Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

MENCIÓN EN INGENIERÍA DEL SOFTWARE

Reconocimiento biométrico de la forma de andar mediante ponibles inteligentes usando el Modelo FMM (Frecuency Modulated Möbius)

Alumno: Guillermo Jáñez Lagüéns

Tutores: Carlos Enrique Vivaracho Pascual,

María Aránzazu Simón Hurtado

Agradecimientos

Este Trabajo de Fin de Grado no podría haberlo llevado a cabo sin el apoyo de varias personas, a las que les estaré muy agradecido por haber hecho posibles este proyecto.

A mi familia, porque, aunque no están familiarizados con el tema, siempre han estado ofreciendo su ayuda para lo que fuese necesario, de diferentes maneras, me han hecho mantener la motivación durante todo este tiempo, para poder enseñarles el resultado final.

A mis tutores del TFG, que, a pesar del tiempo transcurrido, siempre han estado disponibles para resolver las dudas que aparecían, para poder continuar con el trabajo independientemente de las circunstancias.

A mis amigos y a mi compañera de piso, que durante todo el proceso han estado presentes, pero que sobre todo en la parte final me han aguantado y siempre tenían un consejo para ayudar y animarme, o simplemente hacían que me despejase un rato.

A todos los compañeros de la carrera, que se han convertido en amigos, que desde el primer año hasta el último han conseguido hacer memorable la etapa de la universidad.

A la Universidad de Valladolid y sus profesores, que durante todo el grado me han aportado el conocimiento y los recursos necesarios para poder sacarlo todo adelante.

A todos vosotros, os estoy realmente agradecido, porque sin duda habéis sido una parte más de este trabajo, sin la que no habría salido de igual manera, gracias de nuevo.

Resumen

Con el presente trabajo se busca desarrollar un nuevo sistema de reconocimiento biométrico, a partir de otro anterior ya existente, que reconoce a los usuarios en función de su forma de andar, mediante dispositivos ponibles o wearables. A este sistema, se le va a aplicar el uso del modelo FMM o Frecuency Modulated Mobius para la extracción de características. Este modelo ha sido desarrollado para realizar el ajuste de conjuntos de datos oscilantes. A este desarrollo se le añade la dificultad de adaptarlo para poder ejecutar los experimentos de forma paralela, debido a la carga de trabajo que requieren. Finalmente, se analizan los resultados y se comparan con los de la versión anterior del sistema para comprobar cómo afecta esta modificación al funcionamiento del mismo.

Abstract

The aim of this work is to develop a new biometric recognition system, based on a previous existing one, which recognizes users based on their gait, using wearable devices. To this system, the use of the FMM or Frequency Modulated Mobius model will be applied for feature extraction. This model has been developed to perform the adjustment of oscillating data sets. This development has the added difficulty of adapting it to be able to run the experiments in parallel, due to the workload required. Finally, the results are analyzed and compared with those of the previous version of the system to check how this modification affects the performance of the system.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

La existencia de gran cantidad de trabajos de investigación y desarrollo de diferentes tipos de sistemas en el ámbito universitario hace que la probabilidad de que dos de estos trabajos puedan combinarse para mejorar el funcionamiento de alguno de ellos crezca, y es precisamente esto lo que se va a plantear con este trabajo.

Por un lado, gracias al auge de los dispositivos ponibles o wearables, como las pulseras de actividad, que cuentan con una gran cantidad de sensores para tomar diferentes medidas del usuario que los utiliza y, por otro lado, la posible aplicación de la inteligencia artificial como mejora de una situación cotidiana hoy en día como es el reconocimiento del usuario que utiliza un dispositivo móvil mediante biometría (desbloqueo facial o mediante huella dactilar), dio la idea para crear un sistema de reconocimiento biométrico de la forma de andar del usuario mediante dos trabajos diferentes [27] y [28].

El otro trabajo que se combinará, también era una continuación de otros proyectos académicos [21], [22], [25] y [26], que crearon un modelo de ajuste que, dados los valores de una función oscilatoria, permite generar un ajuste para esos datos, independientemente de que este conjunto de datos sea simétrico o asimétrico. Este modelo de ajuste, llamado Frecuency Modulated Mobius, o FMM, se usó como base para desarrollar una biblioteca pública, y poder aplicarlo en diferentes ámbitos. En concreto, en [26] se trató de resolver el problema de ajustar un electrocardiograma haciendo uso de este modelo, y se obtuvieron buenos resultados.

Con estos dos proyectos tan interesantes desarrollados, se planteó la posibilidad de tratar de mejorar el sistema de reconocimiento biométrico utilizando la biblioteca creada para usar el modelo FMM. Ya que, para entrenar el sistema de reconocimiento biométrico se usaba un vector de características que contenía un conjunto de valores estadísticos (máximo, mínimo, media, etc.) de los datos que se generaban al andar, y estas señales eran ondas oscilatorias, se vio la posibilidad de aplicar el ajuste del modelo FMM a estas ondas para generar los vectores de características y comprobar si el sistema de reconocimiento biométrico mejoraba con esta modificación o no.

1.1 Motivación

La motivación principal que presentó este trabajo fue la de poder entrar en contacto con un sistema autónomo que, tras una fase de entrenamiento, pudiese tomar decisiones basadas en la información que había recogido en ese entrenamiento. La relación de este trabajo con la inteligencia artificial y el desarrollo de sistemas autónomos fue sin duda gran parte de la motivación para comenzar, pues en los últimos años del grado fue un tema que empezó a interesarme, y este trabajo permitía conocer algo más en profundidad este tema.

Por otro lado, pese a pertenecer a la mención de ingeniería de software, en la que gran parte de los trabajos de fin de grado tienen como tema el desarrollo de una aplicación, pasando por todas sus fases, yo decidí llevar a cabo otro tipo de trabajo, un trabajo en el que, por supuesto, habría que desarrollar diferentes partes de un sistema, pero la parte principal era la de experimentación y análisis de los resultados obtenidos tras realizar el desarrollo planteado.

El hecho de que fuese un trabajo de naturaleza experimental, y no tan centrado en la ingeniería de software, fue la segunda razón que me hizo decidirme para llevarlo a cabo, pues durante los años del grado, este tipo de trabajos no habían sido tan numerosos y quería probar a realizar un trabajo más grande de este tipo.

1.2 Objetivos

Como ya se ha comentado, este trabajo es continuación de otros anteriores en los que se desarrolló un sistema de reconocimiento biométrico de personas a partir de la forma de andar mediante ponibles inteligentes [27] y [28]. En esos trabajos se capturaron las señales de los sensores acelerómetro y giróscopo, y a partir de esas series temporales se extrajeron las características en el dominio del tiempo y de la frecuencia. Se ha querido en este TFG aplicar una propuesta novedosa.

El objetivo general de este trabajo consiste en desarrollar un sistema de reconocimiento biométrico a partir del anterior, investigando ahora el uso del Modelo FMM (Frecuency Modulated Möbius) para la extracción de características, y comparando su rendimiento con el obtenido a partir del otro sistema.

Para lograr ese objetivo general, se han establecido una serie de objetivos específicos, que son los siguientes:

- Modificar el sistema de reconocimiento biométrico existente incluyendo el uso de la biblioteca que permite el ajuste de modelos FMM para modificar los vectores de características que el sistema utiliza. Esta biblioteca pública es la que se desarrolló en el otro trabajo en el que se apoya este [21].
- Realizar los experimentos utilizando el nuevo sistema aplicado a una base de datos recogidos en trabajos anteriores.
- Comparar los resultados obtenidos con los anteriores.
- Modificar los parámetros del sistema para buscar configuraciones que funcionen mejor.

1.3 Estructura de la memoria

Este trabajo, contando con el capítulo actual de Introducción se ha estructurado de la siguiente forma:

2. Entorno teórico: Este capítulo busca establecer las bases teóricas necesarias para comprender el trabajo realizado. Se explican varios conceptos teóricos mencionados en el trabajo y posteriormente se hace un resumen de los dos trabajos previos utilizados como base para el actual.
3. Tecnologías, Metodología y Planificación: Este capítulo agrupa diferentes apartados en los que se explican la metodología de trabajo que se va a aplicar, las tecnologías utilizadas, la planificación de las tareas, y su posterior modificación al llevar a cabo el trabajo.
4. Desarrollo: En este capítulo se explican las fases del proceso de modificación con sus diferentes etapas.
5. Resultados: Este capítulo se usa como resumen de los resultados obtenidos, divididos según los clasificadores utilizados, mostrando estos datos y sacando las conclusiones oportunas sobre ellos.
6. Conclusiones y Trabajo Futuro: Se describen los objetivos cumplidos junto con la valoración personal del trabajo, y algunas líneas de trabajo que se podrían abordar en un futuro.

2 Entorno teórico

Para realizar este trabajo se parte de la existencia de otros dos proyectos realizados que se explicarán más en profundidad a continuación. Un sistema biométrico que permite identificar al usuario mediante los sensores de una pulsera de actividad, registrando su forma de andar [28]. Y una biblioteca del programa R que, partiendo de los puntos de una función con sus valores temporales, devuelve unos parámetros que permiten representar la gráfica que se ajusta a los puntos de esa función utilizando el modelo Frecuency Modulated Möbius [21].

2.1 Conceptos teóricos

En este apartado se van a explicar los conceptos que se consideran necesarios para comprender mejor el trabajo.

Paralelización

La computación paralela, hace referencia al modo de ejecutar un programa. Esta busca dividir un programa en secciones de código más pequeñas, que puedan ejecutarse simultáneamente, para ahorrar tiempo y recursos de la máquina que lo lleve a cabo. En concreto, este proceso por el que se transforma un programa para convertirlo en un programa que se pueda ejecutar en paralelo es la paralelización. Estas ejecuciones en paralelo son posibles gracias a las unidades de cómputo que tienen las máquinas actuales. En un programa secuencial (los que no están paralelizados) cada instrucción del programa se ejecuta, y cuando termina se pasa a ejecutar la siguiente, y así hasta llegar al final del programa, de manera que la unidad de cómputo solo se utiliza para ejecutar una instrucción en cada instante. Con la paralelización se consigue ejecutar varias instrucciones al mismo tiempo en las diferentes unidades de cómputo disponibles, y reducir así el tiempo de ejecución necesario para llevar cabo el programa inicial.

Biometría

La biometría es el estudio estadístico de los fenómenos o procesos biológicos. Su uso más extendido es el de reconocer a un determinado usuario mediante alguna característica que tenga este como, por ejemplo, la huella dactilar, el reconocimiento facial, la voz o la forma de andar.

Los sistemas biométricos se pueden clasificar según el tipo de características que reconozcan:

- Tipo fisiológico: cuando se reconocen características físicas del usuario, o algún rasgo de su cuerpo, que son únicos de cada persona. Las más comunes son la huella dactilar, el iris del ojo, la retina o el rostro.
- Tipo conductual: cuando se reconocen patrones de comportamiento, la manera en la que cada usuario realiza diferentes acciones, y las características que se obtienen al realizarlo. Algunos ejemplos serían la firma o la forma de andar del usuario.

Estas características deben cumplir ciertos parámetros, ya que deben estar presentes en todos los seres humanos y no variar con el tiempo, esta característica se debe poder medir cuantitativamente y debe ser única para cada individuo, ya que no puede ser posible que dos individuos tengan el mismo valor de esa característica.

Por otro lado, la biometría se puede utilizar para dos acciones claras, la de identificación o la de verificación [24].

- Identificación: cuando el propósito del sistema biométrico es reconocer al usuario que está utilizándolo, entre el conjunto de usuarios que estén registrados en la base de datos del sistema. Para

ello recibe como entrada alguna de sus características (fisiológicas o conductuales) y las compara con la base de datos para encontrar la coincidencia y completar la identificación.

- Verificación: también llamada autenticación, cuando el propósito del sistema biométrico es confirmar o no, si el usuario que está utilizando el sistema es el usuario que está registrado para ello

2.2 Sistema biométrico de referencia

Este sistema biométrico ha sido desarrollado por Irene Salvador Ortega como trabajo de fin de máster en el Máster Universitario en Inteligencia de Negocio y Big Data en Entornos Seguros de la Universidad de Valladolid. Este trabajo se puede encontrar en [28].

2.2.1 Sensores y medición

La biometría para la verificación de usuarios se ha aplicado de diferentes modos en función de las capacidades de la tecnología existente hasta ese momento.

A la hora de usar la forma de andar para el reconocimiento biométrico, hay que tener en cuenta la tecnología que se usa para medir este proceso. Como se menciona en 75[28], inicialmente podría surgir la idea de usar cámaras para grabar al usuario y reconocer ciertos movimientos con lo que se entrenaría un sistema para reconocer estas características en diferentes individuos.

Se podrían usar otros sensores como son los de presión y fuerza y medir cada pisada del usuario, aunque por sí solos podrían ser más fácil de falsificar que otras opciones.

Hoy en día, con los smartphones y wearables actuales, que son dispositivos portátiles que llevamos encima continuamente, junto con la cantidad de sensores con los que cuentan, los convierte en una gran herramienta para realizar estas mediciones sobre la forma de andar de un individuo. Los sensores que tienen estos dispositivos que se consideran más adecuados para este fin son el acelerómetro y el giróscopo ya que miden el movimiento del dispositivo en las tres dimensiones, junto con la aceleración que sufren en cada uno de los ejes. Realizando esta medición en la muñeca, que es donde se sitúan estos dispositivos ponibles, se obtiene un buen método de recogida de datos para poder caracterizar la forma de andar.

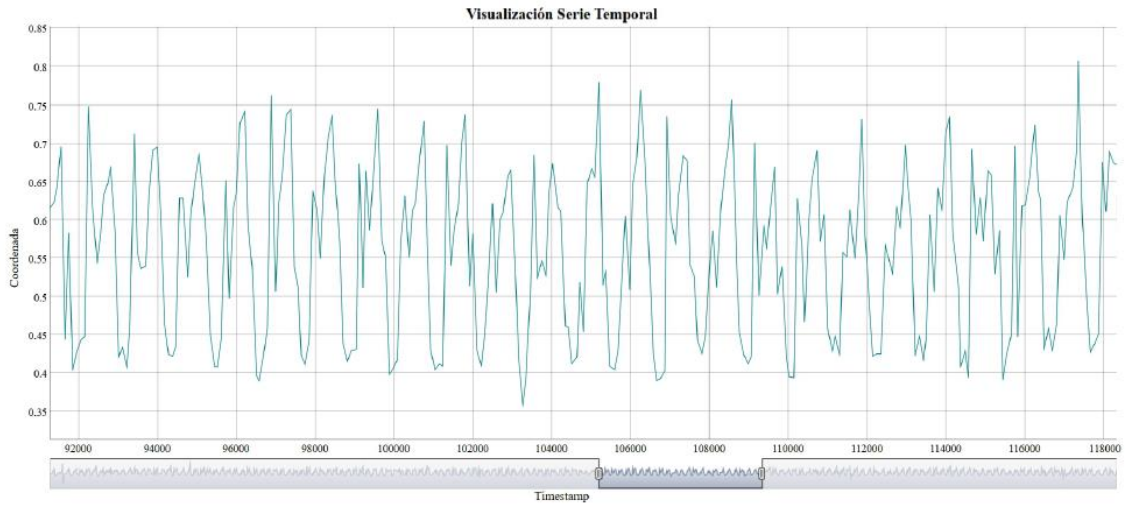

Figura 2.1 Ejemplo de señal periódica

Contando con estos sensores lo siguiente que se plantea es qué parte del movimiento se monitoriza. Viendo la forma que tienen las señales periódicas (Figura 2.1) de los sensores durante la marcha, y usando diferentes trabajos ([18], [29]), la conclusión es que lo mejor es dividir la señal que se obtiene de los sensores en ciclos.

Estos ciclos están formados por un paso de la pierna izquierda y un paso de la pierna derecha, pues esto genera señales periódicas, que se repiten con el tiempo durante la marcha y por lo tanto son similares entre ellas.

Estos ciclos comienzan desde que se despega un pie del suelo, hasta que se va a volver a despegar ese mismo pie del suelo (Figura 2.2).

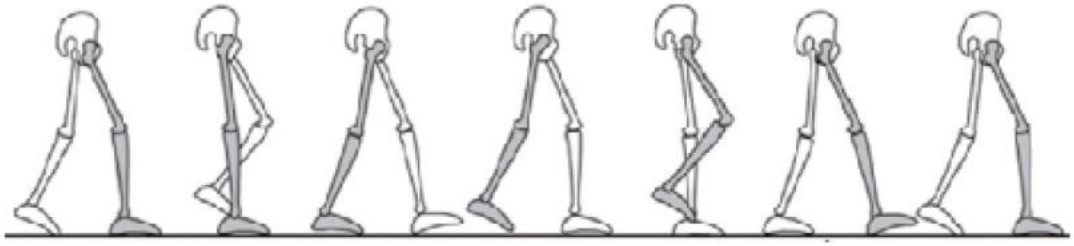

Figura 2.2 Esquema de un ciclo de la marcha

2.2.2 Base de datos biométrica

En este apartado se va a explicar la composición de los datos biométricos capturados que se van a utilizar en este trabajo.

Los datos fueron capturados en el trabajo [28], usando dos *wearables* o ponibles: una pulsera de Microsoft, a la que haremos referencia más adelante como Micro y un reloj de Motorola, al que nos referiremos como Moto. Los datos capturados cuentan con las siguientes características:

- Recorrido andando con una duración de tiempo de 5 minutos aproximadamente.
- Se controla la mano de uso del dispositivo y la orientación de los mismos.
- Se capturan datos en 2 sesiones en días diferentes. Para poder estudiar la variación a lo largo del tiempo del patrón biométrico se adquirieron dos sesiones distintas, con una separación mínima entre ambas de dos semanas.
- Cada día (sesión) se realizan 6 recorridos: 3 con cada dispositivo, los dos primeros utilizando la muñeca de la mano de uso habitual del reloj o portadora, y el último utilizando la opuesta, para comprobar si la mano en que se lleva el dispositivo modifica los resultados.

Para cada usuario al que se le toman los datos, se almacenan los siguientes metadatos, que se pueden ver en la Figura 2.3.

El número asignado al usuario, el sexo, el rango de edad, la mano dominante, la mano portadora y el número de muestras o datos que se han recogido

Usuario	Sexo	Edad	Mano Dominante	Mano Portadora	Nº de datos
usuario1	Hombre	27-45	Derecha	Izquierda	6
usuario2	Hombre	19-26	Derecha	Izquierda	6
usuario3	Hombre	27-45	Derecha	Izquierda	6
usuario4	Hombre	19-26	Derecha	Izquierda	6
usuario5	Hombre	19-26	Izquierda	Derecha	6
usuario6	Hombre	27-45	Derecha	Izquierda	6
usuario7	Hombre	19-26	Derecha	Izquierda	6
usuario8	Mujer	46-65	Derecha	Izquierda	6
usuario9	Mujer	19-26	Derecha	Izquierda	6
usuario10	Mujer	46-65	Derecha	Izquierda	6
usuario11	Hombre	27-45	Derecha	Izquierda	6
usuario12	Hombre	46-65	Derecha	Izquierda	6
usuario13	Mujer	46-65	Derecha	Izquierda	6
usuario14	Hombre	46-65	Derecha	Izquierda	6
usuario15	Hombre	46-65	Izquierda	Izquierda	6
usuario16	Hombre	27-45	Derecha	Izquierda	6
usuario17	Mujer	46-65	Derecha	Izquierda	6
usuario18	Hombre	46-65	Derecha	Izquierda	6
usuario19	Hombre	27-45	Derecha	Izquierda	6
usuario20	Hombre	27-45	Derecha	Izquierda	6
usuario21	Hombre	27-45	Derecha	Izquierda	6
usuario22	Mujer	19-26	Derecha	Izquierda	6
usuario23	Mujer	19-26	Derecha	Izquierda	6
usuario24	Hombre	19-26	Derecha	Izquierda	6
TOTAL MUESTRAS DE DATOS DISPONIBLES					144

Figura 2.3 Metadatos de los usuarios en la Base de Datos de [28]

De manera resumida, al realizar cada recorrido, se va guardando en la Base de Datos la siguiente información:

- Identificador del usuario.
- International Mobile Equipment Identity (IMEI) del teléfono móvil utilizado para la adquisición de los datos. El IMEI es un código que identifica al aparato de forma exclusiva a nivel mundial.
- Dispositivo que se está utilizando: pulsera inteligente Microsoft (Micro) o reloj inteligente Motorola (Moto).
- Tipo de sensor al que pertenece el dato: acelerómetro (ACC) o giróscopo (GYR).
- Timestamp: contiene tanto la fecha, como la hora con una precisión en milisegundos.
- Las coordenadas X, Y y Z del sensor indicado.
- Identificador del usuario.
- Número de la tarea, la sesión y la muestra para distinguir entre las diferentes tomas de datos del mismo usuario.

Para cada toma de datos se genera un fichero .csv con los datos que se van a usar en el sistema biométrico para generar los vectores de características, y esos son:

- El tiempo relativo, que refleja el tiempo que ha pasado desde la captura de los valores de X, Y y Z actuales y la anterior medición.
- Las 3 componentes del sensor, que son X, movimiento hacia la izquierda o derecha; Y, movimiento hacia adelante o hacia atrás; Z, movimiento hacia arriba o hacia abajo.

2.2.3 Sistema existente

A continuación, se describirá brevemente el sistema biométrico ya creado que se va a modificar en este trabajo, explicando primero las partes que lo forman (Figura 2.4Figura 2.4)

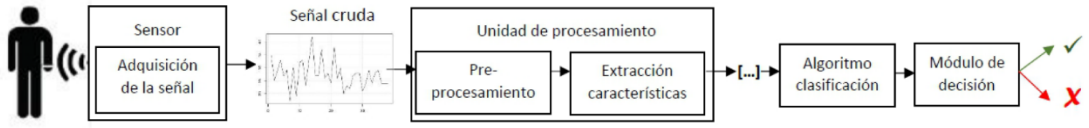

Figura 2.4 Esquema del camino que siguen los datos en el sistema

Sensores

Para los dos dispositivos ponibles, los sensores inerciales utilizados fueron el acelerómetro y el giroscopio, que devuelven en cada instante de tiempo (cada 100 milisegundos aproximadamente) valores de sus medidas en cada una de las tres dimensiones X, Y y Z.

Ventaneo de las muestras de datos

Como se ha mencionado en el apartado 2.2.1, la muestra de datos obtenida se divide en ciclos de la marcha, ya que la señal en estos ciclos es periódica y más fácil de utilizar. Para extraer estos ciclos normalmente se usan algoritmos basados en la localización de picos en la señal, pero estos algoritmos, en señales reales, que tienen zonas con ruido, introducen muchos errores. Por eso, se usó una técnica diferente para dividir la señal en ciclos:

- Se calcula el coeficiente de autocorrelación de la señal. Este valor da una idea de la correlación entre los datos de la señal, sería parecido a dividirla en porciones y obtener el grado de similitud que tienen estas entre sí. Lo que se busca es la división en porciones que se puede hacer de la señal, que genere el mayor grado de correlación (usando el coeficiente de autocorrelación), y así, con estas divisiones de la señal, conseguimos la duración que llamaremos “normal” de un ciclo en esa señal.
- Se divide la señal en intervalos de duración igual al periodo en el que la autocorrelación alcanza su máximo. Cada uno de esos intervalos o conjuntos de muestras será un ciclo.

Sin embargo, un ciclo por sí solo no es suficientemente representativo de la forma de andar del individuo como para poder usarlo. Por eso se decidió realizar el ‘ventaneo’ de la señal. Este proceso agrupa varios de estos ciclos en ventanas o marcos temporales, que se convierten en la cantidad mínima usada tanto en el sistema de limpieza como en la parte de extracción de características para su posterior procesamiento y uso en el reconocimiento del usuario.

En los estudios posteriores se analizó la dependencia del rendimiento del sistema con respecto al tamaño de la ventana (número de ciclos), llegando a la conclusión de que un tamaño óptimo estaba entre 6 y 10 ciclos. El sistema propuesto en [28] usaba 8, por lo que será el número de partida en este trabajo.

Pre-Procesamiento

Una vez que se cuenta con los datos de la señal de los sensores durante la sesión de captura de datos, se le aplica un conjunto de procedimientos, cuyo fin es limpiar esta señal y realzarla para mejorar su rendimiento en las siguientes etapas.

Entre los problemas hallados al revisar los datos se encontraron los siguientes:

- Instantes iniciales y finales en los que no se recogían datos mientras se establecía o terminaba la conexión con el dispositivo móvil (Figura 2.5**¡Error! No se encuentra el origen de la referencia.** y

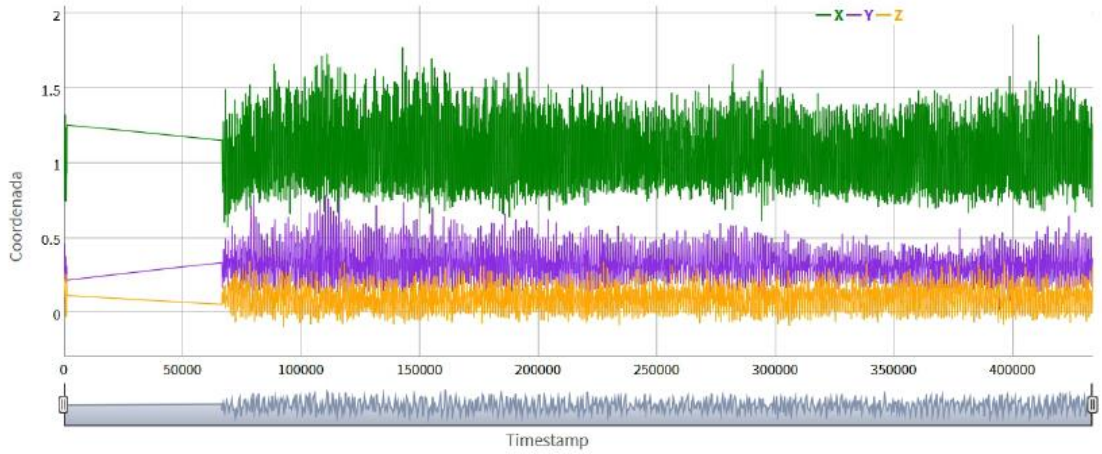

Figura 2.5 Datos crudos con desconexión en instantes iniciales.

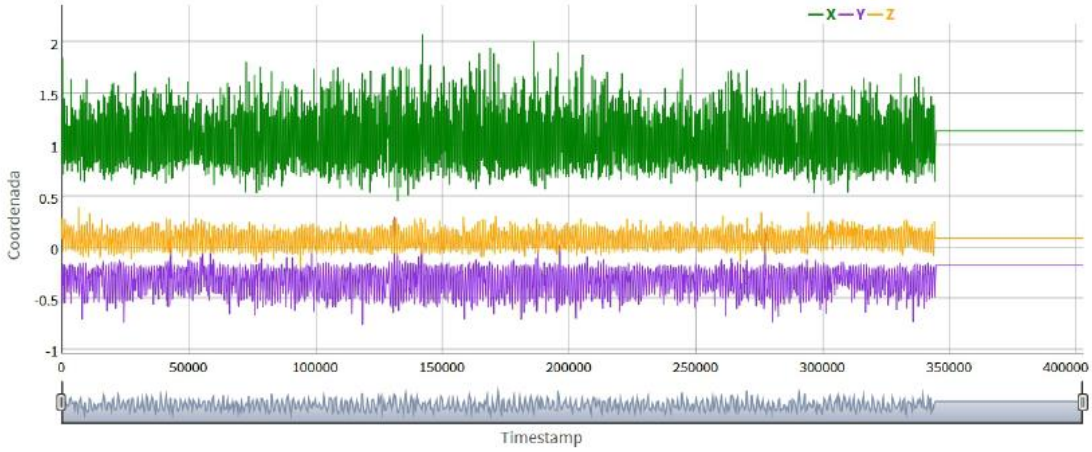

Figura 2.6 Datos crudos con desconexión en instantes finales.

  Figura 2.6**¡Error! No se encuentra el origen de la referencia.**).
- Intervalos en medio de la muestra en los que la conexión falla y no se recogen datos (Figura 2.7).
- Señales cuya duración es menor de lo esperado por algún cierre forzoso de la aplicación (Figura 2.8)

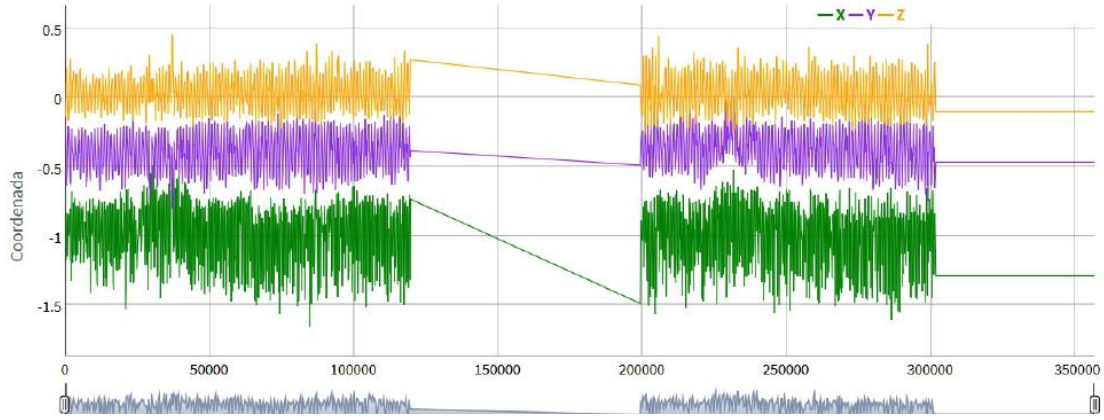

Figura 2.7 Datos crudos con desconexión en instantes intermedios.

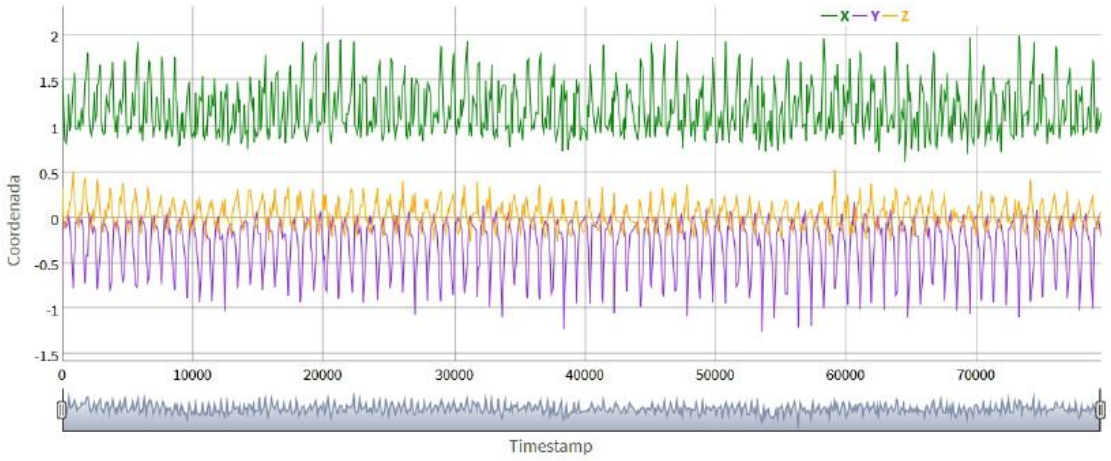

Figura 2.8 Recogida de datos crudos con una duración menor.

- Conjunto de datos desplazados en el tiempo, sin razón aparente (Figura 2.9).

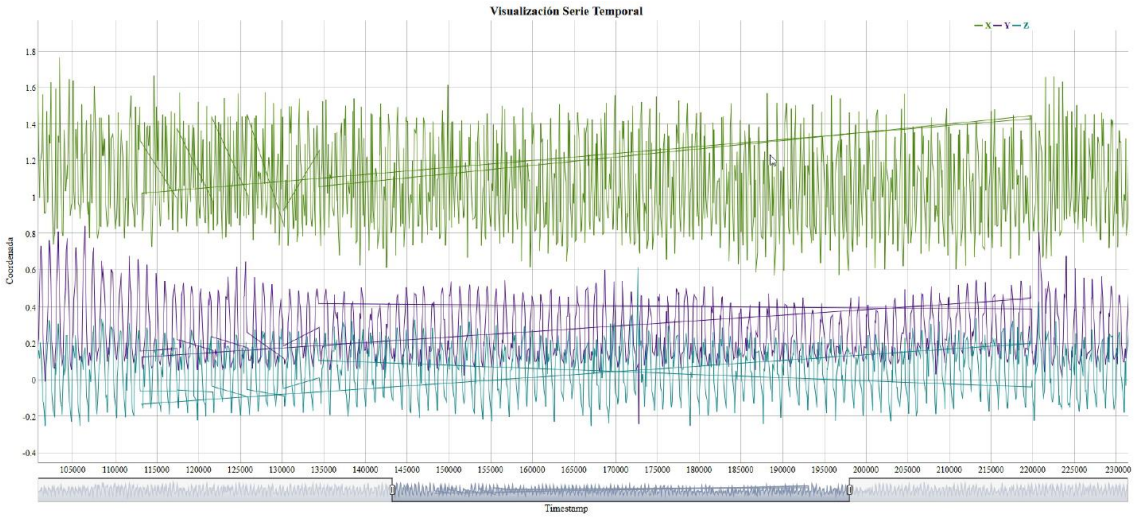

Figura 2.9 Datos crudos con valores desplazados en el tiempo.

Partiendo de la observación de estos problemas, se automatizó el proceso de limpieza de los datos para la eliminación de las zonas sin señal.

Para llevar a cabo esta automatización se utilizan cuatro valores a los que se les da un umbral de aceptación (que depende del valor medio de los datos de la muestra), de modo que, si alguno de estos cuatro valores de la ventana está fuera del límite del umbral, toda ella se invalida y no se utilizará en ninguna fase posterior.

Los cuatro valores utilizados se describen a continuación:

- El índice de correlación. Mide la periodicidad de la señal. Este índice, que puede adquirir valores entre -1 y 1, indica el grado de correlación de sus valores. El 0 indica que no hay ninguna correlación entre los valores (lo que consideramos ruido en nuestras muestras, los valores que no representan la muestra correctamente). Mientras que el 1 y el -1 indican que los valores tienen una correlación y anticorrelación perfecta respectivamente. Se rechaza la ventana si su índice de correlación es menor al 25% de la media de la muestra.
- Tiempo de adquisición entre dos datos consecutivos. Intenta captar las pérdidas de conexión. Se mide el tiempo que pasa entre cada una de las mediciones que realizan los sensores y si este valor es más alto de 550 ms se rechaza.

- Cruces por el valor medio de la señal. Indica si la señal es plana (cruza poco la línea media) o tiene un comportamiento más irregular (cruza frecuentemente la línea media) Se toma como valor medio de la señal el 0, y si el número de veces que la señal de la ventana lo atraviesa es menor al 25% de la media de la muestra, se rechaza.
- La energía de la señal: Nos da información acerca de la amplitud media de los datos de una ventana: normalmente las zonas de baja energía están asociadas a zonas sin señal (las que no nos interesan) y a la inversa. Si este valor es inferior al 10% del valor medio de la muestra, la ventana se rechaza.

Los desplazamientos de secciones de conjuntos de muestras ocurrieron en muy pocas muestras. Este problema se solucionó "a mano" moviendo los conjuntos de puntos desplazados en el tiempo, a su posición correcta dentro de la muestra.

Para ver en más detalle esta automatización y pre-procesamiento diseñados, así como los razonamientos llevados a cabo para su configuración, se puede consultar el apartado 4.3 de [28]

Extracción de características

Tras contar con las muestras de datos sin errores, se extrae de esta el conjunto de características que permitan caracterizar a cada individuo y diferenciarlo del resto. Para ello, de nuevo se procesaron los datos para adaptarlos y así permitir una extracción de características más precisa.

Los siguientes procesos se aplicaron a los datos para realizar el tratamiento previo a la extracción:

- Normalización del periodo: Ya que los intervalos de tiempo que separan dos datos consecutivos de las muestras son variables, se remuestrea la señal estableciendo un periodo [14] fijo entre datos de 83,3 ms. Este valor se obtuvo tras consultar bibliografía sobre el tema y tras realizar un análisis frecuencial de la señal.
- Normalización de la amplitud: Se aplica por la necesidad de tener los datos en una escala común.
- Filtrado: Con el fin de suavizar la señal, se comprobó que la media móvil ponderada (WMA) cumplía esta misión.

$$WMA = \frac{D_{t-1} + D_t + D_{t+1}}{3}$$

  Este valor se calcula usando los datos adquiridos en el instante t (D_t), en el instante inmediatamente anterior (D_{t-1}) y en el inmediatamente posterior (D_{t+1}).

Tras aplicar los procedimientos, se pasa a la extracción de las características como tal. Estas características se pueden extraer en el dominio del tiempo, o en el de la frecuencia:

- Dominio del tiempo: Para cada coordenada (valores X, Y y Z) se divide la muestra en ventanas que se superponen y, para cada ventana, se extraen varias características: media, mediana, máximo, mínimo, desviación estándar, rango máximo (máximo-mínimo), curtosis (cómo de achatada o apuntada está la función en comparación con una distribución normal), percentil 25, percentil 75, coeficiente de asimetría (grado de simetría o asimetría tomando al eje como la paralela al eje de ordenadas que pasa por la media de la distribución), energía (valor obtenido en conjunto con las tres coordenadas y da una idea de la posición tridimensional), valor máximo de la autocorrelación (representa la relación de cada coordenada consigo misma) y la relación media de los componentes XY, XZ y YZ (usando la media de todos los cocientes de las coordenadas seleccionadas).
  En el proyecto se experimentó utilizando estas características de cada una de las coordenadas (X, Y, Z) y también calculando el módulo de las tres coordenadas y obteniendo las características especificadas en el párrafo anterior para ese conjunto de datos.

- Dominio de la frecuencia: También divide la muestra en ventanas, aplica la transformada de Fourier para convertir la señal al dominio de la frecuencia y se utiliza únicamente el módulo de los valores (números complejos) obtenidos. Este módulo es la amplitud de los componentes de frecuencia de la señal.

  De estos datos se extraen las mismas características que en el dominio del tiempo, excepto el máximo y el mínimo, y se añaden tres nuevos valores:

  - Primera y segunda amplitud dominante, que son los dos valores más altos obtenidos entre las amplitudes resultantes del Análisis de la transformada de Fourier en cada una de las componentes de los datos.
  - Primera y segunda frecuencia dominante, que son los dos valores de la frecuencia correspondientes a los dos puntos donde se consiguen las amplitudes anteriores.
  - Área bajo la curva de Fourier (AUC) utilizando interpolación de splines.

Clasificación

Una vez obtenido el vector de características para cada ventana de datos, se debería pasar este vector con las características al clasificador seleccionado para que le asigne la puntuación determinada al conjunto de datos y así, si esta puntuación supera un umbral determinado, considerar al usuario como el usuario propietario, o en caso contrario como un impostor.

En el trabajo usado como referencia [28] se comparó el rendimiento de diferentes clasificadores, tanto supervisados como no supervisados. La lista de clasificadores es la siguiente: árboles de decisión, bagging, k-vecinos (KNN), random forest, reglas RIPPER, naive bayes, máquinas de vectores soporte (SVM) y perceptrón multicapa.

Para más información sobre cada uno de estos clasificadores se puede consultar el apartado 5.6 de [28].

Tras la experimentación se comprobó que el clasificador que mejor funcionaba era el SVM.

Post-procesamiento

Teniendo la puntuación que le asigna cada clasificador a cada vector de características de cada una de las ventanas de la muestra, se observó que estas ventanas se trataban de forma independiente unas de otras, por lo que no mantenían la relación temporal que en realidad sí que tienen.

Por ello se decidió realizar la fusión de ventanas tras la etapa de los clasificadores. Para ello, se unifican las puntuaciones de varias ventanas de la misma muestra que sean consecutivas en el tiempo, y se obtiene así una puntuación para el conjunto de varias puntuaciones de varias de las ventanas.

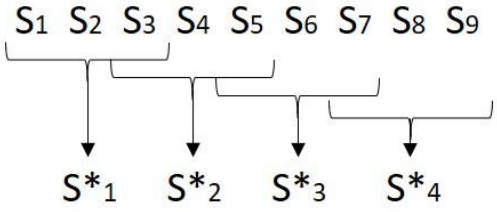

*Figura 2.10 Ejemplo del post-procesamiento realizado, siendo 'S i' la salida del clasificador para la ventana i de la muestra de prueba, y 'S*n' la puntuación fusionada de varias ventanas.*

En la Figura 2.10 se puede apreciar la fusión de 3 ventanas con un solapamiento de 1 ventana, pues en este caso el resultado de la ventana de datos número 3, se utiliza para calcular la puntuación fusionada de las tres primeras ventanas (S1, S2 y S3) pero también para la segunda puntuación fusionada (S3, S4 y S5).

Si el solapamiento fuese de dos ventanas, el valor de S*1 se obtendría usando las puntuaciones S1, S2 y S3, y no se parecían cambios, pero para obtener el valor de S*2, se usarían las puntuaciones S2, S3 y S4, para la puntuación S*3 se usarían S3, S4 y S5, y así hasta fusionar todas las ventanas.

2.3 Modelo FMM

El trabajo que vamos a usar como referencia fue desarrollado por Adrián Lamela Pérez como trabajo de fin de máster en el Máster Universitario en Inteligencia de Negocio y Big Data en Entornos Seguros de la Universidad de Valladolid. Este trabajo se puede encontrar en [21].

El objetivo del trabajo consistió en poder dar una solución más sencilla que las existentes al problema del ajuste de una gráfica parametrizada al conjunto de datos de un electrocardiograma, pues la mayoría de estas soluciones requieren un gran coste computacional o bien no tienen unos resultados lo suficientemente satisfactorios. Para ello, se utilizó un modelo creado por el grupo de investigación, en el que el autor del trabajo participó, llamado Frecuency Modulated Möbius, o FMM [26]. Posteriormente, se creó una biblioteca en el lenguaje de programación R que permitiría a los usuarios obtener los parámetros que representan la gráfica que mejor se ajusta al conjunto de datos dados introducido.

2.3.1 El modelo

Para resolver el problema que se planteaba inicialmente, un modelo muy extendido para ajustar patrones cíclicos era el Cosinor, que utiliza la fase y la amplitud de los datos para dar un ajuste. Sin embargo, este modelo resultaba insuficiente en ciertos conjuntos de datos, en concreto en los que mostraban un mayor grado de asimetría.

Por ello, se decidió aplicar otro modelo, el Frecuency Modulated Möbius, desarrollado por el Grupo de

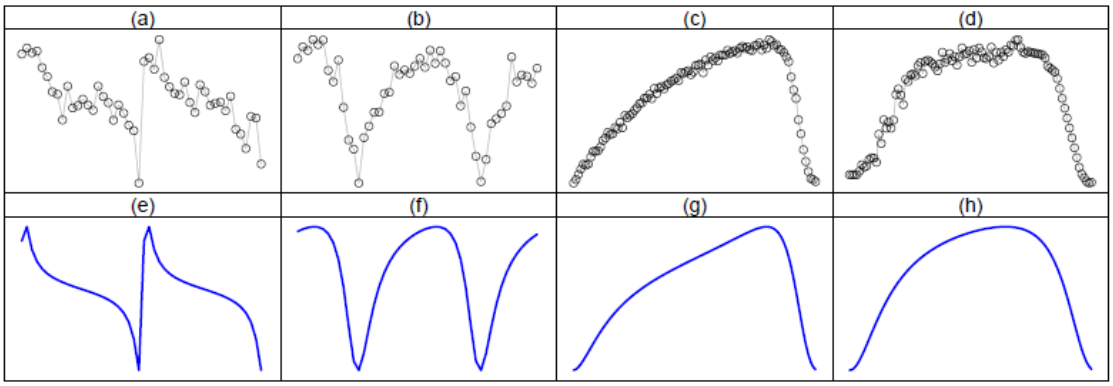

Figura 2.11 Ejemplos de expresiones de genes con dos períodos (a,b), y patrones de luz emitidos por estrellas (c,d), junto con el ajuste producido por el modelo FMM para dichos datos.

Investigación Reconocido "Inferencia con Restricciones", que se explica en más detalle en [26]Este modelo, añadiendo únicamente dos parámetros más que el modelo Cosinor, permite incluir estas formas asimétricas gracias a la utilización del enlace Möbius, en lugar del coseno. Un ejemplo de estos ajustes se puede ver en la Figura 2.11.

Los cinco parámetros que utiliza son:

- M: el intercept del modelo, donde se cruza la gráfica con el eje X.
- A: la amplitud del modelo.
- α: es el parámetro de fase.
- β: define la asimetría del conjunto.
- Ω: define el apuntamiento de la señal. Si es 0 sería una señal completamente apuntada, mientras que si es 1 sería una señal sinusoidal (Figura 2.12).

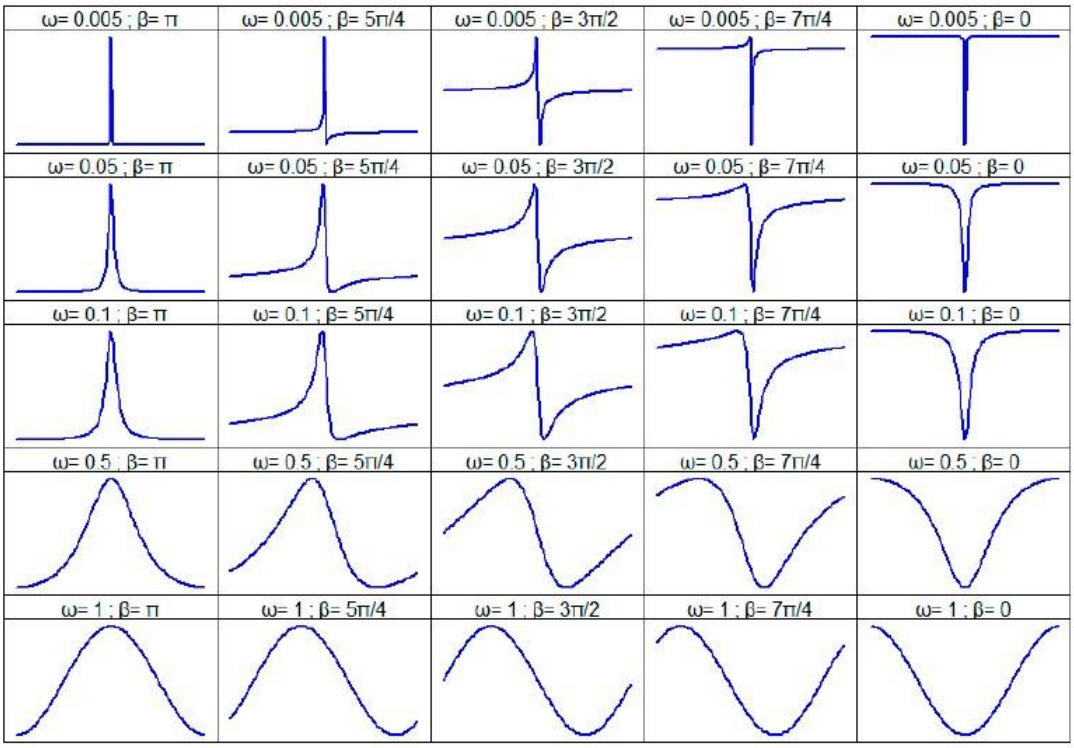

Figura 2.12 Influencia de los parámetros β y ω en un modelo FMM con M = 0, A = 1, α = 0

Con este modelo y el código en R que se creó para utilizarlo, se podían ajustar conjuntos de datos que únicamente contasen con un máximo y un mínimo. Por lo tanto, para ajustar conjuntos de datos más complejos, como por ejemplo el de un electrocardiograma, con varios valles y picos, era necesario extender este modelo.

2.3.2 El modelo multicomponente

Para resolver este nuevo problema, el autor del trabajo creó un modelo FMM multicomponente, al que se le podía indicar ahora el número de ondas componentes (aquellas formadas solamente por un máximo y mínimo) que forman el conjunto de datos, junto con el número máximo de iteraciones en las que encontrar el resultado del ajuste, mediante la técnica del backfitting [1].

El proceso simplificado es el siguiente: Se obtienen los valores de un modelo FMM que ajuste parte del conjunto, tras esto se deja de tener en cuenta esta parte de los datos ya ajustada y se busca un nuevo modelo FMM que ajuste el conjunto de datos restante que está sin ajustar. Se repite este proceso hasta tener los n componentes que se hayan indicado ajustados. Una vez se cuenta con este ajuste parcial, se va refinando poco a poco, hasta que se cumpla alguno de los requisitos de parada, o se llegue al número máximo de iteraciones

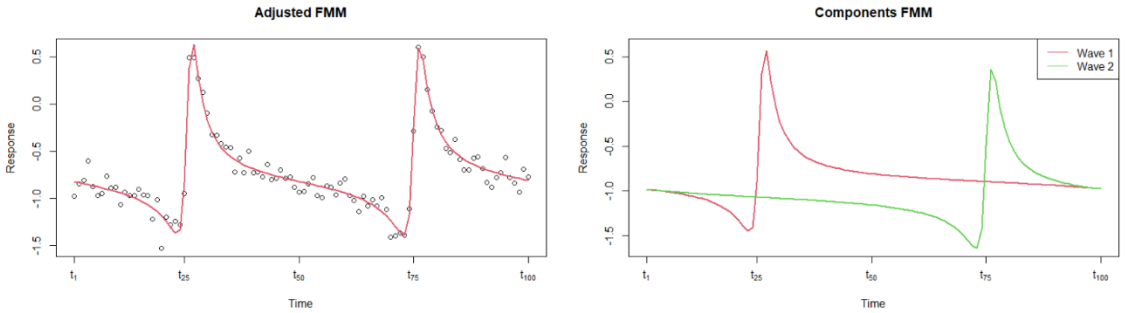

Figura 2.13 Ajuste final usando la función FMM con dos componentes

especificado previamente. Se puede observar un ejemplo de este ajuste en la Figura 2.13.

También se vio la necesidad, y se incluyó la posibilidad de añadir restricciones sobre los parámetros β y ω del ajuste de datos que se busca obtener, de manera que se pueda indicar si las ondas de los diferentes componentes son similares en cuanto al nivel de asimetría y apuntamiento y así obtener un ajuste mucho más preciso.

2.3.3 Biblioteca en R

Tras todo el análisis del problema y la toma de decisiones sobre los modelos y operaciones a implementar, se creó la biblioteca en R que permitiese poner en uso este trabajo. Para ello, se crearon tres funciones principales, según el tipo de ajuste que se necesite:

- Ajuste monocomponente
- Ajuste multicomponente
- Ajuste multicomponente con restricciones.

Función principal

La biblioteca utiliza una única función para realizar el ajuste, *fitFMM*, que encapsula las tres subfunciones mencionadas anteriormente que se ejecutarán dependiendo de los parámetros que reciba la función principal. La lista de parámetros que utiliza la función es la siguiente:

- El conjunto con los valores a ajustar.
- El número de periodos que contienen estos datos.
- Los instantes de tiempo en los que se recoge cada dato.
- El número de componentes que forman el modelo.
- Las restricciones sobre los parámetros β y ω.
- El número máximo de iteraciones a realizar.
- La función que se usará como criterio de parada.
- La precisión del grid interno que se usa para asignar los valores de los parámetros α y ω.
- El número de iteraciones sobre estos grids que se van a realizar.

Para finalizar se pueden establecer tres flags para poder visualizar el progreso de la función, el tiempo que le lleva realizarlo y otra flag por si se quiere realizar el ajuste en paralelo, siempre que la máquina lo permita.

El resultado de la función es un objeto de tipo FMM que incluye:

Los cinco parámetros que devuelve el modelo FMM explicado en el apartado dedicado al modelo (M, A, α, β y ω), el número de periodos del conjunto de datos, los instantes de tiempo del conjunto, el conjunto de datos tras haberles aplicado la sumarización, el conjunto de datos ya ajustados, el error cuadrático medio del ajuste, el conjunto de datos de entrada y el porcentaje de variabilidad.

Otras funciones

Para completar la biblioteca, también se añadieron otras funciones adicionales:

- Función de representación: ayuda con la visualización de los datos, utilizando el objeto FMM que devuelve la función principal para generar dos gráficos: el gráfico del ajuste completo y el gráfico con la división en sus componentes.
- Función de generación: permite generar un conjunto de datos utilizando los parámetros del modelo FMM que los representa. Se le puede añadir ruido gaussiano para mayor similitud con un conjunto de datos real.
- Funciones de resumen: devuelve un resumen del modelo ajustado.
- Función de extracción de coeficientes: para obtener los parámetros del modelo resultado.
- Función de datos ajustados: devuelve los datos ajustados por el modelo para los valores de tiempo especificados.
- Función de predicción: permite predecir los siguientes instantes temporales.

Acceso a la biblioteca

Para poder hacer uso de la biblioteca, se puede acceder a ella desde el CRAN, una red de servidores que funciona como archivo online. Esta red mantiene código y documentación de R actualizada, entre los que se encuentran diferentes bibliotecas de este lenguaje de programación a disposición de los usuarios. En este archivo se puede encontrar la biblioteca FMM que es la utilizada en este trabajo.

2.3.4 Adecuación del modelo a nuestro problema

Como se ha mencionado, el desarrollo de la biblioteca que realiza el ajuste usando el modelo FMM, tenía como objetivo ajustar la gráfica de un electrocardiograma. Este tipo de prueba médica genera una onda que se separa en cinco componentes, y que el modelo FMM consigue ajustar satisfactoriamente. Un ejemplo de este ajuste

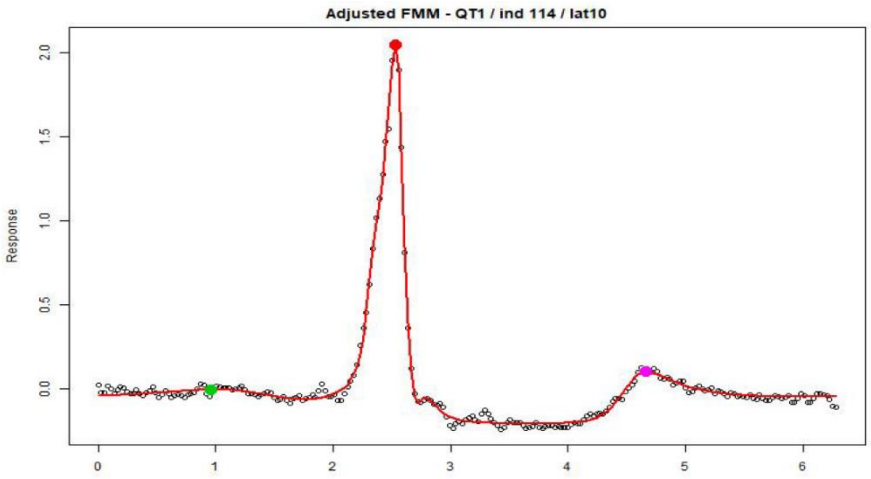

Figura 2.14 Ajuste de un ECG realizado por el modelo FMM.

se puede ver en la Figura 2.14, donde se puede apreciar el ajuste realizado sobre un ECG, y en la Figura 2.15 se observa la división en componentes realizada para que el modelo FMM sea capaz de ajustarlo.

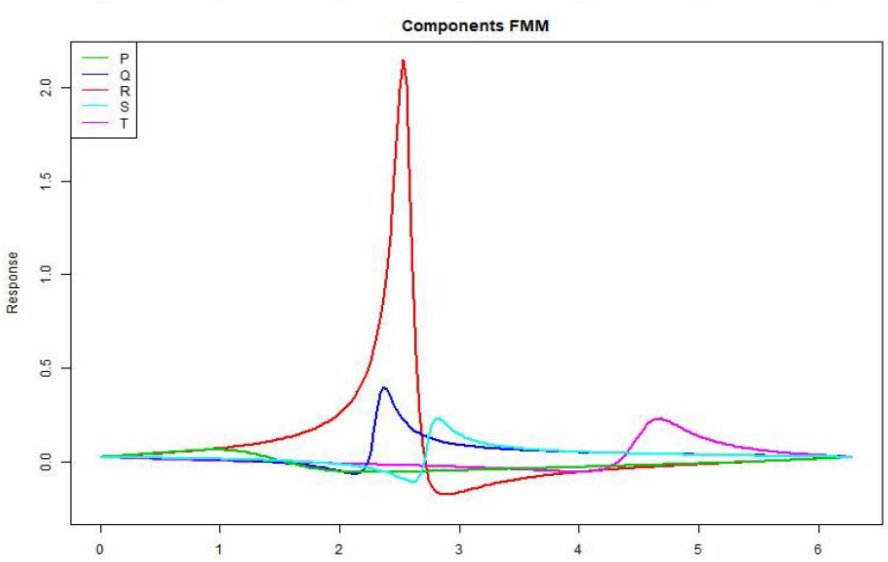

Figura 2.15 Ajuste de un ECG realizado por el modelo FMM dividido en los componentes utilizados

Por otro lado, si se observan los datos que se obtienen en los sensores de los dispositivos ponibles usados en el sistema del reconocimiento biométrico, se aprecia la periodicidad de la señal, pues esta se puede dividir en ciclos que se repiten a lo largo de toda ella. Estos ciclos están formados por dos puntos máximos y dos puntos mínimos, como se puede apreciar en la Figura 2.16.

En este momento, se puede apreciar cierta similitud con la gráfica de un electrocardiograma, ambas tienen una forma parecida, y son fácilmente divisibles en componentes para ser ajustadas por el modelo FMM, que tan buenos resultados había dado en el problema del ECG, lo que hizo plantear que podría ser también adecuado a nuestro problema.

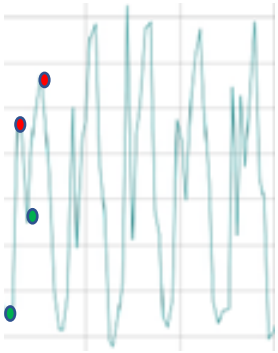

Figura 2.16 Gráfica de los datos obtenidos durante la marcha con los puntos máximos (rojo) y los mínimos (azul) del patrón que se repite marcados.

3 Tecnologías, Metodología y Planificación

3.1 Introducción

Tras explicar los trabajos y sistemas de partida, vamos a pasar a comentar cómo se llevó a cabo el trabajo de este TFG, incluyendo las tecnologías utilizadas, la metodología aplicada y las etapas en las que se ha dividido el proyecto, junto con el contenido de cada una.

3.2 Tecnologías

A continuación, se explicarán las tecnologías usadas, tanto software como hardware:

Lenguaje de programación R

Como se ha mencionado en el apartado 2, los sistemas de partida, tanto el sistema de reconocimiento biométrico como la biblioteca de ajuste de modelos FMM fueron desarrollados con el lenguaje R [15]. Este lenguaje está diseñado especialmente para el ámbito estadístico, pues cuenta con una gran cantidad de herramientas que facilitan operaciones en este campo: modelos lineales y no lineales, tests estadísticos, análisis de series temporales, algoritmos de clasificación y agrupamiento, además de la capacidad para representar estos datos.

Herramienta Visual Studio

Se ha utilizado este entorno de desarrollo integrado (IDE), desde el que se han añadido las modificaciones hechas para cambiar el sistema biométrico. Las razones fueron la familiarización previa que se disponía con el mismo, así como la disponibilidad en la máquina utilizada.

Control de versiones GitHub

Para evitar modificar archivos y perder alguna versión anterior se decidió utilizar Github como control de versiones, que permite mantener un control de los cambios realizados en el proyecto.

RStudio

Este IDE se utilizó principalmente para ejecutar los programas de R, por la facilidad a la hora de hacerlo y por la disponibilidad en la máquina utilizada.

Hardware

El principal hardware utilizado para desarrollar el trabajo fue una máquina virtual proporcionada por la Escuela de Ingeniería Informática de Valladolid, a la que se accedía de manera remota. Esta máquina contaba tanto con el sistema de reconocimiento biométrico y los datos de los usuarios tomados en el trabajo, como con las diferentes herramientas de software mencionadas previamente en este apartado.

Posteriormente, al ver la necesidad de mayor capacidad de cómputo, como se explica en el apartado 4.4Iteración 3. Paralelización., el Departamento de Informática de la Universidad de Valladolid puso a nuestra disposición dos máquinas virtuales con mayor capacidad de cómputo, que posteriormente fueron

redimensionadas para contar con más cores de procesamiento y aumentar así la velocidad de trabajo. Esto permitió, como se mencionará, paralelizar el sistema.

3.3 Metodología de trabajo

A la hora de tomar la decisión sobre qué metodología usar, se deben tener en cuenta las características que definen este trabajo, así como los objetivos especificados en el capítulo 1.2. En primer lugar, hay que tener en cuenta que, la propia naturaleza del trabajo, que es de tipo experimental, hace que una planificación detallada de todo el proyecto desde el inicio hasta su fin no sea posible. Pese a que las fases iniciales del proyecto se tienen claras y están bien definidas, pues se conocen los sistemas que se van a utilizar, así como los cambios que son necesarios para llevar a cabo las modificaciones, no es suficiente, ya que, tras obtener los primeros resultados del sistema modificado, es imposible conocer con exactitud los pasos a seguir a partir de ese momento, pues dependerán de los propios resultados conseguidos y su correspondiente análisis.

Esto hace que la decisión sobre la metodología se encontrase entre dos opciones. La primera opción era la metodología de prototipos evolutivos [12], que en cada iteración desarrolla un sistema nuevo, utilizando la información obtenida en las iteraciones anteriores. La segunda opción era la metodología en cascada [3], en la que se repiten las fases de análisis, diseño, implementación y pruebas con cada iteración.

Al imaginarse el flujo de trabajo, podría ocurrir el caso de que, tras las modificaciones iniciales, con los parámetros que se estimen oportunos, se consigan unos resultados tan buenos que no sea necesario llevar a cabo nuevas iteraciones. Sin embargo, también se puede dar el caso contrario, en el que no se escojan bien estos parámetros y se tengan que repetir todos los experimentos modificando estos parámetros, para volver a analizar los datos, por lo que tampoco se conoce con exactitud el número de iteraciones que se van a llevar a cabo, otra muestra más de la incertidumbre del proyecto en sus fases más avanzadas.

Teniendo esta incertidumbre final en cuenta, se decidió aplicar la metodología en cascada, en la que se van repitiendo las siguientes fases (Figura 3.1) hasta que se está satisfecho con los resultados obtenidos:

- Fase de análisis: en esta fase se reúnen todos los requisitos que ha de cumplir el sistema objetivo, para lo que se deben tener en cuenta los objetivos establecidos en el trabajo
- Fase de diseño: en esta fase, con los requisitos se decide la estructura que va a tener el sistema, planificando la implementación posterior.
- Fase de implementación: en esta fase se escribe el código necesario para añadir la funcionalidad definida previamente para formar el sistema objetivo.
- Fase de pruebas: tras la implementación, se debe probar que todo el sistema funciona correctamente, y que cumple con los requisitos establecidos en la fase de análisis.
- Fase de despliegue: ahora que se cuenta con el sistema funcional, se debe instalar y realizar las operaciones requeridas para permitir que lleve a cabo las funciones para las que fue creado.

- Fase de mantenimiento: en esta fase se mantiene un control sobre el sistema para evitar que ocurran imprevistos por su uso y se pueden realizar pequeñas modificaciones o arreglos que no se hayan manifestado hasta este punto.

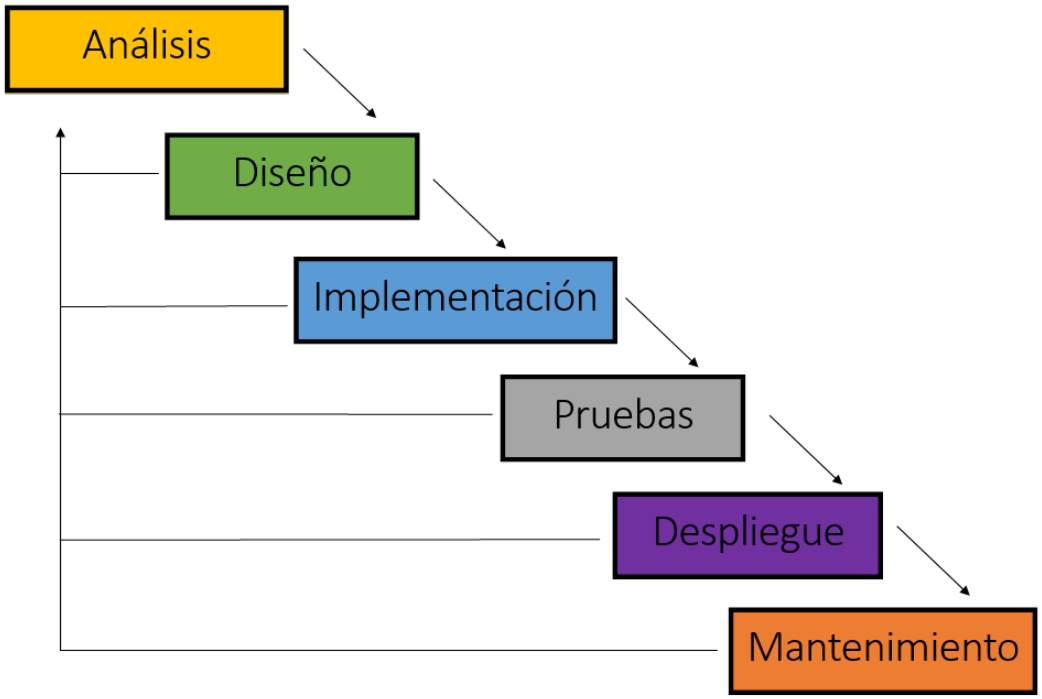

Figura 3.1Resumen del flujo del desarrollo en cascada

Hay que mencionar que las fases de análisis y diseño se realizarán sobre la parte del sistema que se modifica en el este trabajo, es decir, sobre la parte de extracción de características.

3.4 Planificación inicial

En este apartado se van a enumerar las tareas que se planificaron inicialmente en las que se debería dividir el trabajo, junto con el tiempo estimado que debería llevar cada una de ellas.

	Tarea	**Duración**
1	*Documentación trabajo sistema biométrico*	20 h
2	*Documentación trabajo librería en R*	20 h
3	*Familiarización con lenguaje R*	10 h
4	*Instalación de software necesario para el desarrollo*	5 h
5	*Pruebas con la librería en R*	10 h
6	*Familiarización con los programas y adaptación del sistema biométrico*	10 h
7	*Ejecución para obtener los resultados del sistema original*	10 h
8	*Desarrollo del nuevo sistema con los vectores de características*	75 h
9	*Preparación de los experimentos*	5 h
10	*Ejecución de los experimentos*	15 h
11	*Análisis de resultados*	10 h
12	*Documentación memoria trabajo*	55 h

Tabla 3.1 División del trabajo inicial en tareas junto con su estimación de tiempo

La suma total de horas tras la división en tareas de la Tabla 3.1 resulta en 245 horas. Esto es debido a que las tareas 9, 10 y 11, son las que puede darse el caso de que se tengan que realizar varias veces. Debido a la naturaleza experimental de este trabajo, hay ciertas fases del mismo que tienen un grado de incertidumbre sobre ellas que obliga a plantearse diferentes escenarios. Tras hacer las tareas de la 1 a la 8, llega el caso en el que se llevan a cabo la preparación de los experimentos, la ejecución de los mismos y, tras el análisis nos damos cuenta de que los parámetros establecidos no son lo suficientemente buenos, por lo que debemos repetir de nuevo estas tres tareas (9, 10 y 11) para que, dependiendo de la calidad de los resultados obtenidos, se plantee realizar otra vez estas tres tareas con otros parámetros para ver los resultados.

Teniendo en cuenta de nuevo este grado de incertidumbre que se tiene sobre las fases finales del trabajo, se decidió que un buen número de repeticiones de las tareas 9, 10 y 11 sería tres, por lo que sumando las horas estimadas en ese caso se obtienen 305 horas de trabajo estimado.

3.5 Riesgos

En este apartado se van a enumerar los riesgos que pueden tener lugar y que afectarían en mayor o menor medida a la realización del proyecto. En la Tabla 3.2 se incluyen para cada riesgo los siguientes campos:

- Descripción del riesgo
- La probabilidad de que ocurra. Pudiendo tomar tres valores, probabilidad baja, media o alta.
- Impacto sobre el proyecto. En caso de que se diese un riesgo, de qué manera afectaría al proyecto. Se usarán los valores impacto bajo, medio y alto.
- Medidas de mitigación. Las medidas que se van a tomar durante el trabajo para reducir en lo posible las probabilidades de que el riesgo ocurra.
- Medidas de contingencia. Las medidas que se tomarían si el riesgo ocurriese.

Nº	**Descripción**	**Probabilidad**	**Impacto**	**Medidas de mitigación**	**Medidas de contingencia**
1	Pérdida de los datos del trabajo.	Bajo	Alto	- Mantener copias de seguridad en la nube. - Actualizar las copias de seguridad tras cada jornada de trabajo	- Tratar de restaurar las copias de seguridad existentes. - Intentar recuperar las últimas versiones existentes de cada fichero
2	Enfermedad del alumno	Bajo	Medio	- Tratar de llevar una vida saludable. - Evitar poner en riesgo la salud con actividades peligrosas.	- Iniciar las ejecuciones de los experimentos que se puedan realizar para evitar perder más tiempo. - Añadir más horas a la planificación estimada.

3	Tiempo excesivo para realizar los experimentos	Medio	Medio	- Calcular el tiempo estimado de los experimentos en lugar de esperar a que terminen. - Realizar experimentos iniciales no muy exigentes para hacerse a la idea de los tiempos de ejecución.	- Reducir la cantidad de datos con los que se realizan los experimentos. - Dividir las ejecuciones en diferentes máquinas.
4	Ausencia por enfermedad de uno o ambos tutores	Bajo	Medio	- Establecer varios métodos de comunicación para mantener el contacto y prever la situación. - Preguntar las dudas según vayan apareciendo. -Planificar con anterioridad las distintas tareas a llevar a cabo.	- Continuar el trabajo e ir apuntando las dudas para más adelante.
5	Dificultad en el uso de las tecnologías necesarias	Media	Bajo	- Escoger con antelación las tecnologías con las que se esté más familiarizado. - Buscar documentación adecuada para el nivel necesario para el trabajo.	- Buscar nueva documentación adecuada. - Preguntar a algún experto en la materia sobre la manera de proceder.
6	Fallo en la planificación estimada	Alto	Medio	- Consular con los tutores la planificación antes de llevarla a cabo.	- Reducir el número de iteraciones a realizar en el trabajo. No repetir las fases de preparación, ejecución y análisis de los experimentos.

Tabla 3.2 Tabla con los riesgos del trabajo

3.6 Planificación real

En este apartado se mostrarán las tareas en las que se ha dividido realmente el trabajo, así como su duración, en comparación con la duración estimada.

	Tarea Real	Duración estimada	Duración real
1	*Documentación trabajo sistema biométrico*	20 h	25 h
2	*Documentación trabajo librería en R*	20 h	20 h
3	*Familiarización con lenguaje R*	10 h	10 h
4	*Instalación de software necesario para el desarrollo en las máquinas virtuales*	5 h	15 h
5	*Pruebas con la librería en R*	10 h	10 h
6	*Familiarización con los programas y adaptación del sistema biométrico*	10 h	10 h
7	*Ejecución para obtener los resultados del sistema original*	10 h	20 h
8	*Desarrollo del nuevo sistema con los vectores de características*	75 h	45 h
9	*Paralelización del sistema **	-	40 h
10	*Desarrollo de la normalización de los resultados **	-	15 h
11	*Preparación de los experimentos*	5 h	25 h
12	*Ejecución de los experimentos*	15 h	184 h
13	*Análisis de resultados*	10 h	10 h
14	*Documentación memoria trabajo*	55 h	75 h

Tabla 3.3 División del trabajo en las tareas realizadas, junto con su duración estimada y la real

En la Tabla 3.3 se muestran las tareas llevadas cabo durante el trabajo, que sí que han sido diferentes a las tareas planificadas inicialmente. Los mayores cambios vienen con las tareas 9 y 10, marcadas con un *, ya que han sido las tareas nuevas introducidas que no estaban contempladas en la planificación inicial.

Por otro lado, la mayor diferencia de horas se aprecia en la tarea 12, ya que la ejecución de los experimentos llevó un tiempo mucho mayor del esperado, pues como se explicará en el capítulo del Desarrollo, la librería FMM añadía un gran tiempo de ejecución para cada usuario.

El resultado de las horas totales del trabajo son 504 horas.

3.7 Presupuesto estimado

Para calcular el presupuesto estimado del trabajo se han considerado necesarios los siguientes elementos:

- Desarrollador de software: la persona encargada de realizar las modificaciones al sistema existente y encargada de analizar los resultados. El sueldo medio de un desarrollador en España, según [16] es de 2.292€ al mes, o de 14,10€ la hora. Por lo que, empleando a esta persona las 305 horas que se ha estimado para el trabajo en el apartado 3.4**¡Error! No se encuentra el origen de la referencia.**, se obtiene un coste de 4300,50€.
- Equipo informático en el que se llevan a cabo el desarrollo del software y las ejecuciones de los experimentos. Se ha usado como ejemplo el portátil del alumno, su precio se ha consultado en [7] y es de 711,37€. Si se estima la vida útil del mismo en 5 años o 60 meses, aunque hay casos en los que este número puede ser mayor o menor, el precio del producto al mes sería de 11,85€/mes. En la planificación estimada del trabajo, la duración de tiempo era de cinco meses, por lo que, el coste del ordenador portátil durante este periodo de tiempo sería de 59,25€.

Para la duración final del trabajo, usando la misma proporción con la que se ha establecido que las 305 horas planificadas iban a durar 5 meses, obtenemos que las 504 horas llevadas a cabo equivalen a 8,2 meses de trabajo, por lo que el gasto del portátil en este tiempo sería de 97,17€.

- El software necesario para llevar a cabo el desarrollo y el posterior análisis de los resultados. El software utilizado se especificó en el apartado 3.2. Hay que destacar que el único software de pago entre los utilizados es el Microsoft Office, que ofrece una licencia anual por 69€, y el resto son gratuitos. El gasto mensual de esta licencia por lo tanto quedaría en 5,75€/mes, por lo que, para la duración de 5 meses estimada inicialmente, el gasto proporcional de esta licencia sería de 28,75€, mientras que para los 8,2 meses de trabajo realizado sería de 47,15€.

En la Tabla 3.4 se observa el resumen del presupuesto estimado y en la Tabla 3.5 el del presupuesto real, modificando las horas de trabajo y por tanto el gasto en la contratación del desarrollados.

Nª	Elemento	Coste/tiempo	Tiempo de uso	Total
1	Desarrollador de R	14,10€/hora	305 horas	4300,50€
2	Portátil HP HP Pavilion 15-bc500ns	11,85€/mes	5 meses	59,25€
3	Licencia Microsoft 365 Personal	5,75€/mes	5 meses	28,75€
Total				**4388,50€**

Tabla 3.4 Resumen del presupuesto estimado necesario para el trabajo

Nª	Elemento	Coste/tiempo	Tiempo de uso	Total
1	Desarrollador de R	14,10€/hora	504 horas	7106,40€
2	Portátil HP HP Pavilion 15-bc500ns	11,85€/mes	5 meses	97,17€
3	Licencia Microsoft 365 Personal	5,75€/mes	5 meses	47,15€
Total				**7250,72€**

Tabla 3.5 Resumen del presupuesto real necesario para el trabajo

4 Desarrollo

En este apartado se van a explicar las distintas iteraciones en las que se ha dividido el trabajo.

4.1 Iteración 0: familiarización

Esta primera iteración ocurre tras el primer contacto con los sistemas existentes, en la que se buscaba conocer en profundidad el funcionamiento del sistema de reconocimiento biométrico.

Al ser la primera iteración, en la que el objetivo es únicamente de documentación del sistema y no se realiza ningún desarrollo de código, no se va a dividir en las fases de la metodología en cascada.

Para llevar a cabo esta iteración, al saber que el sistema fue desarrollado en el lenguaje de programación R, del que no se tenía apenas ningún conocimiento, primero se consideró necesario familiarizarse con el lenguaje en cierta medida, pues, aunque no vaya a ser requerido un gran conocimiento de este, ya que en principio únicamente habrá que utilizar la biblioteca creada, conocerlo facilitará esta tarea y el entendimiento del sistema, así como la resolución de los errores que se puedan encontrar. En Bibliografía se pueden encontrar los enlaces a las fuentes utilizadas para este fin [30].

Tras esto, se repasó y comprendió todo el código del sistema de reconocimiento biométrico y, mientras se consultaban las operaciones o funciones que se utilizaban en cada parte, se iban modificando ciertas variables. Estas modificaciones que se realizaron eran necesarias para preparar el sistema para ser ejecutado en la máquina virtual que se desplegó con el fin de tener acceso, tanto a los datos del sistema, como a su código, ya que, por ejemplo, había que cambiar las rutas que se usaban (necesarias para consultar los datos de los usuarios o para retornar los ficheros con los resultados) ya que los directorios ahora eran diferentes.

A medida que se iba realizando esta revisión de cada fichero del código, se iban ejecutando, para así comprobar que todo estaba preparado para realizar los experimentos posteriores.

4.2 Iteración 1: experimentación inicial

Una vez conocido el entorno, el sistema y su funcionamiento, pasamos a realizar los experimentos iniciales utilizando los datos originales del TFG de Irene Salvador Ortega [27], que servirán como base para comparar los resultados más adelante.

Análisis

Requisitos funcionales:

RF-01	El sistema debe permitir obtener los vectores de características originales sobre los datos del módulo de las tres coordenadas de los sensores.
RF-02	El sistema debe permitir obtener los vectores de características originales sobre los datos de las tres coordenadas de los sensores (X, Y y Z).

Requisitos no funcionales:

RNF-01	Los experimentos se realizarán solo para el conjunto de valores de entrada especificados a continuación.

Los valores que se iban a asignar a los diferentes parámetros de entrada del sistema en los experimentos son los siguientes:

- El escenario experimental utilizado: se usan los valores 'MonoMono' y 'MultiMono', que indican qué grupo de datos de cada usuario se utiliza y para qué fin, ya que se tienen dos sesiones de tomas de datos S1 y S2, en las que se toman dos muestras de datos en cada una M1 y M2. La configuración experimental indicada en cada uno de esos casos es:
    - El entrenamiento "MonoMono" (Monosesión-Monomuestra) utiliza únicamente las muestras de la primera sesión. La muestra uno para entrenar al sistema y la muestra dos para las pruebas en las que el sistema debe reconocer correctamente al usuario registrado (pruebas auténticas). Las pruebas en las que debe reconocer a los usuarios como impostores, utilizan la muestra dos de la sesión uno del resto de usuarios. En este escenario las muestras de entrenamiento y prueba pertenecen a la misma sesión, es decir, son tomadas el mismo día.
    - El entrenamiento "MultiMono" (Multisesión-Monomuestra) usa la muestra uno de la sesión uno para entrenar al sistema (como en el escenario anterior) y las pruebas de reconocimiento del usuario registrado (pruebas auténticas) se hacen con las dos muestras de la segunda sesión. Para la prueba en la que debe reconocer a los usuarios como impostores se usa la misma muestra que en el caso anterior. En este escenario las muestras de entrenamiento y prueba pertenecen a distintas sesiones, es decir, son tomadas distintos días; el objetivo es ver cómo se comporta el sistema con la modificación del patrón de comportamiento del usuario con el tiempo. Es un escenario más realista que el anterior.
- El dispositivo utilizado. Valores: pulsera Microsoft, reloj Motorola
- El sensor utilizado. Valores: acelerómetro, giróscopo.
- El dominio utilizado para la extracción de características. Valores: tiempo, frecuencia
- El tipo de datos utilizado. Valores: coordenada X, coordenada Y, coordenada Z, módulo de las tres
- El clasificador utilizado: Se han seleccionado sólo algunos de los clasificadores con los que se hizo el trabajo anterior, rechazando los que no generaban resultados aceptables. Se usan el clasificador k vecinos más cercanos con el valor de k = 1, el clasificador SVM con kernel con función de base radial, el SVM con kernel con función de base polinomial de grado 5, el árbol de decisión y el random forest.
- Los valores para la fusión de scores final utilizados (post-procesamiento): Para fusionar los scores finales se estableció un solapamiento de ventana igual a 2, y para el número de ventanas consecutivas cuyas salidas se fusionan, inicialmente fueron los valores 1, 4, 8 y 16, pero para la ejecución de algunos usuarios, al llegar a la ejecución con el valor 16 y no contar con suficientes ventanas, los resultados eran escasos e incluso nulos, por lo que este valor máximo lo sustituimos por 12, con el que las ejecuciones funcionaban correctamente.

Diseño

Al no necesitar realizar una implementación de ningún cambio, más allá de modificar los valores de los parámetros de entrada, esta fase de diseño se omitió.

Implementación

Los únicos cambios que se realizaron fueron los parámetros especificados en la fase de diseño para obtener los resultados con esos parámetros. Los datos obtenidos se generaron mediante un bucle en el que se iban dando diferentes valores a los parámetros especificados en la fase de diseño.

Pruebas

Se comprobó que, al modificar los parámetros, los experimentos sí devolvían los resultados usando los valores especificados en la fase de diseño correctamente.

Despliegue

En esta fase no se llevó a cabo ninguna acción adicional, pues el sistema ya estaba funcionando previamente.

Resultados

Los resultados obtenidos fueron los esperados, así que se continuó con el trabajo planeado

4.3 Iteración 2

Contando ahora con los datos con los que comparar, el siguiente paso es modificar el sistema existente para que, utilizando la biblioteca que ajusta los datos a un modelo FMM, obtener un nuevo vector de características para cada ventana de datos de las muestras producidas por los diferentes usuarios.

Análisis

En cuanto a este primer análisis de las modificaciones a realizar, se encontraban claros los siguientes requisitos a cumplir.

Requisitos funcionales:

RF-01	El sistema debe permitir obtener los vectores de características a partir de la biblioteca FMM aplicado sobre los datos del módulo de las tres coordenadas de los sensores.
RF-02	El sistema debe permitir obtener los vectores de características a partir de la biblioteca FMM aplicado sobre los datos de las tres coordenadas de los sensores (X, Y y Z)

Requisitos no funcionales:

RNF-01	La salida del programa debe mantener el formato de los vectores de características obtenidos tal y como se tiene en el sistema previo a la modificación. (Una tabla con una fila para cada muestra de datos de cada usuario, con los valores devueltos por la función de obtención de características)

El requisito RNF-01 se añade debido a que su implementación facilita la utilización de los programas originales del sistema de reconocimiento biométrico, sin necesidad de añadir modificaciones adicionales a estos ficheros para ser capaces de interpretar los nuevos datos obtenidos.

Sobre los casos de uso de la aplicación, hay que mencionar que no se va a modificar el flujo original de interacciones con el usuario, pues la modificación a realizar únicamente altera una parte interna del funcionamiento del sistema, de tal manera que tanto la entrada como la salida mantienen su estructura, haciendo que el único cambio apreciable entre las dos versiones del sistema sean los valores de los resultados al terminar todas las ejecuciones experimentales.

Diseño

La división original del sistema en diferentes ficheros que se encargan de las diferentes etapas del sistema (detección del ruido, pre-procesamiento de los datos, obtención de características, etc.) hace que la modificación a realizar no sea compleja, ya que esta afecta únicamente al fichero que se encarga de esta obtención de características. Los pasos que se consideraron necesarios para añadir la modificación son los siguientes:

- Eliminar las operaciones de obtención de características existentes. Estas utilizaban tanto funciones nativas del lenguaje R para obtener algunos valores como el máximo, la media o la varianza de los datos, como también funciones creadas por Irene Salvador Ortega para obtener otros datos más complejos como la autocorrelación de los valores o el número de cruces que tiene cada ventana de datos.
- Llamar a la función para obtener el ajuste de los datos con el modelo FMM.
- Almacenar estos datos obtenidos con la misma estructura que se almacenan los vectores de características en el sistema original, para evitar modificaciones en las etapas posteriores. Los programas originales devuelven una tabla con la extensión *.csv* con el vector de características de cada ventana de cada muestra de cada usuario en una fila. Una simplificación se muestra en la Tabla 4.1.

Usuario	**Sesión**	**Muestra**	**Tiempo-inicio**	**Tiempo-fin**	**Media**	**Máximo**	**…**
1	1	2	1	30	--	--	
1	1	2	24	54	--	--	
1	2	1	1	30	--	--	
1	2	2	24	54	--	--	

Tabla 4.1 Ejemplo del formato de la tabla original que almacena los datos de los vectores de características

Los tres primeros valores representan el usuario (valor entre 1 y 18), la sesión (valor entre 1 y 2) y la muestra (valor entre 1 y 2) de los datos que se están utilizando en cada ventana y que forman la entrada que utiliza la función FMM. Los valores tiempo-inicio y tiempo-fin indican los instantes de tiempo en los que empieza y acaba cada una de las ventanas de datos cuyos valores se han utilizado. Por último, en lugar de las columnas de la media y el máximo, se añadirán los valores que devuelve la función FMM y que representan el ajuste de los datos al modelo.

Otra decisión de diseño fue la del parámetro de los componentes que la función FMM multicomponente iba a utilizar para intentar ajustar los datos de las ventanas de los usuarios.

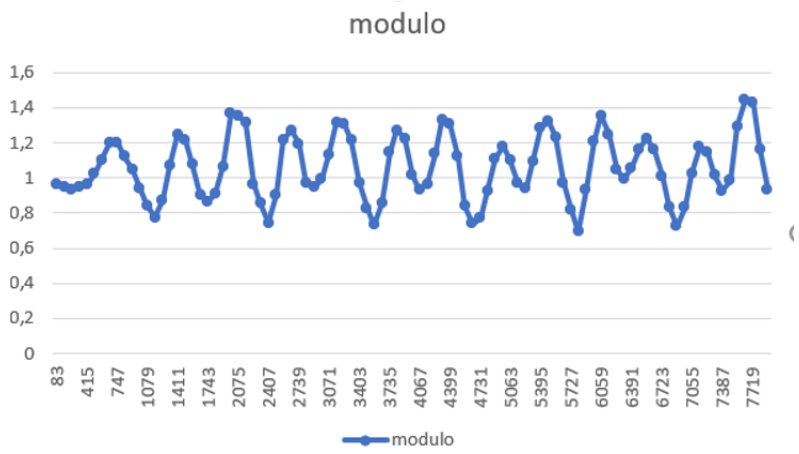

Figura 4.1 Representación gráfica de los datos de la primera ventana de datos del usuario 1.

Para tomar la decisión se necesitaba comprobar si el ajuste de la función FMM con diferentes números de componentes era lo suficientemente satisfactorio como para llevar a cabo la ejecución. Estas pruebas se realizaron con los datos del módulo de las tres coordenadas de la muestra 1 de la sesión 1 del usuario 1 con la pulsera de Microsoft y el sensor acelerómetro. Para realizar las pruebas con los mismos datos que se iban a usar en la ejecución, se dividió este fichero de datos en las mismas ventanas que utiliza el programa para obtener los vectores de características. Usando la primera ventana de datos, se observaron los datos representados en la Figura 4.1.

El principal dato que se obtiene al observar la Figura 4.1 es el número de periodos (entendiendo como tal la parte de la onda que incluye un máximo y un mínimo) que se pueden contar en la onda asociada a una ventana: 13. Se pasó a comprobar el número de periodos apreciables en el resto de ondas de las demás ventanas de datos, resultando la media obtenida un 13. Teniendo en cuenta que la función FMM, para realizar su ajuste, utiliza un componente para cada periodo de la onda que ajusta, se decidió probar a ajustar la ventana de datos con 8 componentes, 14 y 16 para observar sus diferencias.

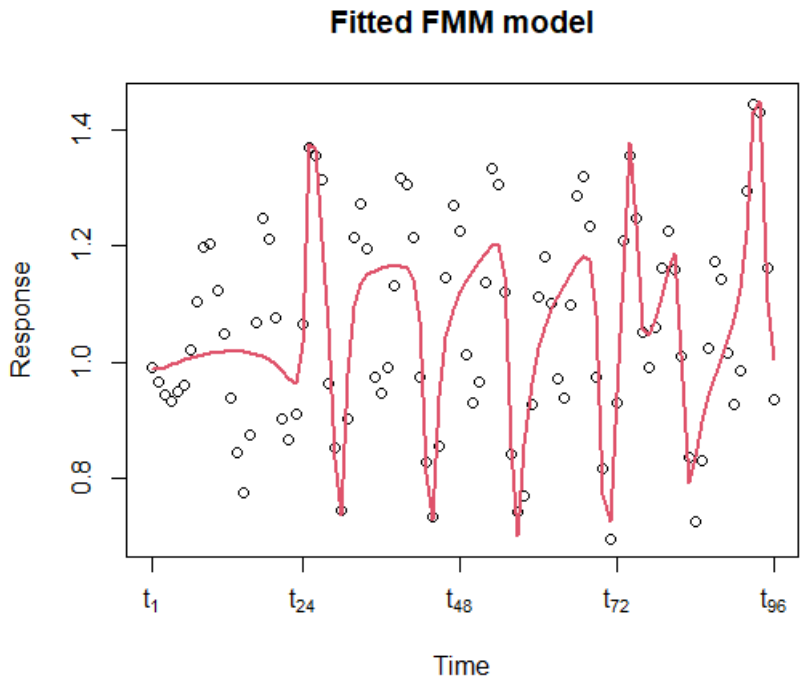

Figura 4.2 Ajuste de la función FMM con 8 componentes

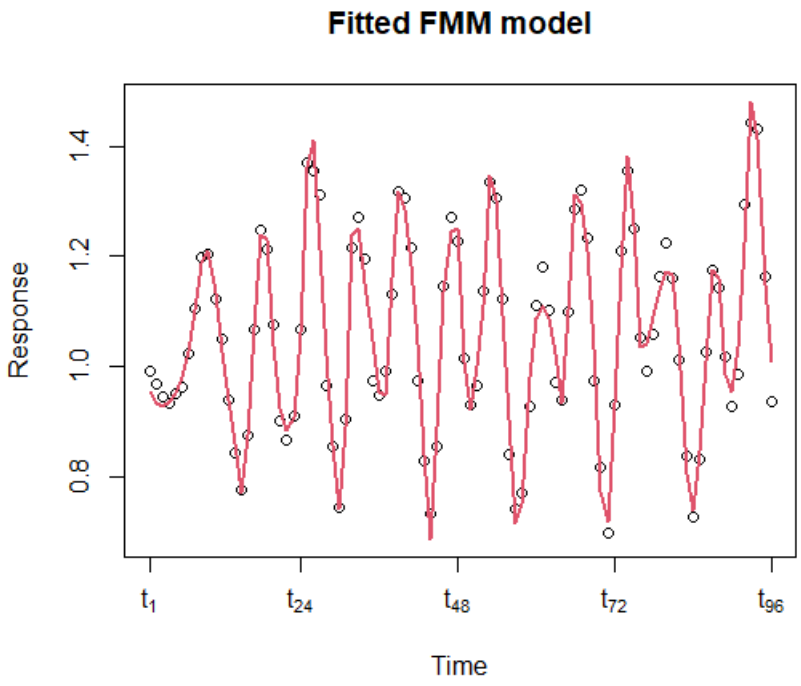

Figura 4.3 Ajuste de la función FMM con 13 componentes

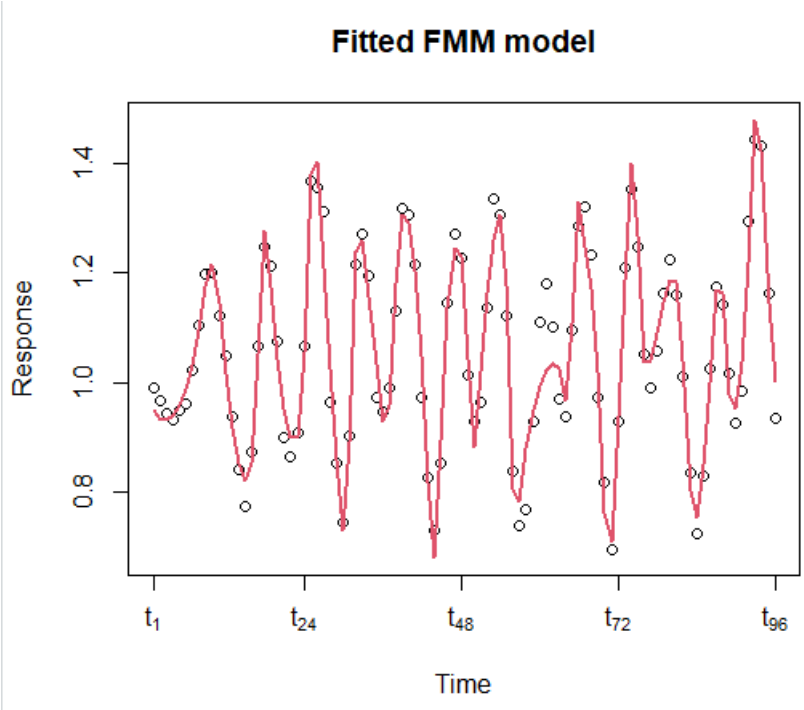

Figura 4.4 Ajuste de la función FMM con 16 componentes

Al comparar las tres opciones se aprecia rápidamente que el ajuste de la función con 8 componentes (Figura 4.2) no es suficiente, pues no tiene en cuenta todos los puntos y, por lo tanto, no obtiene la misma forma que se apreciaba en la gráfica inicial de los datos (Figura 4.1).

Por otro lado, tanto el ajuste con 13 (Figura 4.3) y con 16 (Figura 4.4) componentes a primera vista parece que es suficientemente bueno, pues la forma general del ajuste es muy similar a la gráfica inicial de los datos (Figura 4.1), por lo que es un resultado bastante preciso. La función FMM, al realizar este ajuste, devuelve un

conjunto de datos entre los que hay algunos que también han afectado a la hora de decidir sobre este parámetro. Uno de ellos es el valor de R cuadrado, también conocido como coeficiente de determinación [2], que indica el error medio del ajuste respecto al conjunto de datos usado, siendo 1 un ajuste perfecto y 0 el peor ajuste.

El otro valor que se ha tenido en cuenta ha sido el tiempo que ha tardado en llevar a cabo el ajuste de las cinco primeras ventanas.

Antes de tomar la decisión se comprobó cuánto costaría aumentar la precisión del ajuste (reducir el coeficiente de determinación) en comparación con el tiempo que necesitaría el programa para conseguirlo. Con ese fin, se incluyó en las pruebas el ajuste utilizando 22 componentes.

Nº de componentes	**8**	**13**	**16**	**22**
Coeficiente de determinación	0.655	0.8615	0.9227	0.97
Tiempo en segundos de ejecución	24 segundos	1 minuto y 13 segundos	2 minutos y 26 segundos	3 minutos y 21 segundos

Tabla 4.2 Resumen ajuste con diferente número de componentes

Con los datos de la Tabla 4.2 Resumen ajuste con diferente número de componentesse aprecia que el aumento del tiempo es considerable al incrementar el número de componentes en el ajuste y, sin embargo, la mejora del coeficiente de determinación no lo es tanto. Atendiendo a la relación entre el tiempo invertido y la mejora en la calidad de los resultados, se tomó la decisión de utilizar la función FMM con un ajuste de 16 componentes.

Implementación

A la hora de implementar la modificación, lo que más tiempo consumió fue el último paso, pues el formato de los datos que devuelve la función FMM no permitía almacenar de manera simple los datos en la estructura existente, por lo que hubo que recorrer mediante un bucle esta estructura de datos, para ir añadiéndolos a la estructura resultado y así garantizar que se mantenía el formato original.

Pruebas

Una vez que se contaba con la modificación hecha, y antes de pasar a realizar la ejecución completa, se debían llevar a cabo las pruebas necesarias para verificar el correcto funcionamiento de los cambios realizados.

La lista de comprobaciones que se tuvieron en cuenta fue la siguiente:

- Al haber borrado la parte del código previa al cálculo de los vectores de características, hubo que verificar que los datos introducidos a la función FMM fuesen los correctos, y los mismos que entraban en la antigua función de obtención de características.
- Al modificar la obtención de las características y la forma en que se almacenan, se comprobó que los datos devueltos por la función FMM se almacenasen de manera correcta en la estructura resultado, respetando el mismo formato que en la versión original.

Cuando se tenía garantizado que todo funcionaba correctamente se pasó a iniciar la primera ejecución del sistema con la nueva obtención de características. Ejecutando primero la versión que calcula los vectores de características usando los datos del módulo de las coordenadas de los sensores y después la versión que usa directamente los valores de las coordenadas de los sensores.

Despliegue

La fase de despliegue no requiere nada más que ajustar al entorno de ejecución las rutas del programa que indicaban el directorio desde el que se obtenían los datos crudos iniciales, y el directorio en el que se devolverán las tablas de datos *.csv*.

Resultados

Realmente no se llegó a terminar la ejecución de las pruebas, pues al cabo de tres días de estar ejecutando sin parar las pruebas en la máquina virtual, se comprobó que el tiempo que tardaba en calcular cada uno de los nuevos vectores era excesivo. Para cada ventana de datos, al ejecutar la función FMM de ajuste, con 16 componentes como se había decidido en la fase de diseño, tardaba 20 minutos que, en comparación con los dos minutos y veinte segundos que tardaba de media en la máquina en la que se hicieron las pruebas (la propia del alumno), era bastante alto. Al ver esto, se calculó una aproximación de las iteraciones que iba a realizar el programa y multiplicando por los 20 minutos que empleaba en completar cada una, se obtenía un tiempo de ejecución de 80 días.

Al no ser viable mantener encendida y realizando los cálculos la máquina personal del alumno durante todo el tiempo que durase la ejecución, se paró esta y se pasó a buscar nuevas soluciones.

4.4 Iteración 3. Paralelización.

Soluciones para reducir el tiempo de ejecución

Al conocer la gran cantidad de tiempo que iba a llevar la ejecución en la máquina virtual, se contemplaron las siguientes soluciones:

- Reducir la cantidad de datos a utilizar: una de las primeras soluciones que surgieron fue la más simple, la de reducir el número de iteraciones que se realizaban, disminuyendo el tamaño del conjunto de datos. Para ello se pensó utilizar únicamente los datos relativos a las muestras tomadas con el reloj Motorola y solo los del sensor acelerómetro, reduciendo así en cuatro el tiempo, se obtenían unos 20 días de ejecución, que seguía siendo algo alto.
- Ejecutar las pruebas en otra máquina con mayor capacidad de cómputo. Al observar la diferencia de tiempo al ejecutar el mismo código en dos máquinas diferentes, se contempló la opción de solicitar nuevas máquinas virtuales con una mayor capacidad de cálculo para ver si era posible reducir el tiempo necesario para realizar el ajuste del modelo en cada iteración.
  El Departamento de Informática nos proporcionó dos máquinas con una mejor capacidad de cómputo, para comprobar si el tiempo se reducía y, en caso de ser así, poder dividir las pruebas para llevar a cabo diferentes ejecuciones en cada máquina al mismo tiempo con conjuntos de datos diferentes. Tras instalar el software necesario y preparar las máquinas para las pruebas, se comprobó que el tiempo empleado en el ajuste en cada iteración se reducía de 20 minutos a 1 minuto con 40 segundos de media. Con este nuevo valor, la ejecución pasaría de tardar 80 días a 6 o 7 días.
  Este valor ya era suficientemente bueno en comparación con el inicial, pero si, además, contamos con las dos máquinas a las que nos dieron acceso, este tiempo se dividía entre dos. Además, estaba la opción de reducir los datos con los que realizar las pruebas, como se ha explicado en el punto anterior, por lo que se podría reducir el tiempo de ejecución incluso a un día.
  Sin embargo, hay que mencionar que todos estos cálculos de tiempo necesario para la ejecución hacen referencia únicamente a la ejecución del programa que usa como valores de entrada el módulo de las

coordenadas de los sensores en cada instante, y es necesario ejecutar también el programa que utiliza los valores para cada coordenada de manera individual (X, Y y Z).
La diferencia en la ejecución del programa al usar los datos del módulo y los de las coordenadas es que, con el módulo, en cada iteración solo se ajusta un único vector de datos por ventana, el del módulo, por lo que solo se llama una vez a la función de ajuste FMM. Mientras que, en la ejecución de los datos de las coordenadas, en cada iteración no se calcula el módulo de estas coordenadas y, por lo tanto, se realiza la función FMM de ajuste tres veces, una por cada coordenada existente (X, Y, Z). Por esto, el tiempo necesario se multiplica por tres, obteniendo de nuevo duraciones algo más altas.

- Paralelización: pese a que las soluciones propuestas hasta este punto parecían suficientes, se nos ocurrió una más, paralelizar el código para ejecutarlo de manera simultánea en el procesador de la máquina, como se explica en el apartado 2.1. Se hizo una pequeña tarea de documentación sobre cómo llevar a cabo esta paralelización en el lenguaje R y, a primera vista, no pareció que requiriese una gran complejidad ni conocimientos avanzados. Por ello, se decidió llevar a cabo esta adaptación del código para ver si se podía reducir el tiempo de ejecución, a la vez que se mantenía el uso de todos los datos existentes, sin excluir ningún conjunto de estos. El Departamento de Informática en ese momento se ofreció a sustituir una de las dos máquinas a las que ya teníamos acceso, por otra con un número de núcleos mayor para que, en caso de ser capaces de paralelizar el código, al ejecutarlo se aprovechase el tiempo de una manera óptima y, por lo tanto, obtener una mayor reducción de la duración de la ejecución, tanto para el programa que utiliza los valores de las coordenadas, como para el que usa el módulo de estas.

Teniendo en cuenta todas estas soluciones, se decidió paralelizar el programa y, si se consideraba necesario, reducir la cantidad de datos a utilizar, pero esta opción como último recurso si el tiempo seguía siendo elevado incluso al realizar la paralelización

Análisis

En esta fase, los requisitos se mantenían iguales respecto a la primera iteración, pues el objetivo del sistema es el mismo, aunque se añadió el nuevo requisito que establecía que la ejecución se pudiese realizar de forma paralela. Por lo tanto, la nueva tabla de requisitos quedaba así:

Requisitos funcionales:

RF-01	El sistema debe permitir obtener los vectores de características a partir de la biblioteca FMM aplicado sobre los datos del módulo de las tres coordenadas de los sensores.
RF-02	El sistema debe permitir obtener los vectores de características a partir de la biblioteca FMM aplicado sobre los datos de las tres coordenadas de los sensores (X, Y y Z)
RF-03	El sistema deberá permitir realizar la obtención de vectores de características de forma paralela.

Requisitos no funcionales:

RNF-01	La salida del programa debe mantener el formato de los vectores de características obtenidos tal y como se tiene en el sistema previo a la modificación.

	(Una tabla con una fila para cada muestra de datos de cada usuario, con los valores devueltos por la función de obtención de características)

Para facilitar la tarea de la implementación de esta paralelización, se efectuó primero una fase de análisis de la viabilidad de este método, pues no se tenía experiencia previa que confirmase que esta aproximación para resolver el problema del tiempo de ejecución fuese adecuada.

En esta fase, el primer paso fue un trabajo de investigación para conocer la manera y el método más adecuado para aplicar la paralelización en un programa en R. Se dio uso principalmente a la información de [20], que presenta diferentes métodos para aplicarla:

- La que, tras un primer análisis, se consideró la opción más sencilla y adecuada, fue la que utilizaba la función *mclapply()* [10] del paquete de R *'parallel'*. A esta función se le pasaba el número de cores disponibles en la máquina para dividir la ejecución, y la función que se iba a ejecutar de forma paralela. Sin embargo, al tratar de ejecutar una pequeña prueba para comprobar que esto funcionase así, se descubrió que la orden *mclapply()* no funcionaba y generaba un error. Se investigó y se encontró la causa, la orden *mclapply()* hace uso de la función *fork()* [23] (Figura 4.5), que permite, a partir de la ejecución principal de un proceso, crear un duplicado de este proceso para ejecutar ambos de forma simultánea, pero el problema es que solo se puede utilizar en sistemas Unix, por lo que en nuestras máquinas con Windows quedó descartada esta opción.

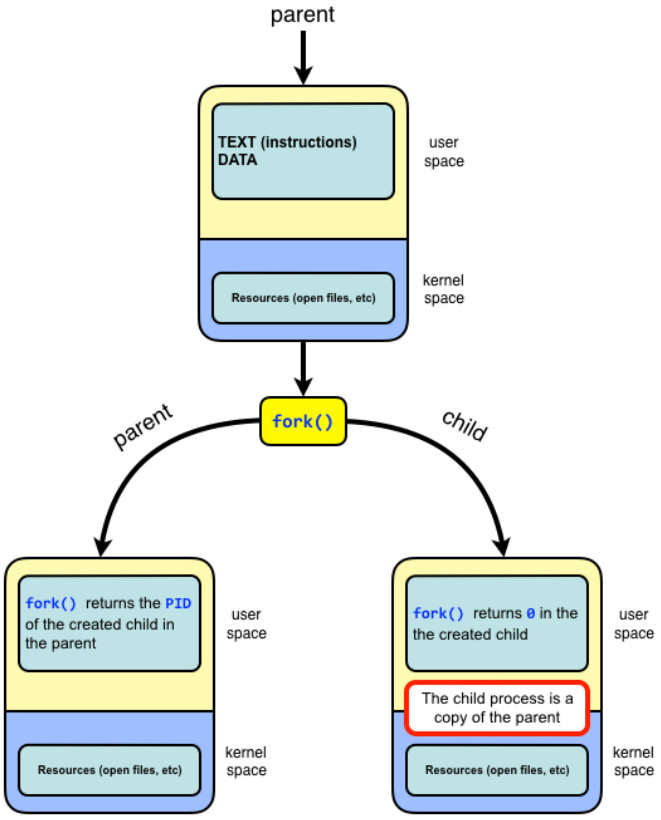

Figura 4.5 Diagrama que representa el funcionamiento de la función fork()

- Se buscó una nueva vía para aplicar la paralelización, y en este caso fue la función *foreach()* [6] [31], a la que se le puede añadir un parámetro que activa la ejecución en paralelo de las diferentes iteraciones. Se hizo una pequeña prueba con un conjunto de datos reducido en la que se ejecutaba cinco veces la función FMM con cuatro componentes. Se comprobó que el tiempo de ejecución con

el código sin paralelizar era de 58,22 segundos, pasando a 17,92 segundos utilizando la función *foreach()* paralelizada.

Contando ya con la seguridad de que este método iba a reducir en gran medida el tiempo de ejecución se pasó al diseño de la implementación, sabiendo ahora cómo usar la función *foreach()*.

Diseño

Tras ver la documentación de la función y, teniendo en cuenta el código existente que se iba a modificar, había que plantear hasta qué punto era necesario modificar el programa para implementar la paralelización.

En el programa original, la ejecución de la sección en la que se calculan los vectores de características sigue el siguiente esquema:

for (nº de usuarios) {
*for (nº de muestras recogidas en cada sesión * nº de sesiones) {*
while (ventana actual < ventana final) {
-calcula el tamaño de la ventana que se va a usar en la iteración
-calcula el conjunto de valores que forman el vector de características
}
-se van almacenando los vectores de características de la toma de datos que ha terminado en una estructura temporal
}
}
-se exporta la tabla con los vectores de características a una tabla .csv

Según este esquema, el punto en el que nosotros vamos a añadir la función FMM es el segundo bloque dentro del bucle *while* (resaltado en rojo), el que calcula los valores de los vectores de características, pues este es el principal código que nosotros queremos paralelizar. Al añadir la función foreach con el parámetro *%dopar%* para que se realice de forma paralela, observamos que si se sustituía el bucle *while* por la nueva función *foreach()*, a la hora de realizar la ejecución en paralelo iba a dar problemas, pues el bloque en el que se calcula el tamaño de la ventana (resaltado en verde) con los datos que se van a usar en esa iteración, usa datos que dependen de la ejecución anterior.

Por ello se decidió hacer lo siguiente:

Como el bloque que calcula el tamaño de la ventana actual (texto en verde) es fijo y no depende de ninguno de los datos que hay dentro del bucle *while*, se optó por sacarlo fuera del mismo. Para calcular el inicio y el fin de la ventana actual, únicamente utiliza el parámetro que se le pasa a la función principal, que indica el tamaño de la ventana, y se va sumando recursivamente hasta llegar al final del fichero con los datos de la muestra actual. Por lo tanto, se modificó para que se cree un nuevo bucle que obtiene en un vector los instantes de inicio y fin de cada ventana, y este vector se utiliza dentro del bucle *while*.

Con este cambio, sumado a la modificación del bloque que calcula el conjunto de valores que forman el vector de características que se hizo en la Iteración 2, en la que se añade la función FMM y se elimina las operaciones que no se usen, ya se puede añadir la función *foreach()*, y el esquema del código queda de la siguiente manera:

for (nº de usuarios) {
*for (nº de muestras recogidas en cada sesión * nº de sesiones) {*
while (ventana actual < ventana final) {
-calcula los instantes inicio y fin de cada ventana y lo devuelve en el vector tamaño_ventanas

}

foreach(longitud del vector tamaño_ventanas)dopar% {

-calcula el conjunto de valores que forman el vector de características

}

-se van almacenando los vectores de características de la toma de datos que ha terminado en una estructura temporal

}

}

-se exporta la tabla con los vectores de características a una tabla .csv

Cabe mencionar que la longitud del vector *tamaño_ventanas* es igual al número de ventanas que tiene la muestra de la iteración actual, luego sirve perfectamente para indicar a la función *foreach()* el número de veces que se tiene que ejecutar su contenido (función de obtención del vector de características de las ventanas).

Contando ahora con la estructura básica en pseudocódigo para aplicar la paralelización al código existente, se pasa a su implementación.

Implementación

En esta fase de implementación, se siguió el esquema explicado en la fase de diseño anterior, las dificultades se encontraron al tratar de devolver correctamente cada estructura de datos de cada iteración del bucle *foreach()* para almacenarla en la tabla final, ya que había que generar cada vector de características en cada iteración, ir almacenándolos y al terminar todas las ejecuciones añadirlos al fichero final fila a fila.

En esta ocasión, para poder llevar a cabo la implementación, hubo que preparar las nuevas máquinas virtuales cedidas por el departamento con el software necesario para desarrollar las modificaciones, ejecutar los programas y realizar el control de versiones (R, RStudio, Visual Studio Code, Git). También se instalaron las bibliotecas que utiliza el programa, entre las que se encuentran la que incluye la función de ajuste, 'FMM' y la que es necesaria para implementar las operaciones de paralelización, 'doParallel'.

Hay que mencionar que para que la función *foreach()* funcione, es necesario iniciar un clúster con la función *makeCluster()* [8], y eliminar este clúster al terminar la ejecución del bucle *foreach()*. La función *makeCluster()* crea copias del proceso R que está corriendo para poder dividir la ejecución. A esta función se le pasan dos parámetros, el primero es el número de cores de la máquina entre los que se podrá dividir la ejecución (se usó la función *detectCores()* [5], que obtiene el número de cores disponibles automáticamente). El otro parámetro que recibe indica el tipo de clúster que se va a crear, pudiendo elegir la versión que utiliza la función *fork()* ("FORK"), o la alternativa si no se está en un sistema UNIX ("PSOCK"), que es la que se ha utilizado.

Una vez que se tiene el clúster creado, se añade la función *foreach()* con el parámetro *%dopar%* para activar esta ejecución en paralelo. Esta función necesita que se le especifiquen como parámetros un par de elementos:

- Necesita el nombre de las bibliotecas que sean necesarias para utilizar las operaciones que tenga dentro el bucle. En nuestro caso se añadió el paquete 'FMM', para poder utilizar la función que ajusta los datos.
- El otro parámetro necesario es el *.export*, en el que se especifican las variables que no están definidas en el mismo entorno del bucle *foreach*, pero que sí que son usadas en el interior de este. En nuestro programa, la estructura de datos que contiene los instantes de tiempo y los valores recogidos por los sensores en cada instante son las variables que se especificaron en este parámetro, ya que son necesarias como entrada para la función de ajuste FMM.

Pruebas

Cuando estaban listos todos los preparativos para la ejecución, primero hubo que verificar que el funcionamiento con la paralelización fuese el correcto.

Para ello, se planteó la siguiente prueba: se utilizaría la primera muestra de cada sesión (en total dos muestras) de los dos primeros usuarios, realizando el ajuste con dos componentes. Se definió de forma que no requiriese una gran cantidad de tiempo, pero permitiese comprobar que todo funcionaba correctamente. Tras ello, se ejecutó la prueba anterior con la versión previa del programa, la que ya incluye el uso de la función FMM, pero sin paralelizar el código, para obtener los datos que se consideran los correctos, con los que comparar posteriormente. Tras esto, se ejecutó la misma prueba, esta vez sobre la versión del programa que contaba con la paralelización ya hecha, y se compararon las tablas resultado obtenidas utilizando el comando *fc* [19] en Windows, que permite encontrar las diferencias entre el contenido de dos ficheros. Cabe mencionar que las pruebas se realizaron tanto en el fichero que utiliza los datos de las coordenadas de los sensores y también en el programa que usa el módulo de estas coordenadas.

Al pasar estas pruebas, también se pudo verificar, de nuevo, que esta modificación reducía el tiempo de la ejecución al paralelizar en ambos casos, para el programa del módulo y el de las coordenadas (Tabla 4.3).

	Sin paralelizar	**Paralelizado**
Programa con el módulo de los datos	1 minuto y 52 segundos	41 segundos
Programa con las coordenadas de los datos	6 minutos y 55 segundos	2 minutos y 24 segundos

Tabla 4.3 Diferencia de tiempo de ejecución entre versiones con paralelización y sin paralelización

Despliegue

En esta ocasión el despliegue fue similar al de la Iteración 2: se repitió el trabajo de ajuste de las rutas de entrada y salida de los datos a las nuevas máquinas.

Resultados

En cuanto se contó con todo el código y el entorno listo para la ejecución, se tomó la decisión de ejecutar primero en una máquina el programa que usa los datos del módulo de las coordenadas únicamente con los datos del sensor acelerómetro del reloj Motorola, para hacernos una idea del tiempo requerido.

Esta ejecución tardó alrededor de siete horas en las máquinas del departamento que contaban con dos cores. En este momento se pidió la ampliación de una de las máquinas para que contase con más cores y poder aumentar la velocidad de ejecución. Tras esta solicitud, se redimensionó una de estas máquinas a 16 cores y se pudo ejecutar otra de las pruebas para el otro sensor del mismo reloj en 2 horas y media.

Teniendo estos tiempos en cuenta, se consideró viable realizar las ejecuciones de todos los datos, pues para el módulo se tardaría alrededor de 10 horas, puesto que se realizan cuatro ejecuciones, dos por cada sensor (acelerómetro y giróscopo) de cada dispositivo (Motorola y Microsoft), multiplicado por dos horas y media que tardaba cada ejecución.

Para las ejecuciones que usan los datos sin calcular el módulo (cálculo de características sobre cada coordenada X, Y y Z del sensor), sobre las que ya se ha explicado que realizan tres ajustes en lugar de uno (como ocurriría si se utilizase el módulo de las coordenadas), el tiempo se multiplica por tres. Con el mismo cálculo que en el párrafo anterior se obtiene esta vez un tiempo aproximado de 30 horas.

4.5 Iteración 4. Normalización

Mientras se esperaba a que terminasen de realizarse las ejecuciones mencionadas en la iteración anterior, se decidió añadir una nueva variación al programa. Al observar los datos que devuelve la función de ajuste, se puede apreciar que, en algunos casos, los valores para cada uno de los parámetros del modelo tienen grandes variaciones.

	A	**Alpha**	**Beta**	**Omega**
Wave 1	0.3550	4.7197	2.4915	0.0563
Wave 2	0.2856	2.2121	5.1578	0.0629
Wave 3	0.3590	1.4012	5.9559	0.0630
Wave 4	0.2593	5.5498	1.4736	0.0545
Wave 5	0.2104	4.2901	3.1686	0.0469
Wave 6	0.3447	2.9087	3.2761	0.0511
Wave 7	0.2111	1.0338	1.0445	0.0363
Wave 8	0.2081	0.2758	2.7769	0.0379
Wave 9	0.1828	3.7208	3.1937	0.0806
Wave 10	0.3329	5.1199	1.4883	0.0455
Wave 11	0.2519	1.6900	4.2038	0.0401
Wave 12	0.2246	2.5456	3.9016	0.0505
Wave 13	0.2678	0.5699	0.4314	0.0513
Wave 14	0.3251	5.9730	0.3837	0.0533
Wave 15	0.1183	4.1002	0.7212	0.0321
Wave 16	0.1824	6.2793	5.0827	0.0350

Tabla 4.4 Muestra de los datos que devuelve la función FMM de ajuste con 16 componentes

La Tabla 4.4 muestra los valores que devuelve una ejecución de la función FMM con 16 componentes, y se aprecia que, para el parámetro Omega, estos datos fluctúan entre 0.806 como valor máximo y 0.0321 como valor mínimo, una variación pequeña en comparación con la que se aprecia entre los valores del parámetro Alpha, por ejemplo, que oscilan entre 6,2793 y 0.02758.

Por este motivo, se decidió incluir en el programa un mecanismo para normalizar los valores de estos parámetros. La normalización [13] busca reducir esta variación de los valores de un conjunto de datos, y añadir unos límites entre los que oscilen estos valores, que serán diferentes según el tipo de normalización escogida.

Los tipos de normalización que se consideró añadir fueron los siguientes:

- Normalización Z: Se realiza obteniendo la media (μ) y la desviación típica o desviación estándar (σ) [4] de cada conjunto de datos que se quiera normalizar. Para cada valor (X) se le resta el valor de la media del conjunto y después se divide entre la desviación típica.

$$\frac{X - \mu}{\sigma}$$

- Normalización mínimo-máximo: En este caso, se calculan los valores máximo (Max) y mínimo (Min) del conjunto de datos a normalizar. Para cada valor (X) se le resta primero el valor mínimo y se divide entre la diferencia del valor máximo y el valor mínimo del conjunto.

$$\frac{X - Min}{Max - Min}$$

- Normalización entre el máximo: Este tipo de normalización para cada valor (X) se divide entre el máximo valor (Max) que exista en el conjunto de datos a normalizar.

$$\frac{X}{Max}$$

Análisis

Teniendo la nueva modificación presente, se mantienen los requisitos de las anteriores iteraciones, pero se añade el RF-04 sobre la normalización, las tablas de requisitos quedan de la siguiente forma:

Requisitos funcionales:

RF-01	El sistema debe permitir obtener los vectores de características a partir de la biblioteca FMM aplicado sobre los datos del módulo de las tres coordenadas de los sensores.
RF-02	El sistema debe permitir obtener los vectores de características a partir de la biblioteca FMM aplicado sobre los datos de las tres coordenadas de los sensores (X, Y y Z).
RF-03	El sistema deberá permitir realizar la obtención de vectores de características de forma paralela.
RF-04	El sistema deberá permitir especificar si se desea aplicar uno de los tres tipos de normalización (Z, mínimo-máximo, máximo) a los valores de los parámetros que devuelve la función de ajuste FMM (A, Alpha, beta y omega).

Requisitos no funcionales:

RNF-01	La salida del programa debe mantener el formato de los vectores de características obtenidos tal y como se tiene en el sistema previo a la modificación. (Una tabla con una fila para cada muestra de datos de cada usuario, con los valores devueltos por la función de obtención de características).

Diseño

En esta fase, al tener creada y funcionando la paralelización de los datos, no se observó ningún problema a la hora de añadir el código que se encargase de calcular los nuevos valores normalizados justo después de ejecutar la función de ajuste FMM.

Se decidió incluir un nuevo parámetro en la función general *'tipoNormalizacion',* con el que se especifique el tipo de normalización que se desea realizar. Se comprobará tras la ejecución de la función de ajuste FMM el valor de esta variable, y si coincide con alguno de los tipos definidos en el requisito RF-04, se realizará la normalización correspondiente; si no coincide, se realizará la ejecución básica, sin normalizar los datos. Además, si se realiza algún tipo de normalización, se añadirá el tipo seleccionado al nombre del fichero final para poder diferenciarlos.

Implementación

Al implementar esta modificación, se añadió el nuevo parámetro a la función y, mediante un bloque switch en el que, si el contenido de este parámetro coincide con alguna de las cadenas que representan los tipos de normalización ('Z', MIN-MAX', 'MAX'), se aplica la fórmula correspondiente, como se ha explicado al inicio de esta iteración, sobre cada uno de los parámetros que devuelve la función de ajuste FMM.

Este bloque switch se añadió justo después de la llamada a la función FMM, y antes de recorrer la estructura de datos que devuelve esta función para leer los valores y añadirlos a la estructura de datos temporal en la que se guardan los valores (manteniendo el formato original del programa). Para ello, se aplican las fórmulas de la normalización, descritas previamente en esta iteración, sobre los datos de cada parámetro que contiene el objeto que devuelve la función de ajuste FMM, y se almacenan en otra estructura de datos temporal que es la que se utiliza posteriormente en el bucle que la recorre para guardar los datos con el formato correcto.

Se han utilizado las siguientes funciones incluidas en R *mean()* [11] para calcular la media del conjunto de datos a normalizar, *sd()* [17] para calcular la desviación estándar, y se usaron las funciones *max()* y *min()* [9] para calcular el máximo y el mínimo respectivamente.

El mismo bloque switch se añadió tanto al programa que usa el módulo de las coordenadas de los sensores, como el que utiliza las coordenadas directamente. Sin embargo, en este último se añadió otra pequeña modificación, buscando simplificar este proceso y evitar duplicar el código. Se trata de incluir un bucle en el que se entra tres veces (una por coordenada existente), y se realiza la función de ajuste FMM, la normalización si está activada y se guarda el resultado en la estructura de datos temporal cumpliendo con el formato del programa original.

Pruebas

Primero se comprobó que las funciones que normalizaban los datos funcionaban correctamente, aplicando los tres tipos de normalización a un conjunto de datos pequeño para poder hacer los cálculos a mano fácilmente y comprobar que devolvía los valores correctos en cada caso.

En este caso, las pruebas debían verificar que la normalización funcionaba correctamente y que se cumplían las fórmulas definidas en el inicio de esta iteración. Para ello, se realizó una ejecución con el programa paralelizado de la Iteración 3. Paralelización., en la que el número de componentes de la función de ajuste FMM era 4. Sobre estos datos se realizaron los tres tipos de normalización y se comprobó que los datos que devolvía la nueva versión del programa eran iguales y devolvía los mismos valores. Hay que mencionar que estas pruebas se hicieron de igual forma tanto para el programa que utiliza las coordenadas de los sensores, como para el que utiliza el módulo de estas coordenadas.

Despliegue

Tras comprobar el correcto funcionamiento de la modificación, el despliegue, de nuevo, se limitó a ajustar las rutas con las que se introducían y devolvían los datos.

Resultados

La ejecución se inició en la máquina virtual que contaba con 16 cores. En primer lugar, se ejecutó el programa que usa los datos del módulo de las coordenadas, para el que se aplicaron los tres tipos de normalización sobre los datos de los dos sensores, de los dos dispositivos. El tiempo necesario para ejecutarlo todo fue de un día, nueve horas y treinta minutos. Posteriormente se iniciaron las mismas pruebas para los datos que usan las coordenadas y en este caso el tiempo de ejecución fueron tres días y veintidós horas.

4.6 Iteración 5. Obtención de resultados.

Una vez se cuenta con los vectores de características que ha generado la biblioteca FMM se pasó a usar estos vectores de resultados en el sistema de reconocimiento biométrico para obtener los resultados.

Sin embargo, al intentar ejecutar el programa que entrena al sistema y genera los resultados se observó que este programa utilizaba los datos de la base de datos biométrica obtenida durante la realización del TFM de Irene Salvador Ortega como parte del entrenamiento, ya que en este trabajo posterior se añadieron más usuarios a la base de datos y haciendo uso de ellos el sistema tendría un entrenamiento más completo.

Por lo tanto, para poder ejecutar este entrenamiento con los mismos usuarios que se habían usado en la ejecución inicial con los vectores de características originales, había que generar los vectores de características nuevos, usando la biblioteca FMM para los usuarios de la base de datos biométrica del TFM.

Análisis

RF-01	El sistema debe permitir obtener los vectores de características originales sobre los datos del módulo de las tres coordenadas de los sensores. (Para los usuarios del TFM).
RF-02	El sistema debe permitir obtener los vectores de características originales sobre los datos de las tres coordenadas de los sensores (X, Y y Z). (Para los usuarios del TFM).

Requisitos no funcionales:

RNF-01	Los experimentos se realizarán solo para el conjunto de valores de entrada especificados en el apartado 4.2

Diseño

Al igual que en la iteración 1, al no necesitar realizar una implementación de ningún cambio, ya que solo se cambiaba el conjunto de datos de entrada, esta fase de diseño se omitió.

Implementación

La única implementación necesaria fue modificar la ruta de los datos a utilizar.

Pruebas

Al hacer las pruebas, el programa sin modificaciones adicionales (aparte de la ruta de los datos de entrada), justo después de haber obtenido los vectores de características normalizados de los usuarios del TFG, no podía ejecutarse para los usuarios del TFM porque devolvía un error. Se estuvo buscando el motivo, ya que, en principio, no hay diferencia en la recogida y el almacenamiento de los datos de los usuarios de los dos trabajos, pues se usaron ambos conjuntos de datos en el sistema del TFM de Irene S. y también se usaron ambos al generar los resultados iniciales con los vectores de características originales en este propio trabajo.

El error persistía pese a que los datos parecían correctos, y se pensó durante un tiempo que podría darse debido a un problema con la ejecución paralela de los datos, pero quitando este apartado se mantenía el error. También se comprobó si el problema estaba relacionado con un agotamiento de los recursos de la máquina virtual, pero ejecutando uno a uno los usuarios, seguía ocurriendo.

Debido a la falta de más tiempo para seguir investigando el origen de este problema, se decidió eliminar los usuarios que diesen error, dando como resultado que, de los 24 usuarios registrados en el TFM, se usasen únicamente 11 de ellos. Se tuvo en cuenta que habría que ejecutar de nuevo los programas originales con los datos originales para generar los resultados del sistema usando los vectores de características sin modificar, para poder comparar con los datos que se obtendrían usando los nuevos vectores de características.

Despliegue

El despliegue no requirió acciones adicionales.

Resultados

Una vez que se tenían ya los vectores de características de todos los usuarios, se comenzó a adaptar el programa que entrena los distintos clasificadores y calcula los errores sobre los ficheros de prueba, para usar los nuevos vectores de características. Principalmente se cambiaron las rutas de las carpetas de entrada de los datos y se cambiaron los nombres de las variables de los vectores de características para que usase las nuevas. Durante este proceso se advirtió la necesidad de ejecutar un programa más sobre los vectores de características de los usuarios. Había que obtener los vectores de características eliminando las ventanas de tiempo en las que el ruido en los datos fuese demasiado alto. Este proceso no supuso ningún problema más allá de adaptar las rutas de entrada y salida de los datos, y los nombres de las variables de los vectores de características modificados.

Con esta iteración completada se pudo comenzar con las ejecuciones de los diferentes experimentos para la obtención de los resultados finales, que se muestran en el siguiente capítulo.

5 Resultados

En este capítulo de resultados se van a mostrar las tablas con los datos de las medidas de error del sistema, para cada uno de los escenarios de pruebas.

El problema biométrico abordado aquí es verificación del usuario, donde el objetivo es indicar si la entrada al sistema pertenece o no a un determinado sujeto. Este es un problema de clasificación binario, para el que se definen dos tipos de error:

- Tasa de Falsos Negativos (TFA): es la proporción de muestras de prueba auténticas que han sido incorrectamente clasificadas como no pertenecientes al usuario.
- Tasa de Falsos Positivos (TFP): es la proporción de muestras de pruebas de impostores que han sido incorrectamente clasificadas como pertenecientes al usuario

El valor de estas tasas de error depende del valor del umbral de decisión. Por eso, para dar una visión completa del rendimiento de un sistema se suelen usar curvas ROC (Receiver Operating Characteristic) (Figura 5.1), que muestra la evolución de estos errores con respecto al umbral.

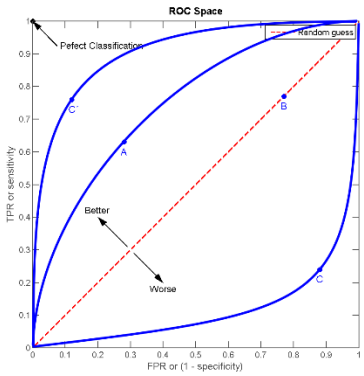

Figura 5.1

Sin embargo, cuando el número de comparaciones es alto, como pasa en este trabajo, el uso de este tipo de curvas no es práctico y lo que se hace es buscar un punto de funcionamiento (umbral) del sistema y mostrar el error en él, resumiendo, así, el rendimiento del sistema en un único valor. En biometría uno de los valores más utilizados es la Tasa de Equierror o Equal Error Rate (EER), que es el error del sistema cuando la TFP se iguala a la TFN; en la curva ROC, el valor del punto donde la curva se corta con la diagonal (puntos rojos en la (Figura 5.2). Cuanto más cercano sea este valor a 0, mejor será el rendimiento del sistema.

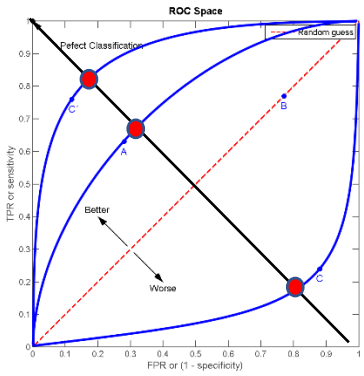

Figura 5.2

Este será el valor usado en este trabajo, medido en %. La forma de operar será la siguiente:

1. A partir de las salidas del clasificador para las muestras de pruebas auténticas e impostoras de un determinado usuario se calculará la EER para ese usuario. Tendremos, entonces, una medida de error para cada usuario de la base de datos biométrica,
2. El valor mostrado en las tablas será la media de esos valores de EER individuales.

Los datos de error van a ser mostrados para cada uno de los clasificadores. Cada una de ellas muestra los resultados con respecto a:

- El escenario de entrenamiento*: MonoMono* (entrenamiento y prueba de la misma sesión) y *MultiMono* (entrenamiento con la primera sesión y prueba con la segunda sesión). Explicados en la fase de análisis del apartado 4.2.
- El dispositivo: *Moto* (reloj Motorola) y *Micro* (pulsera Microsoft).
- El sensor: *ACC* (acelerómetro) y *GYR* (giróscopo).
- La coordenada del sensor: X, Y, Z y Módulo (*M*).
- El número de ventanas fusionadas en el post-procesamiento, en el apartado titulado Post-procesamiento: 1 (no se usa post-procesamiento),4,8 y 12 ventanas.
- El tipo de normalización aplicada: *Sin norm* (Sin normalizar), *Z* (Z-Norm) y *MAX* (dividiendo por el máximo) y *Min-max.*

En todas las tablas y para todos los casos, hay una fila que muestra los resultados del sistema de referencia [28], fila *Original*, para así poderlos comparar con el resto que son los obtenidos con las nuevas características.

Para una mejor visualización de los resultados, se va a poner cada clasificador en una página distinta, de manera que toda la tabla se muestre en esa página y, así, evitar su división entre varias, que siempre dificulta la comparación.

En cada una de las tablas, para cada combinación de los parámetros (entrenamiento, dispositivo, sensor, coordenada y solapamiento de ventana) se va a resaltar en negrita el menor valor del equierror obtenido entre las 4 diferentes modificaciones realizadas (modificación sin normalización, y la modificación con los tres tipos de normalización) y la versión original del sistema. Ya que así se aprecia mejor cuál de las cinco opciones es la que funciona mejor.

En caso de que varias opciones tengan el mismo valor mínimo, se señalarán en este orden (dato del sistema original – dato del sistema modificado sin normalización - dato del sistema modificado con normalización Z - dato del sistema modificado con normalización máximo - dato del sistema modificado con normalización mínimo-máximo.

5.1 1-NN

		Monomono															
		Moto								Micro							
		ACC				GYR				ACC				GYR			
		X	Y	Z	M	X	Y	Z	M	X	Y	Z	M	X	Y	Z	M
1	Original	**17.00**	**22.99**	**25.04**	**20.34**	**20.12**	**22.58**	**18.87**	**21.89**	**23.25**	**21.07**	**18.28**	**16.65**	**24.30**	**18.97**	**24.76**	**20.76**
	Sin norm	40.57	40.84	38.26	41.30	26.22	26.72	26.49	26.51	45.10	44.59	39.09	41.41	29.88	25.76	30.43	36.41
	Z	38.98	41.09	38.34	39.63	25.69	26.39	26.32	26.43	43.77	44.03	39.29	41.75	28.13	28.14	28.14	37.27
	MAX	38.94	38.86	37.24	37.76	25.67	25.51	25.67	26.21	43.45	44.16	40.38	39.02	28.11	28.11	28.11	37.31
	Min-max	38.84	39.42	38.35	38.21	25.67	25.51	25.67	26.21	43.99	44.74	40.24	39.94	28.11	28.11	28.11	37.31
4	Original	**13.44**	**19.83**	**23.13**	**15.85**	**14.42**	**18.17**	**12.73**	**17.24**	**20.10**	**17.10**	**13.89**	**11.77**	**20.63**	**13.64**	**19.99**	**15.87**
	Sin norm	36.50	37.56	39.00	39.96	25.92	25.93	25.53	25.93	44.40	44.03	37.66	38.90	29.50	26.35	29.42	33.07
	Z	33.63	37.96	38.82	37.70	25.74	25.83	25.67	25.75	39.97	44.77	38.84	39.86	29.67	29.66	29.71	34.98
	MAX	32.64	35.80	35.44	35.92	25.83	25.67	25.46	25.79	41.25	42.06	37.94	34.09	29.64	29.67	29.68	34.98
	Min-max	34.99	36.23	37.74	34.35	25.83	25.64	25.54	25.79	41.55	43.08	38.15	36.48	29.63	29.68	29.70	34.98
8	Original	**10.78**	**17.36**	**16.50**	**12.00**	**11.47**	**14.49**	**8.77**	**13.19**	**17.00**	**14.18**	**9.93**	**8.70**	**16.01**	**10.65**	**18.23**	**12.43**
	Sin norm	29.98	33.02	33.83	35.29	23.88	23.41	23.39	23.46	41.52	40.31	34.09	35.07	28.40	24.09	27.23	33.65
	Z	27.48	34.29	33.61	33.87	23.55	23.55	23.51	23.43	38.91	40.87	33.89	35.53	28.47	28.99	28.47	36.37
	MAX	26.27	31.80	31.41	32.35	23.71	23.44	23.34	23.47	37.75	39.02	33.95	29.51	28.63	28.71	28.52	36.37
	Min-max	27.68	32.30	37.62	30.00	23.62	23.51	23.34	23.47	39.70	38.84	32.49	32.50	28.67	28.83	28.48	36.37
12	Original	**8.54**	**15.68**	**14.17**	**9.51**	**7.29**	**12.02**	**4.88**	**9.19**	**15.91**	**12.54**	**7.16**	**6.29**	**14.12**	**9.51**	**16.24**	**10.31**
	Sin norm	25.65	28.46	29.41	31.66	23.26	22.80	22.80	23.11	40.91	39.39	29.86	31.25	25.13	23.88	26.04	27.55
	Z	24.96	32.09	30.04	31.14	23.01	22.80	22.98	22.78	34.65	39.33	31.08	33.27	28.79	29.18	29.00	31.03
	MAX	21.78	27.43	27.85	27.93	23.08	22.87	22.80	22.92	34.78	36.44	29.25	23.94	29.08	28.92	28.83	31.03
	Min-max	25.13	28.52	27.83	26.35	22.98	22.80	22.80	22.92	35.11	37.02	27.35	29.80	29.21	29.14	29.06	31.03
		MultiMono															
1	Original	**16.53**	**23.24**	**25.48**	**21.86**	**25.46**	**22.31**	**27.58**	**29.67**	**29.44**	**31.64**	**25.13**	**20.98**	**30.93**	**26.58**	**30.24**	**29.08**
	Sin norm	43.09	44.58	41.04	43.36	35.84	35.74	35.78	35.76	42.35	41.20	41.19	41.18	35.95	30.68	34.29	42.44
	Z	41.93	42.97	42.75	40.10	35.74	35.64	35.63	35.72	39.96	42.03	41.13	39.01	34.11	34.11	34.05	43.26
	MAX	42.05	48.94	40.37	40.50	35.60	35.62	35.62	35.71	42.67	42.44	41.47	39.20	34.03	33.98	33.96	43.26
	Min-max	42.00	44.79	41.16	39.87	35.64	35.60	35.58	35.71	40.73	42.24	40.88	39.12	34.03	33.99	33.95	43.26
4	Original	**14.17**	**20.42**	**23.04**	**17.54**	**22.10**	**18.98**	**25.02**	**27.37**	**25.50**	**29.05**	**22.47**	**16.62**	**28.65**	**24.57**	**27.86**	**27.41**
	Sin norm	41.98	44.67	38.49	40.83	35.33	35.29	35.32	35.28	42.13	39.32	39.28	38.79	33.72	30.15	33.36	40.56
	Z	41.73	43.14	39.74	38.70	35.35	35.32	35.29	35.29	38.77	40.28	39.27	36.88	34.41	34.44	34.35	41.30
	MAX	40.75	44.82	37.06	37.54	35.24	35.29	35.29	35.18	42.51	42.15	39.44	34.90	34.34	34.33	34.30	41.29
	Min-max	41.67	44.14	38.55	36.14	35.26	35.34	35.27	35.19	39.75	41.69	38.96	33.47	34.34	34.34	34.30	41.29
8	Original	**11.69**	**18.05**	**19.91**	**15.09**	**20.50**	**17.19**	**24.61**	**26.02**	**23.00**	**26.67**	**19.10**	**14.24**	**24.87**	**22.73**	**24.71**	**24.70**
	Sin norm	39.37	43.25	36.17	39.42	34.16	34.05	34.07	33.99	40.61	36.54	37.68	35.70	32.45	28.15	31.63	39.31
	Z	38.65	40.26	37.04	34.51	34.26	34.09	34.11	33.99	34.40	40.25	37.42	33.22	34.00	33.89	33.94	40.28
	MAX	38.93	43.73	34.23	32.85	34.11	34.08	34.10	33.96	40.91	40.71	36.33	31.92	33.88	33.80	33.83	40.28
	Min-max	39.23	43.13	34.98	31.08	34.12	34.12	34.07	33.96	37.97	40.15	36.24	29.28	33.89	33.80	33.87	40.28
12	Original	**10.42**	**15.76**	**17.98**	**12.76**	**17.99**	**15.37**	**23.05**	**24.20**	**22.51**	**25.94**	**17.36**	**12.24**	**22.31**	**21.45**	**22.47**	**23.21**
	Sin norm	40.98	46.74	37.64	42.29	33.42	33.13	33.25	33.18	39.21	34.30	39.23	33.74	30.94	26.81	30.82	37.89
	Z	44.92	43.61	33.04	31.04	33.44	33.26	33.26	33.23	32.75	37.14	34.66	30.78	33.99	33.88	34.12	38.50
	MAX	41.07	51.29	32.10	35.91	33.25	33.18	33.30	33.20	39.88	39.79	34.38	28.75	33.96	33.99	33.92	38.50
	Min-max	40.68	46.30	30.73	27.97	33.30	33.23	33.21	33.21	35.59	42.75	32.30	25.56	33.92	33.83	33.94	38.50

Para el clasificador 1NN se observa que, en todos los casos, la tasa de equierror es menor en la versión original del sistema. Los nuevos vectores de características no mejoran este dato con ninguna de las normalizaciones aplicadas para el clasificador 1-NN.

5.2 SVM kernel Radial

		Monomono															
		Moto								Micro							
		ACC				GYR				ACC				GYR			
		X	Y	Z	M	X	Y	Z	M	X	Y	Z	M	X	Y	Z	M
1	Original	9.92	0.77	0.40	0.97	1.89	3.61	1.40	1.91	7.34	2.65	5.71	3.07	2.24	1.98	1.29	1.04
	Sin norm	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	Original	7.29	**0.00**	**0.00**	0.42	0.07	0.29	0.07	1.14	4.17	2.26	4.89	0.25	0.82	0.12	0.06	0.07
	Sin norm	**0.00**	0.00	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	Original	6.22	**0.00**	**0.00**	0.18	**0.00**	**0.00**	**0.00**	0.22	2.76	1.64	4.57	**0.00**	0.38	**0.00**	**0.00**	**0.00**
	Sin norm	**0.00**	0.00	0.00	**0.00**	0.00	0.00	0.00	**0.00**	**0.00**	**0.00**	**0.00**	0.00	**0.00**	0.00	0.00	0.00
	Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12	Original	6.43	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.15	1.27	0.82	3.54	**0.00**	0.13	**0.00**	**0.00**	**0.00**
	Sin norm	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.00**	**0.00**	**0.00**	0.00	**0.00**	0.00	0.00	0.00
	Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		MultiMono															
1	Original	7.44	0.69	0.62	0.86	0.70	2.21	1.14	0.97	1.81	0.06	2.70	1.54	1.08	0.36	0.57	0.65
	Sin norm	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	Original	4.67	0.19	0.11	0.23	0.29	0.10	0.75	0.16	0.51	**0.00**	1.78	0.13	0.16	**0.00**	0.03	0.06
	Sin norm	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.00	**0.00**	**0.00**	**0.00**	0.00	**0.00**	**0.00**
	Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	Original	4.19	0.03	**0.00**	0.16	**0.00**	**0.00**	0.43	**0.00**	**0.00**	**0.00**	1.27	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Sin norm	**0.00**	**0.00**	0.00	**0.00**	0.00	0.00	**0.00**	0.00	0.00	0.00	**0.00**	0.00	0.00	0.00	0.00	0.00
	Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12	Original	3.57	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.14	**0.00**	**0.00**	**0.00**	0.95	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Sin norm	**0.00**	0.00	0.00	0.00	0.00	0.00	**0.00**	0.00	0.00	0.00	**0.00**	0.00	0.00	0.00	0.00	0.00
	Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Al ver estos resultados del clasificador SVM con Kernel Radial se aprecia que, para los nuevos vectores de características, la media del error es 0 en todos los experimentos, mejorando al sistema de referencia en todos los casos.

5.3 SVM Kernel Lineal Orden 5

		Monomono															
		Moto								Micro							
		ACC				GYR				ACC				GYR			
		X	Y	Z	M	X	Y	Z	M	X	Y	Z	M	X	Y	Z	M
1	Original	32.21	10.83	3.51	16.14	9.60	3.02	0.61	9.08	37.66	35.98	4.00	17.35	1.30	6.42	7.96	5.11
	Sin norm	**0.00**	0.62	0.82	0.18	0.57	3.67	2.27	0.14	**0.00**	0.41	0.68	**0.00**	0.07	1.09	5.33	1.14
	Z	0.61	**0.00**	**0.00**	0.34	**0.00**	**0.00**	3.12	**0.00**	1.87	**0.00**	**0.07**	0.40	0.06	0.85	3.20	**0.00**
	MAX	0.35	0.00	0.20	**0.06**	0.07	0.00	1.13	0.00	1.42	0.00	0.07	0.20	**0.00**	**0.13**	**2.01**	0.00
	Min-max	0.96	0.00	0.14	0.20	0.00	0.24	3.08	0.00	2.02	0.00	0.07	0.31	0.23	0.13	3.77	0.11
4	Original	36.82	7.75	2.71	13.42	8.12	2.26	**0.00**	5.37	37.32	34.83	3.22	16.00	**0.00**	2.93	4.56	3.62
	Sin norm	**0.00**	**0.00**	0.14	**0.00**	0.53	2.10	2.01	0.07	**0.00**	**0.00**	0.06	**0.00**	0.00	0.20	1.70	0.25
	Z	0.00	0.00	**0.00**	0.00	**0.00**	**0.00**	1.50	**0.00**	0.00	0.00	**0.00**	0.00	0.00	**0.00**	0.98	**0.00**
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.91**	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	2.30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.77	0.00
8	Original	36.89	7.36	2.27	13.73	5.74	1.71	**0.00**	4.26	37.42	38.92	3.09	14.98	**0.00**	3.09	3.21	3.10
	Sin norm	**0.00**	**0.00**	0.07	**0.00**	0.33	2.22	2.11	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.20	1.49	**0.00**
	Z	0.00	0.00	**0.00**	0.00	**0.00**	**0.00**	0.43	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	0.18	0.00
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.13**	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.73	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.35	0.00
12	Original	36.91	5.77	3.28	13.45	5.06	1.30	**0.00**	3.93	37.51	38.92	3.15	13.51	**0.00**	3.77	2.24	2.68
	Sin norm	**0.00**	**0.00**	**0.00**	**0.00**	0.07	1.39	1.32	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.00	**0.00**	1.40	**0.00**
	Z	0.00	0.00	0.00	0.00	**0.00**	**0.00**	0.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	0.00
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	1.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.86	0.00
		MultiMono															
1	Original	31.45	12.21	5.83	18.02	5.85	4.84	1.01	5.48	21.64	30.84	3.03	20.30	2.82	7.21	11.29	6.80
	Sin norm	**0.00**	0.17	0.37	0.46	0.10	0.48	1.56	0.06	**0.07**	0.59	0.74	0.35	0.28	4.69	4.89	4.05
	Z	0.86	**0.00**	**0.07**	0.21	**0.00**	0.06	3.69	**0.00**	1.03	**0.00**	**0.03**	0.40	**0.00**	1.04	1.48	**0.00**
	MAX	0.48	0.21	0.13	0.25	0.73	**0.00**	**0.80**	0.03	0.70	0.06	0.03	**0.21**	0.00	**0.20**	**1.05**	0.00
	Min-max	0.88	0.03	0.13	**0.20**	0.03	0.10	2.76	0.04	1.57	0.33	0.03	0.36	0.00	0.58	1.92	0.20
4	Original	35.55	10.25	4.11	15.62	2.45	4.35	0.30	2.75	20.98	30.13	1.85	18.17	1.28	3.53	8.60	5.00
	Sin norm	**0.00**	0.07	0.03	0.07	**0.00**	0.25	0.06	**0.00**	**0.00**	0.29	0.37	0.07	0.21	2.74	2.94	2.38
	Z	0.00	**0.00**	**0.00**	**0.00**	0.00	0.00	0.03	0.00	0.00	**0.00**	**0.00**	**0.00**	**0.00**	0.10	**0.00**	**0.00**
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.17	0.00	0.00	0.00	0.00	0.00	0.12	0.00
8	Original	35.24	9.82	4.90	15.13	1.95	3.71	0.33	1.29	20.80	29.68	1.45	16.98	0.71	2.75	8.31	4.63
	Sin norm	**0.00**	0.07	**0.00**	**0.00**	**0.00**	0.15	**0.00**	**0.00**	**0.00**	0.30	0.30	0.03	0.21	2.16	2.18	1.80
	Z	0.00	**0.00**	0.00	0.00	0.00	**0.00**	0.00	0.00	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12	Original	34.88	9.26	4.75	15.22	1.68	3.41	0.30	0.49	20.87	29.63	1.33	16.88	0.37	1.58	6.90	4.13
	Sin norm	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.33	0.34	**0.00**	0.21	1.04	1.60	0.98
	Z	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.00**	0.00	**0.00**	**0.00**	**0.00**	**0.00**
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

El clasificador SVM con kernel lineal de orden 5 también ha mejorado los resultados del sistema original: los nuevos vectores de características funcionan mejor que los originales. A la hora de decidir cuál de las normalizaciones tiene un mejor rendimiento, el resultado depende del caso, pues todas ellas trabajan bien y varían dependiendo del escenario. Igual pasa con el resto de los parámetros (dispositivo, sensor y coordenada) mostrados.

5.4 Random forest

		Monomono															
		Moto								Micro							
		ACC				GYR				ACC				GYR			
		X	Y	Z	M	X	Y	Z	M	X	Y	Z	M	X	Y	Z	M
1	Original	45.49	12.20	5.03	20.67	14.00	12.98	12.09	14.46	22.32	28.15	13.38	21.66	11.03	7.78	8.75	8.75
	Sin norm	0.49	1.07	0.44	0.27	1.63	0.70	3.94	1.39	0.82	0.41	1.76	0.43	1.20	1.55	2.25	2.51
	Z	**0.00**	0.12	0.06	**0.00**	**0.00**	**0.07**	0.85	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.39	**0.00**
	MAX	0.00	0.07	**0.00**	0.00	0.00	0.13	0.85	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.62	0.06
	Min-max	0.00	**0.00**	0.00	0.00	0.00	0.07	**0.59**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.36**	0.00
4	Original	43.10	9.40	3.55	18.18	9.44	9.43	7.43	9.79	20.86	28.00	12.32	17.72	8.52	4.98	4.78	5.89
	Sin norm	**0.00**	**0.00**	0.07	**0.00**	0.47	0.13	0.69	0.14	0.36	**0.00**	0.26	**0.00**	**0.00**	0.06	1.12	0.91
	Z	0.00	0.00	**0.00**	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.00	**0.00**	0.00	0.00	**0.00**	**0.00**	**0.00**
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	Original	41.94	7.33	2.21	16.77	6.38	7.98	4.88	5.75	20.35	28.02	11.19	16.47	6.25	3.49	3.45	4.70
	Sin norm	**0.00**	**0.00**	**0.00**	**0.00**	0.40	**0.00**	**0.00**	**0.00**	0.14	**0.00**	0.17	**0.00**	**0.00**	**0.00**	0.40	0.58
	Z	0.00	0.00	0.00	0.00	**0.00**	0.00	0.00	0.00	**0.00**	0.00	**0.00**	0.00	0.00	0.00	**0.00**	**0.00**
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12	Original	44.42	6.51	1.79	15.04	5.35	6.83	3.78	3.75	19.29	28.06	9.40	15.07	5.28	2.97	2.85	3.50
	Sin norm	**0.00**	**0.00**	**0.00**	**0.00**	0.27	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.07	**0.00**	**0.00**	**0.00**	0.13	0.14
	Z	0.00	0.00	0.00	0.00	**0.00**	0.00	0.00	0.00	0.00	0.00	**0.00**	0.00	0.00	0.00	**0.00**	**0.00**
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	MultiMono																
1	Original	36.84	10.42	2.57	15.42	9.66	7.52	5.76	8.37	10.89	11.48	9.16	19.38	8.49	6.62	6.53	7.91
	Sin norm	0.49	0.09	0.47	0.45	1.00	0.96	1.45	0.53	0.35	0.79	1.04	0.73	1.04	1.43	1.53	2.54
	Z	**0.00**	**0.03**	0.03	0.03	**0.00**	0.03	0.68	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.06	0.32	**0.00**
	MAX	0.06	0.17	0.06	**0.00**	0.00	**0.00**	0.57	0.00	0.03	0.00	0.00	0.14	0.00	**0.03**	0.09	0.00
	Min-max	0.00	0.08	**0.00**	0.00	0.00	0.00	**0.17**	0.03	0.00	0.00	0.00	0.00	0.00	0.09	**0.06**	0.00
4	Original	31.22	9.27	1.35	12.87	6.85	6.04	4.07	4.54	9.95	10.64	6.57	15.71	4.56	5.16	4.36	5.18
	Sin norm	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.13	0.54	0.03	0.03	**0.00**	0.03	**0.00**	0.06	0.10	0.41	1.90
	Z	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.00**	**0.00**	**0.00**	0.00	**0.00**	0.00	**0.00**	**0.00**	**0.00**	**0.00**
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8	Original	34.07	8.54	0.96	11.53	6.11	4.09	3.09	3.02	8.16	9.60	5.75	14.13	3.20	3.90	3.08	3.80
	Sin norm	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.10	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.04	0.18	1.35
	Z	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.00**	**0.00**
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12	Original	29.64	7.48	0.90	10.95	5.44	3.24	2.28	2.47	7.16	9.36	4.81	12.69	1.81	3.18	2.66	3.10
	Sin norm	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.07	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.87
	Z	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**
	MAX	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Min-max	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Para el clasificador Random Forest también podemos confirmar que los nuevos vectores de características funcionan mejor que los vectores de características originales, pues la tasa de equierror es menor en todos los escenarios. Ahora, la normalización Z-Norm parece tener un mejor rendimiento que el resto, así como usar el módulo en vez de cualquiera de las coordenadas individuales, si nos fijamos en el caso donde más variabilidad hay que es cuando no se usa post-procesamiento (filas agrupadas con valor "1" a su izquierda); no mostrando, en este caso, ninguna diferencia entre dispositivos y sensores.

5.5 Árboles de decisión

		Monomono															
		Moto								Micro							
		ACC				GYR				ACC				GYR			
		X	Y	Z	M	X	Y	Z	M	X	Y	Z	M	X	Y	Z	M
1	Original	57.71	34.08	6.66	20.79	12.06	9.27	**9.65**	9.57	27.51	44.90	21.32	13.49	12.11	**5.30**	**7.59**	6.55
	Sin norm	10.30	**7.83**	**6.85**	8.58	8.64	**6.70**	12.34	11.41	8.62	6.59	**6.62**	**6.68**	7.67	6.77	8.24	8.31
	Z	**9.23**	9.91	8.36	8.01	7.22	13.12	16.02	9.80	8.34	7.31	10.46	8.53	**5.21**	10.66	12.86	7.88
	MAX	10.40	9.69	7.70	8.43	7.76	9.29	11.59	**7.30**	9.27	**5.34**	10.41	9.06	6.05	8.00	13.49	7.84
	Min-max	10.37	8.39	9.39	**7.38**	**6.06**	10.71	11.43	9.43	**8.19**	8.32	9.24	8.38	5.40	9.27	11.79	**5.93**
4	Original	58.14	33.96	5.50	21.92	10.26	10.72	9.77	7.77	28.48	43.94	22.52	13.75	11.61	6.32	6.58	4.73
	Sin norm	10.53	8.10	4.11	7.68	6.52	**6.21**	12.98	10.78	6.81	4.85	6.57	4.99	6.09	6.44	**6.33**	7.72
	Z	7.09	4.88	3.52	3.11	**2.14**	7.07	11.40	3.95	6.40	2.96	7.04	4.59	1.81	5.16	8.45	**1.81**
	MAX	6.92	4.50	**3.13**	3.33	2.05	6.07	10.20	**3.13**	7.96	**1.79**	**4.33**	6.13	1.34	3.49	9.17	3.35
	Min-max	**5.79**	**2.20**	4.32	**2.73**	2.34	7.61	**8.04**	3.85	**3.50**	2.75	5.49	**3.85**	**0.92**	**3.10**	6.89	2.30
8	Original	57.96	33.38	4.19	20.95	8.06	8.26	6.84	3.93	28.26	43.91	21.93	12.04	8.92	4.44	4.74	3.13
	Sin norm	6.23	4.28	2.54	4.04	3.28	4.25	9.80	7.08	4.19	3.30	4.10	2.74	3.28	5.64	4.52	4.77
	Z	3.24	1.54	**0.53**	1.30	0.13	3.13	7.10	0.44	4.24	1.53	4.12	**1.63**	**0.07**	2.08	5.44	**0.07**
	MAX	4.14	1.82	0.65	1.48	0.57	**2.48**	6.16	**0.37**	6.14	**0.00**	2.03	4.04	0.52	1.59	6.77	0.93
	Min-max	**3.20**	**0.83**	2.39	**0.32**	0.**07**	2.58	**2.90**	1.56	**2.07**	0.49	**1.01**	2.22	0.23	**0.99**	**3.31**	0.68
12	Original	55.40	32.09	3.91	19.96	6.90	6.43	4.49	3.23	28.22	43.80	21.28	10.02	7.07	3.85	4.12	2.59
	Sin norm	4.29	2.16	1.70	2.89	2.85	3.63	8.79	5.39	2.49	2.67	2.81	1.77	1.08	5.08	4.09	3.62
	Z	**1.47**	**0.00**	**0.00**	**0.00**	**0.00**	**0.59**	5.88	**0.00**	2.19	0.75	2.62	**0.91**	**0.00**	1.89	2.34	**0.00**
	MAX	3.38	0.35	0.27	0.49	0.00	0.89	2.36	0.07	5.04	**0.00**	0.47	2.11	0.39	**0.13**	4.17	0.00
	Min-max	1.76	0.00	0.69	0.00	0.00	0.64	**1.14**	0.84	1.49	0.00	**0.20**	1.08	0.00	0.27	**0.98**	0.07
		MultiMono															
1	Original	53.52	23.43	**4.36**	15.26	9.99	7.19	9.79	**6.92**	17.93	35.37	14.20	14.51	16.08	7.25	**7.41**	**7.43**
	Sin norm	8.70	**6.93**	6.02	6.96	**7.31**	**6.11**	**6.54**	7.15	**7.06**	**6.12**	**7.12**	**6.78**	7.64	6.88	8.16	7.88
	Z	8.54	8.60	9.49	**6.85**	8.79	13.17	16.34	9.50	8.33	7.23	9.98	6.80	**6.25**	12.55	13.38	7.87
	MAX	8.47	8.64	9.40	7.12	7.92	10.30	11.93	8.10	7.23	8.33	9.67	8.13	7.69	**6.86**	12.19	8.73
	Min-max	**8.36**	7.72	9.72	7.78	7.62	11.60	12.40	7.56	7.09	8.40	9.39	7.90	7.78	7.74	12.36	7.78
4	Original	53.58	23.21	**3.97**	16.78	9.75	6.55	9.40	4.55	18.02	34.11	13.79	15.43	16.69	8.37	**7.40**	5.70
	Sin norm	6.83	5.66	4.52	5.69	5.61	5.02	**6.82**	5.47	4.73	5.18	6.40	4.69	6.49	6.77	8.00	7.99
	Z	5.11	4.08	4.35	2.54	3.31	9.04	12.47	4.98	6.07	2.21	6.94	**2.14**	**1.50**	9.29	9.59	**3.14**
	MAX	4.61	3.37	4.27	**2.26**	2.50	5.08	8.68	**2.47**	4.12	**1.88**	**4.59**	4.45	2.14	**3.16**	8.79	4.01
	Min-max	**4.05**	**1.94**	4.49	2.74	**2.36**	**4.88**	8.47	3.75	**3.42**	3.96	4.73	2.73	2.65	3.77	8.55	4.30
8	Original	52.33	23.30	2.87	16.65	7.82	4.87	7.77	3.02	16.74	33.60	13.05	13.86	14.56	6.79	5.85	5.15
	Sin norm	4.27	4.07	2.67	3.72	3.36	3.46	**4.65**	3.80	2.70	3.30	4.84	2.55	4.29	4.95	6.19	6.05
	Z	2.39	1.40	1.86	0.75	0.99	6.25	9.47	1.63	3.88	0.35	3.64	0.92	0.27	5.86	7.55	**0.33**
	MAX	2.75	1.17	**1.66**	0.68	**0.66**	1.86	5.20	**0.57**	2.00	**0.31**	**1.72**	2.63	**0.22**	**1.04**	5.96	1.82
	Min-max	**1.39**	**0.22**	1.78	**0.43**	0.91	**1.83**	4.68	1.51	**1.72**	1.00	1.76	**0.57**	0.76	1.16	**5.38**	2.05
12	Original	52.18	27.33	2.47	20.30	6.46	2.77	6.74	1.88	16.22	29.33	12.46	13.21	13.15	5.66	5.66	4.75
	Sin norm	3.14	3.15	1.82	2.94	2.09	2.73	3.10	1.88	2.00	2.06	3.71	1.31	3.67	4.02	4.36	4.56
	Z	1.93	0.67	1.07	0.19	**0.07**	4.44	7.31	0.79	2.14	**0.00**	2.08	0.49	0.07	4.33	5.47	**0.11**
	MAX	1.69	0.24	1.02	**0.10**	0.31	1.10	3.35	**0.33**	1.08	0.06	0.66	1.09	**0.00**	0.51	4.44	0.65
	Min-max	**0.93**	**0.00**	**0.71**	0.17	0.19	**0.78**	**2.77**	0.65	**0.76**	0.42	**0.58**	**0.35**	0.16	**0.07**	**3.30**	0.64

Para el clasificador de Árboles, también se confirma que los nuevos vectores de características funcionan mejor que los vectores de características originales, aunque en este caso no es tan claro como en los dos clasificadores anteriores. La razón para esta afirmación es que, pese a que, en algunos escenarios, el error es menor con los vectores de características originales, estos casos son los menos.

5.6 Resumen resultados

En este apartado final del capítulo de resultados se va a realizar un compendio de estos, para poder ver más claramente cómo han ido los experimentos realizados. Para ello se van a utilizar unos gráficos, pero no de todos los datos, sino de un subconjunto de ellos que se van a explicar a continuación.

Se van a comparar los valores del equierror del mismo escenario experimental para los vectores de características originales (color azul), frente a los valores del sistema que ha utilizado los vectores de características modificados (color naranja), en concreto a los que se ha aplicado la normalización Z porque en general ha mostrado buenos resultados. Se recogen los datos de ambos dispositivos ponibles, el reloj Motorola (Moto) y la pulsera de Microsoft (Micro), y para ambos sensores tanto el acelerómetro (ACC) como el giróscopo (GYR). Se recogen los resultados del escenario *MultiMono* ya que es el más realista desde un punto de vista práctico al tener en cuenta la variación del comportamiento del individuo con el tiempo. Y, por último, los gráficos se han dividido en dos tablas, una en la que a los datos no se ha aplicado ningún post-procesamiento la Figura 5.3, y otra tabla en la que sí se ha aplicado post-procesamiento, la Figura 5.4, en concreto el de fusionar los datos de 8 ventanas (apartado Post-procesamiento). Se escogió este post-procesamiento porque es el que se considera con la mejor relación entre los resultados y el tamaño de la señal de prueba.

En las gráficas, SVM3 es el de Kernel Radial y SVM es el de Kernel líneal de orden 6.

Sin post-procesamiento

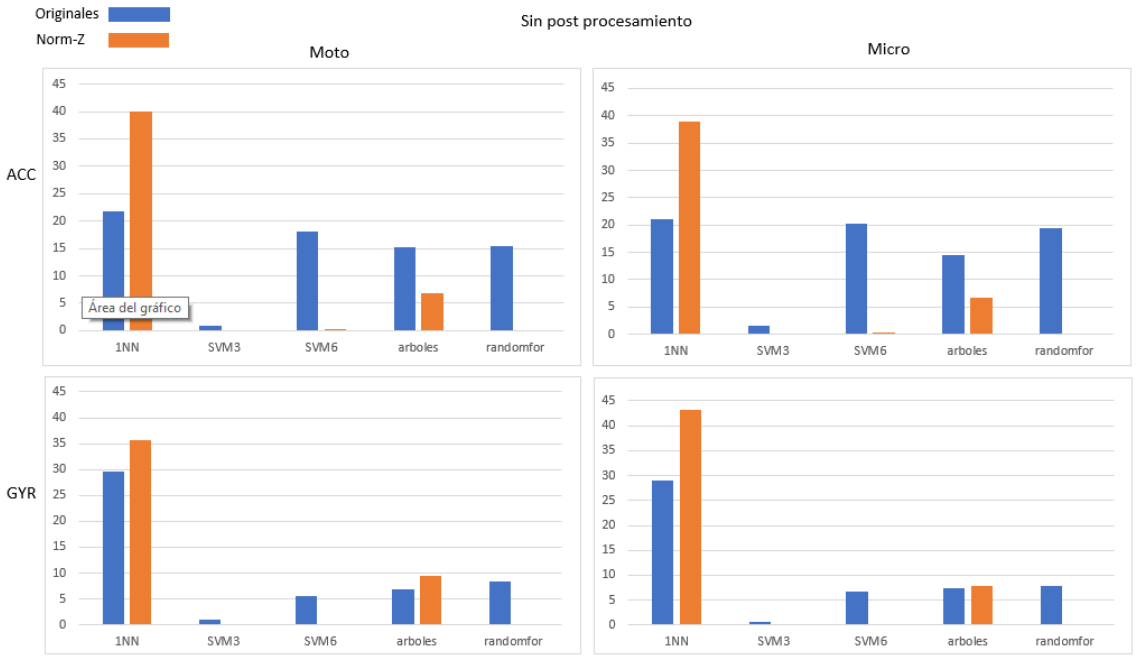

Figura 5.3 Comparación del equierror de los diferentes escenarios experimentales realizados sin ningún post-procesamiento

Con post-procesamiento nº de ventanas=8

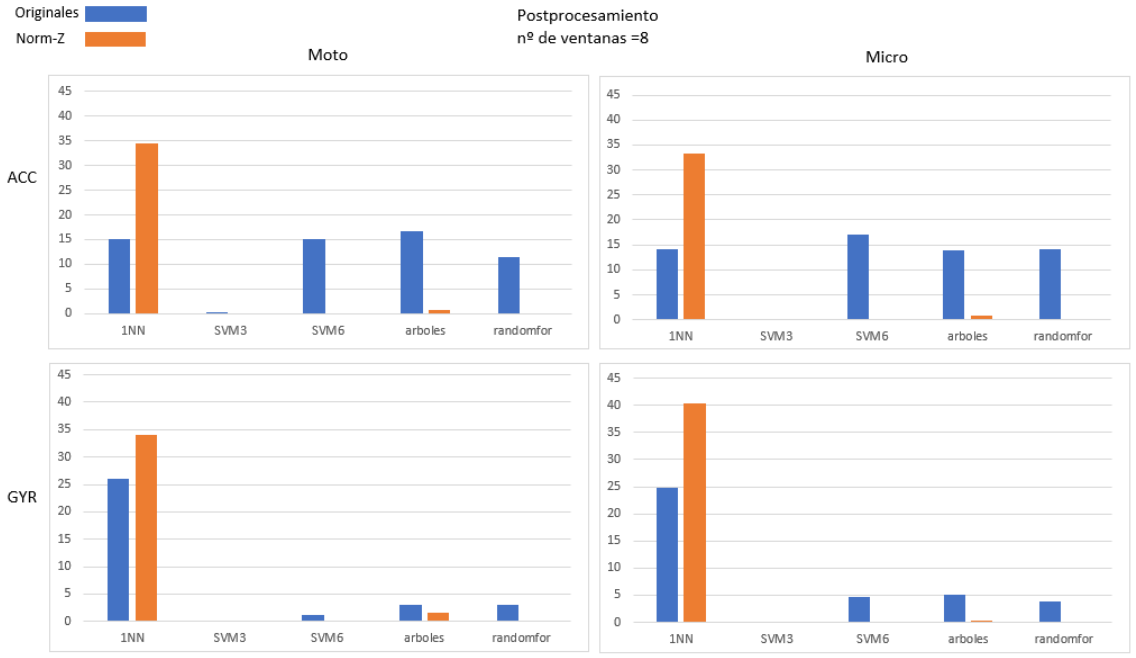

Figura 5.4 Comparación del equierror de los diferentes escenarios experimentales realizados aplicando el post-procesamiento de la fusión de 8 ventanas

6 Conclusiones y Trabajo Futuro

El resumen del trabajo ha sido que el objetivo de modificar el sistema de reconocimiento biométrico existente, para que use la biblioteca de ajuste FMM, y comparar los datos, se ha cumplido satisfactoriamente. Tomando esto como base, se van a presentar las conclusiones, se explicarán los objetivos específicos cumplidos, las líneas de trabajo futuras y las valoraciones personales.

6.1 Objetivos cumplidos

De la lista de objetivos que se estableció al inicio del trabajo en el apartado 1.2, se han cumplido tres de los cuatro objetivos. Sí que se ha conseguido modificar el sistema de reconocimiento biométrico, en concreto la obtención de los vectores de características, para incluir el uso de la biblioteca. Otro objetivo cumplido fue el de llevar a cabo los experimentos utilizando el nuevo sistema modificado, con los datos de la base de datos existente. El último objetivo cumplido ha sido el de analizar los resultados de estos experimentos para realizar la comparación con los datos del sistema original sin la modificación.

Por último, el objetivo que buscaba modificar los parámetros del sistema para encontrar alguna configuración que funcionase mejor no se ha llevado a cabo. Ya que, como se comentó en la planificación inicial, apartado 3.3, se pretendía repetir la fase de experimentación varias veces, con diferentes configuraciones para poder compararlas, objetivo que por falta de tiempo no fue posible realizar. Sin embargo, se podría decir que sí que se han comparado diferentes conjuntos de resultados del sistema modificado. Pues como se explica en el apartado 3.8.5, la normalización de los datos que generaba el sistema modificado nos permitió tener diferentes datos que comparar con el sistema original para ver cuál de ellos daba un mejor resultado y tratar de aclarar si alguna de las normalizaciones trabajaba mejor que otras.

Gracias al cambio de planificación que hubo, el aprendizaje obtenido del trabajo ha sido mayor del esperado inicialmente, pues la paralelización del programa en R así como la búsqueda de soluciones a los problemas de ejecución han contribuido a hacer de este trabajo una fuente de aprendizaje en diferentes ámbitos.

6.2 Trabajo futuro

Como se ha visto en la fase de desarrollo, el trabajo ha podido completarse, pues se ha modificado con éxito el sistema de reconocimiento biométrico, y se ha podido comparar con el original. Sin embargo, han quedado varias líneas de investigación abiertas para futuros trabajos:

- Se puede usar este trabajo como punto de partida para modificar diferentes parámetros tanto del sistema de reconocimiento biométrico (como podría ser el tamaño de las ventanas, o la cantidad de solapamiento de estas) como de la biblioteca del ajuste con FMM (se podría modificar el número de componentes con los que ajusta el modelo) y obtener los resultados para encontrar configuraciones que funcionen mejor todavía.
- Otra línea de trabajo sería la investigación de los resultados obtenidos para el clasificador SVM3, pues, como se ha comentado en el apartado 4.6, al obtener los resultados del sistema modificado, todos los usuarios obtenían el mismo valor para la distancia hasta los impostores y hasta los auténticos. Se podría buscar unos parámetros diferentes para la obtención de datos de ese clasificador y ver si hay variaciones en los datos.

- Se podría investigar también el error que se encontró a la hora de ejecutar el sistema para obtener los nuevos vectores de características con los usuarios del TFM. Pues con algunos usuarios fallaba sin razón aparente pues los datos no tienen diferencia con respecto a los usuarios del TFG.
- Por último, se podría trabajar de nuevo en otra modificación de la obtención de los vectores de características del sistema de reconocimiento biométrico, pero está vez usando otro trabajo académico de interés para el trabajo y comparar con los datos obtenidos en este.

6.3 Valoración personal

Este trabajo, como se puede apreciar, ha sido algo diferente a lo que yo me imaginaba al empezar a realizarlo. Ya sea por los imprevistos surgidos, por el tipo de trabajo experimental realizado o por el desconocimiento de las tecnologías utilizadas en los trabajos previos, usados como base para este. El resultado ha sido una etapa de trabajo diferente a la esperada, en la que obligatoriamente ha habido que adaptarse a los problemas que aparecían, pero que ha permitido conocer en mayor medida un área de la informática que ha conseguido atraparme de una manera inesperada.

Todas las características del trabajo han hecho que salga de mi zona de confort, pues en absoluto ha sido un trabajo más dentro de mi vida académica, sin duda lo recordaré durante mucho tiempo por lo que me ha aportado como estudiante y como persona, pues considero que me ha hecho crecer en ambos aspectos.

Bibliografía

[1] *Backfitting algorithm*. (26 de Junio de 2021). Recuperado el 3 de Enero de 2022, de Wikipedia, La enciclopedia libre: https://en.wikipedia.org/w/index.php?title=Backfitting_algorithm&oldid=1030586496

[2] *Coeficiente de determinación*. (20 de Noviembre de 2020). Recuperado el 11 de Diciembre de 2021, de Wikipedia, La enciclopedia libre: https://es.wikipedia.org/w/index.php?title=Coeficiente_de_determinaci%C3%B3n&oldid=131073673

[3] *Desarrollo en cascada*. (13 de Agosto de 2021). Recuperado el 7 de Julio de 2021, de Wikipedia, La enciclopedia libre: https://es.wikipedia.org/w/index.php?title=Desarrollo_en_cascada&oldid=137652816

[4] *Desviación típica*. (10 de noviembre de 2021). Recuperado el 11 de Diciembre de 2022, de Wikipedia, La enciclopedia libre: https://es.wikipedia.org/w/index.php?title=Desviaci%C3%B3n_t%C3%ADpica&oldid=139630295

[5] *detectCores: Detect the Number of CPU Cores*. (s.f.). Recuperado el 20 de Diciembre de 2021, de RDocumentation: https://www.rdocumentation.org/packages/parallel/versions/3.6.2/topics/detectCores

[6] *foreach: foreach.* (s.f). Recuperado el 20 de Diciembre de 2021, de RDocumentation: https://www.rdocumentation.org/packages/foreach/versions/1.5.1/topics/foreach

[7] *HP Pavilion 15-bc500ns - Ordenador portátil de 15.6" FullHD (Intel Core i5-9300H, 8GB RAM, 1TB HDD + 128GB SSD, NVIDIA GeForce GTX 1050-3GB, FreeDOS) negro - teclado QWERTY Español*. (s.f.). Recuperado el 6 de Julio de 2022, de Amazon.es: https://www.amazon.es/HP-Pavilion-15-bc500ns-Ordenador-port%C3%A1til/dp/B07X5DFXHJ/ref=sr_1_4?keywords=HP+Pavilion+15&qid=1657057047&refinements=p_n_feature_seven_browse-bin%3A8179048031&s=computers&sr=1-4

[8] *makeCluster: Create a Parallel Socket Cluster*. (s.f.). Recuperado el 20 de Diciembre de 2021, de RDocumentation: https://www.rdocumentation.org/packages/parallel/versions/3.6.2/topics/makeCluster

[9] *max: Maxima and Minima*. (s.f.). Recuperado el 11 de Diciembre de 2021, de RDocumentation: https://www.rdocumentation.org/packages/madness/versions/0.2.7/topics/max

[10] *mclapply: Serial versions of mclapply, mcmapply and pvec*. (s.f.). Recuperado el 20 de Diciembre de 2021, de RDocumentation: https://www.rdocumentation.org/packages/parallel/versions/3.4.1/topics/mclapply

[11] *mean: Arithmetic Mean*. (s.f.). Recuperado el 11 de Diciembre de 2021, de RDocumentation: https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/mean

[12] *Modelo de prototipos*. (13 de Agosto de 2021). Recuperado el 7 de Julio de 2021, de Wikipedia, La enciclopedia libre: https://es.wikipedia.org/w/index.php?title=Modelo_de_prototipos&oldid=139152201

[13] *Normalización (estadística)*. (16 de octubre de 2020). Recuperado el 11 de Diciembre de 2021, de Wikipedia, La enciclopedia libre: https://es.wikipedia.org/w/index.php?title=Normalizaci%C3%B3n_(estad%C3%ADstica)&oldid=130123339

[14] *Período (física)*. (30 de Julio de 2021). Recuperado el 1 de Diciembre de 2021, de Wikipedia, La enciclopedia libre: https://es.wikipedia.org/w/index.php?title=Per%C3%ADodo_(f%C3%ADsica)&oldid=137357248

[15] *R (lenguaje de programación)*. (13 de septiembre de 2021). Recuperado el de Junio de 2022, de Wikipedia, La enciclopedia libre: https://es.wikipedia.org/w/index.php?title=R_(lenguaje_de_programaci%C3%B3n)&oldid=139594815

[16] *Salario medio para Programador en España, 2022*. (2022). Recuperado el 6 de Julio de 2022, de Talent.com: https://es.talent.com/salary?job=programador

[17] *sd: Standard Deviation*. (s.f.). Recuperado el 11 de Diciembre de 2021, de RDocumentation: https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/sd

[18] Derawi, M. O. *Accelerometer-Based Gait Analysis, A survey.* (2010).

[19] Gerend, J. *fc*. (12 de Agosto de 2021). Recuperado el 11 de Diciembre de 2021, de Documentación técnica de Microsoft: https://docs.microsoft.com/es-es/windows-server/administration/windows-commands/fc

[20] Jones, M. *Quick Intro to Parallel Computing in R*. (2017 de Julio de 25). Recuperado el 20 de Diciembre de 2021, de https://nceas.github.io/oss-lessons/parallel-computing-in-r/parallel-computing-in-r.html

[21] Lamela Pérez, A. *Implementación de un paquete de software para el ajuste de modelos FMM. Aplicación a la interpretación automática de la señal del electrocardiograma*. (2020). 14 de Marzo de 2021, de http://uvadoc.uva.es/handle/10324/44944

[22] Lamela A., Fernández I., Larriba Y., Rodríguez-Collado A., and Rueda C. *FMM: An R package for modeling rythmic patterns in oscillatory systems.* (2020). Recuperado el 6 de Julio de 2022, de arXiv.com: https://arxiv.org/abs/2105.10168

[23] Michael Kerrisk. *fork(2) — Linux manual page*. (27 de Agosto de 2021). Recuperado el 18 de Junio de 2022, de Linux man pages online: https://man7.org/linux/man-pages/man2/fork.2.html

[24] Misfud, E. *Sistemas físicos y biométricos de seguridad | Observatorio Tecnológico.* (2012). Recuperado el 6 de Julio de 2022, de http://recursostic.educacion.es/observatorio/web/ca/cajon-de-sastre/38-cajon-de-sastre/1045-sistemas-fisicos-y-biometricos-de-seguridad}{http://recursostic.educacion.es/observatorio/web/ca/cajon-de-sastre/38-cajon-de-sastre/1045-sistemas-fisicos-y-biometr

[25] Rueda, C., Larriba, Y., and Lamela, A. *The hidden waves in the ECG uncovered: A multicomponent model for the Cardiac Rhythm*. (2020*)*. Recuperado el 5 de Julio de 2022, de arXiv.com: https://arxiv.org/abs/2005.10173

[26] Rueda, C., Larriba, Y. and Peddada, S. *A Frequency Modulated Mobius Model Accurately Predicts Rhythmic Signals in Biological and Physical Sciences*. (2019). Scientific Reports, 9, 1. pag 1-10.

[27] Salvador Ortega, I. *Investigación y desarrollo en técnicas de reconocimeinto biométrico mediente dispositivos ponibles (Wearables)*. (2019). Recuperado el 14 de Marzo de 2021, de uvadoc.uva.es: http://uvadoc.uva.es/handle/10324/38772

[28] Salvador Ortega, I. *Investigación y desarrollo de un sistema de reconocimiento biométrico mediante dispositivos ponibles (Wearables)*. (2020). Recuperado el 14 de Marzo de 2021, de uvadoc.uva.es: http://uvadoc.uva.es/handle/10324/44947

[29] Thang, H. M. *Gait identification using accelerometer on mobile phone.* (2012).

[30] Vega, J. B. *R para principiantes.* (s.f.). Recuperado el 26 de Junio de 2022, de https://bookdown.org/jboscomendoza/r-principiantes4/}{https://bookdown.org/jboscomendoza/r-principiantes4

[31] Weston, S. *Using the foreach package*. (s.f.). Recuperado el 20 de Diciembre de 2022, de The Comprehensive R Archive Network: https://cran.r-project.org/web/packages/foreach/vignettes/foreach.html

Apéndice A: Contenido del repositorio utilizado

En la dirección https://gitlab.inf.uva.es/ponibles/gillermojannez_fmm se puede encontrar el repositorio en el que se han almacenado los ficheros y los datos utilizados para la realización de este trabajo. Su estructura se puede apreciar en la Figura A 1.

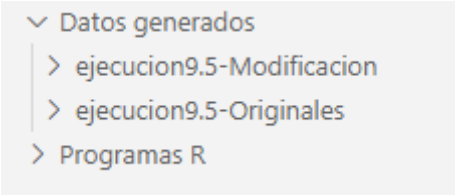

Figura A 1 Vista gráfica de la estructura de directorios del repositorio utilizado.

Cada una de las carpetas contiene lo siguiente:

- **Datos generados**: en esta carpeta se guardan los datos que se han usado para comparar en la sección de resultados. En la carpeta 'ejecucion9.5-Originales' están los datos generados usando el sistema de reconocimiento biométrico original, y en la carpeta 'ejecucion9.5-Modificacion' están los datos del sistema de reconocimiento con la modificación añadida. Estas dos carpetas tienen la misma estructura interna, y se puede usar la Figura A 2 para clarificar la jerarquía de carpetas. Primero cuatro carpetas que hacen referencia al post-procesamiento aplicado con el nombre 'Xscore2solap', donde X es el número de ventanas solapadas (1,4,8,12). Dentro de estas carpetas hay cinco nuevas carpetas con el nombre de cada uno de los clasificadores utilizados. Dentro de las carpetas de los clasificadores hay una carpeta llamada 'ninguna' porque no se realiza ningún tipo de limpieza de la señal. Dentro de estas hay dos nuevas carpetas, una para cada sensor utilizado acelerómetro o giróscopo. Dentro de esta carpeta hay otras dos carpetas, una para cada tipo de entrenamiento aplicado, el monomono o el multimono y dentro de estas carpetas hay tres más 'dist_aut', 'dist_imp' y 'ROC'. La carpeta 'ROC' es la que contiene un fichero con el equierror medio y otro con el área bajo la curva de Fourier (AUC) para cada dispositivo utilizado, para el módulo, y cada una de las tres coordenadas de los datos (X, Y, Z), y para cada tipo de normalización aplicada en caso de ser la carpeta del sistema con esa modificación y no la del sistema original.

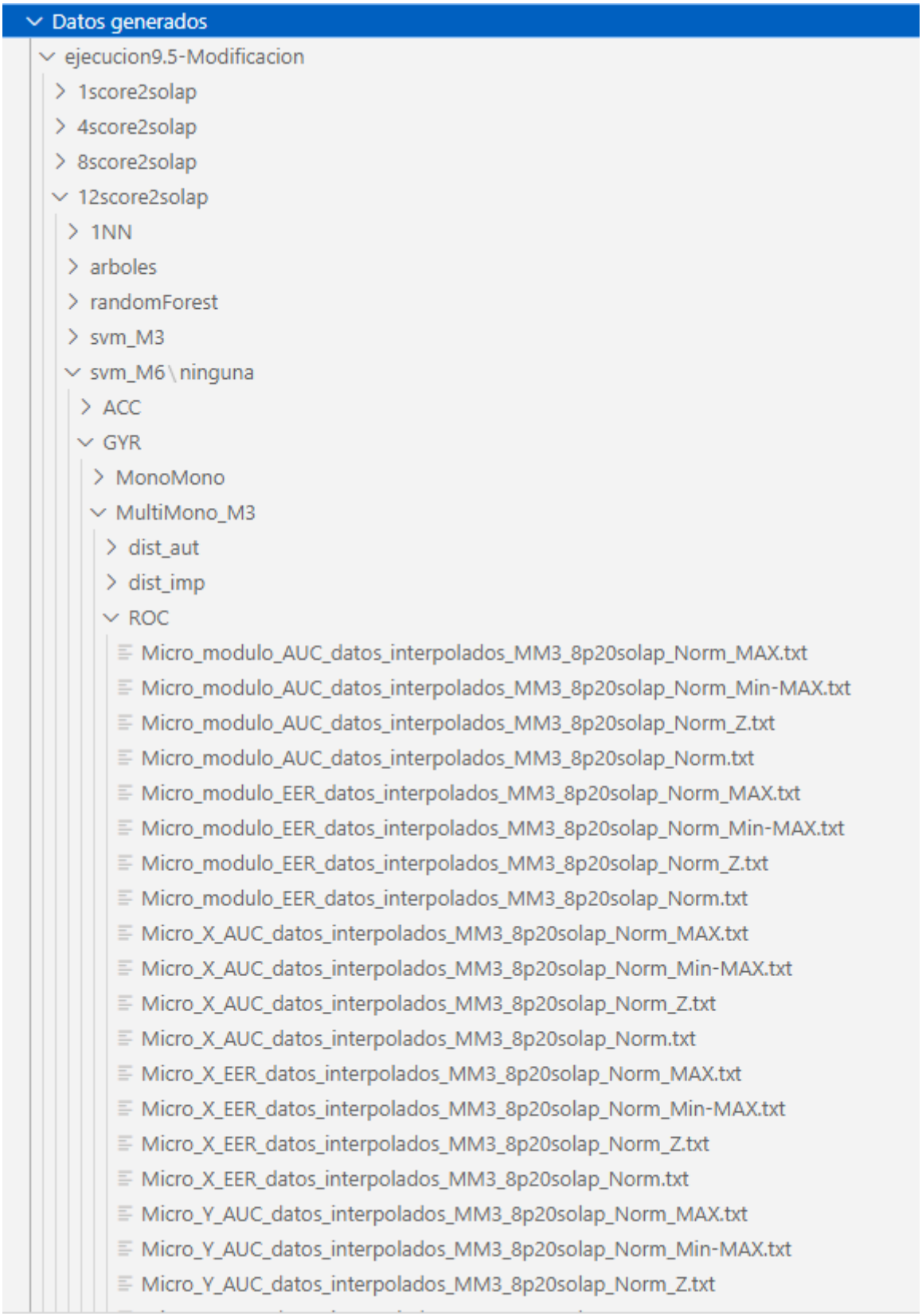

Figura A 2 Vista gráfica de la estructura de directorios de las carpetas 'ejecucion9.5-Originales' y 'ejecucion9.5-Modificacion'.

- **Programas R**: contiene todos los programas que forman parte del sistema de reconocimiento biométrico sobre los que se ha trabajado. Se va a explicar qué contiene cada uno de ellos:

- *9_1_Deteccion_ZonasRuido_datosTFG_modulo*: versión original que se encarga de dividir en ventanas la muestra e indicar las zonas en las que hay ruido.
- *9_2_DatosTFG_EliminacionRuido*: versión original que lee el fichero generado en el programa anterior y, dependiendo de los umbrales establecidos, genera un nuevo fichero donde se indica el principio y final de cada tramo considerado ruidoso en tiempo acumulado.
- *9_3_TablasCaract_modulo_V2*: versión original del fichero que crea los vectores de características para cada ventana, con solapamiento, tanto en el dominio de la frecuencia como en el del tiempo para el módulo.
- *9_3_TablasCaract_modulo_V2_Paralleliza*: versión modificada del programa anterior, que incluye la paralelización para la obtención de los nuevos vectores de características obtenidos a partir del uso de la biblioteca FMM para el módulo de los datos.
- *9_3_TablasCaract_modulo_V2_Normalizacion*: versión modificada que añade al programa paralelizado anterior la opción de normalizar los datos de los vectores de características obtenidos.
- *9_3_TablasCaract_XYZ_V2*: versión original del fichero que crea los vectores de características para cada ventana, con solapamiento, tanto en el dominio de la frecuencia como en el del tiempo para cada una de las coordenadas de los datos (X, Y, Z).
- *9_3_TablasCaract_XYZ_V2_Parallelization*: versión modificada del programa anterior, que incluye la paralelización para la obtención de los nuevos vectores de características obtenidos a partir del uso de la biblioteca FMM para las coordenadas X, Y y Z de los datos.
- *9_3_TablasCaract_XYZ_V2_Normalizacion*: versión modificada que añade al programa paralelizado anterior la opción de normalizar los datos de los vectores de características obtenidos.
- *9_4_TablasCaract_quitarRuido*: versión original que abre los ficheros que contienen los vectores de características y las zonas de ruido. Compara el inicio y final de la ventana correspondiente al vector de características y genera un fichero que indique si está o no dentro de una zona de ruido.
- *9_4_TablasCaract_quitarRuido_modifyrutas*: versión modificada del programa anterior que usa los nuevos vectores de características.
- *9_5_V3_NF_RendimientoFinal_KNN_tfijo_Clasif_SVM_Radial_VariasVentanasScores_DatosTFG*: versión original que realiza los experimentos para KNN y SVM (con distintos Kernels). Genera los ficheros de entrenamiento y prueba, entrena el clasificador, genera las salidas y calcula las tasas de EER.
- *9_5_V3_DiferentesClasificadores*: versión original que hace lo mismo que el anterior, pero para los clasificadores árboles de decisión y random forest.
- *9_5_V4_NF_RendimientoFinal_KNN_tfijo_Clasif_SVM_Radial_VariasVentanasScores_DatosTFG*: versión modificada que genera los resultados de los experimentos para KNN, SVM, árboles de decisión y random forest, pero usando los vectores de características modificados.
- *9_8_V2_NF_RendimientoFinal_KNN_tfijo_ClasifNoSup_DatosTFG*: versión original que hace lo mismo que realiza los experimentos para los clasificadores árboles de decisión, random forest, bagging, naive Bayes, jrip y SvmLineal.