid Robots. 15th IEEE-RAS International Conference on Humanoid Robots (Humanoids), Seoul, SOUTH KOREA, NOV 03-05, 2015, págs. 20-25. issn: 2164-0572 (vid. págs. 21, 24, 25, 40).

Semwal, Vijay Bhaskar, Anjali Gupta y Praveen Lalwani (nov. de 2021). «An optimized hybrid deep learning model using ensemble learning approach for human walking activities recognition». English. En: *JOURNAL OF SUPERCOMPUTING* 77.11, págs. 12256-12279. issn: 0920-8542. doi: 10.1007/s11227-021-03768-7 (vid. págs. 21, 40).

Shaikh, Muhammad Faraz, Zoran Salcic y Kevin Wang (2015). «Analysis and selection of the Force Sensitive Resistors for gait characterisation». En: págs. 370-375 (vid. pág. 15).

Shakya, S., A. Taparugssanagorn y C. Silpasuwanchai (2023). «Convolutional Neural Network-Based Low-Powered Wearable Smart Device for Gait Abnormality Detection». English. En: *IoT* 4.2, págs. 57-77. issn: 2624831X (ISSN). doi: 10.3390/iot4020004 (vid. págs. 21, 40).

Sharifi-Renani, Mohsen, Mohammad H. Mahoor y Chadd W. Clary (jul. de 2023). «BioMAT: An Open-Source Biomechanics Multi-Activity Transformer for Joint Kinematic Predictions Using Wearable Sensors». English. En: *SENSORS* 23.13. doi: 10.3390/s23135778 (vid. págs. 8, 13, 21, 38, 40).

Sharma, Abhishek y Eric Rombokas (2022). «Improving IMU-Based Prediction of Lower Limb Kinematics in Natural Environments Using Egocentric Optical Flow». English. En: *IEEE TRANSACTIONS ON NEURAL SYSTEMS AND REHABILITATION ENGINEERING* 30, págs. 699-708. issn: 1534-4320. doi: 10.1109/TNSRE.2022.3156884 (vid. págs. 21, 23, 38, 40).

Sharma, Kanchan et al. (2008). «Design optimization of MEMS comb accelerometer». En: *Department of electrical and computer engineering, University of Bridgeport, Bridgeport, CT* 6604 (vid. pág. 11).

Sharma, Sagar, Simone Sharma y Anidhya Athaiya (2017). «Activation functions in neural networks». En: *Towards Data Sci* 6.12, págs. 310-316 (vid. pág. 33).

Sherratt, Freddie, Andrew Plummer y Pejman Iravani (feb. de 2021). «Understanding LSTM Network Behaviour of IMU-Based Locomotion Mode Recognition for Applications in Prostheses and Wearables». English. En: *SENSORS* 21.4. doi: 10.3390/s21041264 (vid. págs. 21, 40).

Sherstinsky, Alex (2020). «Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network». En: *Physica D: Nonlinear Phenomena* 404, pág. 132306 (vid. págs. 36, 38).

Silva, Marcelo Guimarães, Pedro Vieira Sarmet Moreira y Henrique Martins Rocha (2017). «Development of a low cost force platform for biomechanical parameters analysis». En: *Research on Biomedical Engineering* 33, págs. 259-268 (vid. pág. 14).

Song, Seongmi, Nathaniel J. Fernandes y Andrew D. Nordin (sep. de 2023). «Characterizing Bodyweight-Supported Treadmill Walking on Land and Underwater Using Foot-Worn Inertial Measurement Units and Machine Learning for Gait Event Detection». English. En: *SENSORS* 23.18. doi: 10.3390/s23187945 (vid. págs. 21, 40).

Song, Yan-Yan y LU Ying (2015). «Decision tree methods: applications for classification and prediction». En: *Shanghai archives of psychiatry* 27.2, pág. 130 (vid. págs. 25, 26).

Sorenson, Harold W (1966). «Kalman filtering techniques». En: *Advances in control systems*. Vol. 3. Elsevier, págs. 219-292 (vid. pág. 66).

Stelzer, Andreas, Klaus Pourvoyeur y Alexander Fischer (2004). «Concept and application of LPM-a novel 3-D local position measurement system». En: *IEEE Transactions on microwave theory and techniques* 52.12, págs. 2664-2669 (vid. pág. 10).

Su, B., C. Smith y E. G. Farewik (2020). «Gait Phase Recognition Using Deep Convolutional Neural Network with Inertial Measurement Units». English. En: *Biosensors* 10.109. issn: 20796374 (ISSN). doi: 10.3390/bios10090109. PMID: 32867277 (vid. págs. 17, 21, 40).

Su, Binbin, Yi-Xing Liu y Elena M. Gutierrez-Farewik (nov. de 2021). «Locomotion Mode Transition Prediction Based on Gait-Event Identification Using Wearable Sensors and Multilayer Perceptrons». English. En: *SENSORS* 21.22. doi: 10.3390/s21227473 (vid. págs. 17, 21, 40).

Sunny, S. M. et al. (2023). «Application of Artificial Neural Network for Successful Prediction of Lower Limb Dynamics and Improvement in the Mathematical Representation of Knee Dynamics in Human Locomotion». English.

En: *Lecture Notes in Electrical Engineering* 1066 LNEE, págs. 921-932. issn: 18761100 (ISSN); 978-981994633-4 (ISBN). doi: 10.1007/978-981-99-4634-1_72 (vid. págs. 21, 40).
Szandała, Tomasz (2021). «Review and comparison of commonly used activation functions for deep neural networks». En: *Bio-inspired neurocomputing*, págs. 203-224 (vid. pág. 33).
Taborri, Juri et al. (2016). «Gait partitioning methods: A systematic review». En: *Sensors* 16.1, pág. 66 (vid. págs. 4, 5, 11, 12, 14, 16, 17, 23).
Thong, YK et al. (2002). «Dependence of inertial measurements of distance on accelerometer noise». En: *Measurement Science and Technology* 13.8, pág. 1163 (vid. pág. 11).
Tian, Yingjie, Yong Shi y Xiaohui Liu (2012). «Recent advances on support vector machines research». En: *Technological and economic development of Economy* 18.1, págs. 5-33 (vid. pág. 29).
Titterton, David y John L Weston (2004). *Strapdown inertial navigation technology*. Vol. 17. IET (vid. pág. 10).
Usubamatov, Ryspek (2019). «Physics of Gyroscope's "Antigravity Effect"». En: *Advances in Mathematical Physics* 2019, págs. 1-7 (vid. pág. 12).
– (2020). «Theory of Gyroscopic effects for rotating objects». En: *The Open Access Journal of Science and Technology*, págs. 1-1 (vid. pág. 12).
Van Eck, Nees Jan y Ludo Waltman (2007). «VOS: A new method for visualizing similarities between objects». En: *Advances in Data Analysis: Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation eV, Freie Universität Berlin, March 8–10, 2006*. Springer, págs. 299-306 (vid. pág. 42).
Vu, Huong Thi Thu et al. (nov. de 2022). «Comparison of machine learning and deep learning-based methods for locomotion mode recognition using a single inertial measurement unit». English. En: *FRONTIERS IN NEUROROBOTICS* 16. issn: 1662-5218. doi: 10.3389/fnbot.2022.923164 (vid. págs. 21, 34, 40).
Wahjudi, Fanuel y Fuchun Joseph Lin (2019). «IMU-Based Walking Workouts Recognition». English. En: 5th IEEE World Forum on Internet of Things (IEEE WF-IoT), Univ Limerick, Elect & Comp Engn Dept, Limerick, IRELAND, APR 15-19, 2019, págs. 251-256. doi: 10.1109/wf-iot.2019.8767285 (vid. págs. 21, 40).
Wan, Eric A y Rudolph Van Der Merwe (2000). «The unscented Kalman filter for nonlinear estimation». En: págs. 153-158 (vid. pág. 66).
Wang, Hua, Chris Ding y Heng Huang (2010). «Multi-label linear discriminant analysis». En: págs. 126-139 (vid. pág. 28).
Wang, Z. et al. (2021). «Terrain recognition and gait cycle prediction using IMU». English. En: *2021 IEEE International Conference on Real-Time Computing and Robotics, RCAR 2021*, págs. 602-607. issn: 978-166543678-6 (ISBN). doi: 10.1109/RCAR52367.2021.9517670 (vid. págs. 21, 29, 40).
Wei, Haochen et al. (sep. de 2023). «Gait Phase Detection Based on LSTM-CRF for Stair Ambulation». English. En: *IEEE ROBOTICS AND AUTOMATION LETTERS* 8.9, págs. 6029-6035. issn: 2377-3766. doi: 10.1109/LRA.2023.3303787 (vid. págs. 21, 38, 40).
Wei, Yuzhang y Qingsong Xu (2015). «An overview of micro-force sensing techniques». En: *Sensors and Actuators A: Physical* 234, págs. 359-374 (vid. pág. 16).
Weigand, Florian et al. (2022). «Continuous locomotion mode recognition and gait phase estimation based on a shank-mounted IMU with artificial neural networks». English. En: IEEE International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, JAPAN, OCT 23-27, 2022, págs. 12744-12751. issn: 2153-0858. doi: 10.1109/IROS47612.2022.9981154 (vid. págs. 21, 40).
West, Bruce J y Nicola Scafetta (2003). «Nonlinear dynamical model of human gait». En: *Physical review E* 67.5, pág. 051917 (vid. pág. 66).
Woodman, Oliver J (2007). *An introduction to inertial navigation*. Inf. téc. University of Cambridge, Computer Laboratory (vid. págs. 10, 11).
Xu, Shaochen et al. (jul. de 2023). «A Real-Time Gait Phase Detection Method Based on BiLSTM-Attention Model». En: págs. 1-4. issn: 2694-0604. doi: 10.1109/EMBC40787.2023.10340216 (vid. págs. 21, 40).
Yafei, Ren, Ke Xizheng y Liu Yijie (2007). «MEMS gyroscope performance estimate based on Allan Variance». En: págs. 1-260 (vid. pág. 12).
Yamashita, Rikiya et al. (2018). «Convolutional neural networks: an overview and application in radiology». En: *Insights into imaging* 9, págs. 611-629 (vid. págs. 33-35).
Yan, Lei et al. (ene. de 2020). «Walking Gait Phase Detection Based on Acceleration Signals Using Voting-Weighted Integrated Neural Network». English. En: *COMPLEXITY* 2020. issn: 1076-2787. doi: 10.1155/2020/4760297 (vid. págs. 21, 40).
Yang, L. et al. (2023). «Inertial Sensing for Lateral Walking Gait Detection and Application in Lateral Resistance Exoskeleton». English. En: *IEEE Transactions on Instrumentation and Measurement* 72.4004014. issn: 00189456 (ISSN). doi: 10.1109/TIM.2023.3265105 (vid. págs. 21, 31, 40).
Yang, P. et al. (2019). «The Research of Gait Recognition Based on High Dynamic Force Sensing Resistor». English. En: *ACM International Conference Proceeding Series*. issn: 978-145037661-7 (ISBN). doi: 10.1145/3386164.3386172 (vid. págs. 16, 21, 22, 29, 40).

Yeh, Po-Chen, Hao Duan y Tien-Kan Chung (2019). «A novel three-axial magnetic-piezoelectric MEMS AC magnetic field sensor». En: *Micromachines* 10.10, pág. 710 (vid. pág. 12).

Yi, Chunzhi et al. (ene. de 2021). «Smart healthcare-oriented online prediction of lower-limb kinematics and kinetics based on data-driven neural signal decoding». English. En: *FUTURE GENERATION COMPUTER SYSTEMS-THE INTERNATIONAL JOURNAL OF ESCIENCE* 114, págs. 96-105. issn: 0167-739X. doi: 10.1016/j.future.2020.06.015 (vid. págs. 21, 38, 40).

Yu, Shuangyue et al. (jul. de 2023). «Artificial Neural Network-Based Activities Classification, Gait Phase Estimation, and Prediction». English. En: *ANNALS OF BIOMEDICAL ENGINEERING* 51.7, págs. 1471-1484. issn: 0090-6964. doi: 10.1007/s10439-023-03151-y (vid. págs. 21, 40).

Yu, Yong et al. (2019). «A review of recurrent neural networks: LSTM cells and network architectures». En: *Neural computation* 31.7, págs. 1235-1270 (vid. págs. 36, 37).

Yuan, Meng et al. (2010). «Research of MEMS piezoresistive pressure sensor». En: 1, págs. 536-539 (vid. págs. 15, 16).

Zadeh, Lotfi Asker (1988). «Fuzzy logic». En: *Computer* 21.4, págs. 83-93 (vid. pág. 24).

Zarzycki, Krzysztof y Maciej Ławryńczuk (2021). «LSTM and GRU neural networks as models of dynamical processes used in predictive control: A comparison of models developed for two chemical reactors». En: *Sensors* 21.16, pág. 5625 (vid. págs. 36, 37).

Zhang, Rong, Christian Vogler y Dimitris Metaxas (2007). «Human gait recognition at sagittal plane». En: *Image and vision computing* 25.3, págs. 321-330 (vid. pág. 58).

Zhang, Xiaohui et al. (sep. de 2023a). «Improving Walking Assistance Efficiency in Real-World Scenarios with Soft Exosuits Using Locomotion Mode Detection». En: págs. 1-6. issn: 1945-7901. doi: 10.1109/ICORR58425.2023.10304773 (vid. págs. 21, 40).

Zhang, Xiaohui et al. (oct. de 2023b). «Real-Time Assistive Control via IMU Locomotion Mode Detection in a Soft Exosuit: An Effective Approach to Enhance Walking Metabolic Efficiency». English. En: *IEEE-ASME TRANSACTIONS ON MECHATRONICS*. issn: 1083-4435. doi: 10.1109/TMECH.2023.3322269 (vid. págs. 21, 40).

Zhang, Y. et al. (2022). «Gait Prediction and Assist Control of Lower Limb Exoskeleton Based on Inertia Measurement Unit». English. En: *5th International Conference on Intelligent Robotics and Control Engineering, IRCE 2022*, págs. 44-49. issn: 978-166546995-1 (ISBN). doi: 10.1109/IRCE55557.2022.9963096 (vid. págs. 21, 40).

Zhao, Yang et al. (2016). «Stance Phase Detection for Walking and Running Using an IMU Periodicity-based Approach». English. En: Advances in Intelligent Systems and Computing 392. Ed. por P Chung et al. 10th International Symposium on Computer Science in Sports (ISCSS), Loughborough, ENGLAND, SEP 09-11, 2015, págs. 225-232. issn: 2194-5357. doi: 10.1007/978-3-319-24560-7_29 (vid. págs. 8, 21, 25, 40).

Zhen, Tao, Jian-lei Kong y Lei Yan (oct. de 2020). «Hybrid Deep-Learning Framework Based on Gaussian Fusion of Multiple Spatiotemporal Networks for Walking Gait Phase Recognition». English. En: *COMPLEXITY* 2020. issn: 1076-2787. doi: 10.1155/2020/8672431 (vid. págs. 21, 23, 40).

Zhen, Tao, Lei Yan y Jian-lei Kong (ago. de 2020). «An Acceleration Based Fusion of Multiple Spatiotemporal Networks for Gait Phase Detection». English. En: *INTERNATIONAL JOURNAL OF ENVIRONMENTAL RESEARCH AND PUBLIC HEALTH* 17.16. doi: 10.3390/ijerph17165633 (vid. págs. 21, 23, 40).

Zhong, Zhun et al. (2020). «Random erasing data augmentation». En: 34.07, págs. 13001-13008 (vid. pág. 35).

Zhou, Zhi-Hua (2012). *Ensemble methods: foundations and algorithms*. CRC press (vid. págs. 29-31).

Zhu, Rong y Zhaoying Zhou (2004). «A real-time articulated human motion tracking using tri-axis inertial/magnetic sensors package». En: *IEEE Transactions on Neural systems and rehabilitation engineering* 12.2, págs. 295-302 (vid. pág. 13).