UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

DISEÑO Y VALIDACIÓN DE UN SOFTWARE DE RECONOCIMIENTO DE GESTOS SIRVIÉNDOSE DEL ACELERÓMETRO DE UN RELOJ INTELIGENTE

AUTOR: JORGE HERRERA CRESPO

TUTOR: LEOPOLDO ARMESTO ÁNGEL

Curso Académico: 2020-21

AGRADECIMIENTOS

Quiero aprovechar la ocasión para agradecer el apoyo de mi tutor, el Dr. Leopoldo Armesto Ángel, por sus buenos consejos durante todo el proceso de trabajo.

También al resto de profesores y profesoras del Master Universitario en Ingeniería Industrial, por la excelente formación que nos han dado durante este año.

Y, por último, a mi familia por su apoyo y paciencia, no sólo a lo largo de este curso, sino de todos los años que ha durado mi formación hasta llegar aquí.

RESUMEN

El trabajo propone un software para el microcontrolador ESP32 de un reloj inteligente, que interpreta los movimientos del brazo y los convierte en órdenes visibles en la interfaz del propio reloj. Previamente se analiza el funcionamiento de los acelerómetros y del microcontrolador, como base del hardware en el que nos apoyaremos. Así como la metodología de análisis de componentes principales y los algoritmos de reconocimiento de gestos Dynamic Time Warping (DTW), DTW with AP and CS, y Multilayer Perceptron (MLP), entre otros posibles métodos. También se analizan otros trabajos de investigación que usaron dichos algoritmos en otros dispositivos emisores diferentes a los relojes inteligentes. Y, a partir de ahí, se diseña el software específico para los relojes inteligentes, como resultado principal del trabajo. El algoritmo que se diseña es sometido, por último, a validación utilizando la metodología de matrices de confusión.

Palabras Clave: Acelerómetro, microcontrolador, reconocimiento de gestos, DTW, MLP, relojes inteligentes.

RESUM

El treball proposa un programari per al microcontrolador ESP32 d'un rellotge intel·ligent, que interpreta els moviments del braç i els converteix en ordres visibles en la interfície del propi rellotge. Prèviament s'analitza el funcionament dels acceleròmetres i del microcontrolador, com a base del maquinari en el qual ens recolzarem. Així com la metodologia d'anàlisi de components principals i els algorismes de reconeixement de gestos Dynamic Time Warping (DTW), DTW with AP and CS, i Multilayer Perceptron (MLP), entre altres possibles mètodes. També s'analitzen altres treballs de recerca que van usar aquests algorismes en altres dispositius emissors diferents als rellotges intel·ligents. I, a partir d'ací, es dissenya el programari específic per als rellotges intel·ligents, com a resultat principal del treball. L'algorisme que es dissenya és sotmés, finalment, a validació utilitzant la metodologia de matrius de confusió.

Paraules clau: Acceleròmetre, microcontrolador, reconeixement de gestos, DTW, MLP, rellotges intel·ligents.

ABSTRACT

This mather thesis proposes the development of a firmware for the ESP32 microcontroller in a smart watch, which interprets the movements of the arm and converts them into visible orders on the smart watch interface. Previously, the operation of the accelerometers and the microcontroller is analyzed, as the basis of the hardware that we will use. As well as the principal component analysis methodology and the Dynamic Time Warping (DTW), DTW with AP and CS, and Multilayer Perceptron (MLP) gesture recognition algorithms, among other possible methods. Other research works that use these algorithms are also analyzed. And, from there, the specific software for smart watches is designed, as the main result of the work. Finally, the algorithm that is designed is subjected to validation using the confusion matrix methodology.

Keywords: Accelerometer, microcontroller, gesture recognition, DTW, MLP, smart watches.

ÍNDICE

DOCUMENTOS CONTENIDOS EN EL TFM

ÍNDICE DE LA MEMORIA

ÍNDICE DEL PRESUPUESTO

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

MEMORIA

DISEÑO Y VALIDACIÓN DE UN SOFTWARE DE RECONOCIMIENTO DE GESTOS SIRVIÉNDOSE DEL ACELERÓMETRO DE UN RELOJ INTELIGENTE

AUTOR: JORGE HERRERA CRESPO

TUTOR: DR. LEOPOLDO ARMESTO ÁNGEL

CAPÍTULO 1. INTRODUCCIÓN

1.1. JUSTIFICACIÓN DEL PROYECTO DE TRABAJO FIN DE MASTER

A finales de septiembre de 2020 la IFR (International Federation of Robotics) presentó su último informe *World Robotics 2020 Industrial* Robots (IFR, 2020) en el que se recoge la situación de la Robótica dentro de la industria mundial, con datos y análisis que aún no estaban marcados por la irrupción de la pandemia de la COVID-19, ya que manejaba cifras y estadísticas hasta 2019. En un momento como el actual, debe entenderse que las reflexiones que se hagan sobre el futuro a medio y largo plazo no deberían limitarse a un escenario como este, por muy relevante que haya sido la pandemia a todos los niveles. Analizar más allá, permitirá unas conclusiones más sostenibles en el tiempo. Aunque también es cierto que la crisis económica mundial, generada por la pandemia, tiene también su propio análisis en el ámbito de la robótica, creando oportunidades y asentando modelos que marcarán el futuro. No obstante, muchas de las tendencias ya estaban de manifiesto en el escenario prepandemia.

Según el informe *World Robotics 2020 Industrial* Robots (IFR, 2020), en 2019 se alcanzó la cifra record de 2,7 millones de robots industriales operando en fábricas de todo el mundo, con unas ventas en ese mismo año de 373.000 unidades a nivel mundial.

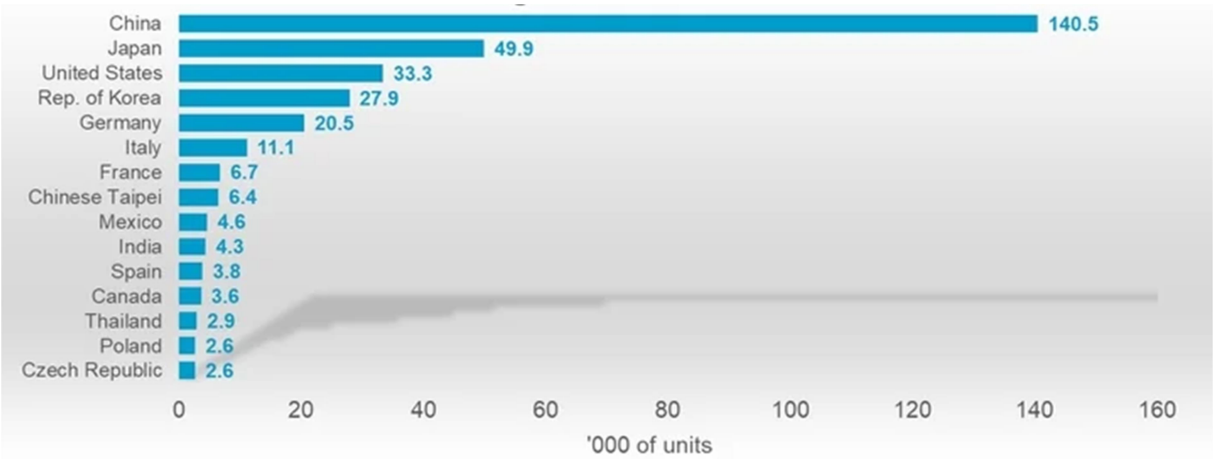


Figura 1.1. Instalaciones anuales de robots industriales TOP 15 países. Fuente: World Robotics 2020.

España, aunque a mucha distancia de los líderes mundiales en este sector, aparece en el selecto grupo de los 15 mayores mercados en la instalación de robots industriales. Aunque desciende un puesto respecto al año anterior, y se sitúa en el decimoprimer lugar, se mantiene en el cuarto puesto a nivel europeo (por detrás de Alemania, Italia y Francia). Aunque a nivel mundial 2019 tuvo una caída de ventas del 12%, España tuvo una caída aún mayor (28%), quedando en las 3.802 nuevas instalaciones, que la situaban en niveles de 5 años atrás.

La salida de la pandemia, la vuelta a la "nueva normalidad" y las oportunidades de los Fondos Europeos Next Generation UE, abren nuevas oportunidades al desarrollo de las Economías de

los países, de las industrias y de ámbitos como la Robótica o la Industria 4.0. La apuesta por utilizar los Fondos de Reconstrucción, Transformación y Resilencia en una ocasión para transformar el modelo económico de toda Europa, abre la oportunidad de que un proceso como el de la robotización de las industrias, y de cada vez más procesos de la vida cotidiana, que ya era imparable, se acelere aún más.

El concepto Industria 4.0., que algunos autores califican también como cuarta revolución industrial (Schwab, 2016), es relativamente reciente. La mayoría de la industria lo sitúa en el año 2013, con la presentación del Informe *Recommendations for implementing the strategic initiative INDUSTRIE 4.0.* (Kagermann et al., 2013) que realiza el Gobierno Alemán con la intención de seguir liderando en Europa la oferta de equipos de producción industrial, y la digitalización creciente de los procesos productivos.

El simbolismo de estos términos trata de mostrar la relevancia que tiene la robotización industrial en las transformaciones económicas y sociales que se están produciendo desde hace unos años.

Si la primera revolución industrial se sitúa, con la aparición del primer telar mecánico y con la máquina de vapor, en la segunda mitad del siglo XVIII y principios del XIX, la segunda revolución industrial se produce ya en el siglo XX, con la primera cadena de montaje, la producción en cadena y la energía eléctrica.

Ya en la segunda mitad del siglo XX se producen una serie de cambios que posteriormente dieron lugar a lo que algunos autores calificaron como la tercera revolución industrial (Rifkin, 2011), donde destacaban: la expansión de las Tecnologías de la Información y la Comunicación (las TICs), su uso generalizado no sólo en la industria sino también a nivel social; la deslocalización de la producción hacia países con costes laborales más bajos; y la aparición de la preocupación por la sostenibilidad del planeta y el uso de sistemas de eficiencia energética. Ya en esta fase, se empiezan a generalizar la automatización y el uso de robots en los procesos productivos, especialmente en sectores con alta inversión en capital.

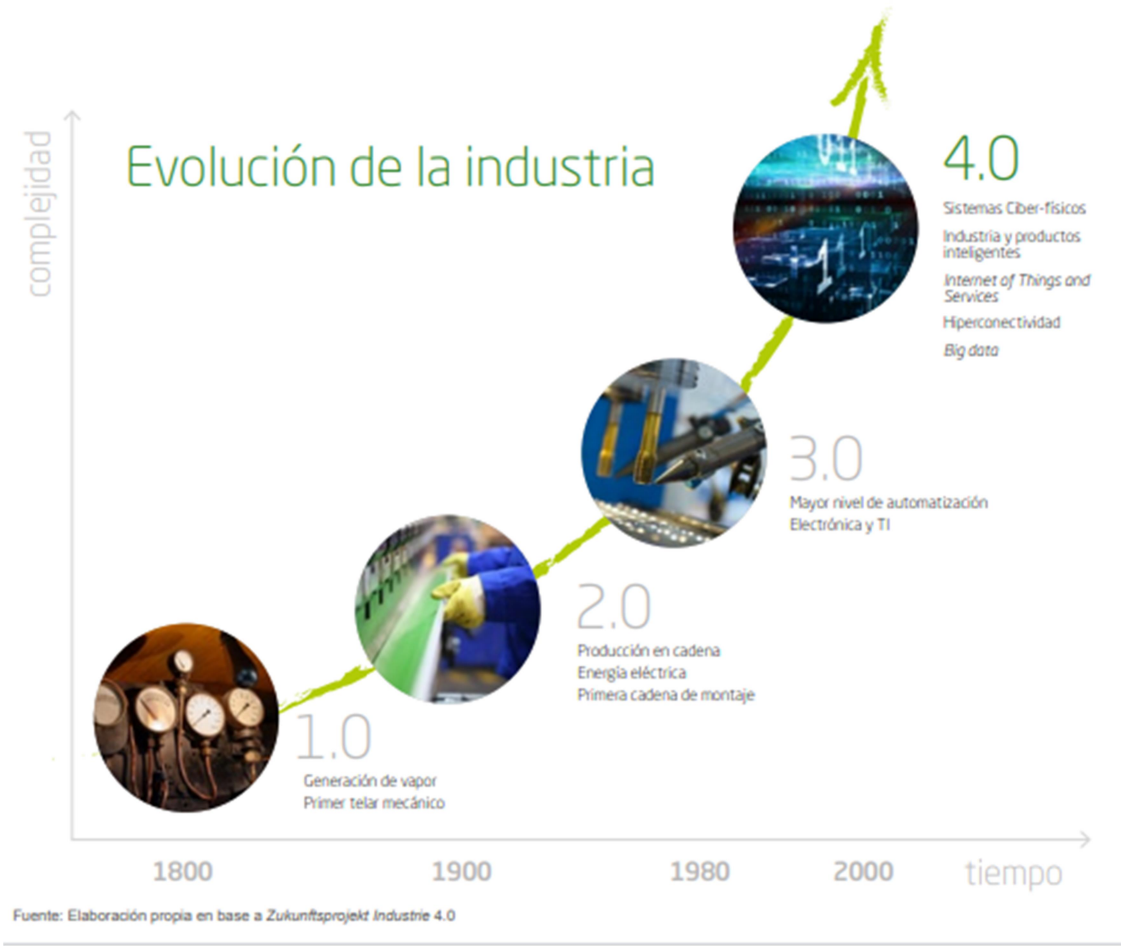


Figura 1.2. Evolución de la Industria a través de las diferentes revoluciones industriales. Tomado del Informe Industria Conectada 4.0. del Ministerio de Industria, Energía y Turismo (MINETUR, 2014; p. 6)[1]

Pero el término Industria 4.0. no es un mayor grado del modelo 3.0. En su planteamiento, parece tener menos sentido la deslocalización (CCOO, 2017), y se plantea como un modelo que no depende tanto del coste de mano de obra, lo que permite mantener un mayor número de puestos de trabajo en la sede original de la empresa.

La robotización incorpora numerosas ventajas: mejoras en la calidad, consiguiendo calidades superiores al trabajo humano; mejoras en el ambiente de trabajo, sustituyendo a trabajadores en tareas pesadas, peligrosas o monótonas, reduciendo así riesgos laborales; y mejoras económicas, el precio de los robots ha disminuido más del 50% en los últimos 20 años, consiguiéndose amortizaciones en promedio en menos de 3 años. (Valera Fernández, 2020b; Diapositivas 6-25)

Sin embargo, también presenta inconvenientes como la eliminación de puestos de trabajo, la introducción de otros nuevos riesgos laborales, suponiendo un temor y un descontento en la sociedad (Valera Fernández, 2020b; diapositivas 26-28).

[1] Se puede consultar en el link: http://www6.mityc.es/IndustriaConectada40/informe-industria-conectada40.pdf

El inconveniente de destrucción de empleo, que implica la sustitución de trabajo humano por robots, ya ha suscitado el debate dentro del Parlamento Europeo y podría solucionarse, por el lado de las cotizaciones al Sistema de la Seguridad Social, con la cotización de la empresa por los robots que hayan sustituido a puestos de trabajo con unas determinadas características (Ispizua, 2018).

Los tópicos están ahí. Es cierto que las industrias están ya bastante robotizadas y que el impacto en el empleo que se tenía que producir ya se ha producido en gran medida. "… la tecnología robótica actual se entiende mejor como la iteración más reciente de tecnologías de automatización industrial que han existido durante mucho tiempo. De hecho, estas tecnologías de automatización posiblemente tuvieron su mayor impacto en el empleo hace generaciones, lo que explica parcialmente los cambios en las estructuras de empleo en los sectores agrícola y manufacturero que se remontan a la Revolución Industrial. Por tanto, los efectos potenciales sobre el empleo de la tecnología robótica actual son a priori limitados". (Fernández-Macías et al, 2020; p. 4)

Además, hay estudios como los de Rivera-Taiba (2019) sobre el impacto de la robotización en el consumo, el empleo, los salarios y el PIB, que indican lo contrario, que **la disminución en inversión en capital robótico de un país produce una caída en el empleo y en los salarios**, ya que lo que se produce es un mayor aumento de las importaciones desde países que sí que progresan en ese sentido. El modelo, llega a cuantificar que una caída del 1% en el capital robótico de un país "produce una caída en el empleo y los salarios de 0,05% si el cambio es transitorio y de 0,25% si es permanente, por un aumento en las importaciones de capital robótico" (Rivera-Taiba, 2019; p. 37)

El modelo de industria 4.0 se caracteriza por la "smartización", los sistemas ciber-físicos, la hiperconectividad, el Big Data, la producción personalizada, la re-configurabilidad, o los robots colaborativos.

Muchos conceptos y muchos campos en los que se avanza desde la Ingeniería, y desde áreas como la de los Sistemas de control, Automatización y la Robótica.

Este Trabajo Fin de Master se va a centrar en una pequeña parte de todo este mundo. En concreto en los sistemas de comunicación persona-robot, y en los lenguajes de programación de robots.

Precisamente, la propia IFR recoge en su web las principales tendencias que se prevén en la robótica de los próximos años[2], con tres ejes destacados:

- La "Programación por demostración" se va a imponer definitivamente. Eso permitirá que la programación y la instalación de robots serán cada vez más fáciles, y llegará a más empresas y sectores. Los sensores digitales unidos al software inteligente permitirán que a través de la lectura del propio movimiento humano se elabore el

[2] Se puede consultar en el link: https://ifr.org/news/top-trends-robotics-2020/

software que manipulará los brazos robóticos, o cualquier otro robot que sustituya tareas humanas repetitivas y programables.

- La adaptación de los sistemas robóticos a entornos cada vez más cambiantes, a través de su flexibilidad. Métodos más ágiles, a la vez que sencillos, para el manejo de los robots, donde se incluyan voz, gestos y reconocimiento de la intención del movimiento humano. De esa forma los sistemas robóticos podrán llegar a muchos más ámbitos, y a empresas de todos los tamaños y sectores. Se plantean las operaciones colaborativas, como el gran potencial de crecimiento de la robótica, para complementar las inversiones en la robótica industrial tradicional de alto coste y tamaño.

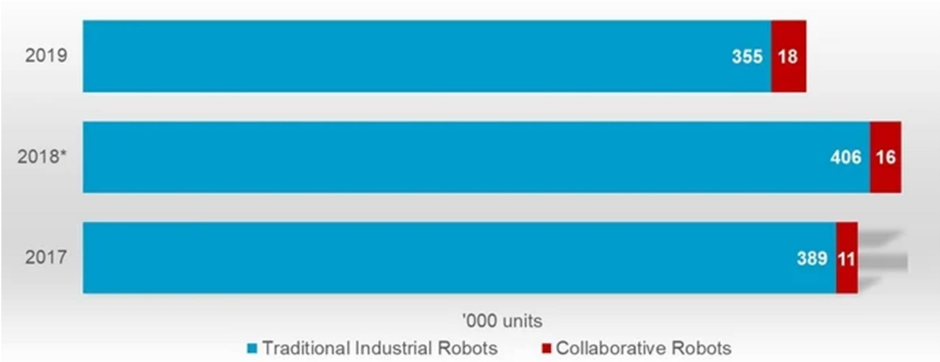


Figura 1.3. Robots en la industria tradicional y colaborativa. Fuente: International Federation of Robotics (IFR).

- La mayor conectividad de los sistemas robóticos, entre sí y con el Internet industrial de las cosas (IIoT). Conforme avancen los sistemas que permitan la comunicación entre sistemas robóticos, eliminando la dependencia de un fabricante u otro. La denominada "Especificación complementaria de robótica", desarrollada de manera conjunta por la VDMA (Asociación Alemana de Fabricantes de Maquinaria) y la *Open Platform Communications Foundation* (OPC), han diseñado una interfaz genérica estandarizada para robots industriales, que hace posible que los robots industriales se conecten al denominado Internet industrial de las cosas (IIoT). Esta tecnología también permite lo que se llama *Robots-as-a-Service*, que no es más que la posibilidad de que pequeñas y medianas empresas puedan alquilar robots para adaptar a sus procesos productivos, a costes mucho más asequibles, y sin necesidades de trabajar con personal de alta cualificación.

Este trabajo profundiza en ese concepto de “Programación por demostración” (PbD), que ha sido objeto de desarrollo a través de diferentes dispositivos de entrada de la información, manipuladores y estrategias de aprendizaje.

Y a la vez, afrontará otro de los grandes retos del sector, según la IFR, para los próximos años. La simplificación y flexibilidad de los procesos para programar los robots, y que permitan democratizar el acceso de la robótica, para que se haga accesible a empresas-usuario de todos los tamaños y sectores.

A continuación, se revisarán algunos trabajos que se pueden considerar antecedentes de este TFM, y que ayudan a identificar mejor dónde se puede aportar algo más en este campo de la Programación por Demostración (con un desarrollo algo mayor de PbD mediante acelerómetros, por tratarse del interfaz en el que se va a centrar el TFM). A partir de ahí, estará mucho más definido el objetivo a buscar por el TFM y la metodología a aplicar para conseguirlo.

1.2. ANTECEDENTES

"Los robots industriales realizan acciones programadas en un entorno especialmente preparado y altamente estructurado" (Pires et al, 2009; p. 73).

La intervención humana en este proceso es la de programación del conjunto de movimientos que realizaría después el o los dispositivos robóticos.

Aún hoy, la mayoría de los robots industriales son "máquinas no inteligentes", diseñadas, equipadas y programadas para realizar tareas específicas.

La robótica industrial implantada en la mayor parte de las industrias del mundo occidental es una tecnología de alto coste y orientada a fabricar muchas unidades. Normalmente requiere a expertos en programación, y las tareas programadas y ejecutadas dependen del acierto de estos especialistas. Cualquier modificación es complicada, y se produce de manera excepcional en los procesos productivos.

Así, este tipo de sistema productivo basado en el uso de robots industriales se caracteriza por su coste y la falta de flexibilidad.

Figura 1.4. Robots en la industria automovilística, como ejemplo de robótica de finales del siglo XX.

Los sistemas de programación robótica de este tipo se comienzan a calificar como clásicos. Y se caracterizan porque requieren programación técnica (profesionales cualificados) y tiempo de programación especializada (normalmente más cara, y poco accesible a pequeñas empresas). En los primeros momentos de robotización de los sistemas industriales esto limitó su uso a grandes compañías y a sectores que producían grandes lotes de producto y que podían afrontar este tipo de inversiones.

Durante tiempo, se ha pensado que una PYME (Pequeña y Mediana Empresa), y/o que un trabajador no experto en programación, difícilmente podrían generar soluciones robóticas para sus procesos productivos. Y, mucho menos, poder interactuar con el robot para para diseñar su programación, o ajustarla (en el fondo reprogramar) para mejorar o redefinir sus usos.

Pero, como se ha indicado en el epígrafe anterior, los retos de presente en la industria del siglo XXI han colocado a los sistemas de "Programación por demostración" (PbD), ante la responsabilidad de llevar la robótica a cada vez más empresas, y a sus trabajadores (usuarios finales) como actores activos en el diseño y rediseño de sus procesos productivos basados en equipos robotizados.

La programación de robots por demostración (PbD), o el aprendizaje desde la demostración (LfD, siglas de Learning from Demostration), forman parte de un proceso en el que la robótica podrá atender un mayor número de tareas de manera flexible y accesible para un mayor número de empresas y usuarios.

La PbD y el LfD son dos técnicas con una fuerte vinculación. Hasta el punto de que no se puede plantear el sentido de una sin que esté vinculada a la otra. Por ello a lo largo del resto del TFM se va a utilizar el término PbD en sentido amplio, con lo que se estarían incluyendo a ambas.

Con ellas el usuario final puede conseguir que el robot realice movimientos y tareas adaptadas a sus necesidades, modificarlos y, en definitiva, programar de manera intuitiva el desempeño del equipo robótico de acuerdo con lo que se pretenda que ejecute.

Los sistemas PbD se han visto como una oportunidad de democratizar el acceso a los sistemas robóticos para las PYMEs y para usuarios no especialistas.

Utilizando una interfaz intuitiva, el usuario podría programar las órdenes con las que hacer trabajar después a los sistemas mecánicos (especialmente robots).

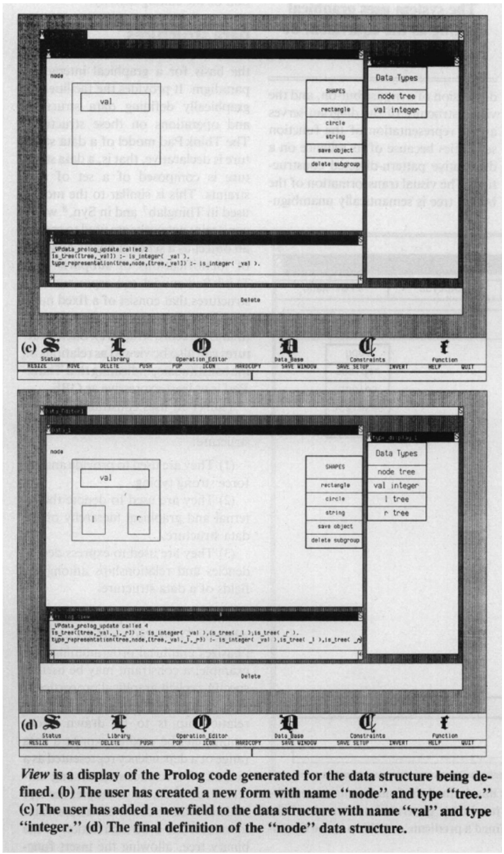
Pero la PbD no es sólo "una forma más fácil para que los usuarios sin conocimientos de programación de robots creen una nueva aplicación (…). Además, facilita la programación del robot acelerando el proceso de programación de nuevos productos, cumpliendo así el requisito de fabricación flexible en esta nueva época industrial" (Zhou et al, 2020; p. 689).

La PbD ha tenido un importante desarrollo en los últimos años, gracias a los avances tecnológicos aparecidos (tanto en hardware, como en software). En una sociedad cada vez más tecnológica, el uso de dispositivos y nuevas tecnologías es cada vez más generalizado, y ya

no sólo en las generaciones jóvenes, sino en el conjunto de la sociedad. La cultura de disponer y usar los dispositivos para cada vez más tareas está muy extendida, y eso favorece la receptividad de cambios también en los procesos industriales de todo tipo de empresa.

Hoy en día, son numerosos los trabajos que plantean avances en los procesos de PbD. Y muestran una gran variedad en cuanto a dispositivos de entrada utilizados, estrategias de aprendizaje, o formas de mejorar los sistemas de programación (bien sea en el diseño algoritmos basados en plantillas, o de algoritmos basados en modelos).

Pero, curiosamente los primeros trabajos que utilizan el término PbD, son de hace más de 35 años (Rubin et al, 1985). En este momento, el concepto se refería a una semántica para modelar programación, ya con la filosofía de los actuales sistemas de PbD. Es decir, utilizar metodologías más sencillas e intuitivas de edición gráfica dinámica, para elaborar un programa complejo, con un sistema de restricciones que se iba generando a través de la inclusión de "almohadillas de razonamiento" (Think Pads).



View is a display of the Prolog code generated for the data structure being defined. (b) The user has created a new form with name "node" and type "tree." (c) The user has added a new field to the data structure with name "val" and type "integer." (d) The final definition of the "node" data structure.

Figura 1.5. Vista de pantallas del sistema PbD de Rubin et al (1985). Tomado de Rubin et al, 1985; p. 75.

La programación tenía una naturaleza mucho más intuitiva que los sistemas de lenguaje máquina, y la metodología Think Pad programaba invocando componentes del sistema de restricciones, configurando así el programa.

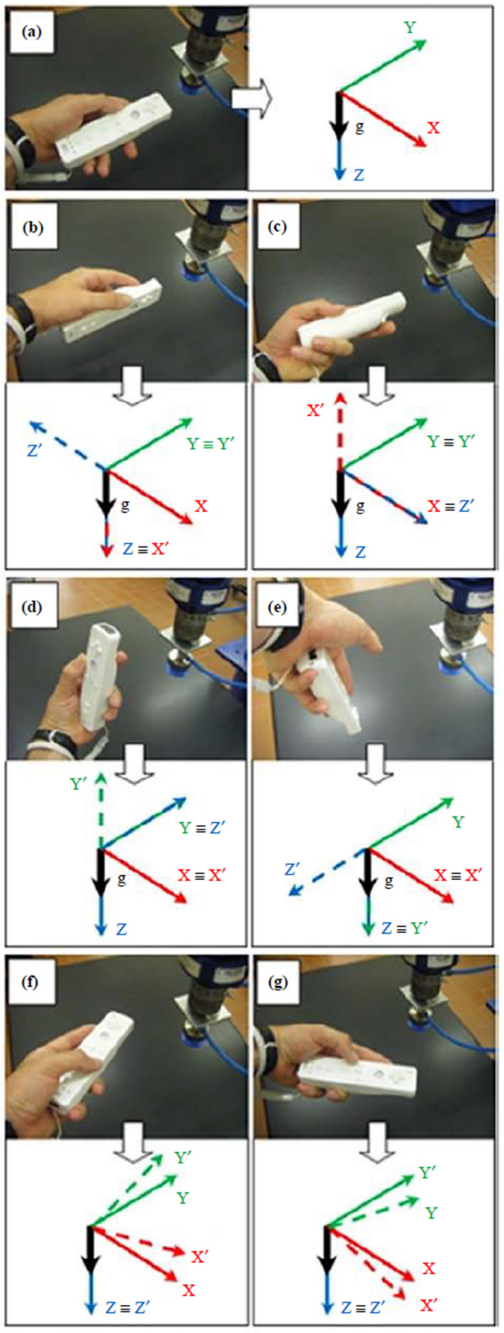
La programación se realiza mediante el encapsulamiento de los indicadores de programación en figuras (almohadillas de razonamiento), que sustentan instrucciones de programación que se incorporan al propio programa.

Su característica de ofrecer una representación dinámica de las funciones a programar es la característica común de los actuales sistemas de PbD. En todo lo demás, los cambios son sustanciales. Ahora, las discusiones y propuestas ya se centran en el mejor tipo de interfaz (pues existen múltiples opciones que en aquel momento ni se podían imaginar), o en las opciones de diseño del algoritmo (destinado, no sólo a elaborar un programa; sino a su uso en la industria robótica).

Las interfaces de interacción entre persona y programación, empezaron siendo los teclados de ordenador, y de ahí pasaron a otros dispositivos conectados físicamente al equipo (ratón, lápices ópticos, …).

El siguiente paso fueron los dispositivos inalámbricos, conectados normalmente por infrarrojos (donde además de las versiones inalámbricas de ratón o lápiz óptico, se incorporan otros dispositivos variados como cámaras o mandos).

En este sentido, se ha trabajado desde dispositivos como los joysticks (Nagata et al, 2001) o los bolígrafos digitales (Pires et al, 2007), que dieron paso a otros dispositivos que trabajaban con infrarrojos y conexión inalámbrica, como el mando de la Wii (Neto et al, 2010).



Notes: (a) – no rotation; (b) – rotation around the *Y*-axis in the negative direction (*Ry* –); (c) – (*Ry* +); (d) – (*Rx* –); (e) – (*Rx* +); (f) – (*Rz* +); (g) – (*Rz* –)

Figura 1.6. Tipos de movimientos utilizados en el sistema de LfD por Neto et al (2010; p. 143)

Dispositivos como el mando de la Wii, joysticks, o el dispositivo Kinect de la Xbox, aparecen en varios de estos estudios, como interfaz de entrada de registros para el aprendizaje del modelo. Dispositivos que no serían ni difíciles de adquirir, ni caros, pero que tienen un uso singular, es decir no forman parte de objetos de uso cotidiano, como sí lo son ya el smartphone, o los smartwatches.

Todos tienen en común el reconocimiento de gestos basado en acelerómetros. Es esta una tecnología que brinda muchas posibilidades. Lleva más de una década siendo estudiada por los autores PbD (Neto et al, 2010; ….), pero la presencia de propuestas sobre los diferentes dispositivos se ha ido produciendo en virtud de la mayor o menor popularización de los mismos, dentro de una limitación de disponer de la tecnología que permitiese parametrizar la información desde los dispositivos a unas coordenadas (se está hablando de sensores tipo acelerómetro, fundamentalmente)

Wang y Li (2016) trabajan con un dispositivo de entrada al sistema, como el teléfono móvil.

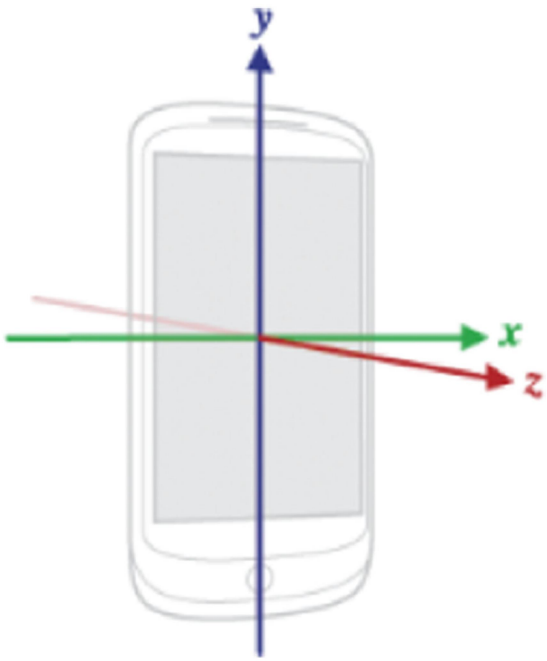


Figura 1.7. Tipos de movimientos utilizados en el sistema de LfD por Wang y Li (2016; p. 8644)

El smartwatch (con el requisito de llevar incorporado un acelerómetro triaxial), cumple con los requisitos que se están considerando para una interfaz óptima en un sistema LfD de aprendizaje por demostración. Y, sin embargo, aún son pocos los trabajos que utilizan dispositivos de pulsera en sus propuestas.

Chowdhury y Chattopadhyay (2020) plantean un algoritmo de detección de actividad a través de microcontroladores de un dispositivo de muñeca.

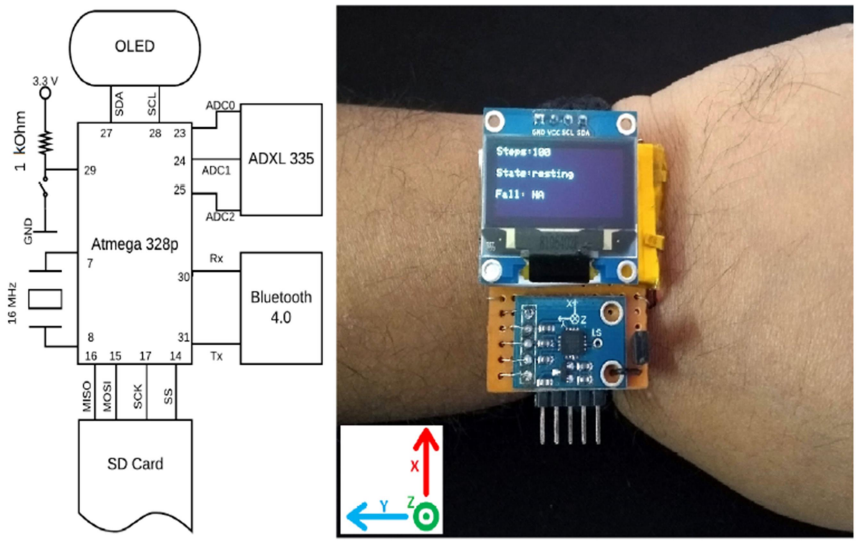


Figura 1.8. Dispositivo de pulsera de Chowdhury y Chattopadhyay (2020; p. 7526)

Aunque no se propone con un uso para la manipulación robótica, sino como una mejora de eficiencia respecto a las opciones de pulseras de medición de ejercicio físico disponibles en el mercado, su interfaz y la transmisión de información a través de un acelerómetro triaxial, lo plantean como un predecesor de este trabajo.

Sin embargo, hasta ahora se está hablando sólo de dispositivos basado en sensores de captura de movimientos. Pero para hacer un análisis completo de esta fase de LfD, y aunque no van a ser objeto del trabajo de este TFM, es conveniente hacer un repaso de otras opciones de interfaces, que permita completar esta revisión de la situación actual en cuanto a la fase de Aprendizaje desde la Demostración LfD.

Si se hiciese una primera distinción entre dispositivos de entrada, se tendrían:

- Por un lado, los de **reconocimiento de voz**, cuyo desarrollo ha sido muy importantes en otros tipos de uso, como la domótica y los ya famosos asistentes virtuales (Siri, Alexa, Google Assistant o Cortana). Para su uso en la robótica industrial, surgen ciertos problemas iniciales (como la eliminación del ruido circundante, o la dificultad de que un usuario estándar domine un lenguaje universal con limitaciones de idiomas, precisión de los términos correctos, …) se han ido solucionando, y se presentan numerosos sistemas basados en instrucciones por voz, o que combinan voz y gestos, como interfaz de programación (Pires, 2005).
- Por otro lado, está el **reconocimiento de gestos**. El lenguaje de gestos es un sistema de lenguaje más reducido, pero con un componente intuitivo mayor. Dentro de la captura de gestos, a su vez, hay varios tipos de interfaces:

- **Interfaces basados en la imagen** (con reconocimiento de gestos y miradas). Que trabajan fundamentalmente a través de cámaras de video (Jha y Chiddarwar, 2017; Zhou et al, 2020).
- **Interfaces de interacción física**. Que utilizan sensores de captura de movimientos y que convierten en órdenes los movimientos humanos. Y que principalmente son los ya comentados (joysticks, mandos de la Wii, smartphones, smartwatches, …).

El gesto de la mano es uno de los más habituales en la vida cotidiana. Además de ser libre, flexible e intuitivo, tiene un carácter más universal que otros tipos de lenguaje, pues no está sometido a las restricciones del idioma, y es accesible incluso a personas con diversidad funcional (como sordomudas, que pueden hacer un uso muy interesante en cualquiera de las opciones de interfaz por gestos, o incluso ciegas, que encontrarían una utilidad en el caso de interfaz por interacción física a través de dispositivos manuales).

En los procesos de trabajo de LfD y PbD se incluyen una serie de fases, cada una de las cuales, a su vez, permite algunas variantes en su diseño:

1. La persona que diseñará el programa de gestión de los movimientos robóticos muestra los comportamientos de movimiento que el robot deberá ejecutar después.
2. El programa de manipulación del dispositivo robótico extrae la información de esas demostraciones, usando las coordenadas XYZ del acelerómetro triaxial del dispositivo, que se almacena configurando los diccionarios de gestos y movimientos.
3. Se elabora el algoritmo de reconocimiento a partir de una de las técnicas (más comúnmente plantillas o modelos) que se haya decidido utilizar. Dicho algoritmo será la base de la posterior programación que se haga a partir de él. Y, además, deberá ser luego ajustado a las especificaciones de funcionamiento robótico en el caso de cada programador-usuario, y a las aplicaciones que necesite desarrollar.

   Las técnicas LfD tienen en común que la programación no se hace por simple programación de movimientos punto a punto, sino por objetivos (agarrar, desplazar hacia delante, …) con lo que el lenguaje de programación se hace más accesible al usuario final.

4. El programa es ajustado en las variables clave de ejecución (restricciones de espacio para cada tarea, velocidad del movimiento, aceleración, celdas específicas de ejecución, requisitos para evitar obstáculos o resolver imprevistos, etc.) para cada aplicación y programador-usuario que lo vaya a utilizar.

   Esta fase PbD es iterativa, y supone la mayor oportunidad de este proceso. Pues puede permitir continuos ajustes por parte del usuario final, de acuerdo con cambios de circunstancias.

5. Transmisión de las órdenes programadas a los mecanismos de actividad del robot.
6. También cabe la posibilidad de que el robot, y el resultado de sus movimientos, retroalimente al programa, mejorando la precisión de la programación (se entraría en

> el campo de la Inteligencia Artificial). Y entonces lo que se haría es la oportuna corrección de los problemas de mapeo, o de modelo, bien mediante correcciones manuales por el usuario, bien mediante "sistemas de aprendizaje y capacitación de circuito cerrado" (Jha y Chiddarwar, 2017)

Incluso, el sistema PbD puede dar un paso más, con la elaboración de modelos cinemáticos y dinámicos (donde el robot actúa como le indica un compañero humano que le da las instrucciones en tiempo real), a veces por interfaz de voz, y a veces de interacción física (Pires et al, 2009).

El presente TFM se va a centrar sólo en las fases 1 a 3. Por ello, se considera interesante revisar los trabajos sobre PbD publicados en los últimos años con incidencia sobre las mismas. Se entiende que ayudarán a desarrollar mejor las fases de LfD y PbD que este TFM desarrolla.

Sistemas de diseño del algoritmo de reconocimiento.

El diseño del programa que debe regular el funcionamiento del sistema robótico tiene una fase inicial que es la de construir toda la estructura de órdenes que luego deberán ejecutarse a partir de información que garantice que el robot hará lo que debe hacer.

Para ello, hay un primer algoritmo que es el "algoritmo de reconocimiento". Este algoritmo debe ser sometido a validación, antes de incorporarse al conjunto del sistema de programación para el funcionamiento del sistema robótico.

Un algoritmo de reconocimiento es un programa que recoge y almacena los datos obtenidos por un dispositivo, los procesa y los clasifica para reconocer gestos tipo.

En la mayor parte de los trabajos de programación PbD, los algoritmos de reconocimiento se pueden situar en una de estas dos categorías:

- Los **métodos basados en plantillas**: como el DTW (Dynamic Time Warping).
- Los **métodos basados en modelos**: como el HMM (Hidden Markov Models), o el ANN (Artificial Neural Networks).

Los DTW han demostrado un rendimiento satisfactorio (Liu et al., 2009). Sin embargo, a la hora de reconocer gestos de diferentes usuarios este rendimiento decrece drásticamente.

Lo que hacen los métodos como el DTW es desarrollar un algoritmo a partir de **plantillas de datos**. Por ejemplo, DTW trabaja con esas plantillas y mide la distancia entre series numéricas deformadas y desplazadas en el tiempo. Agrega diversas observaciones, realizadas en momentos distintos, y de ahí saca su plantilla. Es decir, lo que hace es desarrollar un algoritmo que alinea series numéricas deformadas en el eje tiempo, hasta obtener una coincidencia óptima entre ellas. Este fundamento se utiliza para determinar si las series son más o menos

parecidas, permitiendo así determinar si se corresponden a un mismo gesto o, por el contrario, son gestos diferentes.

Los **métodos basados en modelos**, como su nombre indica, generan un modelo a partir de los datos.

El reconocimiento de gestos basado en acelerómetros se ha utilizado para construir modelos de Redes Neuronales Artificiales (ANN por sus siglas en inglés, Artificial Neural Networks) (Yang et al, 2006; Neto et al, 2009); o los llamados modelos ocultos de Markov (HMM por sus siglas en inglés Hidden Markov Models) (Kela et al, 2006).

HMM pueden funcionar bien en muchas aplicaciones dependientes del usuario. El modelo busca determinar los parámetros ocultos (de ahí su nombre) de una cadena de datos observados.

Neto et al (2010) utilizando un algoritmo ANN, previamente entrenado, lograron una tasa de reconocimiento de gestos y posturas de hasta el 96%, considerada una tasa plenamente válida.

Wang y Li (2016) proponen un método de reconstrucción mejorado llamado MVSAMP que, según ellos, obtiene un mejor rendimiento que los algoritmos tradicionales básicos de DTW y HMM.

Pero no todos los algoritmos de reconocimiento se deben situar en una de estas dos categorías.

Yin et al (2014) proponen un enfoque para el reconocimiento de gestos, mediante acelerómetro en dispositivo manual, que no utiliza ni métodos basados en plantillas, ni métodos basados en modelos. Utilizan los rastros de gestos que se proporcionan habitualmente en los propios dispositivos de reconocimiento de gestos.

Utilizan muestras de entrenamiento, pero no para construir el modelo, o la plantilla, sino para validar su propuesta de modelo.

A partir de los datos recogidos, se procede al diseño del algoritmo de reconocimiento. Y esa fase también incorpora muchos elementos de análisis y de actuación.

En el proceso de trabajo, es una fase importante la del correcto ajuste entre el movimiento de la interfaz y la programación posterior. Zhou et al (2020) proponen un método de inferencia para corregir los problemas de ajuste entre observación y el algoritmo de reconocimiento que dará pie a la posterior programación de usuario.

Zhou et al (2020) utilizan una interfaz de visión por cámara, y trabajan especialmente la correcta interpretación de los gestos y su transmisión al programa que determinará después la ejecución del robot.

Pueden ocurrir desde errores de visión a errores de calibración. Los resultados de los algoritmos pueden ser inexactos, produciendo fallos en la etapa de ejecución del robot (Zhou et al, 2020; p. 690).

Ese problema se corrige utilizando métodos de coincidencia entre la observación y el movimiento pre-programado.

Wang y Li (2016), por su parte, proponen un proceso de trabajo que consiste en:

- Pretratamiento con sistema DTW y programación por afinidad (AP) y representación dispersa (SR).
- A partir de ahí, se propone un algoritmo de compensación para solucionar los problemas causados por el rango de amplitud del acelerómetro.
- Y utilizan la teoría de transformación de coordenadas para aliviar el efecto del ángulo de desplazamiento, esto debido a que empleaban un mando de Wii, y la inclinación del mando altera los resultados.
- Y en la etapa de mapeo hacen una representación de la variedad de trazos de los gestos.
- A partir de ahí, proponen un método de reconstrucción mejorado que llaman MVSAMP (Modified Variable Sparsity Adaptative Matching Pursuit), a partir del método SAMP.

Yin et al (2014) proponen una programación independiente del entrenamiento del usuario. Usando el rastro de gestos del proveedor del dispositivo, y diseñan un algoritmo que llaman MMDTW (Mapping based Modified DTW) al ser un algoritmo de mapeo basado en el sistema DTW modificado.

Lau (2009) indica que el principal problema de todas las propuestas de PbD ha sido su usabilidad.

Y, como algunos otros trabajos, Lau (2009) se ha centrado en analizar los errores más habituales en la PbD para establecer los principales puntos de atención en el desarrollo de un sistema de PbD.

Si se hiciese una revisión de estos riesgos, se destacaría:

- Se suelen introducir un número escaso de registros para el aprendizaje del modelo (training examples) y la elaboración del algoritmo de reconocimiento. Es importante, por ello, ser capaces de manejar el ruido de las muestras de aprendizaje. Ese ruido es el que se produce cuando el programa se ha diseñado con un conjunto de registros que no ha recogido todas las singularidades posibles del usuario final, con lo que la posterior programación de usuario no ejecutaría con precisión la orden al sistema. La heurística podría interpretar de manera incorrecta la acción a realizar. Y cuando la programación pretende parametrizar varios pasos de desempeño de un equipo robótico, sólo el error en uno de ellos inhabilita toda la programación.

- Que el usuario final no entienda las instrucciones, o que no confíe en su pericia para manejar el proceso de programación.
- Que cuando el usuario final quiera ajustar el sistema, porque tiene mayor conocimiento sobre su tarea que el propio sistema, el programa descarrile la automatización, porque el sistema de ajuste no sea lo suficientemente preciso o sencillo.

Como ya se ha comentado con anterioridad, PbD también sirve para el desarrollo de la Inteligencia Artificial. De manera que, con nuevas instrucciones, o con un mayor número de registros de aprendizaje, el programa aprenda y mejore la variabilidad. Eso se llama aprendizaje automático por sistemas de automatización (Machine Learning) (Lau, 2009; p. 67).

1.3. OBJETIVOS DEL TRABAJO FIN DE MASTER.

Como ya se anticipaba en el epígrafe 1.1., este Trabajo Fin de Master se va a centrar en una pequeña parte de todo este mundo de los avances en la Robótica Industrial. En concreto, en los sistemas de comunicación persona-robot, y en los lenguajes de programación vinculados, de alguna manera, con los sistemas robóticos.

De manera más específica, el TFM se va a centrar en desarrollar la fase de LfD dentro de la PbD. Para ello, la tarea crítica es la de elaboración del algoritmo de reconocimiento.

El algoritmo de reconocimiento en este trabajo es:

- un programa (lo que es),
- que recoge y almacena los datos obtenidos por un acelerómetro, los procesa y los clasifica (cómo trabaja)
- para reconocer gestos tipo (para qué sirve).

En su diseño, no se pierde de vista un propósito final, como es que termine comunicándose con los sistemas de programación de los robots industriales dedicados a tareas de pick and place.

No obstante, es muy importante saber delimitar lo que se pretende hacer en cualquier trabajo. En ningún momento este trabajo se plantea entrar en la fase del diseño de programación de los propios sistemas robóticos. No sólo por la diversidad de sistemas de programación existentes (Offline, simuladores, Middleware,…) sino porque incluso cada marca desarrolla su propio software, como ABB con RAPID,… (Valera Fernández, 2020a; diapositivas tema 4).

Una meta que sí que es alcanzable y precisa, es la programación del dispositivo interfaz (en este caso un reloj inteligente con acelerómetro). Que servirá como emisor en un proceso de comunicación entre ese interfaz y los programas de funcionamiento de los diferentes dispositivos robóticos.

Lo que se programa en el dispositivo interfaz es el algoritmo de reconocimiento. Que, una vez instalado, quedaría en el propio software del reloj. La información que contuviese ese software permitiría que un usuario manipulando el reloj (haciendo una secuencia de movimiento de los contenidos en el diccionario de gestos) pudiese comunicarse con el programa que manejaría a un sistema robótico concreto, siempre que dispusiese también del software adecuado. Estaría por tanto, trabajándose dentro de la técnica de PbD.

La tecnología que utilizan los relojes inteligentes es lo suficientemente universal como para que un resultado eficaz del algoritmo propuesto, pueda convertirse en un logro de valor a nivel tecnológico-científico.

El objeto general del trabajo, por tanto, consiste en elaborar un algoritmo de reconocimiento de gestos sirviéndose del acelerómetro de un reloj inteligente.

Para ello, se van a plantear una serie de objetivos específicos, con los que implementar ese objetivo general:

1. Realizar un estado de la cuestión (revisión teórica), para conocer las características de las diferentes opciones de diseño de algoritmos de reconocimiento de gestos, que se están utilizando en otros trabajos sobre PbD y que se podrían ejecutar en el dispositivo elegido.
2. Contrastar aquellas metodologías o técnicas principales, identificadas tras la revisión teórica como más interesantes, y obtener sus porcentajes de precisión para el objetivo general de este TFM.
3. Validar los resultados de reconocimiento, cuando manipule el reloj un usuario distinto a los que generasen la librería de gestos (independiente del usuario[3]), mediante las herramientas que permiten medir la eficacia de reconocimiento de este tipo de algoritmos. En definitiva, contrastar el algoritmo creado.

1.4. METODOLOGÍA DEL TRABAJO FIN DE MASTER.

[3] En el capítulo 2 se introducirá este concepto al hablar de las técnicas DTW y MLP. Se habla de análisis independiente del usuario cuando el algoritmo es capaz de interpretar correctamente los gestos de una persona distinta a las que generaron los registros para elaborar la librería de gestos.

Respecto a la metodología para la revisión teórica, se llevará a cabo una revisión de artículos académicos, TFMs y Tesis publicadas, sobre metodología LfD y PbD, con la que se hará una base documental de trabajo que se sistematizará después en un capítulo de revisión teórica.

Para los objetivos específicos 2 y 3 se trabajará en las siguientes tareas del proceso de PbD:

1. Diseño de un diccionario de gestos que permita sustentar cualquier programación básica de un sistema robótico dedicado a tareas de pick and place. A partir de la grabación de un número suficiente de repeticiones de ese conjunto de gestos se elaborará una librería para cada uno de los gestos del diccionario, que servirá después para entrenar los algoritmos y realizar su testeo.

   Así, se define un diccionario de 11 gestos y se crea una librería con una base de datos de 5.500 registros a partir de 4 participantes.

2. Preprocesado de las señales obtenidas por el acelerómetro y almacenadas en la librería de gestos, obteniendo las características que pueden definir el movimiento que se incluirá en el algoritmo de reconocimiento.

3. Se analizarán después los principales métodos de reconocimiento de gestos. En concreto, aquellos que, según la bibliografía trabajada, aparecen como habitualmente utilizados. En ese sentido se analiza el uso del método DTW (Dynamic Time Warping) como referente de los métodos basados en plantillas. Y del grupo de los métodos basados en modelos, se analizará un método del tipo ANN (Artificial Neural Networks), en concreto el MLP (Multilayer Perceptron). Todos ellos se estudiarán previamente con detalle en el capítulo 2 de este TFM.

4. Por último, se elabora el algoritmo de reconocimiento en base a los resultados del análisis comparado. Ese algoritmo será validado en su eficacia a través de su tasa de reconocimiento, tanto teóricamente (con las matrices de confusión obtenidas de los datos de la librería), como de forma aplicada (mediante el manejo del dispositivo por un usuario distinto sobre la misma librería de gestos, y obteniendo una matriz de confusión entre algoritmo y la ejecución del nuevo usuario).

CAPÍTULO 2. MARCO TEÓRICO

Tal y como se ha ido comentando a lo largo del capítulo 1, el objetivo del presente trabajo consiste en desarrollar las 3 primeras fases del proceso PbD, siendo el resultado final un software que incorpora el algoritmo de reconocimiento de gestos, y que se ejecutará en un reloj inteligente.

Este capítulo trata de recoger las diferentes técnicas que puede emplear el algoritmo, centrándose en aquellas elegidas para ser analizadas en el presente trabajo.

2.1. ANÁLISIS DE COMPONENTES PRINCIPALES

El análisis de componentes principales PCA, por sus siglas en inglés (Principal Component Analysis) es ampliamente utilizado en estadística, informática, clasificación y en identificación y monitorización de procesos de control.

La idea principal del ACP es: reducir la dimensionalidad de un conjunto de datos conservando, dentro de lo posible, la variabilidad existente en dicho conjunto de datos. ACP ayuda a analizar grandes conjuntos de datos con dimensiones muy grandes, además de ofrecer compresión de datos sin mucha pérdida de información.

Este método ha sido visto en el Máster Universitario de Ingeniería Industrial, concretamente en la asignatura “Identificación y Control de Sistemas Complejos”, que se imparte en la especialización de control, automatización y robótica. Y se implementará en este trabajo con la finalidad de determinar la bondad de las características de los datos seleccionadas.

Los pasos a seguir para aplicar el método son (Sabinas, 2013; pp. 18-19):

1. Seleccionar los datos que se van a analizar.
2. Extracción de la media: se resta a cada muestra el valor de la media de su señal, produciendo un conjunto de media cero.
3. Descomposición en valores singulares.
4. Selección de los componentes y evaluación de las características: aquellas componentes con las ganancias más grandes son las que permitirán clasificar mejor, pues presentan mayor varianza, mientras que, si estas ganancias son pequeñas, deberían escogerse otras características, pues las actuales no permiten diferenciar bien los gestos.

2.2. PRINCIPALES GRUPOS DE GESTOS SEGÚN SU NATURALEZA.

Cuando se trabaja con análisis y estudio de gestos, se puede hablar de dos clases de gestos: los gestos estáticos y los gestos dinámicos (Kaâniche, 2009).

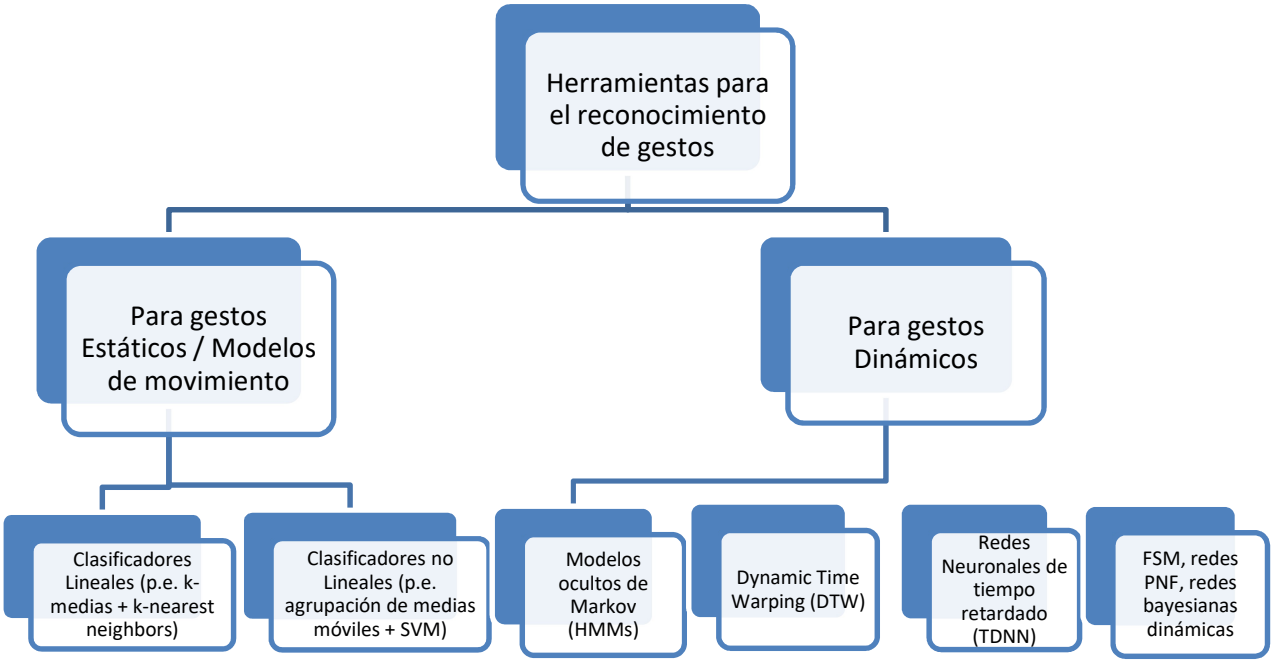


Figura 2.1. Herramientas para reconocimiento de gestos. Tomado de Kaâniche (2009; p. 17)

Por un lado, están los estudios y los métodos que trabajan con gestos estáticos. Estos identifican posturas, posiciones del cuerpo. Suelen emplearse en el reconocimiento basado en imágenes, por ejemplo, detectar el estado de ánimo de una persona en una fotografía o interpretar un mensaje en ella.

Por otro lado, están los estudios y métodos que trabajan con gestos dinámicos. Los gestos dinámicos se caracterizan por el reconocimiento de una secuencia de gestos que varían en el tiempo. Por tanto, hay una variable clave en su análisis, la variable tiempo. Normalmente, el registro de información se realiza mediante el uso de series temporales.

Con independencia de que se trabaje con gestos estáticos o dinámicos, los procesos de PbD después se registran en los diccionarios de gestos para terminar construyendo plantillas o modelos, con los que se elaboran los algoritmos de reconocimiento.

Tal y como se recogía en el epígrafe 1.2., en la mayor parte de los trabajos de programación PbD, los algoritmos de reconocimiento utilizados se pueden situar en una de estas dos categorías:

- Los **métodos basados en plantillas**: como el DTW (Dynamic Time Warping), que comparan los gestos desconocidos con plantillas de gestos conocidas, buscando similitudes entre estos.
- Los **métodos basados en modelos**: como el HMM (Hidden Markov Models), o el ANN (Artificial Neural Networks), que utilizan una serie de gestos conocidos con los que entrenar un modelo de reconocimiento.

Tal y como se comentaba también en el epígrafe 1.4. (al hablar de la metodología de este trabajo), cuando se realiza una revisión bibliográfica general sobre ambas categorías, se observa fácilmente aquellas metodologías que predominan entre los diferentes trabajos, y que son consideradas por ofrecer las mejores tasas de reconocimiento de gestos y posturas. En términos generales, son las que se han elegido para desarrollar el TFM. En concreto, se analiza el método DTW (Dynamic Time Warping) como referente de los métodos basados en plantillas, y del grupo de los métodos basados en modelo se analizará un método del tipo ANN (Artificial Neural Networks), en concreto el MLP (MultiLayer Perceptron).

De su análisis comparado, se terminará eligiendo aquella que mejor resultado dé, en cuanto a tasas de reconocimiento de gestos y posturas, para el diseño del algoritmo de reconocimiento de gestos que se integrará en el proceso de trabajo que se ha planteado para este TFM.

2.3. DYNAMIC TIME WARPING

El método DTW (Dynamic Time Warping) es una herramienta flexible para comparar series temporales en presencia de deformaciones no lineales de tiempo" (Müller, 2007; p. 211).

Como se decía ya en el capítulo 1, los métodos que trabajan con plantillas de gestos, como el DTW, desarrollan el algoritmo de reconocimiento de los gestos a partir de plantillas conformadas en una fase previa en la que se ha construido una base de datos, recogiendo más de una muestra de cada tipo de gesto. Concretamente el algoritmo DTW mide la distancia entre series numéricas de las deformadas y desplazadas en el tiempo.

Las observaciones que se integran en la biblioteca de gestos, o consiguientemente en la plantilla, se realizan en momentos distintos, y al agregarse configuran la plantilla.

Después el algoritmo alinea esas series temporales de datos, y las analiza hasta determinar si existe una coincidencia óptima entre ellas, con las que se configura la plantilla.

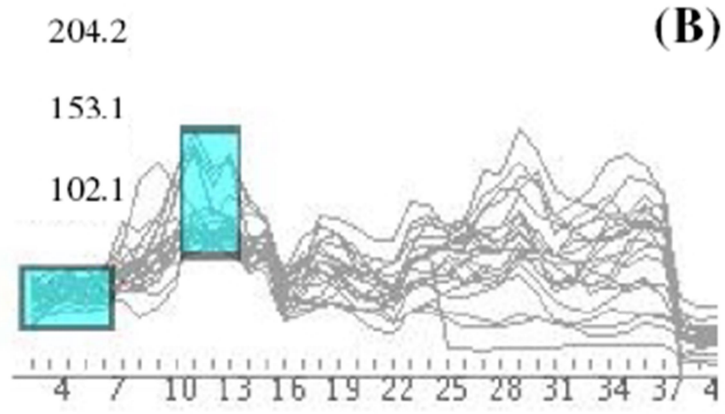


Figura 2.2. Ejemplo de plantilla generada por varios registros de voz (Senin, 2008; p. 15)

El algoritmo determina si la serie de datos registrada es más o menos parecida y, por tanto, se corresponde con un mismo gesto o no. De ahí, el algoritmo de reconocimiento sacará la plantilla con la que se comparará un gesto del usuario-programador final, cuando esté utilizando el mismo para programar un sistema de gestión de un equipo robótico.

Los primeros usos del DTW se plantearon para el reconocimiento de patrones de voz, a través de las ondas sonoras, para usos de reconocimiento de voz. La mejor ruta de deformación, a partir de los datos registrados, refleja el ajuste entre la plantilla y una base y se cuantifica el grado de ajuste entre el ajuste y la base. La puntuación debe permitir comparaciones para poder determinar coincidencias. Se plantea también un límite, como el que se muestra en la Figura 2.3. que determinan los intervalos limítrofes que establecen un límite razonable. (Berndt y Clifford, 1994)

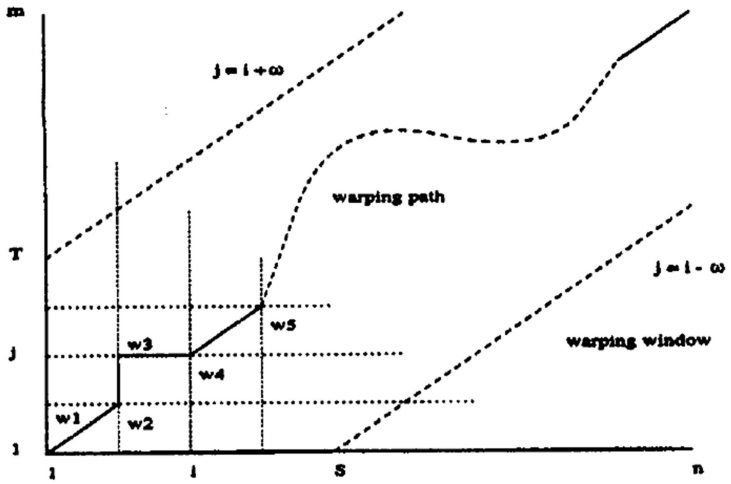


Figura 2.3. Un ejemplo de Warping Path de Berndt y Clifford (1994; p.362)

La indexación en series temporales trabaja considerando que cualquier serie temporal de longitud "n" puede considerarse una "tupla de datos[4]" en un espacio de "n" dimensiones. Una vez se indexa dicho espacio se empieza por realizar comparaciones simples. Pero eso puede suponer que la indexación directa de ese espacio sea ineficiente. Por ello, lo que se plantea en este método es la reducción de dimensionalidad transformando la serie de n elementos en una serie de menor dimensión. (Senin, 2008; p. 16)

Siendo "S" y "T" dos series temporales, cualquier relación entre S y T se puede representar como un camino continuo entre un punto de partida (1,1) al punto final (M, N). Siendo "M" el número de muestras de S, y "N" el número de muestras de T. Un punto cualquiera del camino

[4] En Matemáticas tupla es una lista o secuencia ordenada y finita de elementos.

(i, j) indica que la componente i de S, se empareja a la componente j de T. El coste de emparejar ambas muestras es la distancia entre ellas. A partir de ahí, el siguiente emparejamiento sólo puede ser hacia delante en el tiempo, es decir una muestra posterior a "i" no puede emparejarse con una muestra anterior a "j", y viceversa. La similitud entre S y T se evalúa como la distancia acumulativa mínima de todos los caminos posibles (coste de ajuste) (Liu et al, 2009).

DTW emplea programación dinámica para calcular el coste y encontrar así el camino óptimo. Por tanto, el camino óptimo de (1,1) a (i, j), puede ser cualquiera de los que llegan a sus 3 predecesores (i-1, j), (i, j-1) o (i-1, j-1). Y el coste de ajuste desde (1,1) a (i, j) será la distancia en (i, j) más el menor coste de ajuste de sus predecesores (Liu et al, 2009).

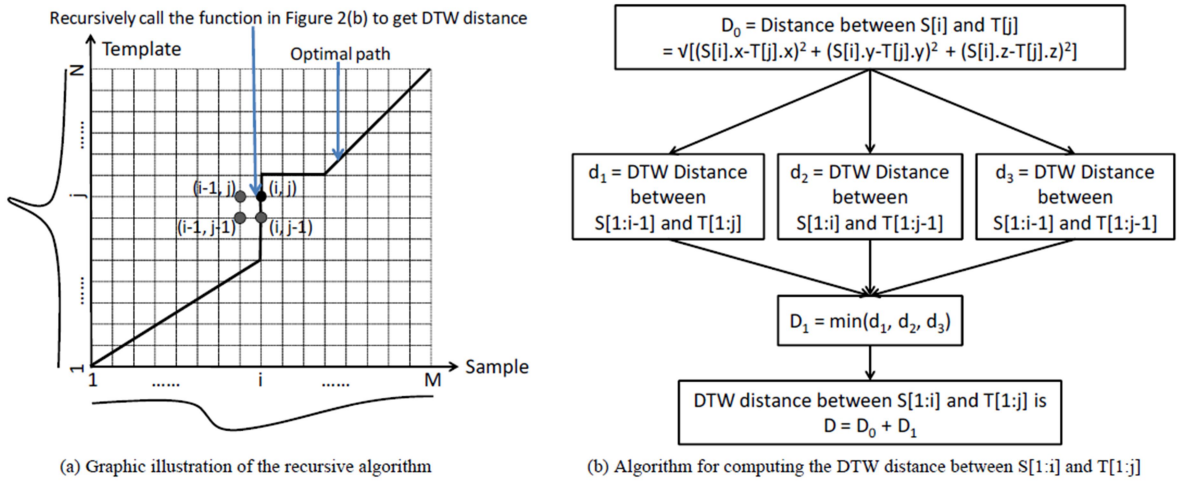


Figura 2.4. Ilustración gráfica del algoritmo DTW propuesto por Liu et al. (2009; p. 4)

2.4. *AFFINITY PROPAGATION* (AP) Y *COMPRESSIVE SENSING* (CS)

El método DTW (Dynamic Time Warping), presenta un desempeño muy eficiente desde el punto de vista del coste computacional y la precisión de reconocimiento, en comparación con otros métodos. Sin embargo, se le considera un método con una alta dependencia del usuario. Es decir, presenta buenos resultados si el programador usuario es la misma persona que realiza las plantillas, limitando así las aplicabilidad y eficacia final del algoritmo.

Para mejorar el algoritmo de reconocimiento elaborado a partir del DTW se pueden, no obstante, utilizar algunas opciones que pasan por depurar sus análisis, complementado su método con otras dos técnicas de *clustering* (Akl y Valaee, 2010; p. 2271). Una de las opciones está más indicada para el caso en el que el modelo es dependiente del usuario y la otra cuando el modelo es independiente del usuario.

Akl y Valaee (2010) proponen el refuerzo del método DTW con la técnica de Propagación de Afinidad (Affinity Propagation, AP) para el aprendizaje del modelo, cuando este es dependiente del usuario. La técnica de reconocimiento dependiente del usuario consiste en comparar mediante DTW los gestos desconocidos con el conjunto de ejemplos obtenidos por alguna de las aplicaciones de Propagación de Afinidad (AP).

Por otro lado, si el reconocimiento es independiente del usuario[5], la comparación por DTW (entre los gestos desconocidos y el conjunto de ejemplos obtenidos por AP) no es suficiente. En ese caso, se recurre a otra de las técnicas, que también se revisa en este epígrafe: la detección por compresión (Compressive Sensing, CS).

Affinity Propagation (AP)

Propagación de afinidad traducido al castellano. Recoge un conjunto de aplicaciones que trabajan en base a un algoritmo de agrupación (*clustering*) que considera todos los datos como posibles ejemplos. Y de forma recursiva[6] busca, hasta encontrar, un buen conjunto de ejemplos y agrupaciones.

La entrada a este algoritmo es la matriz de similitud {S (i, k)}, construida a partir de las comparaciones DTW entre los registros de la plantilla de datos (data points) y los candidatos a ejemplo (candidate exemplars). La técnica de Propagación por afinidad manda: "mensajes de responsabilidad" (llamados r) desde los datos hacia los "candidatos a ejemplos", indicando la idoneidad del dato para ser representado por el ejemplo; y "mensajes de disponibilidad" (llamados a) desde los candidatos a ejemplo hacia los datos, indicando cómo de apropiado sería para el candidato, ser ejemplo del dato.

[5] Se habla de dependiente del usuario cuando la persona de la que se obtuvieron los registros para elaborar la librería es también la que utiliza el después el algoritmo para programar. Y de independiente cuando el método es lo suficientemente fiable para que pueda hacerlo otra persona.

[6] En Matemáticas y en Computación, se utiliza el término recursivo refiriéndose a la definición de los elementos de un conjunto mediante otros elementos del mismo conjunto.

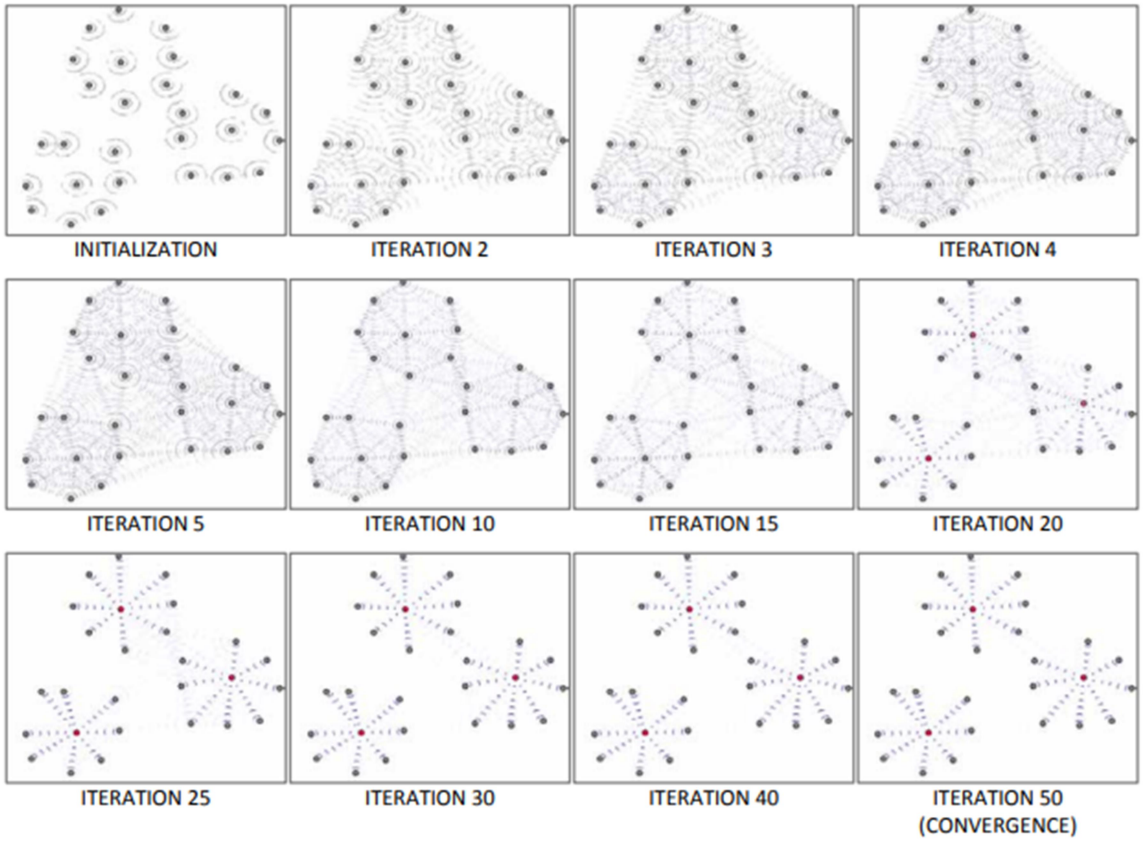


Figura 2.5. Dinámica de Affinity Propagation (Dueck, 2009; p. 44)

El algoritmo utilizado por la técnica de Propagación de afinidad lleva a que los puntos que estén cercanos entre sí se cataloguen como de alta similitud. Mientras que conforme más lejanos aparezcan, la similitud sería más baja (es decir, se parecerían menos).

De esta forma el algoritmo que trabaja con AP genera dos matrices distintas:

- La matriz de responsabilidad, que determina en qué medida cada punto del conjunto de datos es responsable (esto es la importancia que tiene en el proceso de reconocimiento).
- Y la matriz de disponibilidad, que determina la cantidad de otros puntos que cada punto tiene a su alrededor.

A partir de estas dos matrices el algoritmo elegirá sus "exemplars" que son los que estarán en el centro de los distintos clústeres, eligiendo los que tienen el mayor grado de responsabilidad y disponibilidad.

El pseudocódigo que emplea AP es el siguiente (Dueck, 2009; p. 6):

1. Los valores de la diagonal de la matriz de similitud son las preferencias a priori (coste negativo de añadir un clúster), calculadas como la mediana de esa columna.

2. Se inicializan las disponibilidades a cero.
3. Se calculan responsabilidades y disponibilidades hasta encontrar convergencias de la siguiente forma:

$$\forall i,k:\ r(i,k) = s(i,k) - \max_{k':k'\neq k}\left[s(i,k') + a(i,k')\right]$$

$$\forall i,k:\ a(i,k) = \begin{cases} \sum_{i':i'\neq i} \max[0,\ r(i',k)], & \text{for } k = i \\ \min\left[0,\ r(k,k) + \sum_{i':i'\notin\{i,k\}} \max[0,\ r(i',k)]\right], & \text{for } k \neq i \end{cases}$$

4. Las salidas son $\underline{c} = (c_1 \ldots c_N)$, donde c_i indica el ejemplar al que el punto i es asignado, calculado como $c_i = \text{argmax}_k\ [\ a\ (i,\ k) + r(i,\ k)]$.

Compressive Sensing CS

La teoría de Detección por Compresión (Compressive Sensing, CS) fue desarrollada por Candès (2006) y Donoho (2006). Consiste en recuperar una señal con muchas menos mediciones ruidosas que las necesarias para utilizar la tasa de Nyquist[7], que era la que se usaba tradicionalmente.

La señal se transforma en un dominio en el que hay una representación escasa, y entonces la señal se reconstruye utilizando técnicas de optimización (Nahar y Kolte, 2014).

Siendo "y" la repetición de un gesto desconocido, y R la matriz cuyas columnas son repeticiones de la plantilla de los gestos del diccionario, entonces se puede asumir que "y" es una réplica, o una semejanza cercana a una de las columnas de R, y entonces:

$$y = R\theta \tag{2.1}$$

θ es un vector cuyos elementos son ceros, excepto el índice correspondiente a la columna de R de la que "y" es réplica.

La situación de tener una repetición desconocida que replica una repetición de plantilla no existe en un escenario real, entonces para el caso real se tiene:

$$y = R\theta + \epsilon \tag{2.2}$$

Donde ϵ representa al ruido de medida.

Se introduce el preprocesador W, tal que Y=W*y, este parámetro se calcula como

$$W = Q * R^{+} \tag{2.3}$$

Q = orth $(R^T)^T$ es una base ortogonal para el rango de R^T y R^+ es la pseudoinversa de R.

Así el problema de optimización consiste en minimizar θ, sujeto a:

[7] La tasa de Nyquist establece la frecuencia de muestreo mínima que permite reconstruir una señal original sin distorsiones, para el caso de señales cuyo rango se establece entre la frecuencia cero y la frecuencia máxima.

$$Y = Q\theta + W\epsilon \tag{2.4}$$

Al disponer de una señal de 3 componentes, en este caso aceleración en 3 ejes, R serán 3 matrices y θ tres vectores, entonces:

$$\theta_{eq} = \theta_x + \theta_y + \theta_z \tag{2.5}$$

Las técnicas de Detección por Compresión (Compressive Sensing) se basan en dos principios básicos: la dispersión (sparsity), propia de la señal que interesa explicar; y la incoherencia (incoherence), que corresponde al tipo de sondeo que se realiza.

Estas técnicas permiten llevar a cabo simultáneamente el proceso de muestreo y el de compresión de las señales poco densas (reconstruyendo señales reales a partir de un reducido número de proyecciones aleatorias). De forma que se pueden diseñar protocolos de muestreo capturando la información útil de señales dispersas y consiguiendo así guardar toda la información en una cantidad reducida de datos.

Por ello, se puede decir que CS es un protocolo simple para el acopio de datos que, a su vez, es altamente eficiente. Puesto que normalmente las señales reconstruidas prácticamente no muestran diferencias de eficacia de reconocimiento con las señales originales.

Con una muestra reducida en el diccionario de datos y una colección aparentemente incompleta de registros, después la técnica CS es capaz de reconstruir señales reales mediante la implementación de su algoritmo.

2.5. REDES NEURONALES ARTIFICIALES

Las Redes Neuronales Artificiales (ANN Artificial Neuronal Network) son estructuras de tratamiento de la información basadas en los fundamentos de redes neuronales biológicas. Los elementos básicos (las neuronas) se conectan entre sí en diferentes capas mediante conexiones. Estas conexiones adquieren un valor numérico conocido como “peso sináptico”.

Ese peso (W_{ji}) multiplica las señales de entrada y define la importancia de cada una de ellas.

La neurona siguiente se activa si la entrada total supera un cierto umbral. De manera que todo el sistema funciona basándose en una función de activación (Izaurieta y Saavedra, 2000; p. 2).

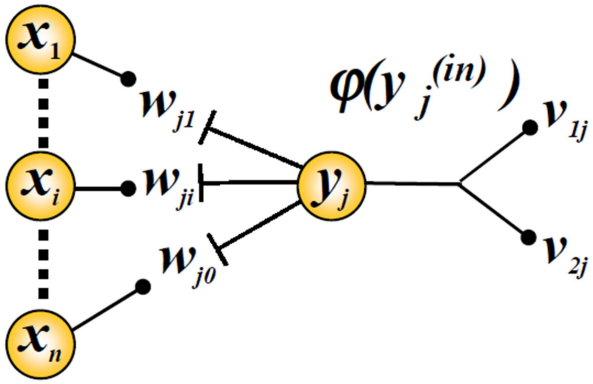


Figura 2.6. Esquema de funcionamiento de una neurona biológica. Tomado de Izaurieta y Saavedra (2000; p. 2)

Las redes neuronales se han aplicado satisfactoriamente en diferentes áreas de conocimiento: biología, medicina, economía, ingeniería, psicología, etc.

La red neuronal, mediante aprendizaje, modifica los valores de los pesos para que, al tener unos valores en las neuronas de la capa de entrada, se consigan los valores deseados en la capa de salida. El valor de una neurona se corresponde a la suma ponderada de los valores de las neuronas que se conectan con la entrada de la neurona, ponderada por los pesos de las conexiones.

Las redes neuronales forman un modelo para encontrar, de forma iterativa, el valor de esos pesos a partir de un conjunto de datos. Una vez la red ha sido entrenada, se puede utilizar para hacer predicciones o clasificaciones.

A partir de ese proceso básico, las redes neuronales artificiales conciben un modelo abstracto y simple de una neurona que replica los procesos de las neuronas naturales.

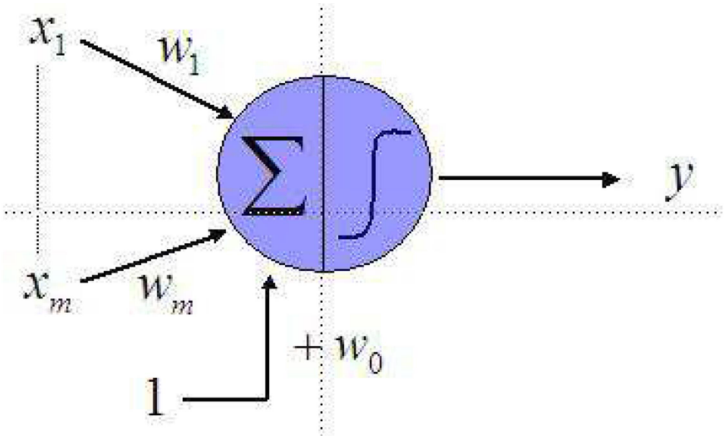


Figura 2.7. Esquema de una neurona artificial (McCulloch y Pitts, 1943). Tomado de Salas (2004; p. 2)

El modelo incluye un vector de pesos $\underline{w}\ (w_1,.., w_d)^T$ equivalente a las conexiones sinápticas de una neurona real.

- w_0 es el umbral de acción o activación,
- el vector $\underline{x}$ es la entrada,
- y el escalar y la salida de la unidad.

La actividad consiste en generar una única salida. La función de activación γ se genera por la suma ponderada entre el vector de entrada $\underline{x}\ (x_1, ..., x_m)^T$ y el vector de pesos $\underline{w}\ (w_1, .., w_d)^T$ más un sesgo w_0. Con todo ello, se genera la siguiente fórmula (Salas, 2004; p. 2):

$$y = \gamma\left(\sum_{i=1}^{m} w_i x_i + w_0\right)$$

Sin embargo, una sola capa no suele dar buenos resultados, por ello conviene construir una red multicapa (FANN, siglas en inglés de Feedforward Artificial Neuronal Network).

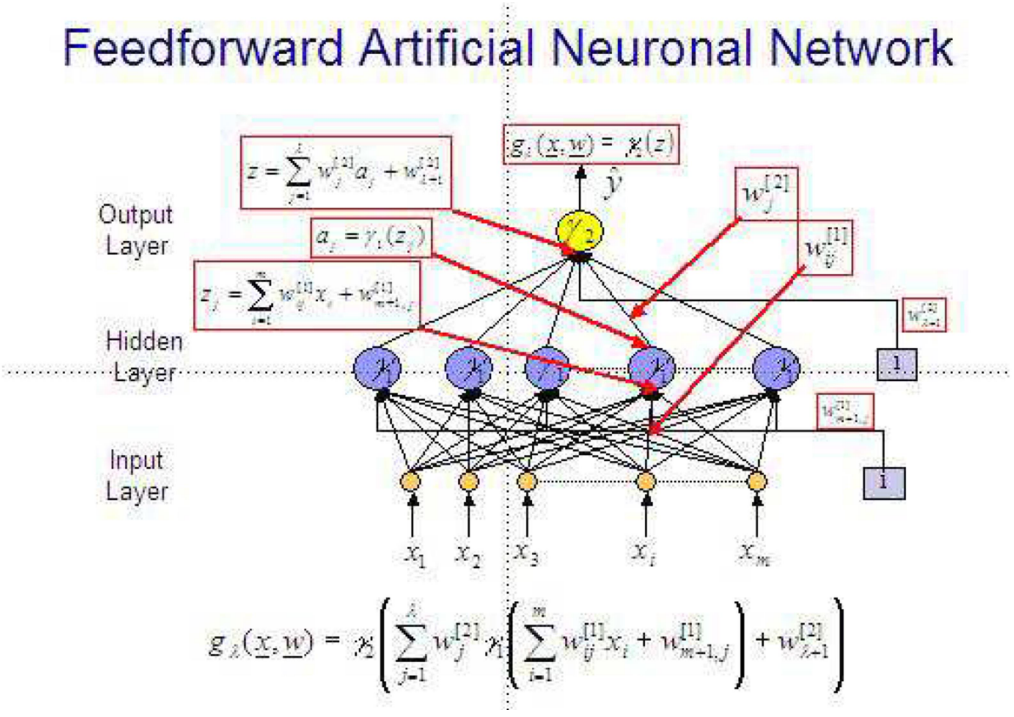


Figura 2.8. Esquema de una red neuronal artificial multicapa FANN (Salas, 2004; p. 4)

Cada tipo de capa (capa de entrada, capa oculta y capa de salida) aprende a encontrar y detectar las características que mejor ayudan a clasificar los datos.

Existen diferentes funciones de activación de neuronas. Estas funciones relacionan el valor de entrada de una neurona con el valor de salida de la misma.

Entre las funciones de activación se puede encontrar (Martín del Brío y Serrano, 1995):

- La función identidad o lineal, que define que la salida es igual a la entrada

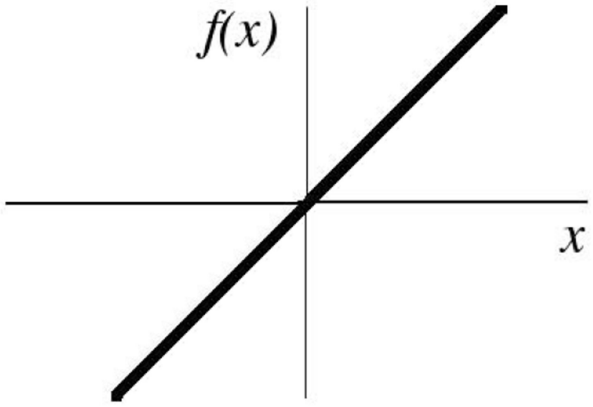


Figura 2.9. Función lineal (Martín del Brío y Serrano, 1995; p. 111)

- La función escalón, que se utiliza si se pretende que las salidas de la red sean variables binarias, pues las salidas de la neurona valen 1 si la entrada sobrepasa cierto umbral y 0 si no se llega.

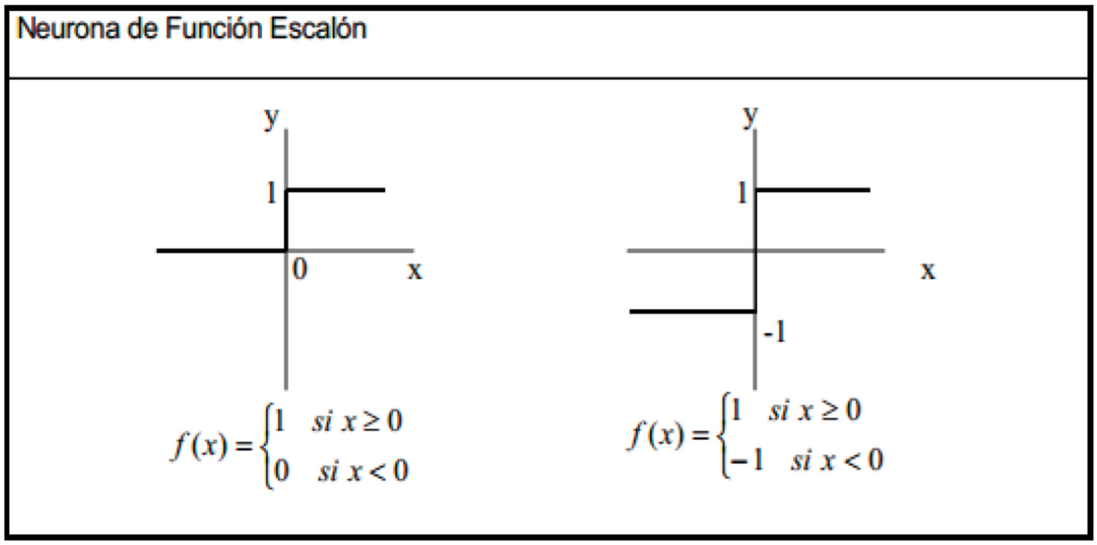


Figura 2.10. Función escalón (Andrade, 2013; p. 17)

- La función lineal mixta, que es apropiada si se desea información analógica a la salida, es una combinación de las funciones anteriores, encontrándose una zona lineal acotada entre un umbral

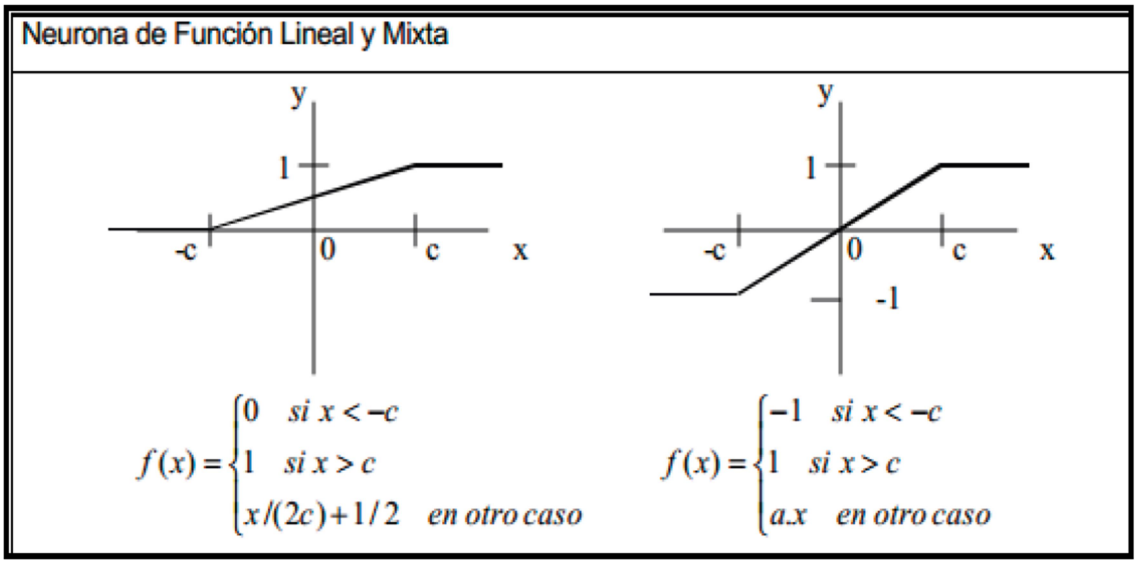


Figura 2.11. Función lineal mixta (Andrade, 2013; p. 17)

- La función sigmoidal, que es la más apropiada si se desea información analógica a la salida, diferenciándose la sigmoidal logística y la tangente hiperbólica. Ambas tienen forma de sigmoide, pero su ecuación es diferente.

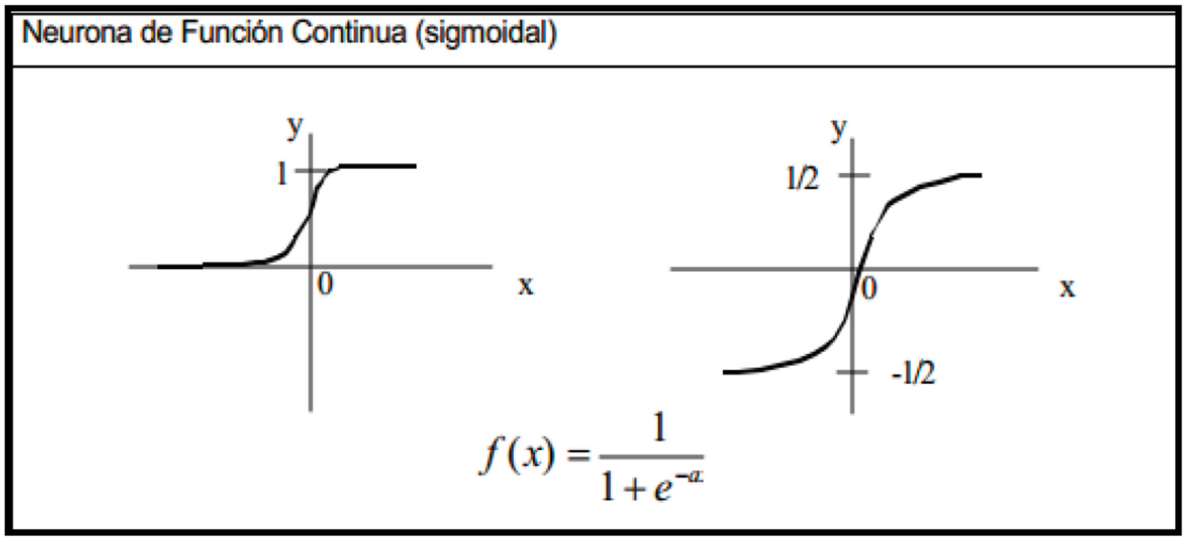


Figura 2.12. Función sigmoidal (Andrade, 2013; p. 20)

- La función Gaussiana, que permite realizar mapeos con un solo nivel de neuronas.

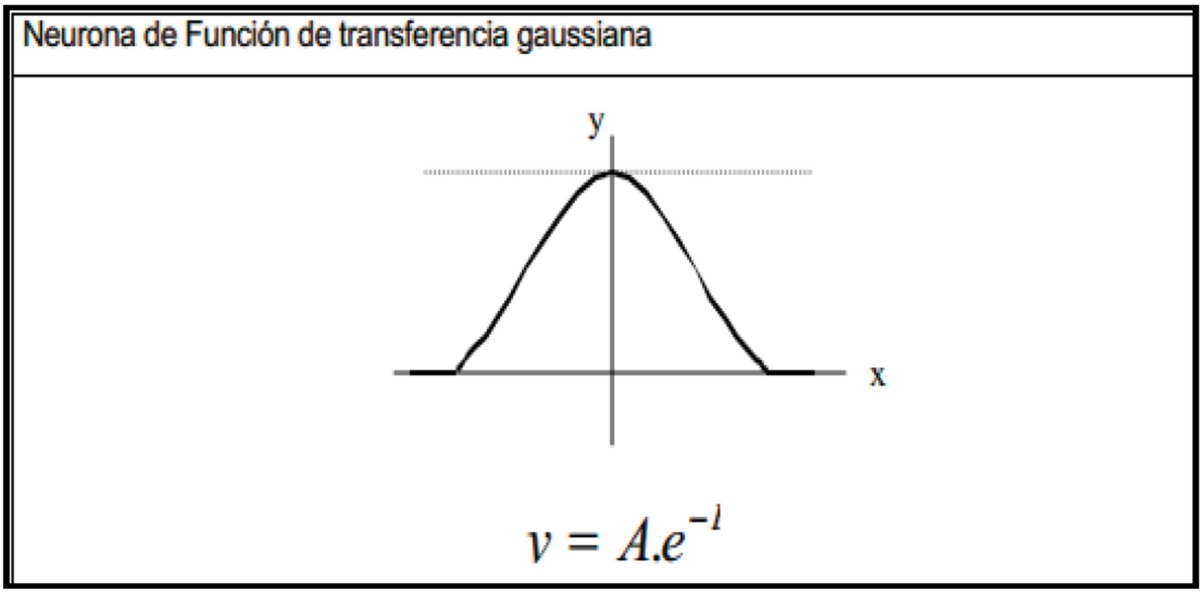


Figura 2.13. Función gaussiana (Andrade, 2013; p. 20)

Tipos de redes neuronales

Existen diferentes tipos de redesneuronales (Andrade, 2013):

Perceptrón simple: Es el tipo de red más antiguo que se conoce, planteada por primera vez en 1943. Consiste en comparar el valor de entrada con un patrón para determinar si se activa o no la neurona, siguiendo una funcion de activación tipo escalón. Utiliza aprendizaje supervisado, necesita conocer el valor de salida esperado para un conjuto de datos de entrada para ser entrenada. Es utilizado en problemas de clasificación y da resultados perfectos cuando los patrones son separables.

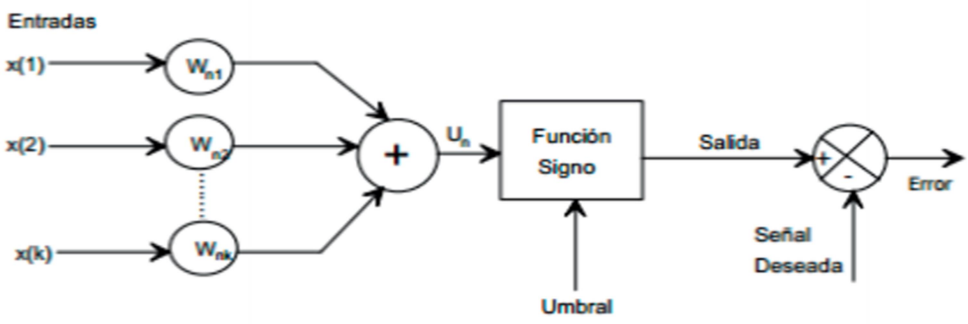


Figura 2.14. Estructura de la red Perceptrón simple (Andrade, 2013; p. 23)

Adaline: Este tipo de red es uno de los elementos principales en el procesado digital de señales. Es muy similar al perceptron, pero utiliza la funcion lineal como función de activación.

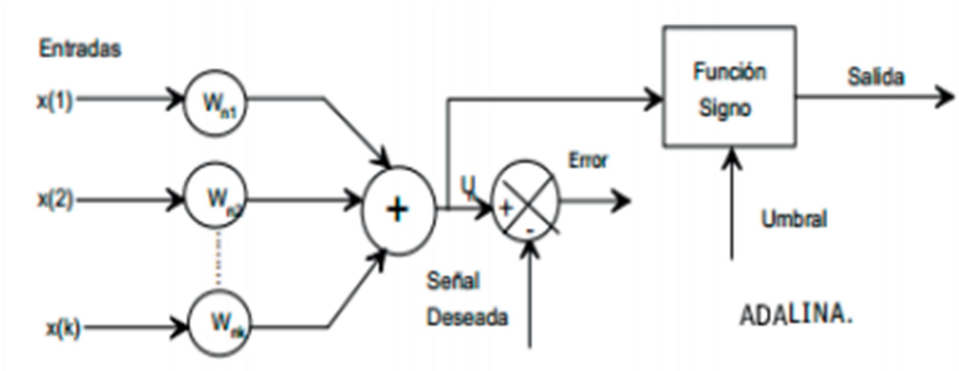


Figura 2.15. Estructura de la red Adaline (Andrade, 2013; p. 26)

Multilayer Perceptron: Esta red está formada por una capa de entrada, al menos una capa oculta y una capa de salida. La característica de este tipo de redes es que usan *Backpropagation*, como función de entrenamiento. Se trata de un modelo altamente no lineal, con tolerancia a fallos y capaz de establecer relaciones entre conjuntos de datos. Este tipo de red es el que en mayor número de aplicaciones se ha utilizado, por ello es el tipo de red que se desarrolla en el presente trabajo y se desarrollará más profundamente en el epígrafe 2.66.

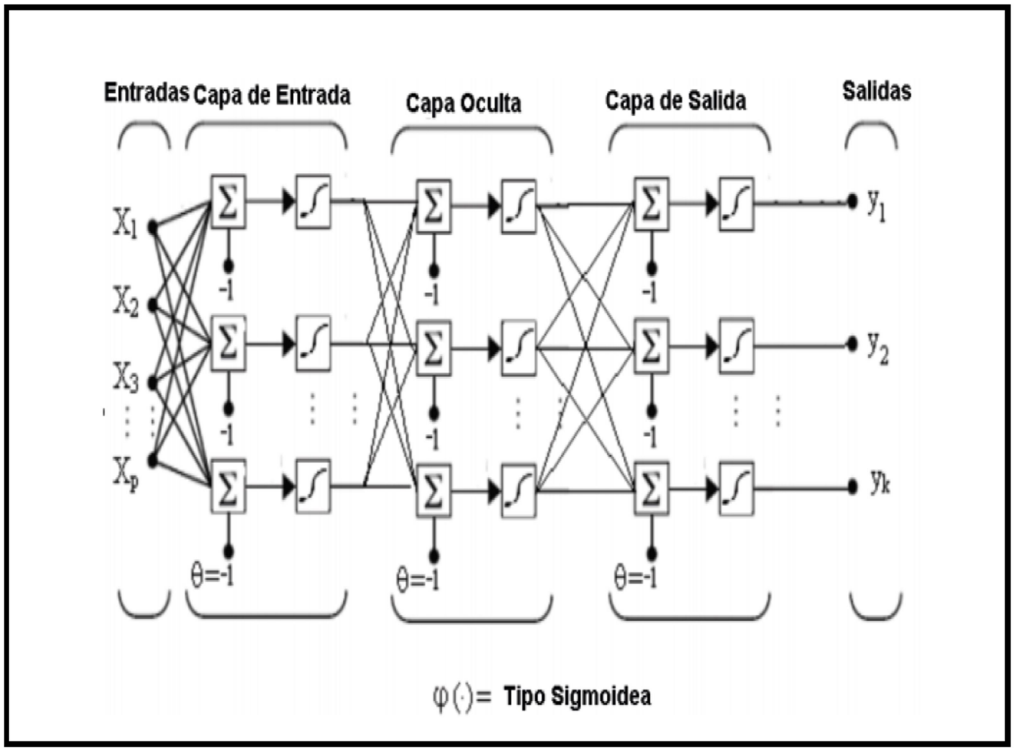


Figura 2.16. Estructura Perceptrón multicapa utilizando funciones de salida Sigmoide (Andrade, 2013; p. 27)

Máquina de Boltzmann: Esta red representa y resuelve problemas complicados y es útil en reconocimiento de patrones. Su aprendizaje y funcionamiento se basa en el "recocido simulado" (simulated annealing), donde las neuronas se conectan mediante conexiones bidireccionales.

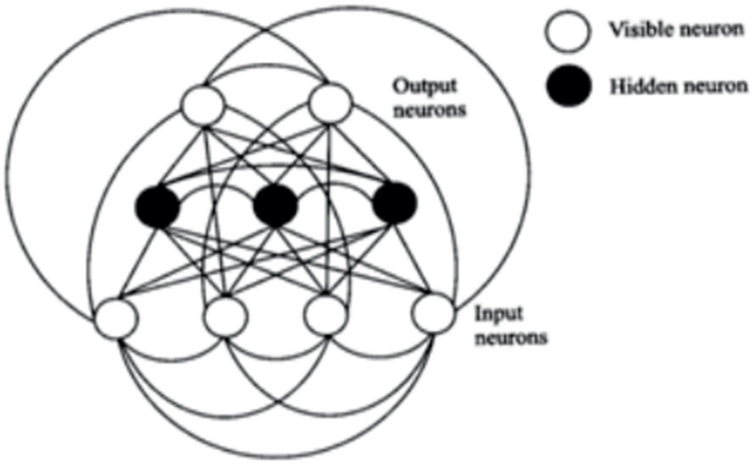


Figura 2.17. Arquitectura de la máquina de Boltzmann (Andrade, 2013; p. 31)

Máquina de Cauchy: es una versión mejorada de la máquina de Boltzmann. Aunque presenta una arquitectura y un funcionamiento idéntico, es más rápida en converger y alcanza siempre el mínimo global de energía. Sin embargo, no es muy habitual encontrarla, pues necesita mucho tiempo en la etapa de aprendizaje y funcionamiento, y además su funcionamiento es mucho menos intuitivo que el de otras redes. (Flórez y Fernández, 2008)

Red de Aprendizaje Asociativo: Este tipo de red utiliza aprendizaje no supervisado, no necesita influencias del exterior para ajustar los pesos. Este tipo de redes no reciben ninguna información de si la respuesta encontrada es o no correcta. Generan varias salidas con diferentes posibilidades de interpretación.

Redes de Kohonen: Este tipo de red fue propuesto por Teuvo Kohonen en 1982 (Kohonen, 1982), posee aprendizaje no supervisado, posee dos capas (entrada y competición) y requiere un gran número de registros de entrenamiento.

Learning Vector Quantization (LQV): Es como una red de Kohonen, pero su aprendizaje es supervisado.

Redes recurrentes: Este tipo de red tiene conexiones que van desde los nodos de salida hacia los nodos de entrada y otras conexiones arbitrarias. Las redes recurrentes más importantes son la red de Elman y la red de Hopfield (Andrade, 2013).

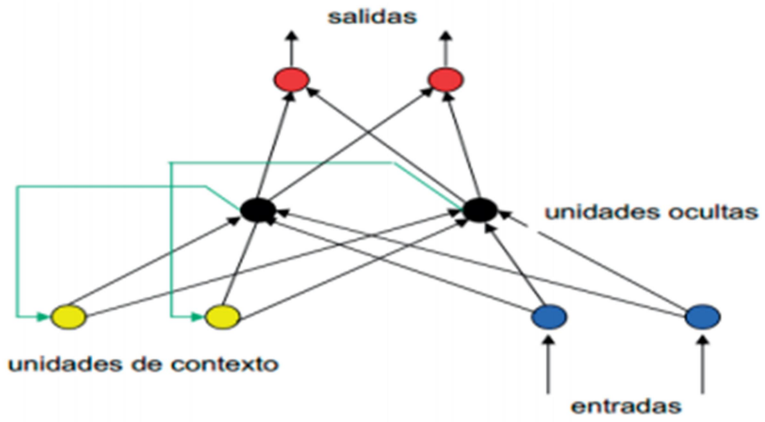


Figura 2.18. Arquitectura de la red de Elman (Andrade, 2013; p. 35)

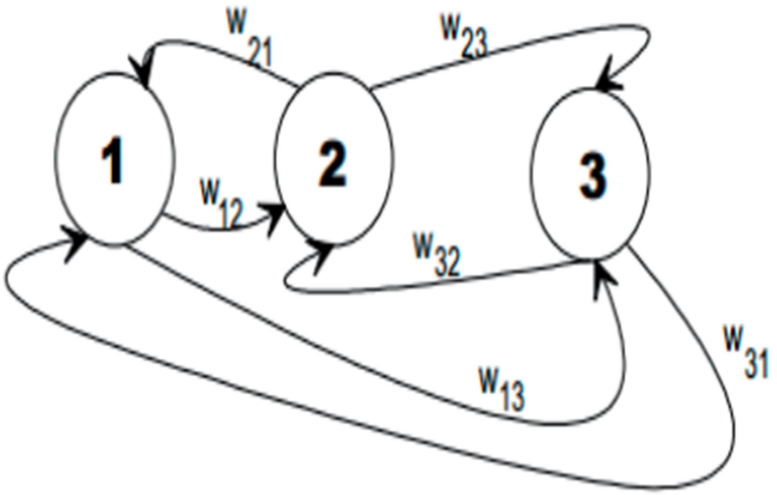


Figura 2.19. Esquema de la red de Hopfield (Andrade, 2013; p. 37)

Los algoritmos de redes neuronales ANN, han logrado buenos niveles de resultados en otros trabajos realizados dentro del grupo de estudios e investigaciones basados en “Programación por demostración” (PbD), como en el caso de Neto et al (2010), que lograron una tasa de reconocimiento de gestos y posturas de hasta el 96%, que se puede considerar una tasa plenamente válida para un algoritmo de reconocimiento.

2.6. MULTILAYER PERCEPTRONS (MLP)

Como se ha comentado en el epígrafe anterior, la característica principal del perceptrón multicapa es que utiliza *Backpropagation* como función de entrenamiento.

Su arquitectura está formada por una capa de entrada, una o varias capas ocultas, y una capa de salida. Las conexiones son siempre hacia delante, desde la entrada hacia la salida, sin realimentación de capa ni conexiones laterales.

En la función *Backpropagation* se pueden encontrar 2 etapas: la etapa de funcionamiento; y la etapa de aprendizaje.

Suponiendo una red con una capa oculta, se define w_{ji} como el peso de la conexión entre la neurona "i" de la capa de entrada con la neurona "j" de la capa oculta, y v_{kj}, como el peso de la conexión entre la neurona "j" de la capa oculta con la neurona "k" de la capa de salida.

Luego, en la **etapa de funcionamiento**, la entrada X_p de N componentes, se transmite a la capa oculta a través de los pesos de forma que la entrada de la neurona "j" de la capa oculta se calcula como:

$$net_{pj} = \sum_{i=1}^{N} w_{ji} x_{pi} + bias_j \tag{2.6}$$

Donde $bias$ es el umbral usado para añadir no linealidades entre la entrada y salida de la neurona de la capa oculta j.

El valor de salida de la capa oculta se calcula a partir del valor de entrada mediante la función de activación elegida de entre las funciones vistas en el epígrafe 2.55, denotándolo como b_{pj}.

Del mismo modo, la entrada de la neurona k de la capa de salida se calcula como:

$$net_{pk} = \sum_{j=1}^{L} v_{kj} b_{pj} + bias_k \tag{2.7}$$

Donde $bias$ es el umbral usado para añadir no linealidades entre la entrada y salida de la neurona de la capa oculta k.

El valor de salida de la capa oculta se calcula a partir del valor de entrada mediante la función de activación elegida de entre las funciones vistas en el epígrafe 2.55, denotándolo como y_{pk}.

En la **etapa de aprendizaje** se pretende conseguir el mínimo error entre la salida de la red, y la salida deseada por el programador, a lo largo de un conjunto de repeticiones conocido como grupo de entrenamiento.

La función del error que se pretende minimizar para cada patrón viene dada por:

$$E_p = \frac{1}{2} \sum_{k=1}^{M} \left(d_{pk} - y_{pk}\right)^2 \tag{2.8}$$

Siendo d_{pk} el valor deseado para la neurona de salida k del patrón p. La medida global del error se consigue como:

$$E = \sum_{p=1}^{P} E_p \tag{2.9}$$

La función *Backpropagation* fundamenta la actualización de los pesos en la técnica conocida como gradiente decreciente (Rumelhart et al., 1985; p. 4).

Al ser E_p función de los pesos de la red, su gradiente forma el vector con las derivadas parciales del error respecto a cada uno de los pesos. De esta forma el gradiente toma la dirección hacia la que aumenta más el error. Por lo que el sentido opuesto determina el decremento más rápido en el error. Para encontrar un mínimo se ajustará cada peso en la dirección resultante.

Un problema que se puede dar, al utilizar este método, es que se caiga en un mínimo local[8] (Rumelhart et al., 1985; p. 5).

La forma de ajustar los pesos de forma iterativa consiste en aplicar la regla de la cadena a la ecuación del gradiente y añadir una tasa de aprendizaje (η) y factor de momento (α) con la finalidad de acelerar el proceso de convergencia, teniendo en cuenta la dirección tomada en la iteración anterior.

La actualización se puede hacer después de haber presentado todos los patrones de entrenamiento, denominado aprendizaje por lotes o modo *batch*. O se puede actualizar tras la presentación de cada patrón de entrenamiento, aprendizaje en serie, o modo *on line*. En este caso el orden de presentación de los patrones debe ser aleatorio, pues si el orden siempre es el mismo, el entrenamiento se dirige a favor del último patrón, predominando sobre el resto (Martín del Brío y Sanz,2006).

Entonces la ecuación de actualización de los pesos es:

$$\Delta v_{kj}(n+1) = \eta\left(\sum_{p=1}^{P} \delta_{pk} b_{pj}\right) + \alpha \Delta v_{kj}(n) \tag{2.10}$$

Donde:

$$\delta_{pk} = \left(d_{pk} - y_{pk}\right) f'(net_{pk}) \tag{2.11}$$

Siendo $f'(net_{pk})$ la derivada de la función de activación de la capa de salida.

$$\Delta w_{ji}(n+1) = \eta\left(\sum_{p=1}^{P} \delta_{pj} x_{pi}\right) + \alpha \Delta w_{ji}(n) \tag{2.12}$$

Donde:

$$\delta_{pj} = f'(net_{pj}) \sum_{k=1}^{M} \delta_{pk} v_{kj} \tag{2.13}$$

[8] Mínimo, no de todo el sistema, sino de una parte de él.

Visto el funcionamiento del algoritmo de *backpropagation*, se plantea el problema de que existe un conjunto de parámetros que no se han definido. Estos parámetros son los valores de los pesos iniciales y los umbrales de error, la arquitectura de la red, el valor de la tasa de aprendizaje y del factor momento, y las funciones de activación. Pues no pueden ser conocidos a priori, y se determinan por ensayo y error. Es necesario encontrar aquella combinación de parámetros que dé buenos resultados.

Al diseñar la red neuronal por primera vez, los **pesos** deben tener algún valor, así como los umbrales. Este valor puede ser aleatorio, pero para reducir la duración del entrenamiento y evitar valores de salida de neuronas extremos, los valores más utilizados se encuentran entre (-0.5) y (+0.5) (Montaño,2002; p. 254).

Respecto a la **arquitectura de la red**, el número de capas y el número de neuronas de cada capa será el valor mínimo que presente buenos resultados (Montaño, 2002).

La **tasa de aprendizaje** controla el tamaño de la variación de los pesos en cada iteración. Una tasa de aprendizaje demasiado pequeña puede reducir enormemente la velocidad de convergencia y fomentar la posibilidad de quedar atrapado en un mínimo local. Una tasa de aprendizaje demasiado grande puede causar inestabilidades en la convergencia, no se alcanzaría nunca el mínimo. Normalmente el valor de la tasa de aprendizaje se toma ente (-0.5) y (+0.5) (Montaño, 2002).

El **factor de momento** permite filtrar las oscilaciones en la convergencia provocadas por la tasa de aprendizaje, acelera la convergencia pues si la dirección de convergencia es similar a la dirección anterior, la aproximación será mayor. Pero si es contraria, esto significa que se ha pasado un mínimo, entonces la aproximación es menor para poder alcanzarlo. Normalmente el valor del factor de momento es próximo a 1 (Montaño, 2002).

De entre las **funciones de activación** vistas en el epígrafe 2.55, el algoritmo *backpropagation* exige una función de activación continua y derivable. Para aprovechar la capacidad de las redes neuronales para aprender relaciones no lineales o complejas, es necesario utilizar funciones no lineales (Rzempoluck, 1998). Por ello, en general se utiliza la función sigmoidal.

CAPÍTULO 3. DESARROLLO EN MATLAB DE LOS MÉTODOS DE RECONOCIMIENTO DE GESTOS

Como ya se ha comentado en el Capítulo 1, el objetivo general del presente trabajo consiste en elaborar un software de reconocimiento de gestos sirviéndose del acelerómetro de un reloj inteligente.

Para ello han sido necesarios una serie de pasos que, partiendo de cero, tienen que conducir a la implementación de los diferentes objetivos específicos, con los que se ha pretendido avanzar en el proceso. Entre esos pasos, una tarea fundamental es la contrastación de los principales métodos de reconocimiento de gestos, elaborándose un análisis de la idoneidad de cada uno de ellos.

Dentro de este subproceso del trabajo, el primer paso ha sido el de **desarrollar un diccionario de gestos**, es decir, elegir los gestos que se van a utilizar para la aplicación sobre la que se va a implementar el software.

A continuación, se debe **elaborar una librería de suficiente amplitud para cada uno de los gestos del diccionario**, y para ello ha sido necesario elaborar un algoritmo que permita recopilar las series temporales de los tres ejes del acelerómetro, relativas a los gestos realizados por los usuarios.

Como se ha visto en ya en los capítulos 1 y 2, es difícil clasificar los gestos a partir de los datos del acelerómetro en bruto. Por tanto, **se ha hecho un preprocesado de los datos** y se han extraído algunas características disminuyendo así los datos con los que se trabajará.

Con las características de los datos procesados ya se puede iniciar el proceso de **implantación del método**. Y testeándolo con los datos que no se utilizan a la hora de entrenar el método se obtienen porcentajes de precisión, indicando la efectividad del método tanto para que el usuario final sea la misma persona que realizó los gestos, como para que el usuario final pueda ser cualquier otra persona.

Una vez implementado, se pueden extraer conclusiones de la idoneidad de cada uno de los métodos analizados según cual sea la finalidad del software.

Todos estos pasos, que se acaban de presentar, se detallan a continuación. Desarrollándose así el conjunto de objetivos específicos 2 y 3 planteados en el epígrafe 1.3.

En todo este proceso, una herramienta fundamental ha sido el software de cálculo MATLAB. Instrumento habitual a lo largo del grado y del master y que, de nuevo aquí, ha sido utilizado como entorno de programación para el desarrollo de los algoritmos. Se considera innecesario desarrollar una presentación de esta herramientas en un ámbito como este.

3.1. ELECCIÓN DE GESTOS

De todas las aplicaciones en las que se puede emplear el reconocimiento de gestos hoy en día, este trabajo ha decidido dirigirse a la utilización de un algoritmo de reconocimiento que permita diferenciar entre un conjunto limitado de gestos, que serán reconocidos como órdenes para la programación de un robot industrial. Más concretamente para una operación de pick and place, porque como se vio en la asignatura de Robótica Industrial del Máster en Ingeniería Industrial, la operación de pick and place se usa en todas las industrias robotizadas.

	TIPO DE APLICACIÓN				
Sector	Manipulación de piezas	Aplicación de material	Mecanizado	Soldadura	Control de calidad
Industrial de alimentación					
Ind. textil					
Ind. madera y mueble					
Ind. papel e imprenta					
Ind. química y plástica					
Ind. vidri o y cerámica					
Ind. metalúrgica					
Ind. electrónica y electricidad					
Ind. ing. control, óptica					
Ind. del automóvil					

Figura 3.1. Aplicaciones de los robots según sector industrial (Valera Fernández, 2020b; diapositiva 30)

Como se ha dicho al plantear los objetivos del trabajo, el TFM se va a centrar en desarrollar la fase de LfD dentro de la PbD. Sin entrar en la programación de los sistemas robóticos con los que luego el algoritmo se comunicaría.

Pero es importante, no perder de vista la última fase de un proceso como este, y entender bien cómo operan los sistemas robóticos de pick and place, a la hora de definir el diccionario de gestos que se tenga que diseñar.

Figura 3.2. Ejemplo de aplicación pick & place. Paletización. (Valera Fernández, 2020b; diapositiva 17)

Las órdenes que se utilizan en cualquier sistema de pick and place son (Valera Fernández, 2020b):

1. Iniciar y finalizar la comunicación con el robot para asignarle la programación.
2. Mover el extremo del robot en el espacio de trabajo en las 3 direcciones cartesianas, esto es arriba, abajo, izquierda, derecha, delante y detrás.
3. Activar el mecanismo que atrapa al objeto y desactivarlo para soltarlo.
4. Y dar la orden de continuar las operaciones programadas en modo automático.

A parir de ellas, se desagregan en este trabajo las 11 órdenes que despliegan la serie de movimientos de cualquier programación de pick and place.

Con la finalidad de que estos gestos sean fáciles de entender y replicar por parte del usuario final, se ha decidido que los gestos utilizados correspondan a la traducción de cada acción al Lenguaje de Signos de la Lengua Española según el diccionario SpreadTheSign.

Las palabras que se han traducido son:

- **Agarrar**: para dar la orden de atrapar el objeto.
- **Derecha**: para indicar al robot que debe moverse de tal forma que su extremo se desplace hacia la derecha de su posición actual, desde el punto de vista del robot.
- **Frente**: para indicar al robot que debe moverse de tal forma que su extremo se desplace hacia delante de su posición actual, alejándose de su base.
- **Hacia abajo**: para indicar al robot que debe moverse de tal forma que su extremo se desplace hacia abajo de su posición actual.
- **Hacia arriba**: para indicar al robot que debe moverse de tal forma que su extremo se desplace hacia arriba de su posición actual.
- **Hacia atrás**: para indicar al robot que debe moverse de tal forma que su extremo se desplace hacia atrás de su posición actual, acercándose a su base.
- **Iniciar**: para dar la orden de que se pretende iniciar la comunicación con el robot.
- **Izquierda**: para indicar al robot que debe moverse de tal forma que su extremo se desplace hacia la izquierda de su posición actual, desde el punto de vista del robot.
- **Liberar**: para dar la orden de liberar el objeto o dejar de agarrarlo.
- **Seguir**: para dar la orden de continuar la operación programada en modo automático.
- **Terminar**: para dar la orden de finalizar la comunicación con el robot.

En el Anexo 1 se recogen los links a los videos de cada uno de los 11 movimientos o gestos, con los que se ha entrenado el algoritmo, y que ejecutaron en series repetitivas las personas encargadas de elaborar la librería.

También se presenta en dicho Anexo 1 una captura de imagen representativa de cada uno de ellos.

3.2. IMPLEMENTACIÓN DEL ALGORITMO DE RECOGIDA DE DATOS

Para leer los gestos, se ha utilizado el reloj inteligente programable de la marca LilyGo®, que incorpora un acelerómetro triaxial BMA423. Este reloj se programa con Arduino.

Figura 3.3. Dispositivo LilyGo con Acelerómetro triaxial BMA423, utilizado en el TFM.

Hoy en día, la mayoría de pulseras y relojes inteligentes incorporan sensores de muy diverso tipo (GPS, giroscopios, altímetros, sensores de proximidad, sensores biométricos, …) y también acelerómetros.

Los acelerómetros de los relojes inteligentes miden parámetros como la aceleración (de ahí su nombre), pero también la dirección del movimiento, la intensidad o la orientación (posición) del propio dispositivo. Junto con el giroscopio permite a la pulsera (o reloj) saber qué tipo de movimiento está realizando la persona que lo lleva.

Se cuenta, por tanto, con un dispositivo con dos requisitos fundamentales: llevar incorporada la tecnología que permite el registro de información que se pediría a cualquier interfaz de entrada de información para un sistema de LfD; y tratarse de un dispositivo que en la actualidad tiene carácter de uso cotidiano en cualquier país occidental.

Con la finalidad de que en posteriores pasos sea más simple la forma de tratar los datos, se ha decidido ajustar los gestos a un número fijo de muestras.

Se programó, entonces, una primera aplicación para gestionar la operatoria de incorporación de registros a la base de datos.

La aplicación, al iniciar, mostraba por pantalla un interruptor y un botón. El botón permitía calibrar el acelerómetro y el interruptor cambiaba la pantalla e iniciaba la recolección de los datos. En la nueva pantalla, únicamente aparecería un interruptor que, al pulsarlo, finalizaba la recolección de datos del acelerómetro y los mandaba por bluetooth al dispositivo al que estaba conectado, junto con el número de muestras que había tomado del gesto.

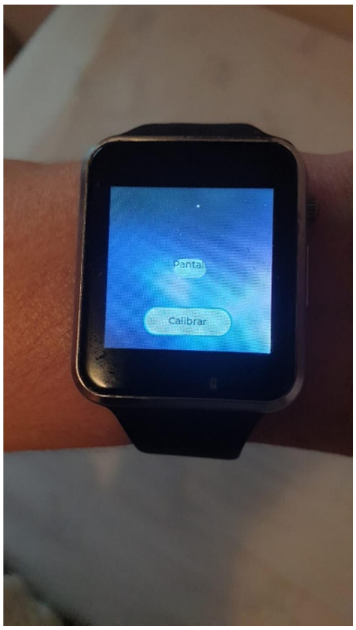


Figura 3.4. Pantalla mostrada al inicio de la aplicación.

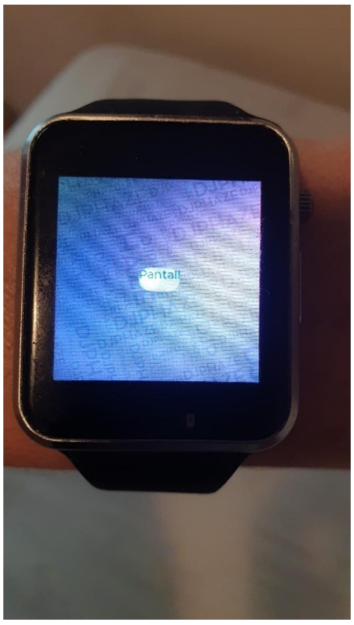


Figura 3.5. Pantalla mostrada mientras se recogen los datos.

Al pulsar el interruptor, se regresaba a la pantalla de inicio.

Esta fase, a su vez, tiene dos subfases: una dirigida a definir el número de muestras necesarias para que el algoritmo pueda leer de forma completa todos los gestos; y otra fase de elaboración del propio algoritmo de recogida de datos.

En la primera subfase, el método es registrar algunos registros (en este caso 5) de cada gesto, para así determinar el número de muestras necesarias para poder leer todos los gestos (que es el objetivo).

Realizando el experimento de esta primera subfase se llegó a la conclusión de que, con 35 muestras de un gesto, éste se podía leer de manera completa.

El pseudocódigo del programa de esta subfase es el siguiente:

1. Reconocer el dispositivo
2. Inicializar el hardware
3. Encender la pantalla
4. Asignar el sensor BMA
5. Inicializar el Bluetooth
6. Iniciar la memoria interna
7. Configurar el acelerómetro a una ratio de datos de salida de 100 Hz, un rango de medida de +- 2g, ancho de banda AVG4 y modo continuo.
8. Calibrar el acelerómetro
9. Incorporar todos los elementos a la pantalla de inicio
10. Inicia el bucle
11. Si se active la segunda pantalla empieza a leer y guardar los datos de cada componente en una matriz e incrementa una unidad un contador
12. Si no está activa la segunda pantalla y el contador tiene un valor diferente al que se ha inicializado, enviará la matriz de datos y el número de muestras por bluetooth, e inicializará el contador a su valor.
13. Fin del bucle.

Una vez fijado el número de muestras, se pasaría a la subfase de elaboración del algoritmo de recogida de datos (no se debe confundir con el de reconocimiento que actúa en una fase posterior). El algoritmo de recogida de datos es muy similar al anterior, y sigue sus pasos. Únicamente hay que indicarle que termine de leer cuando alcance las 35 muestras.

Es momento entonces de crear la base de datos (o librería). Cuantos más datos se recojan para preparar el modelo, más casos se tendrán en cuenta y mayor fiabilidad tendrá el modelo. Por ello se ha decidido que la base de datos cuente con un total de 5500 registros, lo que supone 500 registros de cada gesto. Estos 5500 registros fueron recogidos por 4 personas de diferentes complexiones y edades a lo largo de una semana.

Aunque el número de participantes en este caso se pudiese considerar escaso, lo cierto es que las diferentes fisionomías de estos permiten considerar la librería como suficientemente fiable. Además, hay que considerar que en cualquier momento el tamaño del diccionario se podría ampliar y, con posterioridad bastaría simplemente con aumentar el número de registros y de participantes.

Todos los gestos iniciaban en la misma posición, con el brazo en paralelo al cuerpo y hacia abajo. El usuario entonces calibraba el acelerómetro para reconocer esa como la

aceleración base, y una vez el dispositivo indicaba que se había calibrado, se pulsaba el interruptor y se comenzaba a hacer el gesto. Una vez finalizado el movimiento, se pulsa el interruptor y se monitorizaban a través del bluetooth los datos leídos por el acelerómetro, asegurándose de que se habían recogido todas las muestras.

Los datos recogidos para el gesto que se traduce como agarrar se representan en la siguiente figura:

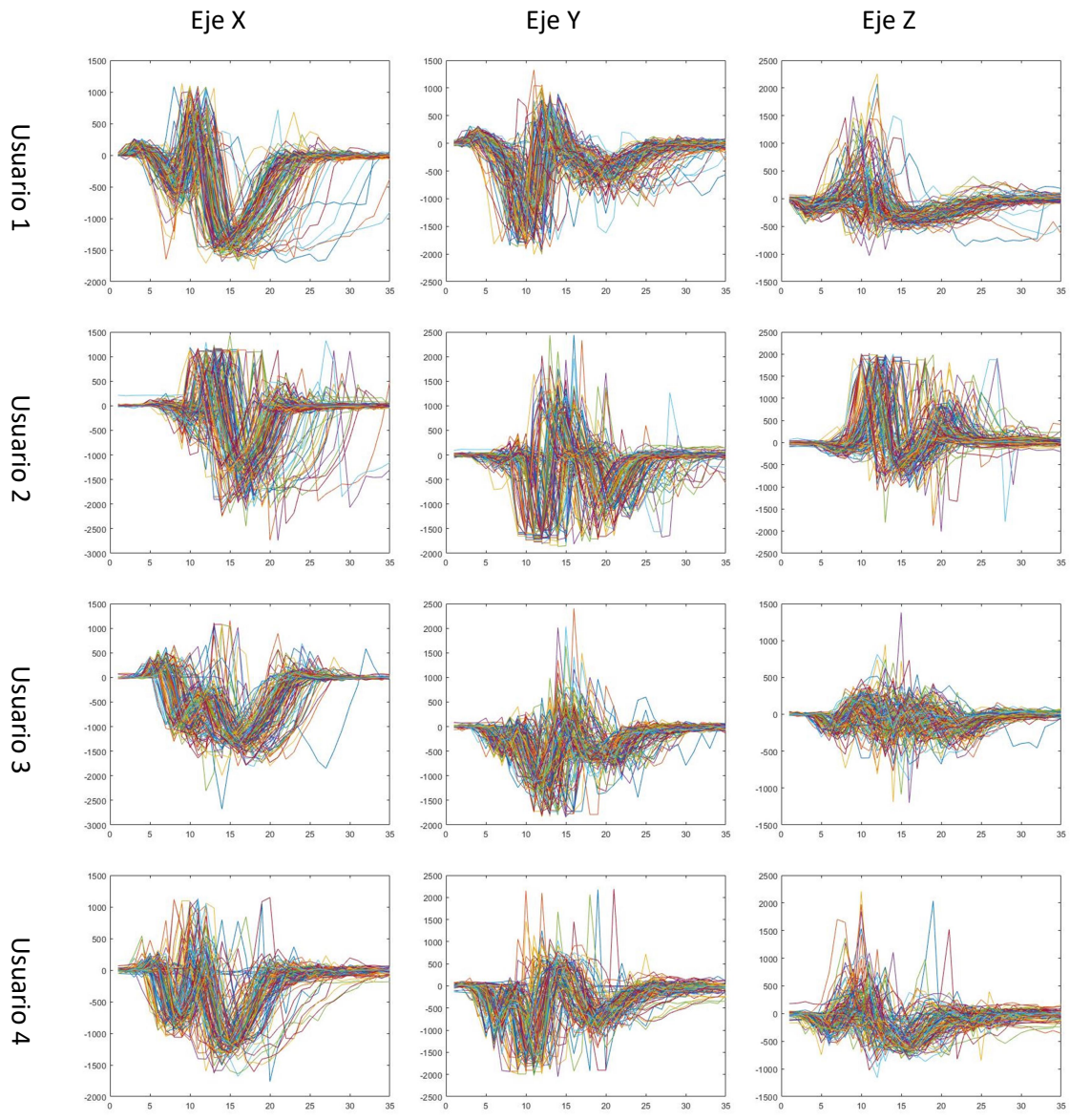


Figura 3.6. Registros recogidos del movimiento Agarrar.

En el Anexo 2 de la Memoria se dispone de la colección de datos para los 11 gestos definidos en el presente trabajo.

3.3. PREPROCESADO DE DATOS

Para tener otra visión de las series temporales que proporcionan los acelerómetros, se ha decidido calcular las derivadas de cada serie como la diferencia entre la muestra siguiente y la muestra actual (técnica del rectángulo posterior con diferencia de paso unidad), en la última muestra se utiliza la técnica del rectángulo anterior, diferencia entre la última muestra y la muestra anterior.

A la hora de realizar el gesto es muy difícil replicar las amplitudes de aceleración entre un movimiento y otro, por ello para el mismo gesto y la misma persona, las señales de los acelerómetros pueden no comprenderse en los mismos límites. Por ello, al escalar las amplitudes de las señales, se consigue hacer que las señales de un mismo gesto, ya sean del mismo usuario o de usuarios distintos, se asemejen más entre ellos.

Para ello de las 16500 series temporales que se han recogido (los 3 ejes de los 5500 registros), se ha dividido cada serie por su máximo valor absoluto. Y al multiplicar por 100, se recogen datos de entre (-100) y 100. Lo mismo se ha hecho para las derivadas de las señales.

3.4. CARACTERÍSTICAS UTILIZADAS PARA CLASIFICAR

Cada registro dispone de 35 muestras para cada uno de los tres ejes, lo que supone 105 datos en total. Sin embargo, esta cantidad se puede reducir, recolectando las principales características que permiten identificar la serie temporal tridimensional.

En este trabajo se ha decidido dividir el registro en 4 zonas: la primera recogería desde la primera a la décima muestra; la segunda, de la undécima a la décimo quinta; la tercera, de la décimo sexta a la vigésimo quinta; y la cuarta, de la vigésimo sexta a la trigésimo quinta. La segunda zona recoge un menor número de muestras porque en esa zona hay mayor variabilidad, y al estrechar la zona es más fácil recoger esas desviaciones.

Esta misma división se hace para los registros de las aceleraciones en los tres ejes, y también para sus derivadas, lo que resulta en 24 zonas en cada movimiento, con lo que, sacando la media y la varianza de cada zona, se obtiene un total de 48 características que permiten definir un movimiento.

Tanto el preprocesado de datos, como la obtención de características, se realizan con el programa MATLAB, pues no requieren cálculos muy complejos y permite ir guardando los resultados en ficheros *.mat*, que se emplearan como entrada de los algoritmos de clasificación.

Ahora bien, ¿es adecuado haber elegido estas divisiones para obtener las características, o deberían elegirse otras características para este conjunto de datos? Para ello se hace un **análisis de componentes principales de las características de los 5500 registros**.

Como se comentó en el epígrafe 2.1, los pasos a seguir para aplicar el método PCA de Análisis de Componentes Principales son:

1. Seleccionar el conjunto de datos: en este caso es la matriz de características de dimensión 48 x 5500
2. Extracción de la media: en Matlab esta operación se hace con el comando *zscore*
3. Descomposición en valores singulares: Para ello se emplea el comando *svd*
4. Seleccionar los componentes y evaluar las características: En este caso, como se muestra en la figura 3.7. donde aparecen los valores propios del conjunto de datos, la ganancia mínima es 14.21. Este valor se considera un valor alto, denotando que en todas las características hay variabilidad suficiente como para poder distinguir un gesto de otro.

```
221.7425
164.0756
145.4942
131.9467
117.3109
111.3077
101.5250
 94.5936
 88.2475
 86.5472
 84.0538
 78.9792
 73.8745
 71.1741
 67.9852
 66.2447
 62.1989
 59.0449
 57.9727
 55.9164
 55.4016
 53.5520
 52.6207
 50.5428
 48.7172
 48.0061
 46.1588
 45.0419
 43.9172
 42.6973
 42.1494
 39.6599
 39.3297
 38.7350
 37.0854
 36.2931
 34.2651
 33.2991
 33.0321
 31.9285
 30.2627
 30.1017
 28.5603
 25.9660
 22.3888
 17.6015
 16.1891
 14.2153
```

Figura 3.7. Listado de valores propios de la matriz de características (48 datos).

3.5. ALGORITMOS DE CLASIFICACIÓN

El algoritmo de clasificación es un programa MATLAB en el que se estudian los diferentes métodos de clasificación que han sido analizados.

El algoritmo de recogida de datos tenía como propósito realizar la librería de registros. El **algoritmo de clasificación** utiliza los principios de cada método para establecer los parámetros que permitirán clasificar los gestos, sin clasificarlos aún. El algoritmo de clasificación trabaja ya con los datos de la librería después de haber sido preprocesados. Y se centra en la fase, que se denomina de entrenamiento, de cualquiera de las técnicas de clasificación existentes (bien sea las que trabajan con plantillas o con modelos).

3.5.1. Dynamic Time Warping

Uno de los métodos que se han estudiado es el Dynamic Time Warping (DTW) o, por su traducción al castellano, la deformación de tiempo dinámica. Este algoritmo se usa para deformar series temporales con el objetivo de que se parezcan más, se puede interpretar como un escalado en el tiempo.

A primera vista parece que este método, más que para clasificar, se utiliza para preprocesar las señales. Sin embargo, tal y como se plantea en Liu et al. (2009), se pueden utilizar los fundamentos de este método con el fin de encontrar similitudes entre señales y así clasificar entre las señales que más se parecen.

De esta forma, conociendo un conjunto de gestos, que en términos generales se presentaban ya en el epígrafe 3.1, y de los que se ha tomado un banco de datos tal y como se ha comentado en el epígrafe 3.2, se forman las plantillas y se asigna el nuevo gesto desconocido a aquel gesto con el que presenta mayor similitud.

Esta similitud se determina por la distancia entre las señales, pues cuanto más se parezcan entre ellas, menor es la deformación temporal. Por tanto, dos series temporales son más similares si la deformación temporal es menor.

Implementando el método en MATLAB, se hacen dos agrupaciones diferenciadas que recogen los gestos: la plantilla y los gestos de prueba, siendo mucho mayor la colección de registros en los gestos de prueba.

El comando que calcula en Matlab el DTW es

```
[dist,ix,iy] = dtw(x,y)
```

Donde x será un registro de la plantilla, e “y” el gesto desconocido. Las salidas de esta función son:

- dist, que es el coste de ajuste que se vio en el epígrafe **¡Error! No se encuentra el origen de la referencia.** ;
- ix e iy contienen la secuencia monótona creciente de los emparejamientos entre las muestras de x e y, que conforman el camino óptimo.

Así el nuevo gesto desconocido realizado por el programador usuario, se comparará con cada uno de los registros que conforman la plantilla y este gesto se reconocerá como aquel con el que el coste de ajuste sea menor.

```
load("DatosEscalados.mat");
Plantilla=[];
Resto=[];
for i=1:11
    Plantilla=[Plantilla DatosEscY(:,(i-1)*1500+1:(i-1)*1500+120)];
    Resto=[Resto DatosEscY(:,(i-1)*1500+121:(i-1)*1500+375)];
end
for i=1:440
    for j=1:935
        [d,x,y]=dtw(Plantilla(:,(i-1)*3+1:(i-1)*3+3),Resto(:,(j-1)*3+1:(j-1)*3+3));
        dis(i,j)=d;
    end
end
```

Figura 3.8. Algoritmo DTW implementado en MATLAB.

3.5.2. Dynamic Time Warping con Propagación de Afinidad y Detección por Compresión

Con la finalidad de obtener mejores resultados se propone incorporar la propagación de afinidad y la detección por compresión al método del Dynamic Time Warping.

La propagación de afinidad trata de encontrar aquel registro que mejor define a un gesto, y lo propone como ejemplo.

A la hora de reconocer gestos, se compara el nuevo gesto desconocido con el conjunto de ejemplos encontrados por el algoritmo de propagación de afinidad. Utilizando la técnica de Dynamic Time Warping.

Por tanto, se encuentran 2 etapas distintas. Por un lado, la etapa de entrenamiento, donde se calcula la matriz de similitud y se aplica el algoritmo de propagación de afinidad. Y, posteriormente, una etapa de prueba o testeo, donde se comparan la plantilla de gestos con los ejemplos obtenidos.

La detección por compresión consiste en reconstruir las señales y se emplea en la fase de prueba independiente del usuario.

Para este modelo se han diferenciado dos algoritmos de entrenamiento, uno dependiente del usuario y otro independiente del usuario. La diferencia entre ellos radica, como ya se ha dicho, en que en uno de ellos el usuario que crea los registros de entrenamiento y el que luego hará la programación PbD, usando el algoritmo, es la misma persona (dependiente del usuario). Y en el otro caso, la persona o personas que generan la librería de gestos pueden ser distintas a las que luego actúan como usuarios programadores con el algoritmo de reconocimiento (independiente del usuario).

Se ha calculado la matriz de similitud S mediante DTW tal y como se indicaba en el epígrafe **¡Error! No se encuentra el origen de la referencia.**. También se han calculado las preferencias a priori como la mediana de las similitudes de entrada (Akl y Valaee,2010), esto es la mediana de la última fila de la matriz S.

El algoritmo de AP se puede calcular mediante la función *apcluster* desarrollada por Wang (2021).

La función tiene la siguiente estructura:

[idx,netsim,i,unconverged,dpsim,expref]=apcluster(s,p)

- Idx es un vector con el índice de los registros que el algoritmo identifica como ejemplos,

- netsim refleja la aptitud del modelo (net similarity),
- i es el número de iteraciones que han sido necesarias para encontrar la convergencia,
- unconverged, si su valor es 1 indica que el modelo no ha convergido,
- dpsim refleja la aptitud de un dato a su ejemplo,
- expref indica las preferencias a los ejemplos seleccionados.

Las entradas a la función son:

- S, que es la matriz de similitud,
- y p que es el vector de preferencias.

Una vez aplicado el modelo de AP, la fase de entrenamiento ha finalizado, pues se han encontrado los ejemplos con los que se comparará el gesto desconocido para, del mismo modo que se hace en Dynamic Time Warping, realizar el reconocimiento del gesto.

```
clear all;close all;clc;
load("DatosEscalados.mat");
M=125;
Entrenamiento=[];
for i=1:11
    Entrenamiento=[Entrenamiento DatosEscY(:,(i-1)*1500+1:(i-1)*1500+(M*3))];
end

NM=size(Entrenamiento,2)/3;

for n=1:NM
    for o=1:NM
        [dx,x,y]=dtw(Entrenamiento(:,(n-1)*3+1),Entrenamiento(:,(o-1)*3+1));
        [dy,x,y]=dtw(Entrenamiento(:,(n-1)*3+2),Entrenamiento(:,(o-1)*3+2));
        [dz,x,y]=dtw(Entrenamiento(:,(n-1)*3+3),Entrenamiento(:,(o-1)*3+3));
        S(n,o)=(-1)*(dx^2+dy^2+dz^2);
    end
end
Entrenamiento=Entrenamiento';
S=S';
p=median(S(:,NM));
N=size(S,1);
if length(p)==1
    for i=1:N
        S(i,i)=p;
    end
else
    for i=1:N
        S(i,i)=p(i);
    end
end
[idx,netsim,i,unconverged,dpsim,expref]=apcluster(S,p);
idx2=unique(idx);
```

Figura 3.9. Algoritmo de entrenamiento de DTW con AP.

3.5.3. Redes Neuronales. Multilayer Perceptron.

Dentro del grupo de los modelos de Redes Neuronales Artificiales (ANN por sus siglas en inglés, Artificial Neural Networks), de los que se ha hablado en el epígrafe 2.5., el algoritmo entrena la red neuronal que se utilizará para reconocer los gestos a partir de sus características.

Para ello se han empleado los comandos de MATLAB: network y train. El comando network define la arquitectura de la red siguiendo la siguiente estructura:

Net = network (numInputs, numLayers, biasConnect, inputConnect, layerConnect, outputConnect)

Donde:

- numInputs es el número de conjuntos de datos de entrada que tendrá la red. En este caso se utiliza un único conjunto de entrada, que será la matriz de características de 48 x 5500 datos.
- numLayers es el número de capas ocultas más la capa de salida, este valor se ha ido modificando hasta encontrar aquel valor que consigue dar mejores resultados.
- biasConnect son los valores que se suman a las entradas de las capas, y se usan para añadir no linealidades entre entrada y salida de las capas. Como los gestos suponen funciones no lineales complejas, se ha decidido poner bias en todas las capas.
- inputConnect indica en qué capa se conectan las entradas, pudiendo no ser necesario conectar todos los conjuntos de entrada a la primera capa, y así crear redes con jerarquías complejas. En este caso se va a utilizar una red simple, el conjunto de entrada se conectará a la primera capa.
- layerConnect indica como están conectadas las capas, pudiendo determinar realimentaciones en capas, conectar la salida de una capa con la entrada de una capa anterior o con una siguiente. En este caso, la red, es una red simple y por tanto la salida de una capa se conectará con la entrada de la siguiente hasta llegar a la capa de salida.
- outputConnect indica qué capa está conectada con la salida, en este caso la cuarta capa es la capa de salida.

Con lo comentado, el comando network recibe las siguientes entradas:

```
net = network(1, 4, [1; 1; 1; 1], [1;0;0; 0], [0 0 0 0; 1 0 0 0; 0 1 0 0; 0 0 1 0 ], [0
0 0 1])
```

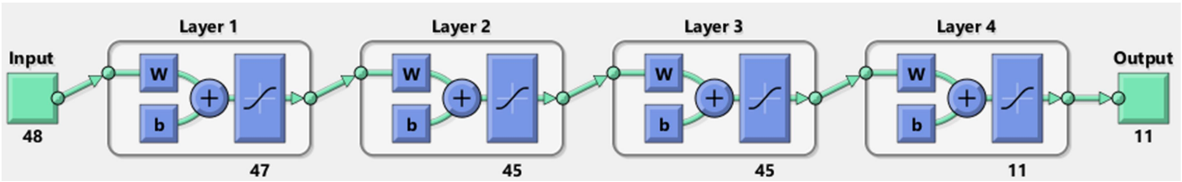


Figura 3.10. Arquitectura de la red MLP.

A continuación se definen las funciones de la red, se le indica que adapte la red con reglas de aprendizaje de pesos y bias net.adaptFcn = 'adaptwb'.

Y se define la función que divide los datos de entrada en: datos de entrenamiento, validación y test net.divideFcn = 'dividerand'. Todo ello divide los datos de entrada de forma aleatoria en datos de entrenamiento, validación y test.

El buen funcionamiento de la red se mide con la media cuadrática normalizada del error net.performFcn = 'mse'.

Y la función que se utiliza en la actualización de los valores de pesos y bias durante el entrenamiento de la red es Resilient Backpropagation. Se eligió esta función, y no otra que utilizara backpropagation, porque al hacer ensayos con las diferentes funciones esta era de la más rápida a la hora de presentar buenos resultados. Por ejemplo, con Levenberg-Marquardt Backpropagation se tardaban unos 6 minutos en entrenar la red, y los resultados de la matriz de confusión eran peores que en el caso de Resilient Backpropagation, que tardaba 13 segundos en entrenar el modelo y obtenía un 97% de precisión.

En este proceso también es necesario definir las capas. El número de capas se definió cuando se inicializo la red, y para cada capa se define el número de neuronas que la componen. Este número debe ser el menor número necesario que consiga buenos resultados.

El número de neuronas de la capa de salida debe coincidir con el número de filas de la matriz Test, que contiene los valores deseados de salida. En este caso son 11 neuronas en la capa de salida porque se pretende que la red saque un valor 1 para el gesto que la red identifica en el movimiento de entrada y saque un valor 0 en el resto de los gestos.

En cada capa se define la función de inicialización de los pesos net.layers{1}.initFcn = 'initnw', y la función de activación de la capa, en este caso la tangente hiperbólica net.layers{2}.transferFcn = 'tansig'.

Una vez se ha definido completamente la red, se entrena con el comando train. El comando train tiene la siguiente estructura: net = train(net,x, t).

Donde net es la red que se ha definido, x es la matriz de entrada, aquí se introduce la matriz de características obtenida en el epígrafe 3.4, de dimensiones 48 (las 48 características del movimiento) x 5500 (los 5500 movimientos almacenados). La matriz t, es la que previamente se ha mencionado como matriz Test. Ella contiene los valores que se desea que se obtengan al aplicar la red sobre la entrada, tiene dimensiones 11x5500,

11 para indicar con un 1 la posición que identifica el gesto que se sabe que es y 5500 movimientos de entrada.

Al ejecutar el comando train se observa la siguiente pantalla:

Figura 3.11. Herramienta de entrenamiento de redes neuronales.

De la pantalla se puede extraer mucha información. Se puede ver cómo está dividida en 4 bloques.

- En el primero, “Neural Network”, se observa la arquitectura de la red, tal y como se ha definido.
- El segundo corresponde a los “Algorithms”, en este bloque se indican las funciones que se están utilizando y que ya han sido descritas.
- El tercer bloque muestra el progreso, el número de iteraciones que lleva, el tiempo transcurrido, el error que tiene, …
- Finalmente, el cuarto bloque, muestra las gráficas con los resultados del entrenamiento, de ellas la más interesante es “Confusion”. Esta función muestra la matriz de confusión del modelo, que permitirá la evaluación y su comparación con otros métodos.

El entrenamiento finaliza bien porque ha alcanzado las 1000 iteraciones, bien porque ha conseguido 6 controles de verificación o bien porque el usuario ha decidido parar la simulación con el botón Stop Training.

```
load BaseDeDatos.mat

net = network(1, 4, [1; 1; 1; 1], [1;0;0; 0], [0 0 0 0; 1 0 0 0; 0 1 0 0; 0 0 1 0 ], [0 0 0 1]);

net.adaptFcn = 'adaptwb';
net.divideFcn = 'dividerand'; %Se divide de forma aleatoria

net.performFcn = 'mse';
net.trainFcn = 'trainrp';

net.plotFcns = {'plotperform', 'plottrainstate', 'ploterrhist', 'plotconfusion', 'plotroc'};

%set Layer1
net.layers{1}.name = 'Layer 1';
net.layers{1}.dimensions = 47;
net.layers{1}.initFcn = 'initnw';
net.layers{1}.transferFcn = 'tansig';

%set Layer2
net.layers{2}.name = 'Layer 2';
net.layers{2}.dimensions = 45;
net.layers{2}.initFcn = 'initnw';
net.layers{2}.transferFcn = 'tansig';

%set Layer3
net.layers{3}.name = 'Layer 3';
net.layers{3}.dimensions = 45;
net.layers{3}.initFcn = 'initnw';
net.layers{3}.transferFcn = 'tansig';

%set Layer4
net.layers{4}.name = 'Layer 4';
net.layers{4}.dimensions = 11;
net.layers{4}.initFcn = 'initnw';
net.layers{4}.transferFcn = 'tansig';



x=[Entrenamiento Validacion Test];
t=[kron(eye(11),ones(1,380)) kron(eye(11),ones(1,20)) kron(eye(11),ones(1,100))];

net = train(net,x, t); %training

y = net(x); %prediction

view(net);
```

Figura 3.12. Algoritmo MLP implementado en MATLAB.

3.6. ALGORITMO DE RECONOCIMIENTO DE GESTOS

Una vez se han analizado los métodos de clasificación, que ya habían sido anticipados en la revisión teórica del Capítulo 2, y desarrollándolos de manera práctica en MATLAB, tal y como queda plasmado en los anteriores epígrafes de este Capítulo 3, los métodos han sido también analizados mediante matrices de confusión, tal y como se expondrá en el capítulo 4.

De los tres métodos que se han trabajado en esta Memoria, se ha implantado la red neuronal del multilayer perceptron en el reloj inteligente programable para construir en base a él el algoritmo de reconocimiento de gestos.

Es ahora, después del trabajo aportado por el algoritmo de recogida de datos (creando la librería de gestos) y del algoritmo de clasificación (utilizando los principios de cada método para establecer los parámetros que permitirán clasificar los gestos, sin clasificarlos aún), cuando el algoritmo de reconocimiento culmina el proceso con la clasificación final que permite reconocer a cada gesto.

El algoritmo de reconocimiento, por tanto, aprovecha e integra las tareas de los otros dos.

Precisamente en el Capítulo 4, los resultados de reconocimiento de los 3 métodos de clasificación son los que justifican la elección de uno de ellos. El algoritmo que se instalará en el dispositivo es el que integre al método que mejores tasas de reconocimiento haya logrado en las pruebas de validación.

Además, la tarea de implementar el algoritmo en otro dispositivo y en un lenguaje de programación diferente, se facilita con el uso del método MLP. Pues como la red ya está entrenada, simplemente es necesario pasar las matrices de los pesos y los umbrales.

El pseudocódigo del algoritmo de reconocimiento es el siguiente:

1. Reconocer el dispositivo
2. Inicializar el hardware
3. Encender la pantalla
4. Asignar el sensor BMA
5. Inicializar el Bluetooth
6. Iniciar la memoria interna
7. Configurar el acelerómetro a una ratio de datos de salida de 100 Hz, un rango de medida de +- 2g, ancho de banda AVG4 y modo continuo.
8. Calibrar el acelerómetro
9. Incorporar todos los elementos a la pantalla de inicio
10. Inicia el bucle

11. Si se active la segunda pantalla empieza a leer y guardar los datos de cada componente en una matriz e incrementa una unidad un contador
12. Si no está activa la segunda pantalla y el contador tiene un valor diferente al que se ha inicializado, enviará la matriz de datos y el número de muestras por bluetooth, e inicializará el contador a su valor.
13. Si se activa la tercera pantalla, ejecutará la red neuronal a los datos que acaba de recoger, realizando el cálculo de características previamente.
14. Una vez reconocido el gesto, muestra por pantalla el nombre del gesto correspondiente.
15. Fin del bucle.

Con el algoritmo ya en el dispositivo, la aplicación, al iniciar, muestra por pantalla un interruptor y un botón (ver secuencia completa en la figura 3.13).

Figura 3.13. Secuencia de pantallas que muestra el software utilizado por el dispositivo.

El botón permite calibrar el acelerómetro y el interruptor cambia la pantalla e inicia la recolección de los datos. En la nueva pantalla, únicamente aparecería un interruptor que, al pulsarlo, finalizaba la recolección de datos del acelerómetro y los mandaba por bluetooth al dispositivo al que estaba conectado, junto con el número de muestras que había tomado del gesto.

Al pulsar el interruptor, se regresaba a la pantalla de inicio, pero ahora había otro interruptor que al pulsarlo llevaba a una pantalla diferente. Esta nueva opción se habilita ahora, ya que no se puede reconocer el gesto sin haber recogido los datos.

En esta pantalla se aplica el algoritmo de reconocimiento del gesto realizado anteriormente. Esta pantalla también dispone únicamente de un interruptor que al pulsarlo vuelve a la

pantalla de inicio, pero también aparece en esa pantalla el nombre del gesto que ha reconocido.

CAPÍTULO 4. ANÁLISIS DE RESULTADOS Y COMPARACIÓN DE MÉTODOS

Aunque en el capítulo 3 ya se han adelantado algunas consideraciones que anticipan resultados del estudio experimental del TFM, se va a ver y conocer el detalle de dichos resultados del trabajo de campo en este capítulo 4.

Los resultados, y su análisis, servirán para comprobar si se han conseguido los objetivos específicos 2 y 3, claves en el objetivo general que se planteó el TFM desde su diseño inicial (elaborar un algoritmo de reconocimiento de gestos sirviéndose del acelerómetro de un reloj inteligente).

Como se puede recordar los objetivos específicos 2 y 3 se plantean comparar las tasas de reconocimiento de los principales métodos utilizados por los trabajos anteriores en el caso de nuestro propio estudio. Y, una vez identificado el método que mejores tasas de reconocimiento ofrece, instalar en el dispositivo el algoritmo que integra al método que haya tenido esos mejores resultados.

Pero para finalmente validarlo se ha realizado, no sólo la comparación de los resultados de los algoritmos de clasificación (matrices teóricas de confusión) de cada método de clasificación, sino que se ha validado el propio algoritmo de reconocimiento finalmente elegido para instalar en el reloj, con una prueba de experimental. Dicha prueba ha consistido en el manejo del dispositivo por un usuario distinto a los que generaron la librería de gestos, ejecutando los gestos del diccionario de gestos. Comprobándose si los gestos realizados quedaran cubiertos por el algoritmo, aplicando también una matriz de confusión, pero en este caso para comparar los gestos reales realizados por el usuario y los resultados que sacaría el algoritmo instalado.

En los tres primeros epígrafes de este capítulo, se van a presentar los resultados obtenidos en cada uno de los diferentes métodos de clasificación de gestos, identificados tras la revisión teórica del capítulo 2, y se evaluará su nivel de eficacia a través de las tasas de reconocimiento que la biblioteca de gestos generada obtendrá en cada uno de dichos métodos (DTW, DTW mejorado y MLP).

De paso, se podrá analizar si los resultados obtenidos corroboran o no las conclusiones de otros trabajos previos que han utilizado dichos métodos.

Pero, el propósito principal es llegar a una conclusión propia sobre cuál de los tres métodos contrastados tiene la tasa de reconocimiento más alta y, por tanto, es el más adecuado.

Las matrices de confusión darán la respuesta. Las mismas comparan los resultados deseados con los resultados obtenidos indicando la cantidad y/o porcentaje de gestos que han sido reconocidos como “i” cuando se esperaba que se reconocieran como “j”.

De esta manera se reconoce el desempeño de los modelos, en la diagonal principal se encuentran los datos correctamente clasificados y el resto de las celdas recogerán los errores de clasificación. (Ducloux et al, 2017)

Al haberse hecho el estudio de eficacia de los tres métodos, que servirá para compararlos entre sí (epígrafe 4 del capítulo), el método que resulte más eficaz ya habrá sido desarrollado y estudiado en gran medida, con lo que no es necesario mucho más trabajo para dar respuesta al objetivo específico 3. No obstante, el epígrafe 5 servirá para someter a una última validación al que será el algoritmo de reconocimiento finalmente instalado en el dispositivo.

4.1. DYNAMIC TIME WARPING

Para extraer una valoración del método visto en el epígrafe 3.5.1 se ha implementado el algoritmo que prueba su fiabilidad con los registros que no forman la plantilla de registros.

Se han hecho dos pruebas diferentes, una con gestos de un único usuario (dependiente del usuario) y otra con los gestos de todos los usuarios (independiente del usuario).

Los resultados obtenidos se comparan con los resultados que obtuvieron Liu et al (2009), cuando desarrollaron su algoritmo uWave.

Ellos consiguieron la matriz de confusión para el caso dependiente del usuario con observaciones en diferentes días, con una tasa de reconocimiento de un 93,5% en términos de promedio.

	92.1	0.1	2.4	1.9	0.1	2.9	0.6	0.1
	1.6	91.6	1.3	1.1	0.7	0.4	2.7	0.6
	0.5	0	95.9	1.2	0.7	1.7	0	0
	0.3	0	1.6	96.2	0.7	1.1	0	0.1
	0.3	0	1.5	0.6	97.0	0.5	0	0.1
	2.4	0	2.4	2.3	1.0	91.7	0.1	0
	3.4	1.9	2.6	1.7	0.4	0.7	89.2	0
	1.1	0.6	1.7	0.9	0.8	0.7	0	94.2

Figura 4.1 Matriz de Confusión para DTW en el caso dependiente del usuario (Liu et al., 2009; p. 6) Donde las columnas son los gestos reconocidos y las filas las identidades reales de entrada.

Si se compara con la matriz de confusión de este trabajo, los resultados para el caso dependiente del usuario son muy parecidos (93,15%).

	Agarrar	Derecha	Frente	Hacia Abajo	Hacia Arriba	Hacia Atrás	Iniciar	Izquierda	Liberar	Seguir	Terminar
Agarrar	100%	15%	0%	0%	11%	0%	9%	0%	0%	7%	0%
Derecha	0%	84%	0%	0%	1%	0%	0%	0%	0%	0%	0%
Frente	0%	0%	100%	0%	0%	5%	0%	1%	0%	0%	0%
Hacia Abajo	0%	0%	0%	100%	1%	0%	0%	0%	0%	0%	0%
Hacia Arriba	0%	0%	0%	0%	79%	0%	0%	0%	0%	0%	0%
Hacia Atrás	0%	0%	0%	0%	0%	85%	0%	0%	0%	0%	0%
Iniciar	0%	0%	0%	0%	0%	0%	88%	0%	1%	0%	0%
Izquierda	0%	0%	0%	0%	0%	8%	0%	99%	0%	0%	0%
Liberar	0%	0%	0%	0%	0%	0%	2%	0%	99%	1%	0%
Seguir	0%	1%	0%	0%	1%	0%	0%	0%	0%	92%	0%
Terminar	0%	0%	0%	0%	7%	2%	0%	0%	0%	0%	100%

Figura 4.2 Matriz de Confusión DTW para el caso dependiente del usuario (Elaboración propia)

La matriz debería en el eje diagonal obtener porcentajes del 100% en todos los gestos del diccionario. Si fuese así, el ajuste sería perfecto y la tasa de reconocimiento del 100%. Cuando esto no ocurre es porque hay un gesto que se ha interpretado como otro. Por ejemplo en un 15% de los registros "ir a la derecha" ha sido confundido con agarrar.

El diseño de esta matriz de confusiones, se hace por el programador comparando lo que en el diseño experimental se definió como que debería salir (gesto que se pretendía realizar), con lo que el algoritmo de clasificación reconoce.

Sin embargo, el DTW no da tan buenos resultados cuando se trata de una situación de usuario independiente. En el caso de Liu et al (2009) la tasa de reconocimiento era tan baja que decidieron no incluirla en su publicación.

En el caso de este trabajo, los resultados de la matriz de confusión para usuario independiente han sido:

	Agarrar	Derecha	Frente	Hacia Abajo	Hacia Arriba	Hacia Atrás	Iniciar	Izquierda	Liberar	Seguir	Terminar
Agarrar	76%	15%	1%	0%	28%	0%	20%	3%	10%	24%	2%
Derecha	4%	65%	0%	0%	8%	2%	13%	3%	40%	16%	0%
Frente	0%	0%	92%	0%	1%	9%	0%	34%	0%	0%	6%
Hacia Abajo	1%	1%	0%	83%	1%	0%	0%	3%	0%	2%	0%
Hacia Arriba	3%	0%	0%	0%	48%	4%	0%	2%	4%	0%	0%
Hacia Atrás	0%	0%	0%	0%	0%	57%	0%	4%	0%	0%	0%
Iniciar	2%	0%	4%	0%	3%	0%	50%	4%	1%	3%	5%
Izquierda	0%	0%	1%	12%	0%	13%	0%	38%	0%	0%	2%
Liberar	2%	3%	0%	0%	2%	0%	13%	1%	31%	8%	2%
Seguir	11%	15%	0%	4%	4%	6%	5%	6%	13%	46%	0%
Terminar	1%	0%	1%	0%	6%	9%	0%	3%	0%	0%	83%

Figura 4.3 Matriz de Confusión DTW para el caso independiente del usuario (Elaboración propia)

Resultando una tasa promedio de reconocimiento del 60,72%.

La forma de hacer esta matriz es la de comparar los resultados que deberían haber salido, según la planificación experimental del programador, en cada gesto, con los datos que salen del algoritmo para la totalidad de personas que han participado en la creación de la librería. La diferencia entre dependiente del usuario e independiente del usuario es que en el primer caso los gestos analizados corresponden a cada usuario por separado, y en el segundo se incluyen la totalidad de los gestos de una manera indiscriminada.

4.2. DYNAMIC TIME WARPING. AFFINITY PROPAGATION AND COMPRESSIVE SENSING

Con la finalidad de mejorar el rendimiento del DTW en el caso de más de un usuario y así poder ser utilizado en más aplicaciones, se ha combinado con los algoritmos AP (propagación de afinidad) y el algoritmo CS (detección de compresión).

De esta forma se incrementan las precisiones tanto del modelo dependiente del usuario como del modelo independiente del usuario, llegando a precisiones destacadas para considerarlo como un método muy eficiente.

En esta ocasión los resultados conseguidos se han comparado con los del trabajo de Akl y Valaee (2010), que para el caso dependiente del usuario consiguen una precisión promedio de 99,79 %, mientras que en el caso independiente del usuario consigue una precisión del 96,89%.

Este trabajo consigue la siguiente matriz de confusión para el caso dependiente del usuario

	Agarrar	Derecha	Frente	Hacia Abajo	Hacia Arriba	Hacia Atrás	Iniciar	Izquierda	Liberar	Seguir	Terminar
Agarrar	99%	2%	0%	0%	8%	0%	0%	0%	0%	1%	1%
Derecha	0%	94%	0%	0%	0%	0%	0%	0%	0%	4%	0%
Frente	0%	0%	98%	0%	0%	5%	0%	2%	0%	0%	0%
Hacia Abajo	0%	0%	0%	98%	0%	0%	0%	0%	0%	0%	0%
Hacia Arriba	0%	0%	0%	0%	86%	0%	0%	0%	0%	0%	0%
Hacia Atrás	0%	0%	2%	2%	0%	95%	0%	1%	0%	0%	0%
Iniciar	0%	0%	0%	0%	6%	0%	100%	0%	0%	0%	0%
Izquierda	0%	0%	0%	0%	0%	0%	0%	97%	0%	0%	0%
Liberar	1%	2%	0%	0%	0%	0%	0%	0%	100%	0%	0%
Seguir	0%	2%	0%	0%	0%	0%	0%	0%	0%	95%	1%
Terminar	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	98%

Figura 4.4. Matriz de Confusión DTW+AP para el caso dependiente del usuario (Elaboración propia)

Consiguiendo una tasa de reconocimiento promedio de 96,5%, algo más alta que la previa tan sólo con DTW.

Pero lo más relevante es el análisis para el caso independiente del usuario, que es el que no se tiene bien resuelto con la aplicación en solitario del DTW.

Tal y como se vio en el capítulo 3, la mejora del DTW para esta situación requiere que además del método de propagación de afinidad (AP), se aplique el de detección de compresión (CS):

	Agarrar	Derecha	Frente	Hacia Abajo	Hacia Arriba	Hacia Atrás	Iniciar	Izquierda	Liberar	Seguir	Terminar
Agarrar	98%	1%	0%	1%	0%	1%	0%	0%	0%	0%	0%
Derecha	2%	91%	0%	0%	0%	3%	0%	0%	0%	3%	0%
Frente	4%	1%	89%	1%	0%	5%	0%	0%	0%	0%	0%
Hacia Abajo	7%	4%	1%	79%	0%	7%	0%	0%	2%	0%	0%
Hacia Arriba	6%	2%	0%	1%	89%	2%	0%	0%	0%	0%	0%
Hacia Atrás	1%	0%	0%	0%	0%	98%	0%	1%	0%	0%	0%
Iniciar	4%	0%	0%	1%	0%	4%	88%	0%	1%	0%	0%
Izquierda	2%	2%	0%	1%	0%	6%	0%	86%	2%	0%	0%
Liberar	4%	0%	0%	0%	1%	1%	0%	0%	93%	0%	0%
Seguir	2%	1%	0%	0%	0%	1%	0%	0%	0%	96%	0%
Terminar	5%	0%	0%	1%	0%	3%	0%	0%	0%	0%	90%

Figura 4.5. Matriz de Confusión DTW+AP+CS para el caso independiente del usuario (Elaboración propia)

Consiguiendo una tasa de reconocimiento promedio de 90,6%. Que se podría considerar ya satisfactoria, y más si se compara con el 60,72% que conseguía el método DTW en solitario.

4.3. MULTILAYER PERCEPTRON

A diferencia de los anteriores métodos basados en plantillas, este método se basa en el entrenamiento de una red para que las conexiones entre los elementos básicos (neuronas) tengan aquellos valores que le permitan clasificar un gesto desconocido.

Este método ya fue utilizado por Ducloux et al. (2017) consiguiendo la siguiente matriz de confusión para el caso independiente del usuario

Salidas		estimada							
real		**0.982**	0	0	0.009	0	0.009	0	0
		0	**0.973**	0	0	0	0.009	0.009	0.009
		0	0	**0.982**	0	0	0.009	0.009	0
		0	0	0.009	**0.991**	0	0	0	0
		0	0	0	0.009	**0.991**	0	0	0
		0.009	0	0	0.009	0.009	**0.973**	0	0
		0	0	0	0	0	0	**1**	0
		0	0	0	0	0	0	0	**1**

Figura 4.6. Matriz de Confusión MLP para el caso independiente del usuario (Ducloux et al, 2017; p. 5)

La tasa de reconocimiento promedio de su estudio es del 98,65%, superior a las de los estudios de Liu et al. (2009) para DTW, y de Akl y Valaee (2010) para DTW+AP+CS.

Los resultados en el caso de este trabajo consiguen tasas de reconocimiento similares.

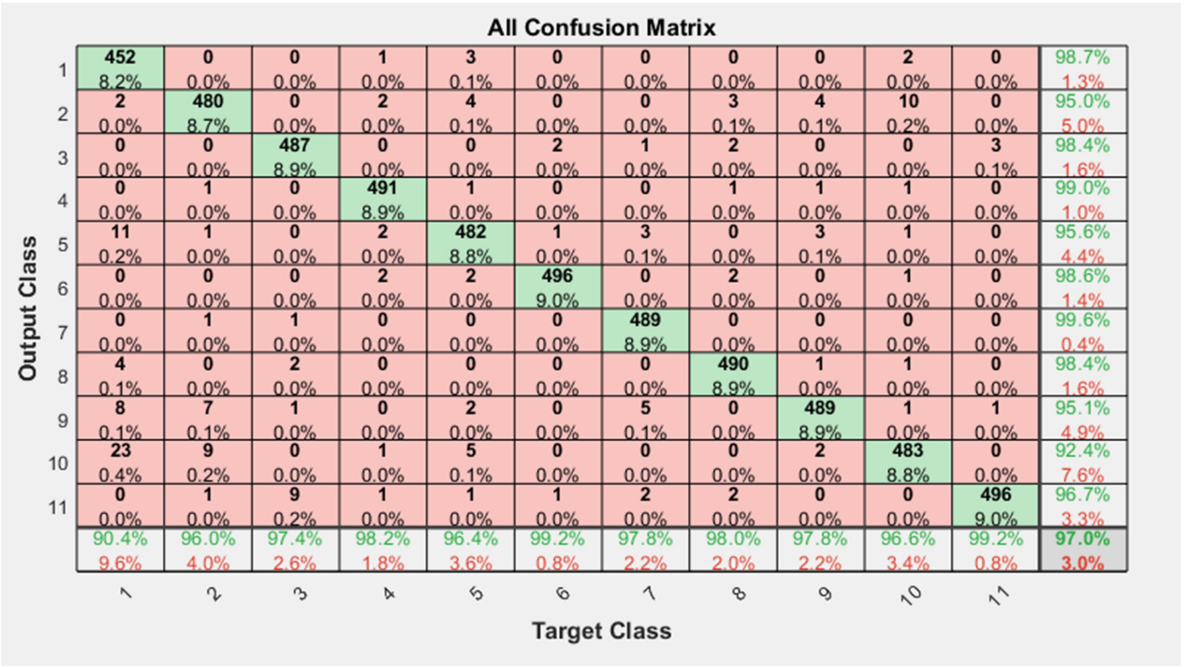


Figura 4.7. Matriz de Confusión MLP para el caso independiente del usuario (Elaboración propia)

La tasa de reconocimiento promedio se sitúa en el 97%, similar a la de Ducloux et al (2017), y como ocurre en la bibliografía, también en el caso del TFM la tasa del método MLP mejora a las otras alternativas analizadas.

4.4. COMPARACIÓN DE MÉTODOS

Analizando los resultados obtenidos no cabe duda de que el método que mejores resultados da es el Multilayer Perceptron.

Trabajando con el escenario de situación independiente del usuario, que es el que se corresponde con el propósito de diseñar un algoritmo de reconocimiento para la programación PbD posterior llevada a cabo por usuarios distintos a los que participaron en la elaboración de la biblioteca de gestos, los resultados son:

- DTW: tasa promedio de reconocimiento: 60,72%.
- DTW+AP+CS: tasa promedio de reconocimiento: 90,6%.
- MLP: tasa promedio de reconocimiento: 97%.

Además, a la hora de implementar el método es el que menos espacio ocupa en la memoria del dispositivo, pues al no utilizar plantillas, el número de datos se reduce considerablemente, aunque sigue siendo alto debido a que tiene muchas capas ocultas y neuronas en cada capa.

Sin embargo, no hay que menospreciar el atractivo que presenta el método basado en Dynamic Time Warping, pues su mal funcionamiento en diferentes usuarios lo compensa con su capacidad de personalización, siendo capaz de modificar la plantilla completa fácilmente para cada usuario que necesite el dispositivo.

Elegido el método MLP, para ser la base del algoritmo de reconocimiento que se instalará en el dispositivo del reloj inteligente. El algoritmo finalmente instalado tiene una dimensión difícil de reproducir en el texto de esta Memoria. En la figura 4.8. se reproduce la parte de este, con los pesos y los umbrales obtenidos del algoritmo de clasificación del método MLP, para formar la red neuroral.

```
for(int a=0; a<47;a++){
  float sum=0.0;
  for(int b=0; b<48;b++){
    sum+=w[a][b]*dato[b];
  }
  incapa1[a][1]=sum+bias1[a-1][1];
}
for (int i=0;i<47;i++){
  outcapa1[i][1]=(exp(incapa1[i][1])-exp(-incapa1[i][1]))/(exp(incapa1[i][1])+exp(-incapa1[i][1]));
}
for(int a=0; a<45;a++){
  float suma=0.0;
  for(int b=0; b<47;b++){
    suma+=v1[a][b]*outcapa1[b][1];
  }
  incapa2[a][1]=suma+bias2[a-1][1];
}
for (int i=0;i<45;i++){
  outcapa2[i][1]=(exp(incapa2[i][1])-exp(-incapa2[i][1]))/(exp(incapa2[i][1])+exp(-incapa2[i][1]));
}
for(int a=0; a<45;a++){
  float sumb=0.0;
  for(int b=0; b<45;b++){
    sumb+=v2[a][b]*outcapa2[b][1];
  }
  incapa3[a][1]=sumb+bias3[a-1][1];
}
for (int i=0;i<45;i++){
  outcapa3[i][1]=(exp(incapa3[i][1])-exp(-incapa3[i][1]))/(exp(incapa3[i][1])+exp(-incapa3[i][1]));
}
for(int a=0; a<11;a++){
  float sumc=0.0;
  for(int b=0; b<45;b++){
    sumc+=v3[a][b]*outcapa3[b][1];
  }
  incapa4[a][1]=sumc+bias4[a-1][1];
}
for (int i=0;i<11;i++){
  outcapa4[i][1]=(exp(incapa4[i][1])-exp(-incapa4[i][1]))/(exp(incapa4[i][1])+exp(-incapa4[i][1]));
}
float aux=0;
for(int i=0;i<11;i++){
  if(outcapa4[i][1]>aux){
    aux=outcapa4[i][1];
    gesto=i;
  }
}
  return gesto;
```

Figura 4.8. Parte del algoritmo de reconocimiento instalado en el dispositivo (Elaboración propia)

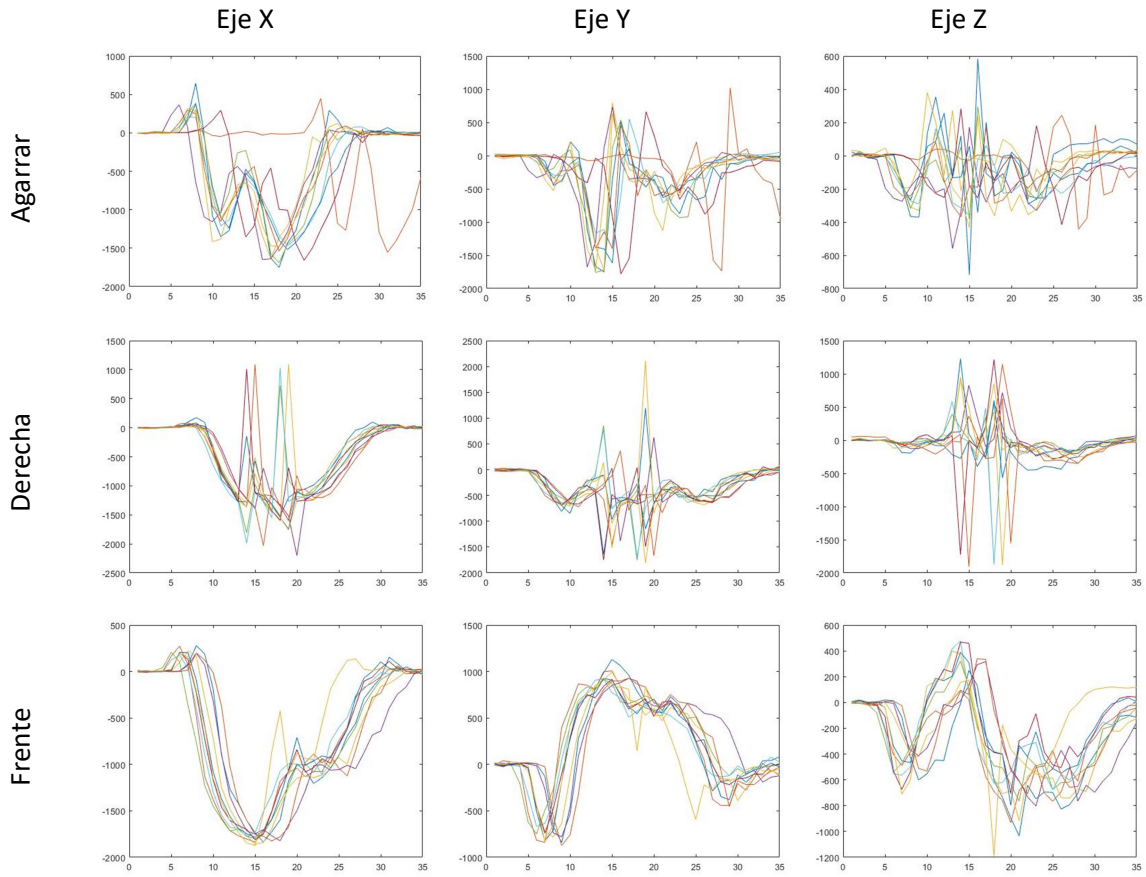
Esta red neuronal será la que, con nuevos datos de entrada en cada ocasión en las que un usuario quiera hacer uso del algoritmo, se ejecuta llevando a cabo el reconocimiento de la secuencia de gestos incluidos en el diccionario de gestos.

4.5. VALIDACIÓN DEL ALGORITMO DE RECONOCIMIENTO

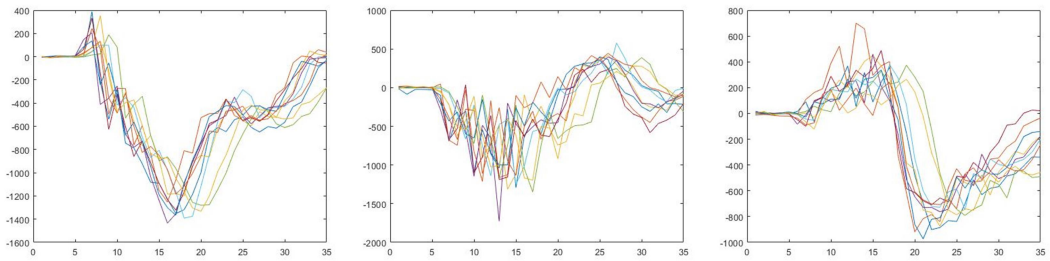
Precisamente, y como un procedimiento de validación de refuerzo, además de la validación por comparación de los resultados de los algoritmos de clasificación (matrices teóricas de confusión) de cada método, se ha realizado una prueba de experimental. Dicha prueba ha consistido en el manejo del dispositivo por un usuario distinto a los que generaron la librería de gestos, ejecutando los gestos del diccionario de gestos.

Para ello se ha contactado con una persona diferente a las que estuvieron trabajando en la elaboración de la librería de gestos, para que hiciera una serie de pruebas con el reloj, contrastando si su programación (la del algoritmo incorporado) daba unos resultados eficaces para usuario-final.

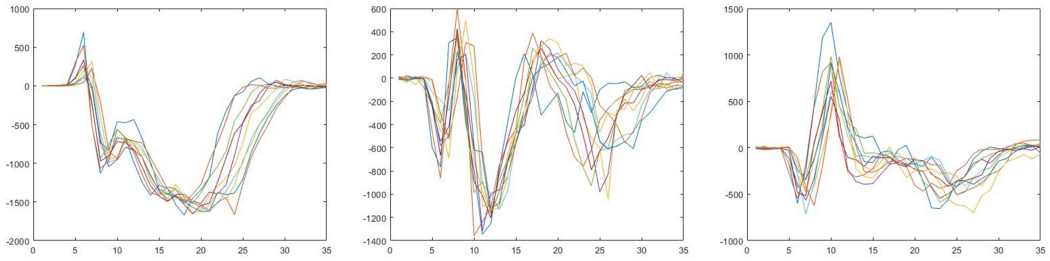
El experimento consistió en repetir cada gesto del diccionario 10 veces. Los resultados obtenidos para cada uno de los gestos son los siguientes:



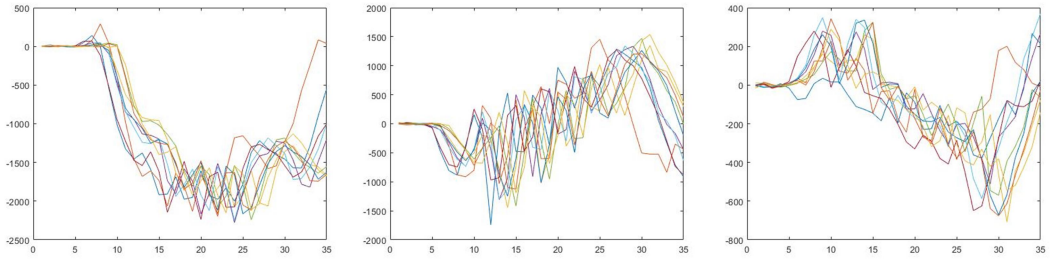
Hacia Abajo



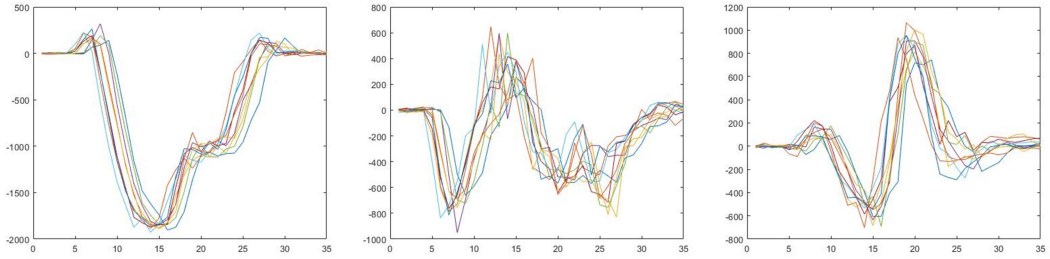
Hacia Arriba



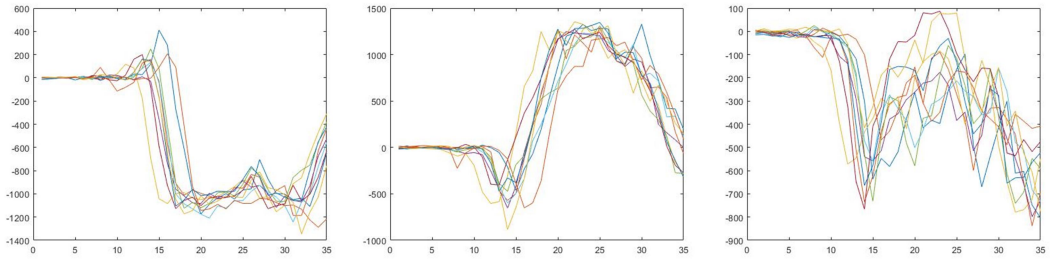
Hacia Atrás



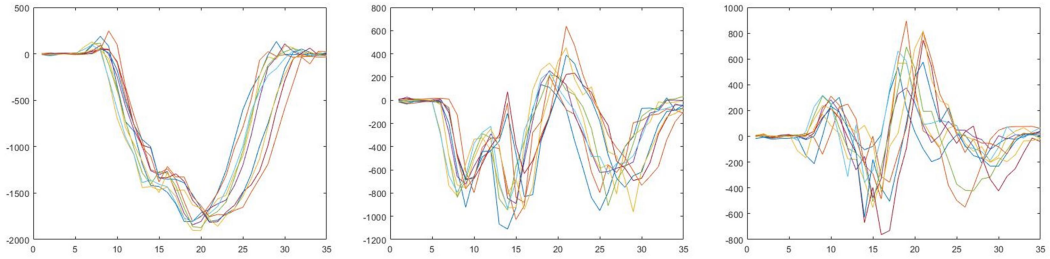
Iniciar



Izquierda



Liberar



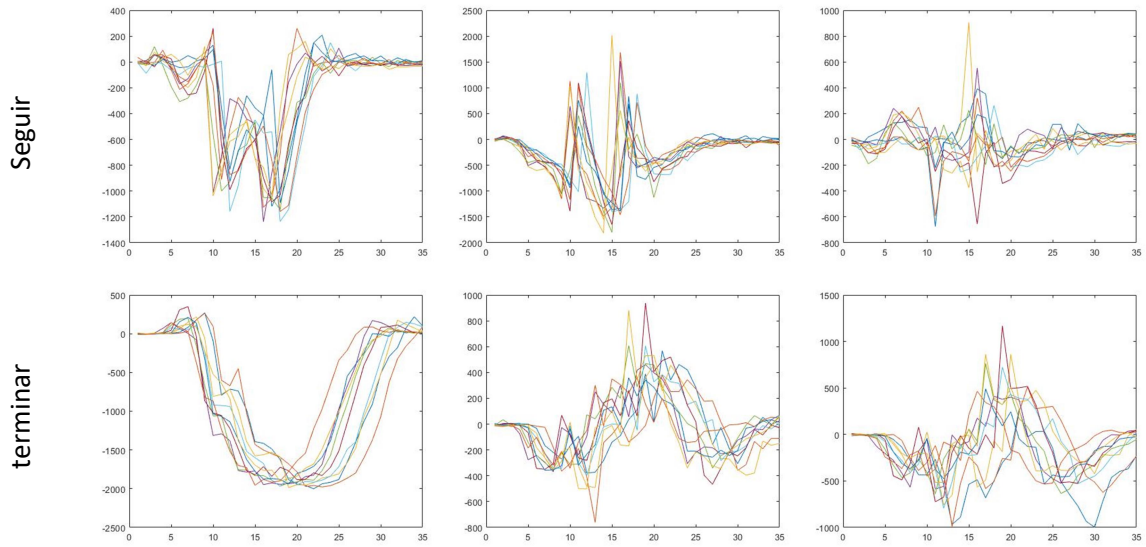


Figura 4.9. Colección de datos para los 11 gestos tras la tarea de validación (Elaboración propia)

Plasmando los resultados en una matriz de confusión, se obtiene una tasa de reconocimiento del 93,63 %.

El gesto seguir es el que ha tenido una tasa de reconocimiento lago más baja, al ser confundido en 2 de las 10 ocasiones con el gesto derecha, y en una con el gesto agarrar. No obstante, una tasa del 93,63 para un número tan reducido de pruebas (tan sólo 10 por gesto) se puede considerar un muy buen resultado.

	Agarrar	Derecha	Frente	Hacia Abajo	Hacia Arriba	Hacia Atrás	Iniciar	Izquierda	Liberar	Seguir	Terminar
Agarrar	90%	0%	0%	0%	10%	0%	0%	0%	0%	10%	0%
Derecha	0%	90%	0%	0%	0%	0%	0%	0%	10%	20%	0%
Frente	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%	0%
Hacia Abajo	0%	0%	0%	100%	0%	0%	0%	0%	0%	0%	0%
Hacia Arriba	0%	0%	0%	0%	90%	0%	0%	0%	0%	0%	0%
Hacia Atrás	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%
Iniciar	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%	0%
Izquierda	0%	0%	0%	0%	0%	0%	0%	100%	0%	0%	0%
Liberar	0%	0%	0%	0%	0%	0%	0%	0%	90%	0%	0%
Seguir	10%	10%	0%	0%	0%	0%	0%	0%	0%	70%	0%
Terminar	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	100%

Figura 4.9. Matriz de Confusión generada para la tarea de validación del algoritmo(Elaboración propia)

Con estos resultados se puede concluir que el algoritmo se considera plenamente validado, y con ello alcanzado el objetivo general del TFM.

CAPÍTULO 5. CONCLUSIONES

Si se recuperasen ahora la lista de los objetivos específicos planteados en el epígrafe 1.3. se podría analizar el grado de logro de estos, según los resultados finales del trabajo.

El trabajo de revisión del marco teórico realizado en el capítulo 2 ayuda a comprender que, dentro de los métodos para el desarrollo de algoritmos de reconocimiento de gestos que han utilizado las investigaciones sobre PbD, el método DTW (Dynamic Time Warping), y sus versiones mejoradas, de DTW+AP (Affinity Propagation) y DTW+AP+CS (Compressive Sensing), ofrecen unos buenos resultados cuando lo que se plantea es trabajar con plantillas de gestos.

Y que, en el caso de trabajar con modelos, la metodología Multilayer Perceptron (MLP) dentro del grupo de métodos que trabajan con Redes Neuronales Artificiales (ANN Artificial Neuronal Network) es capaz de desarrollar algoritmos con tasas de reconocimiento aún superiores. Y, además, con más rapidez y ocupando menos espacio en el dispositivo, al tener una forma de trabajar que reduce considerablemente el número de datos con los que se trabaja.

Los resultados obtenidos por el trabajo de campo realizado en este TFM coinciden en un grado muy elevado con lo que se habían observado en la bibliografía. Y también, en el estudio realizado, los resultados obtenidos, tanto de la aplicación del algoritmo mejorado DTW+AP+CS (con una tasa promedio de reconocimiento: 90,6%), como del MLP (con una tasa promedio de reconocimiento: 97%), se reproduce lo ya observado por trabajos anteriores (Akl y Valaee, 2010; Ducloux et al., 2017).

A partir de ahí, se pueden resaltar las siguientes conclusiones:

1. **El estudio realizado** para crear y analizar un algoritmo de reconocimiento de gestos a partir de los datos generados por el acelerómetro de un reloj de pulsera, sobre un diccionario de 11 gestos y una librería con una base de datos de 5.500 registros, **obtiene tasas de reconocimiento de niveles semejantes a las obtenidas por trabajo que han sido publicados en revistas especializadas y en congresos internacionales**. Con lo que **el TFM ha sido capaz de implementar con éxito un proceso de trabajo como el que se marcó como objetivo**.
2. Que, al contrastar las metodologías para el desarrollo de algoritmos de reconocimiento más utilizadas en la bibliografía especializada, **los algoritmos elaborados con** modelos de redes neuronales artificiales (ANN), y en concreto el que utiliza el sistema **Multilayer Perceptron (MLP)** es el que **obtiene la tasa de reconocimiento más alta (97%)**.
3. Que en el caso de los algoritmos que trabajan con plantillas, **el método DTW funciona bien cuando la persona cuyos gestos han servido para la creación de la biblioteca de gestos, y la que luego usaría el algoritmo para hacer programación de usuario, son la misma persona** (93,15% de tasa de reconocimiento). Y en el caso de usuario

independiente (personas distintas) el modelo mejorado de **DTW+AP+CS también ofrece unos niveles satisfactorios**, además de recoger la ventaja de su capacidad de personalización, siendo capaz de modificar la plantilla completa fácilmente para cada usuario que necesite el dispositivo.

4. **El algoritmo de reconocimiento construido mediante el método MLP es eficaz para hacerlo funcionar con un reloj inteligente con acelerómetro**, en las primeras fases de un proceso de PbD encaminado a manejar un sistema robótico dedicado a tareas de pick and place.

La validez analizada por este trabajo se ha realizado de manera teórica a través de la comparación de las matrices de confusión, pero también se ha sometido a validación de usuario final con un experimento ad-hoc.

Se considera por ello, que el algoritmo planteado es válido para los propósitos buscados que se circunscriben a la elaboración del algoritmo de reconocimiento, y por tanto **el trabajo ha conseguido, tanto sus objetivos específicos como su objetivo general**.

Limitaciones del trabajo y desarrollos futuros

El trabajo presentado, permite otras posibilidades para un desarrollo futuro, como la contrastación del algoritmo en un desempeño real construyendo un programa de manejo de un sistema robótico concreto (finalizando así todas las fases de la Programación por Demostración).

El trabajo se ha centrado en validar el software (algoritmo de reconocimiento) instalable en la interfaz (reloj inteligente). Validado el algoritmo, su programación se podría trasladar a una aplicación descargable, de manera que los usuarios tuviesen instalado el algoritmo en su reloj, con una descarga sencilla y de bajo coste.

Pero en un proceso de comunicación entre dispositivos, faltaría por desarrollar el software a instalar en los sistemas robóticos por parte de las empresas fabricantes de estos. Software que, tomando como emisor el dispositivo con el algoritmo elaborado en este TFM, lo interpretase y lo convirtiese en órdenes de movimiento para el programa de manipulación del robot.

Hay, por tanto, una segunda fase de trabajo por abordar que consiste en cerrar el proceso de comunicación (interfaz – sistema robótico) que sería objeto de un trabajo posterior.

Esta segunda fase, tendrá una complejidad especial. Y es que el módulo de programación que se hiciese debería disponer de un lenguaje suficientemente universal para integrarlo en

cualquier sistema de programación de cualquier tipo de fabricante de sistemas robóticos. Esta hipótesis es difícil de creer, por lo que sería más probable que las empresas de fabricación robótica necesitasen un diseño ad-hoc para la integración del sistema que se elaborase.

La diversidad de sistemas de programación existentes (Offline, simuladores, Middleware,…) e incluso el hecho de que cada marca desarrolla su propio software, como ABB con RAPID,… (Valera Fernández, 2020a; diapositivas tema 4), justifican el anterior razonamiento.

Para que se pueda pasar a esa segunda fase, es muy importante que se haga primero el algoritmo de reconocimiento y que se demuestre eficaz para utilizarse en procesos de PbD con un dispositivo como el elegido. Si eso no existe, o no es eficaz, no hay capacidad de llegar a las empresas de fabricación de sistemas robóticos y ofrecerles la tecnología de interpretación del algoritmo de reconocimiento para que la instalen en sus equipos.

Por tanto, a falta de desarrollar un nuevo proyecto encargado de elaborar el algoritmo (módulo de programación) para que las empresas fabricantes de sistemas robóticos lo incorporen. Y así, puedan ofrecérselo a sus potenciales clientes, como una ventaja para que empiecen a trabajar con sistemas de Programación por Demostración (PbD). Los resultados de este TFM son condición necesaria para pasar a ese segundo proyecto.

Respecto a la generación de ingresos por parte del algoritmo de reconocimiento, el sistema de convertir el algoritmo en una *app* descargable a través de las diferentes plataformas existentes en el mercado (las más conocidas y utilizadas son Google Play para dispositivos Android, y App Store para Apple), se tendrían dos posibilidades: convertirla en una app de pago, en base a su naturaleza profesional; o convertirla en una app gratuita que se financie por la publicidad.

Esta segunda opción, tiene como inconveniente que por el tipo de servicio que ofrece, los ingresos por visualizaciones de anuncios publicitarios es casi imposible que compensen la inversión. Hay que tener en cuenta que sistemas como Google Admob retribuyen 3 euros por cada 1.000 visualizaciones. Sólo si la estrategia fuese generar los ingresos sólo vía el pago de las empresas de robótica, para incorporar esta tecnología, podría tener sentido una app gratuita, con un propósito de promoción de ventas más que de negocio.

La opción de app de pago podría ser más interesante. El precio medio de las apps de pago está entre los 5 y 10 euros.

Según el presupuesto elaborado (20.437,16 €), y con una app con un precio de descarga de 10 €, la inversión para estar amortizada necesitaría con 2.044 descargas. Esto requeriría una empresa con suficiente estructura comercial, tanto para llegar a las empresas fabricantes de sistemas robóticos, como para después comercializar la app entre usuarios finales.

Por todo ello, se ha diseñado el presupuesto con la hipótesis de que el trabajo se realiza como encargo de un tercero (empresa) a la que se le factura como servicio técnico de ingeniería por el trabajo realizado.

CAPÍTULO 6. BIBLIOGRAFÍA

Akl, A. y Valaee, S. (2010): "Accelerometer-based gesture recognition via dynamic–time warping, affinity propagation, & compressive sensing" *Proceedings of the 2010 IEEE International Conference on Acustics Speech and Signal Processing*. Dallas. 14-19 March. Pp. 2270-2273.

Aleotti, J.; Skoglund, A. y Duckett, T. (2004): "Position Teaching of a Robot Arm by Demonstration with a Wearable Input Device." Paper presented at the *International Conference on Intelligent Manipulation and Grasping (IMG04)*. Genoa (Italy). July 1-2, 2004.

Andrade, C. (2013): "Estudio de los principales tipos de redes neuronales y las herramientas para su aplicación". Universidad Politécnica Salesiana. Sede de Cuenca. Pdh Thesis. Febrero.

Berndt, D.J. y Clifford, J. (1994): "Using dynamic time warping to find patterns in time series". *AAAI Workshop on Knowledge Discovery in Databases.* Pp. 359–370.

Candès, E.J. (2006): "Compressive sampling". *Proceedings of the International Congress of Mathematicians*. Madrid. European Mathematical Society. Pp. 1433-1452.

CCOO Industria (2017): *La Digitalización y la Industria 4.0. Impacto industrial y laboral*. Secretaría de Estrategias Industriales de Comisiones Obreras. Madrid. Septiembre de 2017.

Chowdhury, D. y Chattopadhyay, M. (2020): "Development of a Low-Power Microcontroller-Based Wrist-Worn Device with Resource-Constrained Activity Detection Algorithm". *IEEE Transactions on Instrumentation and Measurement*. Vol. 69. Nº 10. October. Pp. 7522-7529.

Dillmann, R. (2004): "Teaching and learning of robot tasks via observation of human performance", *Robotics and Autonomous Systems*. Vol. 47 Nº 2/3. Pp. 109-116.

Donoho, D.L. (2006): "Compressed Sensing". *IEEE Transactions on Information Theory*. Vol 52. Nº 4. April. Pp 1289-1306.

Ducloux, J.; Colla, P.; Petrashin, P.; Lancioni, W. y Toledo, L. (2017): "Sistema de Reconocimiento de Gestos de la Mano basado en Acelerómetro para Interacción en TV Digital" Repositorio Digital Universitario. Centro Regional Universitario Córdoba. Universidad de la Defensa Nacional. https://rdu.iua.edu.ar/handle/123456789/461

Dueck, D. (2009): "Affinity Propagation: Clustering data by passing messages". Graduate Department of Electrical & Computer Engineering. University of Toronto. PhD thesis. Versión disponible en:
DDueck-thesis_small.pdf (columbia.edu)

Fernández-Macías, E.; Klenert, D. y Antón Pérez, J. I. (2020): "Not so disruptive yet? Characteristics, distribution and determinants of robots in Europe". *JRC Working Papers. Series on Labour, Education and Technology. European Commission*. Nº 2020/03. Versión disponible en: http://hdl.handle.net/10419/231335.

Flórez, R. y Fernández, JM. (2008): "Las Redes Neuronales Artificiales, Fundamentos Teóricos y aplicaciones prácticas". Serie Metodologías y Análisis de Datos en Ciencias Sociales. Netbiblo ediciones. A coruña. Versión disponible en:
Las Redes Neuronales Artificiales - Raquel Flórez López, José Miguel Fernández Fernández - Google Libros

IFR (2020): *Presentation World Robotics 2020 Industrial Robots Report*. International Federation of Robotics Press Conference 24th September 2020 Frankfurt. Version disponible en:
https://ifr.org/downloads/press2018/Presentation_WR_2020.pdf

Ispizua, E. (2018): "Industria 4.0: ¿Cómo afecta la Digitalización al Sistema de Protección Social?". *Lan Harremanak Revista de Relaciones Laborales*. Nº 40. Pp. 12-30. Versión disponible en: https://doi.org/10.1387/lan-harremanak.20325.

Izaurieta, F. y Saavedra, C. (2000): "Redes Neuronales Artificiales". Departamento de Física, Universidad de Concepción Chile. Versión disponible en:
https://disi.unal.edu.co/~lctorress/RedNeu/LiRna003.pdf

Jha, A. y Chiddarwar, S.S. (2017): "Robot programming by demonstration using teleoperation through imitation". *Industrial Robot*. Vol. 44. Nº 2. Pp. 142-154.

Kaâniche, M. B. (2009): "Gesture recognition from video sequences. Signal and Image processing". Université Nice-Sophia Antipolis UFR Sciences Ecole Doctorale STIC. PhD Thesis. October.

Kagermann, H.; Wahlster, W. y Helbig, J. (2013): *Recommendations for implementing the strategic initiative INDUSTRIE 4.0. Securing the future of German manufacturing industry*. Final report of the Industrie 4.0 Working Group. Joint Publication ForschungsUnion and Acatech (National Academy of Science and Engineering). 08 april

Kela, J.; Korpipää, P.; Mäntyjärvi, J.; Kallio, S.; Savino, G.; Jozzo, L. y Di Marca, S. (2006): "Accelerometer-based gesture control for a design environment". *Personal and Ubiquitous Computing*. Vol. 10. No. 5. Pp. 285-99

Kohonen, T. (1982): "Self-Organized Formation of Topologically Correct Feature Maps". *Biological Cybernetics*. Vol 43. Pp. 59-69.

Lau, T. (2009): "Why Programming by Demonstration Systems Fail: Lessons Learned for Usable AI". *AI Magazine*. Vol 34. Nº 4. Pp. 65-67.

Liu, J.; Wang, Z.; Zhong, L.; Wickramasuriya, J. y Vasudevan, V. (2009): "uWave: Accelerometer-based personalized gesture recognition and its applications". *Pervasive and Mobile Computing.* Pp. 1-9

Martín del Brío, B. y Sanz, A. (2006): *Redes Neuronales y Sistemas Borrosos*. Editorial RA-MA. Madrid. Tercera Edición. ISBN 978-84-7897-743-7

Martín del Brío, B. y Serrano, C. (1995): Fundamentos de las redes neuronales artificiales: hardware y software. *Scire. Representación y Organización del Conocimiento*. Vol 1. Nº 1. Pp. 103-125.

McCulloch, W.S. y Pitts, W. (1943): "A logical calculus of ideas immanent in nervous activity". *Bulletin of Mathematical Biophysics*. Vol. 5. Pp. 115-133. Cita tomada de: Salas (2004)

MINETUR (2014): *Industria Conectada 4.0. La transformación digital de la Industria Española. Informe Preliminar*. Ministerio de Industria, Energía y Turismo. 23 de julio de 2015.

Montaño, J.J. (2002): "Redes Neuronales Artificiales aplicadas al Análisis de Datos". Universitat de les Illes Balears. Facultad de Psicología. PhD thesis.

Müller, M. (2007): *Information Retrieval for Music and Motion*. Springer – Verlag Berlin Heidelberg. New York. ISBN 978-3-540-74047-6.

Nagata,F.;Watanabe,K.;Kiguchi,K.;Tsuda,K.;Kawaguchi,S.; Noda,Y. y Komino,M. (2001): "Joystick teaching system for polishing robots using fuzzy compliance control". *Proceedings of the 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation*. Canada. 29 July-1 Aug. 2001 Pp. 362-367.

Nahar,P.C. y Kolte, M.T. (2014): "An Introduction to Compressive Sensing and its Applications". *International Journal of Scientific and Research Publications*. Vol. 4. Nº 6. Pp 469-474.

Neto, P.; Pires, J.N. y Moreira, A.P. (2009), "Accelerometer-based control of an industrial robotic arm", *Proceedings of the 18th IEEE International Symposium on Robot and Human Interactive Communication*. Toyama, Japan. 27 Sept.-2 Oct. 2009. Pp. 1192-1197

Neto, P.; Pires, J.N. y Moreira, A.P. (2010): "High-level programming and control for industrial robotics: using a hand-held accelerometer-based input device for gesture and posture recognition". *Industrial Robot*. Vol. 37. Nº 2. Pp. 137–147.

Pires, J.N. (2005): "Robot-by-voice: Experiments on Commanding an Industrial Robot using the Human Voice". *Industrial Robot*. Vol 32. Nº 6. Pp. 505-511.

Pires, J.N.; Godinho, T. y Araujo, R. (2007): "Using digital pens to program welding tasks", *Industrial Robot*. Vol 34. Nº 6. Pp. 476-486.

Pires, J.N.; Veiga, G. y Araújo, R. (2009): "Programming-by-demonstration in the coworker scenario for SMEs". *Industrial Robot*. Vol. 36 Nº. 1. Pp. 73-83.

Rifkin, J. (2011): *The Third industrial revolution. How lateral power is transforming energy, the Economy, and the World*. Palgrave Macmillan. United States of America. ISBN: 978-0-230-11521-7.

Rivera-Taiba, T. (2019): "Efectos de la Automatización en el empleo en Chile". *Revista de Análisis Económico*. Vol. 34. Nº 1. Abril. Pp. 3-49.

Rubin, R.V.; Golin, E.J. y Reiss, S.P. (1985): "Think Pad: A Graphical System for Programming by Demonstration". *IEEE Software*. Tomo 2. Nº 2. March/April. Pp. 73-79.

Rumelhart, D.E.; Hinton, G.E. y Williams, M.J. (1985): "Learning Internal Representations by Error Propagation". Published in Rumelhart y McClelland (Eds.). *Parallel Distributed Processing*: *Explorations in the Microstructure of Cognition*. Vol. 1: Foundations. Cambridge, MA: Bradford Books/MIT Press. Pp. 318-362. Versión disponible en: https://apps.dtic.mil/sti/pdfs/ADA164453.pdf

Rzempoluck, E.J. (1998): *Neural network data analysis using Simulnet*. Springer-Verlag. New York. Cita tomada de: Montaño (2002).

Sabinas, Y. (2013): "Reconocimiento anticipado de gestos". Instituto Nacional de Astrofísica, Óptica y Electrónica INAOE. Mexico. PhD thesis.

Salas, R. (2004): "Redes neuronales artificiales". Universidad de Valparaiso. Departamento de Computación. Materiales. Volumen 1. Pp. 1-7. Versión disponible en: https://d1wqtxts1xzle7.cloudfront.net/50358783/Redes_Neuronales_Artificiales-with-cover-page-v2.pdf?Expires=1626382619&Signature=P0wsFwp5vYAxYqqv-56BvOBkA9KaPSgm78e42jfNzhXVn8FTOBdQBP3W0MUJDgDb8v-iu27kKFug7v0uE0qmj-C5xnRQg0GVyBuKTthtSNF0DiH4NpLM6cumaKBoDQ9OKtFwAjgKutrVVQUD0K1ctLJVtiPH~RRcdW9koUaRYEll1DNtF5yxYDVX7YWHGJqzGEOKBTRYv4hToByA8Gebrb5Q1T4oeUed8EsdeB5CJJvAuYZD6~3zs4WZlA6Yhwg5LavHzW66-6iQ9YywXNX17M-JKWA9KJwtnVmmQrCNhs0RMYPAF2Cmf5Fr0PwUxuxE5tiOlTkxgPqchvAv0nFFSg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

Schwab, K. (2016): *The Fourth Industrial Revolution*. Foro Económico Mundial. Penguin Random House Grupo Editorial. Barcelona. ISBN: 978-84-9992-699-5

Senin, P. (2008): *Dinamic Time Warping Algorithm Review*. University of Hawaii. Manoa, Honolulu. USA. Versión disponible en: https://seninp.github.io/assets/pubs/senin_dtw_litreview_2008.pdf

Valera Fernández, A. (2020a): *Programación y Simulación de Robots Industriales.* Repositorio de Material Master U. Ingeniería Industrial. Escuela Técnica Superior de Ingeniería Industrial. Universidad Politécnica de Valencia. https://poliformat.upv.es/access/content/group/DOC_33687_2020/Teor%C3%ADa/Tema%204%20Programacion%20RAPID%20Poliformat.pdf

Valera Fernández, A. (2020b): *Aplicaciones Industriales de Robots.* Repositorio de Material Master U. Ingeniería Industrial. Escuela Técnica Superior de Ingeniería Industrial. Universidad Politécnica de Valencia.

https://poliformat.upv.es/portal/site/DOC_33687_2020/tool/49b79f52-9f78-4948-8c30-3eb3416fa588?panel=Main

Wang, K. (2021). Fast Affinity Propagation Clustering under Given Number of Clusters (https://www.mathworks.com/matlabcentral/fileexchange/25722-fast-affinity-propagation-clustering-under-given-number-of-clusters), MATLAB Central File Exchange. Retrieved July 16, 2021.

Wang, H. y Li, Z. (2016): "Accelerometer-based gesture recognition using dynamic time warping and sparse representation". *Multimedia Tools and Applications*. Vol 75. Pp 8637–8655.

Yang, J.; Bang, W.; Choi, E.; Cho, S.; Oh, J.; Cho, J.; Kim, S.; Ki, E. y Kim, D. (2006): "A 3D hand-drawn gesture input device using fuzzy ARTMAP-based recognizer". *Journal of Systemics, Cybernetics and Informatics*. Vol. 4. Nº. 3. Pp. 1-7.

Yin, L.; Dong, M.; Duan, Y.; Deng, W.; Zhao, K. y Guo, J. (2014): "A high-performance training-free approach for hand gesture recognition with accelerometer". *Multimedia Tools and Applications*. Vol 72. Pp 843–864

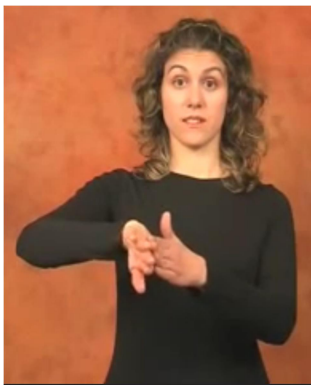
Zhou, Z.; Ji, L.; Xiong, R. y Wang, Y. (2020): "A spatial information inference method for programming by demonstration of assembly tasks by integrating visual observation with CAD model". *Assembly Automation*. Vol 40. Nº 5. Pp 689–701

ANEXO I

Agarrar

https://media.spreadthesign.com/video/mp4/5/475144.mp4

Derecha



https://media.spreadthesign.com/video/mp4/5/14985.mp4

Frente

https://media.spreadthesign.com/video/mp4/5/301144.mp4

Hacia Abajo

https://media.spreadthesign.com/video/mp4/5/152036.mp4

Hacia Arriba

https://media.spreadthesign.com/video/mp4/5/175607.mp4

Hacia Atrás

https://media.spreadthesign.com/video/mp4/5/152037.mp4

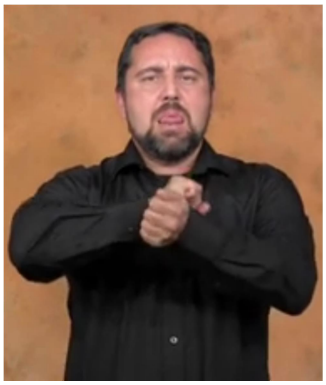
Iniciar

https://media.spreadthesign.com/video/mp4/5/55547.mp4

Izquierda

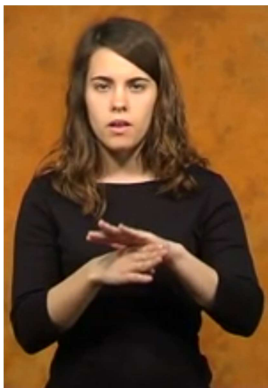
https://media.spreadthesign.com/video/mp4/5/272177.mp4

Liberar



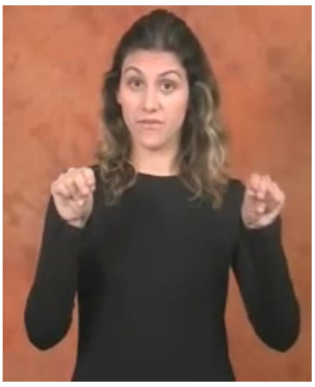
https://media.spreadthesign.com/video/mp4/5/179495.mp4

Seguir



https://media.spreadthesign.com/video/mp4/5/131789.mp4

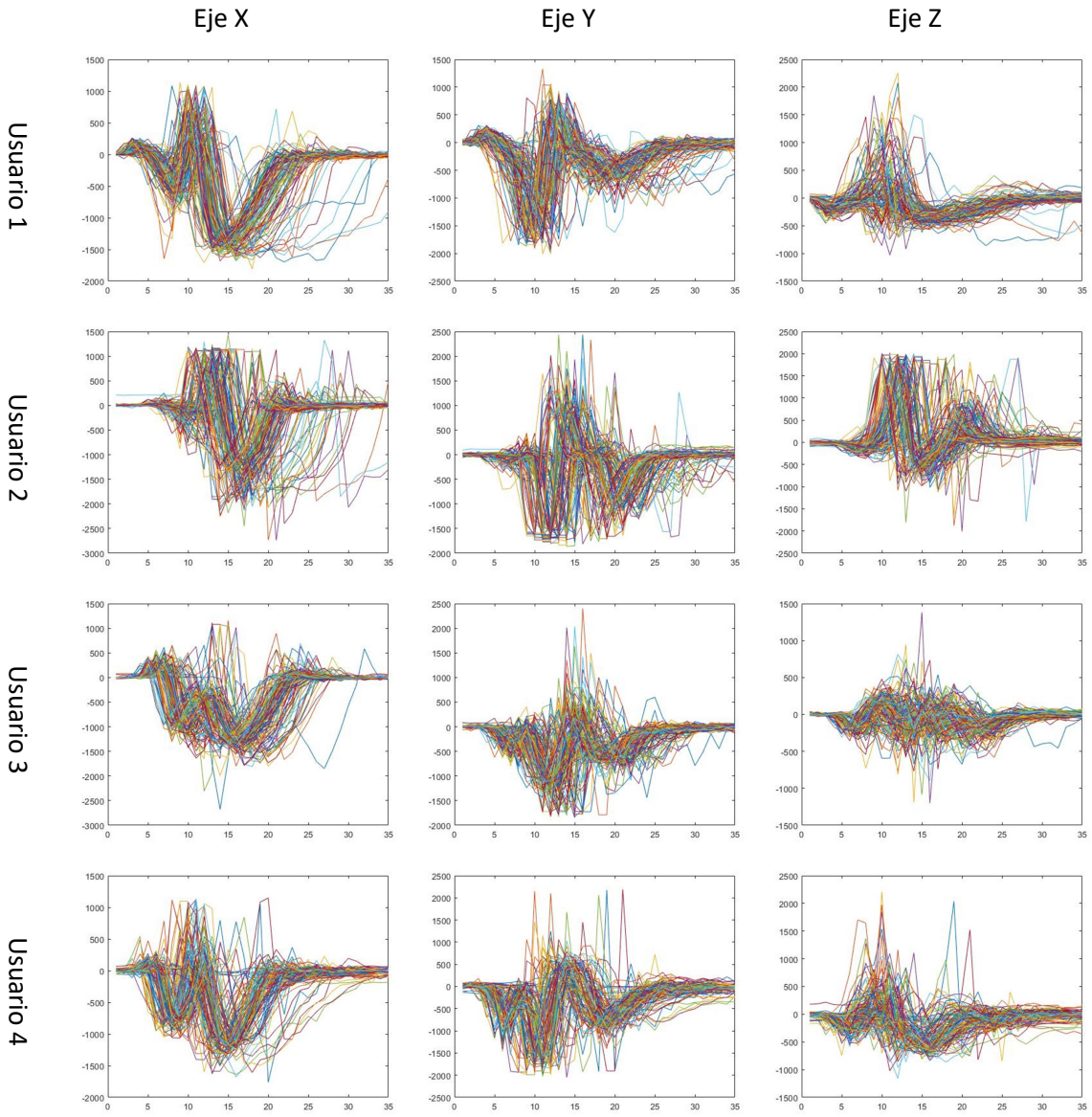
Terminar



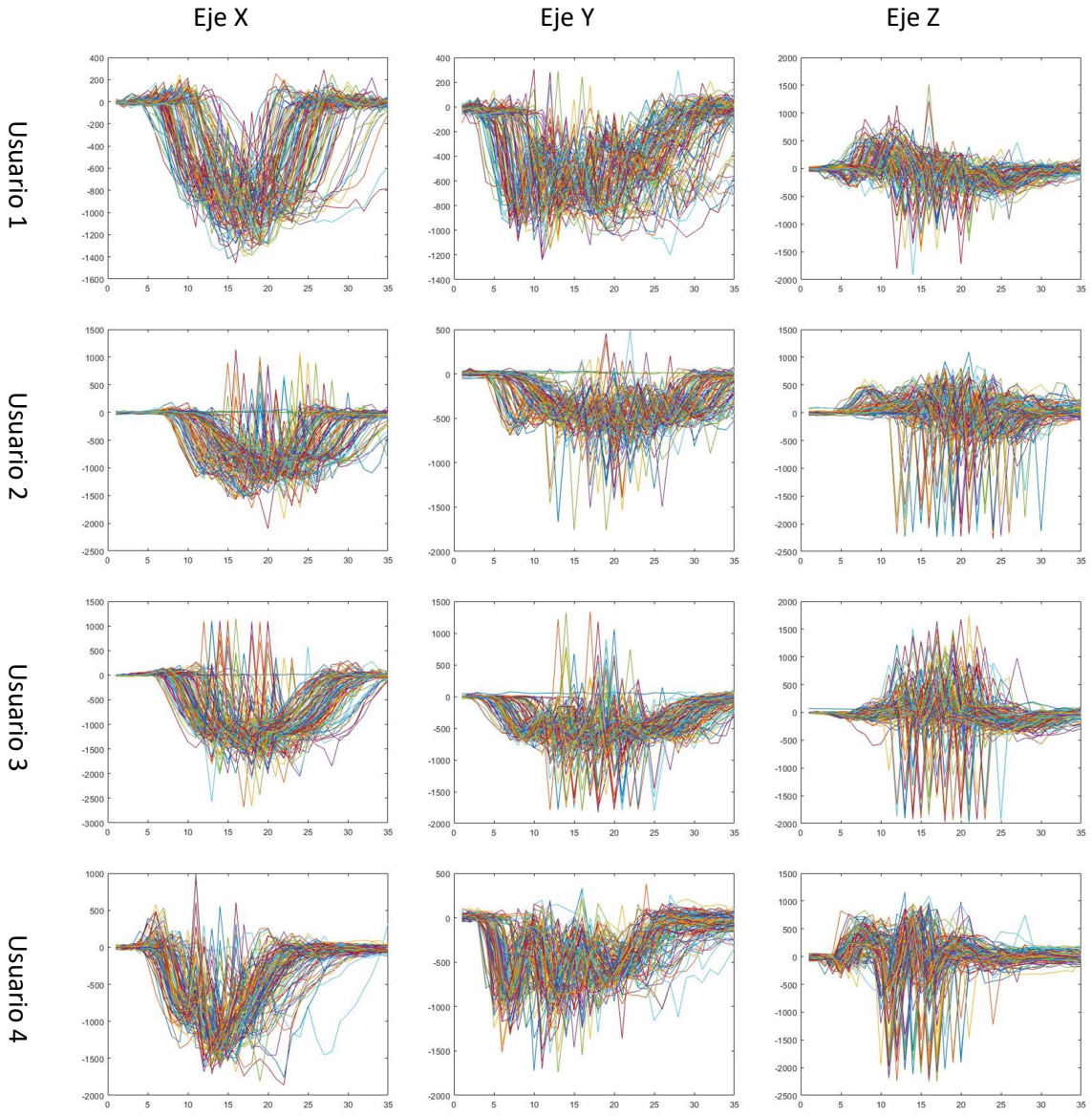
https://media.spreadthesign.com/video/mp4/5/47759.mp4

ANEXO II

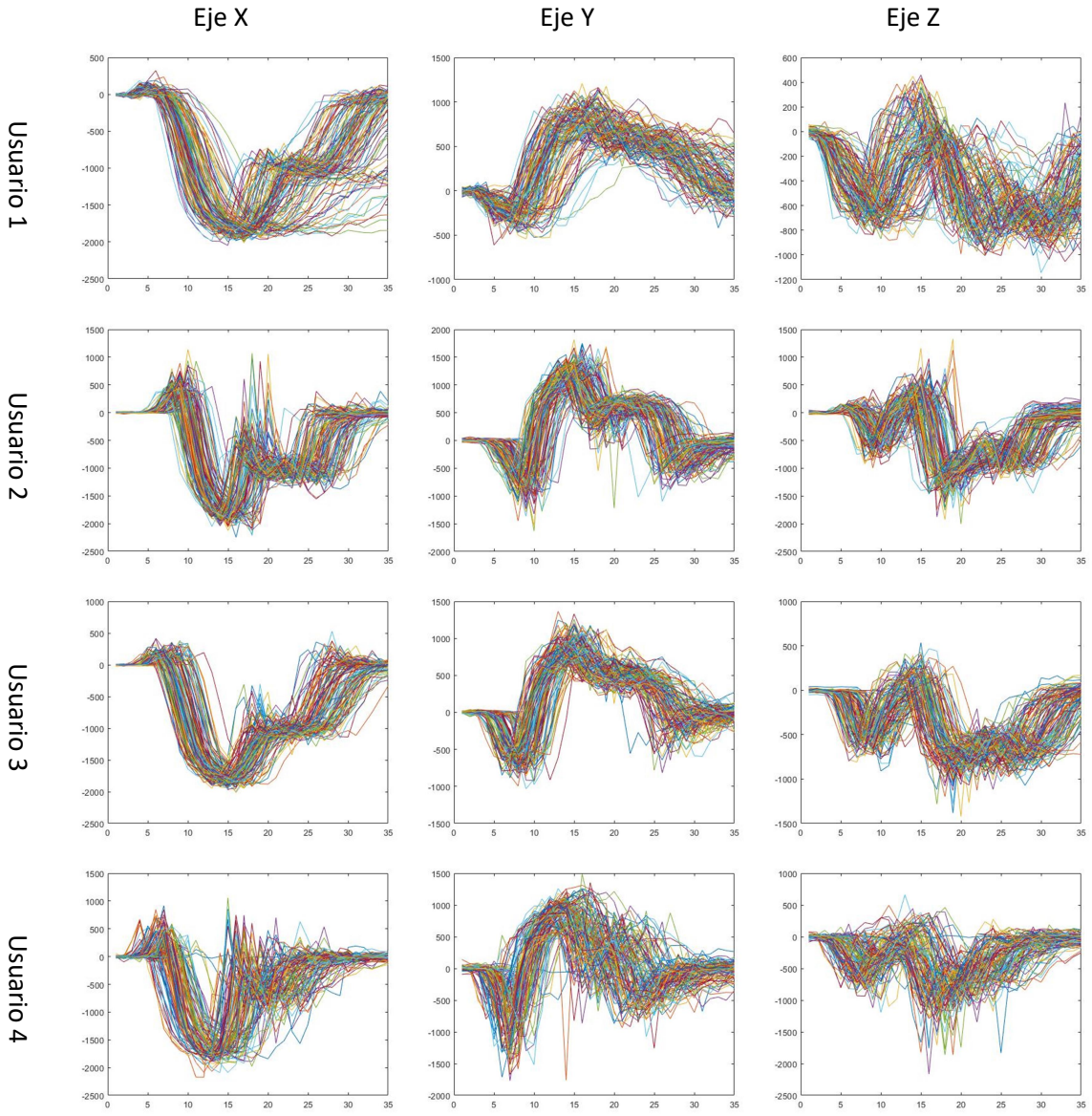
AGARRAR



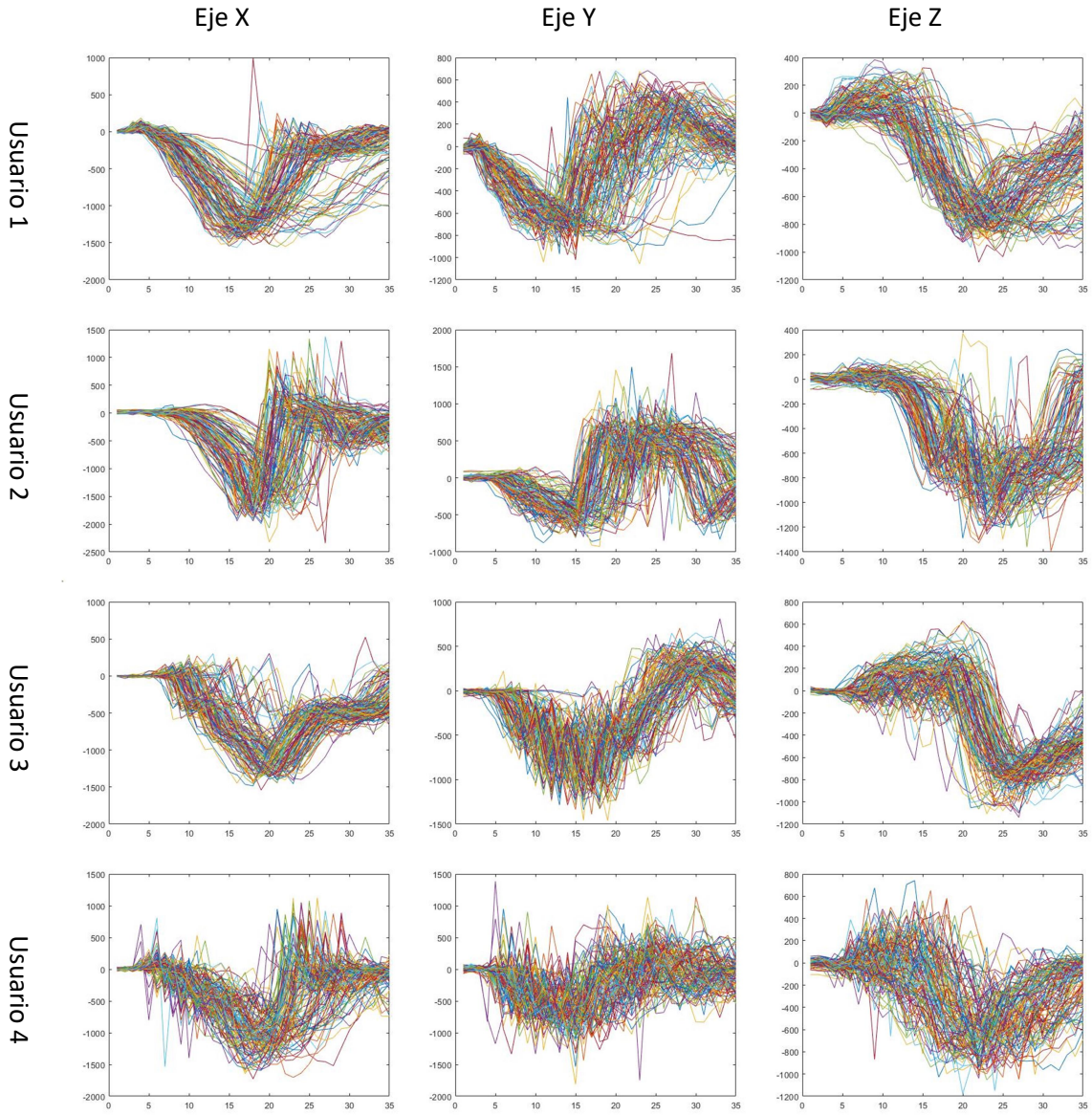
DERECHA



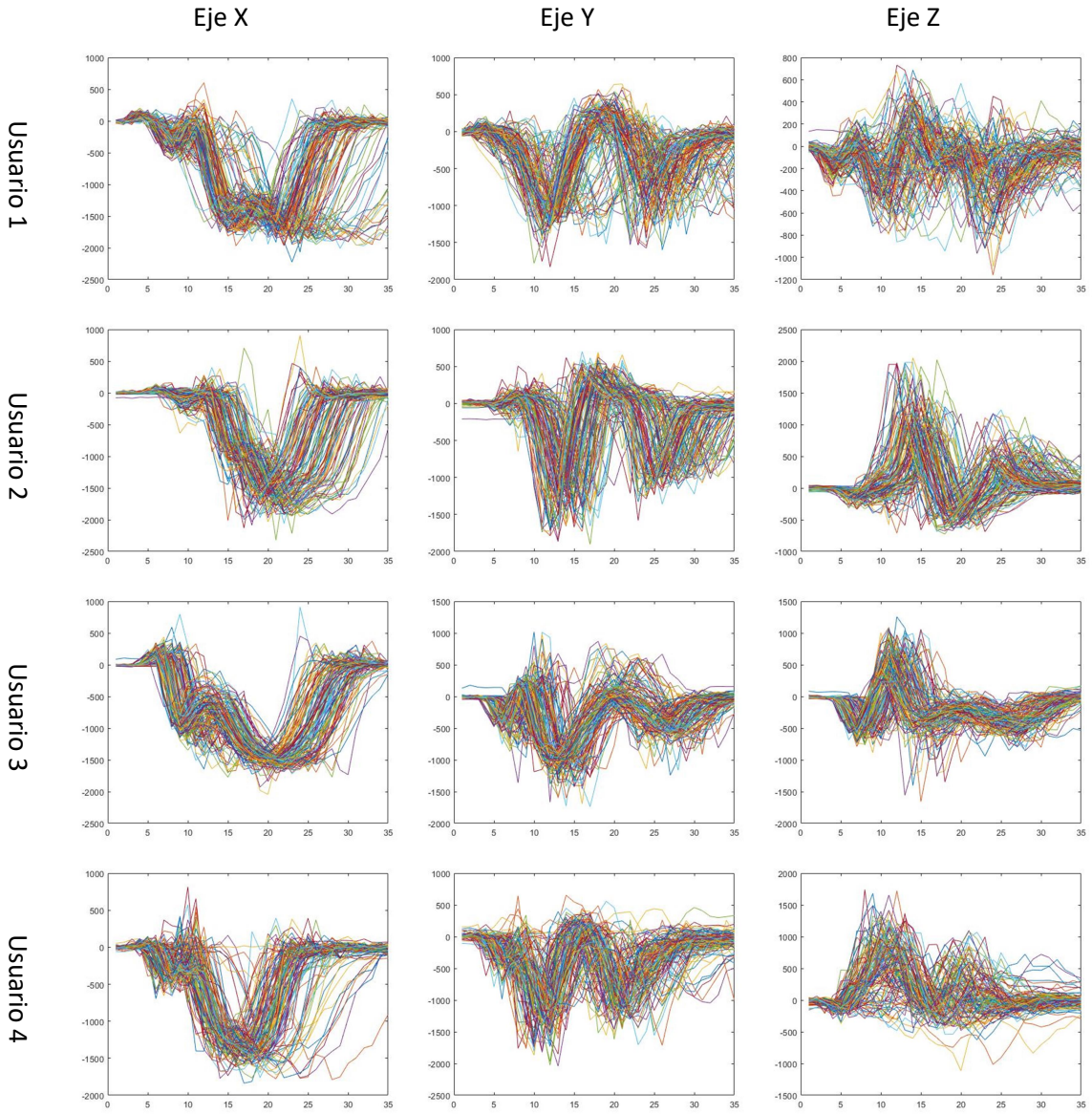
FRENTE



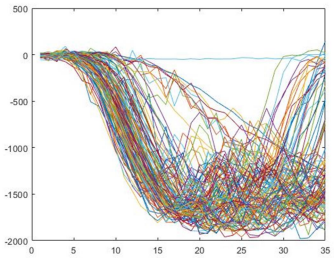
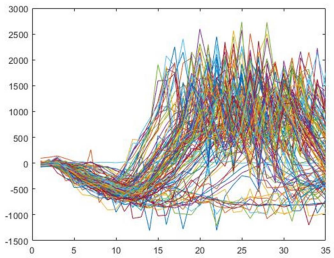
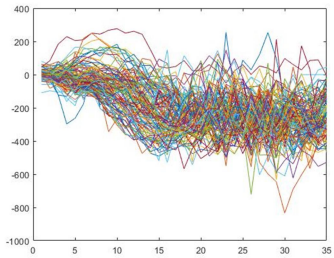
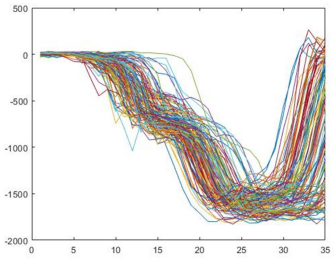
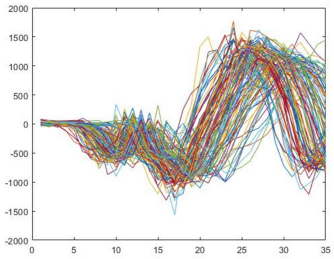
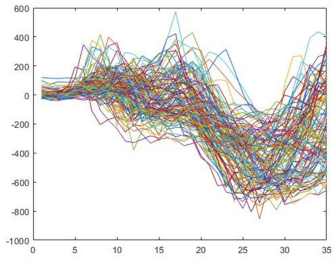
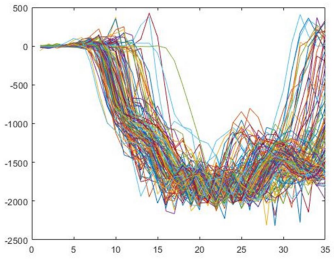
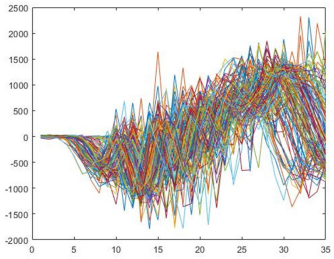
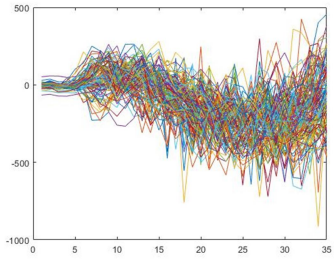
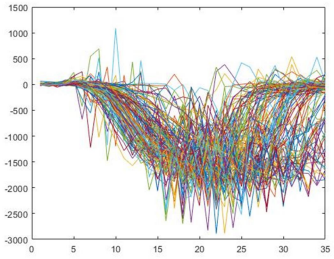
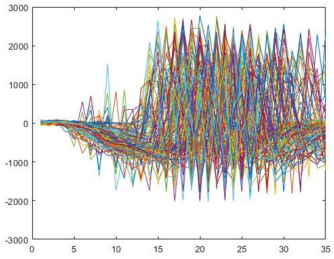
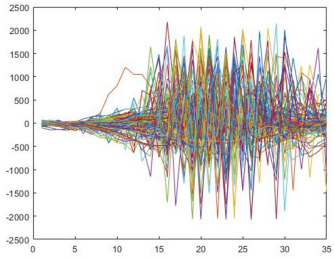
HACIA ABAJO



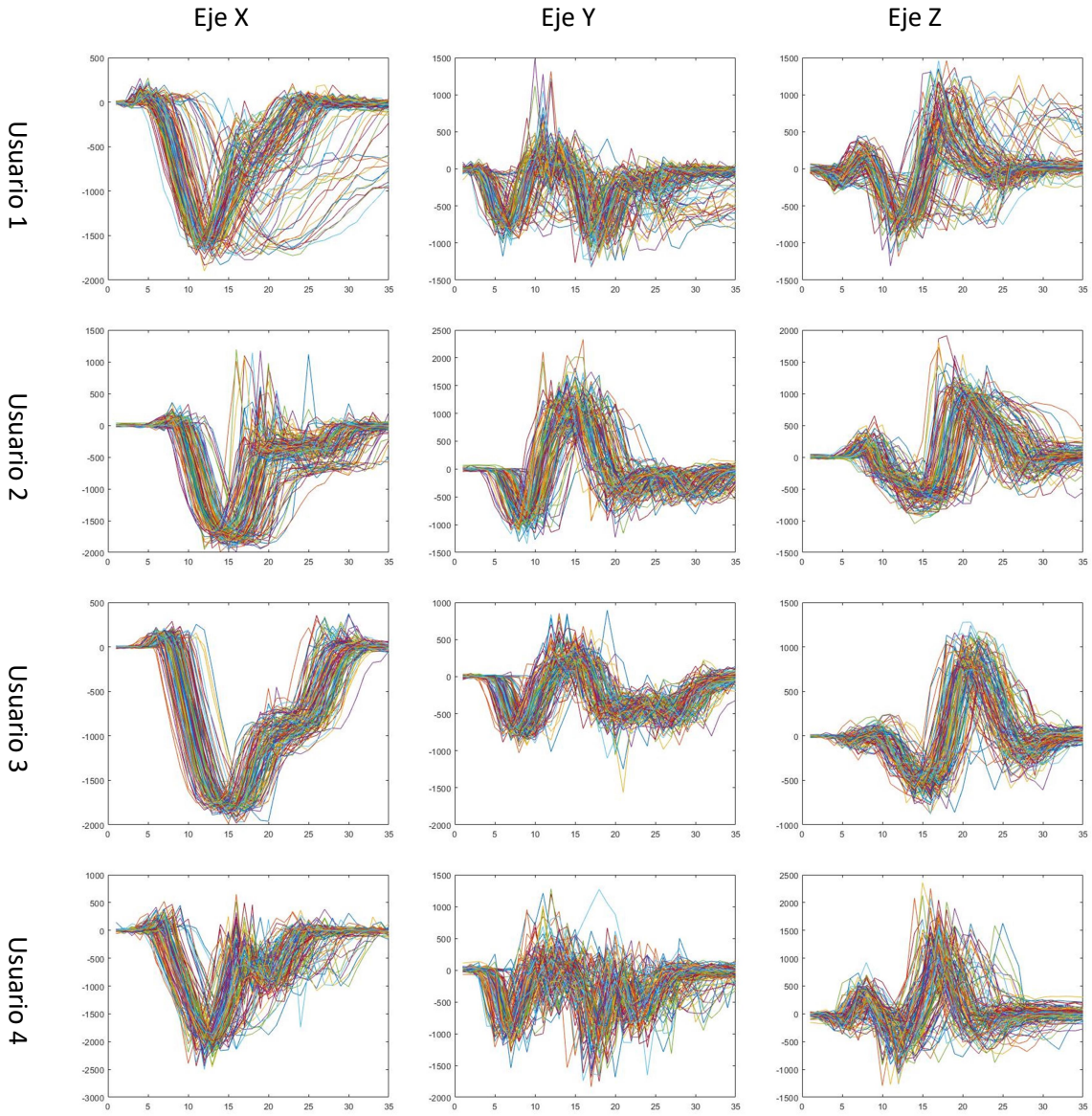
HACIA ARRIBA



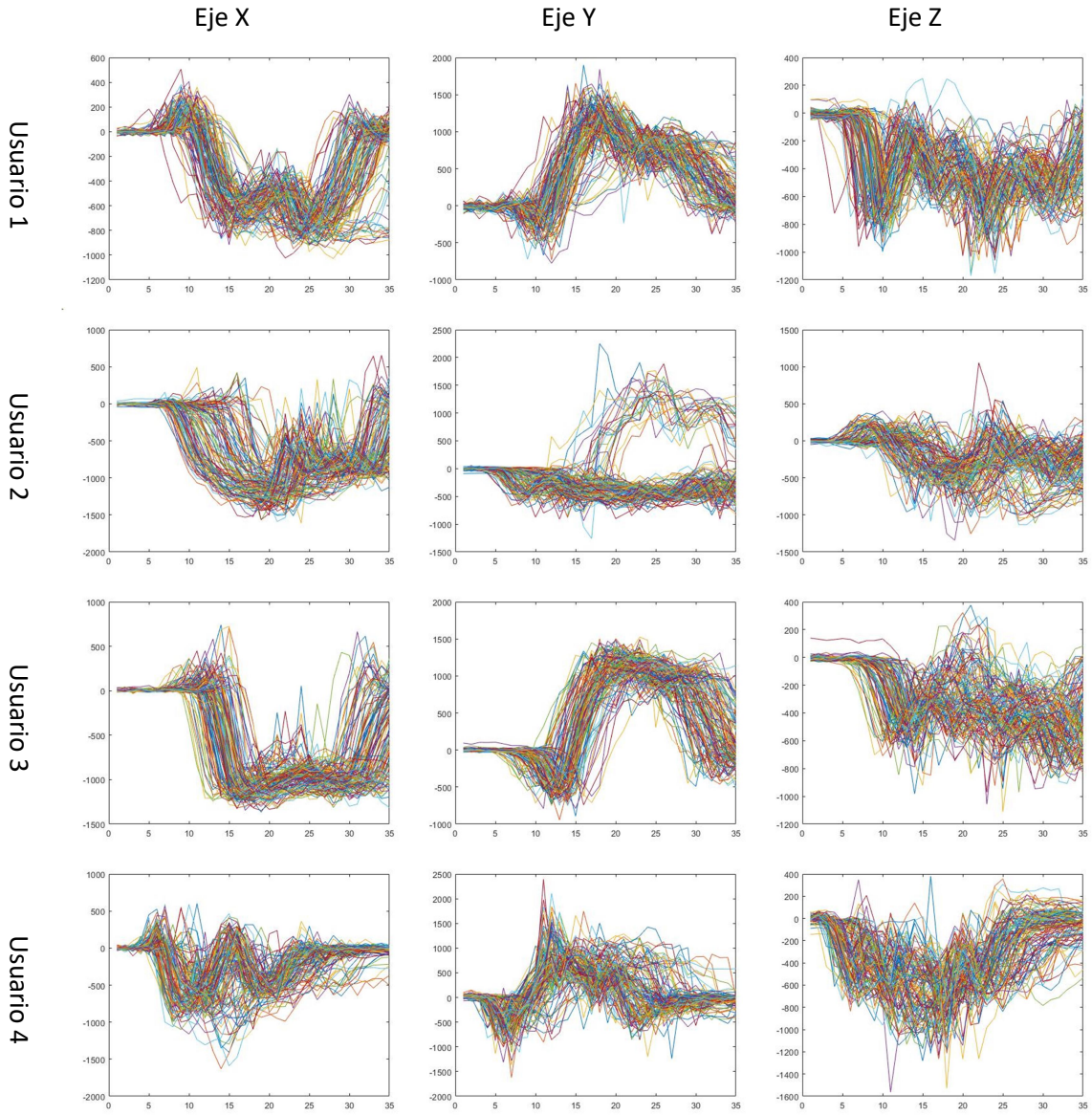
HACIA ATRÁS

	Eje X	Eje Y	Eje Z
Usuario 1			
Usuario 2			
Usuario 3			
Usuario 4			

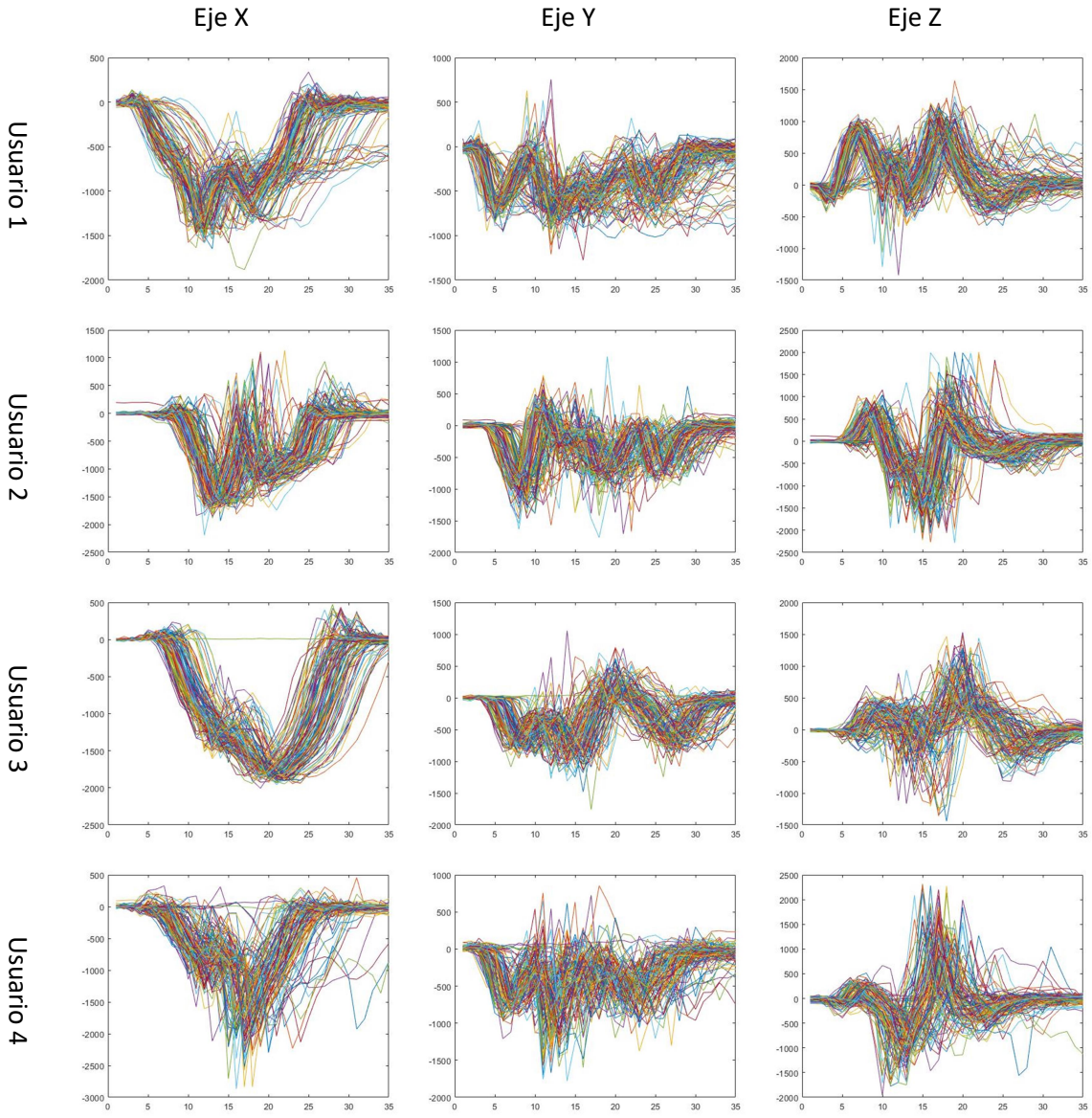
INICIAR



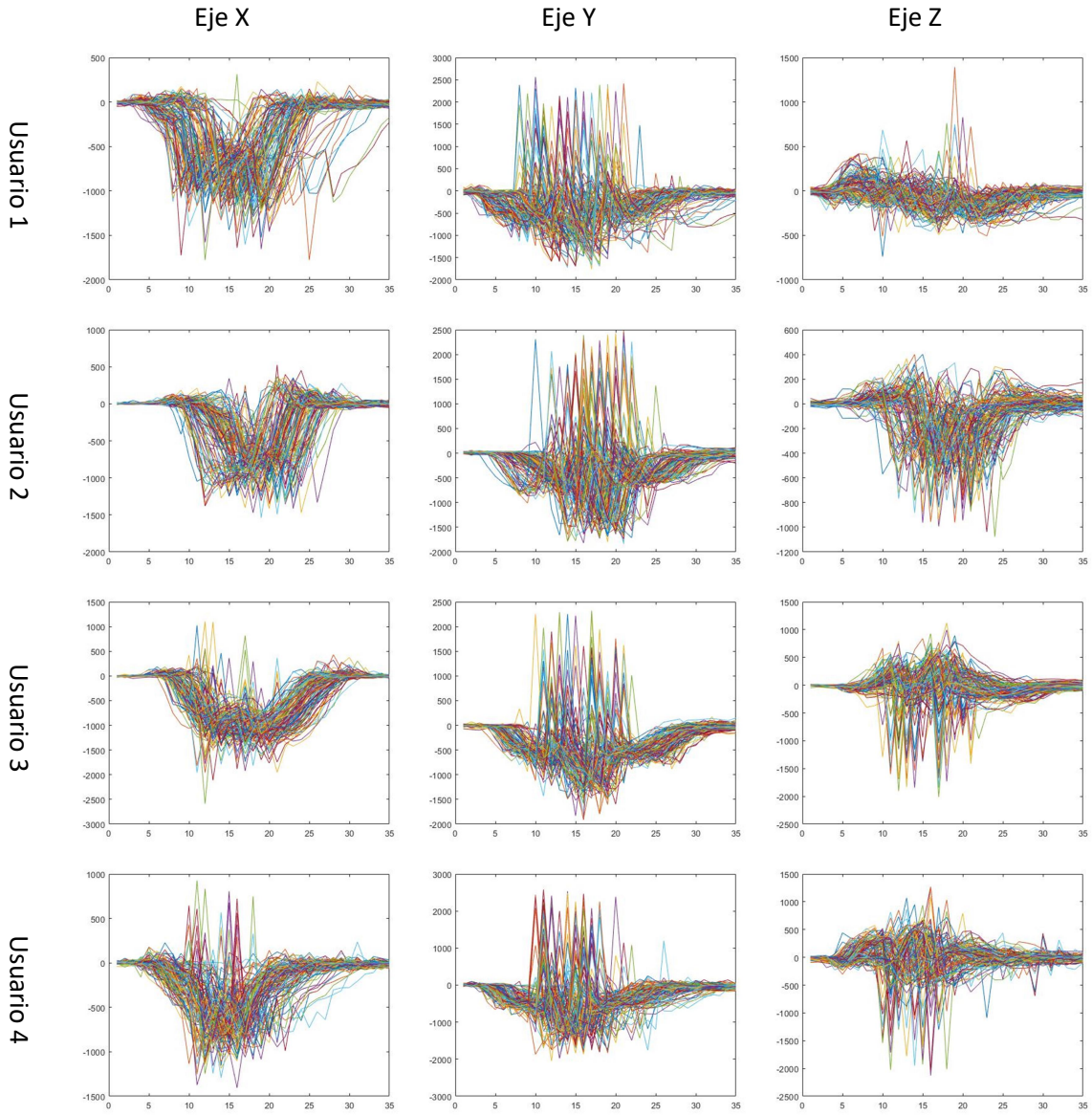
IZQUIERDA



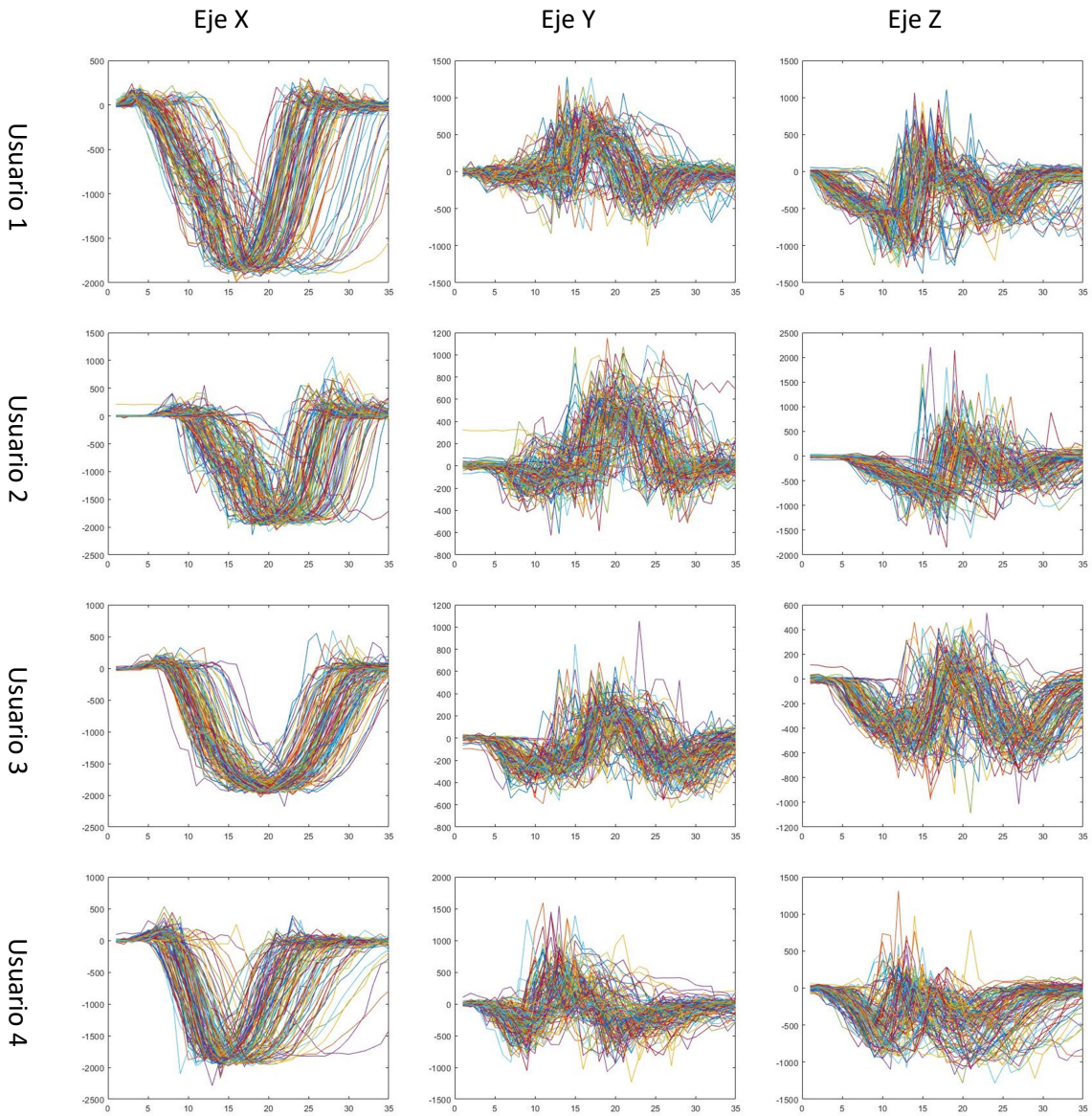
LIBERAR



SEGUIR



TERMINAR



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

PRESUPUESTO

DISEÑO Y VALIDACIÓN DE UN SOFTWARE DE RECONOCIMIENTO DE GESTOS SIRVIÉNDOSE DEL ACELERÓMETRO DE UN RELOJ INTELIGENTE

AUTOR: JORGE HERRERA CRESPO

TUTOR: DR. LEOPOLDO ARMESTO ÁNGEL

Aunque el proyecto desarrollado tiene unos costes de materias primas y equipamiento muy bajos, es importante llegar a tener una valoración precisa del presupuesto económico que el trabajo de "diseño y valoración del software de reconocimiento de gestos, sirviéndose del acelerómetro de un reloj inteligente" requiere para su elaboración.

La hipótesis para el diseño de este presupuesto es que el trabajo se realiza por encargo de una empresa al ingeniero que ha desarrollado el software, y es la empresa la que lo explotará comercialmente.

Los dos elementos más obvios para el cálculo del presupuesto son la compra del reloj inteligente programable de la marca LilyGo®, que incorpora un acelerómetro triaxial BMA423, y el coste de las horas de trabajo empleadas a lo largo del proceso.

Pero hay otros componentes presupuestarios que se deben incluir si esta tarea, en lugar de un trabajo dirigido a la elaboración de un TFM, fuese una tarea contratada por terceros y hubiese que hacer un cálculo del precio del encargo. En ese caso, habría que imputar unos gastos generales (como estimación de los costes de consumo eléctrico, conexión a internet, amortización del equipo informático, costes sociales del autor,…) que habitualmente se recoge en un porcentaje sobre precio de ejecución de un 13% adicional. También se añade un 6% de beneficio industrial. Y tras la suma de todo ello, un IVA del 21%.

1. CONTENIDO DEL PRESUPUESTO

A la hora de calcular el presupuesto de ejecución se van a realizar presupuestos parciales y, finalmente, un presupuesto final.

En el proceso de trabajo, se distinguen las siguientes actividades: Diseño del diccionario de gestos y de la librería; elaboración del algoritmo de reconocimiento; y redacción de la Memoria del Proyecto.

1. DISEÑO DEL DICCIONARIO DE GESTOS Y DE LA LIBRERÍA DE GESTOS

   Se considera incluida en esta fase:

   - La compra del dispositivo LilyGo® con acelerómetro triaxial BMA423.
   - Coste de compensación a las personas que colaboran en la elaboración de la librería, mediante la realización de los gestos que se almacenarán.
   - Coste de horas de trabajo de Ingeniería industrial.

2. ELABORACIÓN DEL ALGORITMO DE RECONOCIMIENTO

En esta fase se incluyen tanto las horas de análisis de la información preliminar (estudio de documentación, trabajos previos, …), como la del diseño del propio algoritmo.

En el diseño de un presupuesto profesional, el estudio de información preliminar no se debería presupuestar (pues se considera que el profesional conoce cualquiera de los elementos de trabajo y, tan sólo si hubiese un coste de acceso a algún documento necesario del proceso se podría imputar. Como no es el caso, se harán visibles estas horas en este presupuesto, pero se facturarán a coste cero.

No así las horas de programación, que sí se incluirán. Por tanto, se considera incluida en esta fase tan sólo:

- Coste de horas de trabajo de Ingeniería industrial para la elaboración del algoritmo de reconocimiento.

3. REDACCIÓN DE LA MEMORIA DEL PROYECTO

La redacción de la Memoria es una tarea también facturable, pues de su calidad depende la correcta implementación y aprovechamiento por parte del cliente del producto elaborado (software de reconocimiento de gestos).
En este caso, igual que en la actividad anterior, se considera incluido:

- Coste de horas de trabajo de Ingeniería industrial.

DISEÑO DEL DICCIONARIO DE GESTOS Y DE LA LIBRERÍA DE GESTOS

Se entiende que el reloj programado, finalmente sí que se entregará al cliente como parte del trabajo realizado. De no ser así, no debería incluirse en los costes a facturar por su precio total (dado que su uso no se consume con la prestación del servicio) y quedaría incluido en el 13% de gastos generales.

Nº Act.	Concepto	Uds	Precio €	Importe (€)
01	Compra del dispositivo LilyGo con acelerómetro triaxial BMA423	1	36,55	36,55
02	Horas ingeniería	70	41,87	2930,90
03	Horas colaboradores	40	2,3	92,00
	COSTES DIRECTOS			3059,45
	Costes directos complementarios 1%			30,59
	COSTE TOTAL			**3090,44**

Tabla 1 Costes de diseño del diccionario de gestos y de la librería de gestos.

El precio por "hora de ingeniería", ha sido obtenido de la Encuesta de Salarios y Actividad Profesional de los Colegios Oficiales de Ingenieros Industriales de Álava, Bizkaia, Gipuzkoa y Navarra[9] (la mejor disponible a la que se ha podido tener acceso).

[9] Disponible en http://www.adconsultores.es/wp-content/uploads/Encuesta-salarios-Ingenieros-Industriales-2018-2019-v0.5.pdf

Se refiere a precios de 2018 y 2019, y para acceder a los precios vigentes requiere colegiación. Como se entiende que en el caso de ser este un proyecto profesional, el titular estaría colegiado y, por tanto, podría tener acceso a los precios actualizados, se considera esta tarea como una simulación de la real.
En el estudio que incluye la encuesta de salarios se establece este importe (65,43 €/hora) para el caso de un ingeniero que trabaja por cuenta propia, y factura por horas, con tareas propias de una Oficina Técnica. De ahí, se ha detraído el 21% de IVA, el 6% de beneficio industrial y el 13% de gastos generales, que se imputarán ya al final del gasto del presupuesto (65,43-21% = 51,69; 51,69 – 19% = **41,87 €/hora**).

Por lo que respecta a la compensación de las personas que colaboran en la elaboración de la librería de gestos, se entiende que su tarea debe ser recompensada. Es cierto que el importe de compensación es difícil de calcular. A nivel científico, los experimentos más habituales tienen que ver con la toma de medicamentos, en ese caso hay un Reglamento Europeo (UE 536/2014) que regula ese tema, pero como norma general no hay retribución para los participantes en los ensayos clínicos. En el caso de este estudio, ni siquiera existen los riesgos de un ensayo clínico, pero se entiende que debe compensarse esa colaboración, y para hacer una estimación se ha utilizado el Procedimiento de elección de Colaboradores Científico-Técnicos de la Agencia Estatal de Investigación[10]. Aunque su perfil, no es exactamente el del caso que se plantea aquí, puede ser un dato que se podría utilizar por analogía. En dicho procedimiento se establecen 4 tipos de módulo de compensación, según el nivel de dificultad de la tarea, siendo el Módulo A el más bajo. Para ese módulo se determina una compensación de 465 €/mes, con lo que entendiendo una correspondencia entre retribución mes y retribución hora de 180 horas al mes (como jornada ordinaria de un trabajo por cuenta ajena), el precio por hora sería 465/180 = **2,6 €/hora**.

ELABORACIÓN DEL ALGORITMO DE RECONOCIMIENTO

Nº Act.	Concepto	Uds	Precio €	Importe (€)
02	Horas ingeniería	100	41,87	4187,00
04	Licencias MATLAB	1	1.363	1363,00
	COSTES DIRECTOS			5550,00
	Costes directos complementarios 1%			55,50
	COSTE TOTAL			**5605,50**

Tabla 2 Costes de elaboración del algoritmo de reconocimiento.

[10] Disponible en: https://www.unav.edu/documents/11314/9085421/2018_Protocolo-eleccion-colaboradoresCT-AEI.pdf

En la fase de elaboración del algoritmo, se contempla también abonar por el uso de licencias MATLAB
No se incluye ningún otro tipo de licencia, proyecto o legalizaciones, porque no se considera que existan para este caso.

REDACCIÓN DE LA MEMORIA DEL PROYECTO

Nº Act.	Concepto	Uds	Precio €	Importe (€)
02	Horas ingeniería	130	41,87	5443,10
	COSTES DIRECTOS			5443,10
	Costes directos complementarios 1%			54,43
	COSTE TOTAL			**5497,53**

Tabla 3 Costes de redacción de la Memoria del proyecto.

2. PRESUPUESTO FINAL

Concepto	Importe (€)
COSTE **DISEÑO DEL DICCIONARIO DE GESTOS Y DE LA LIBRERÍA DE GESTOS**	3090,44
COSTE **ELABORACIÓN DEL ALGORITMO DE RECONOCIMIENTO**	5605,50
COSTE **REDACCIÓN DE LA MEMORIA DEL PROYECTO**	5497,53
PRESUPUESTO DE EJECUCIÓN	14193,47
Gastos Generales (**13%**)	1845,15
Beneficio Industrial (**6%**)	851,60
PRESUPUESTO ANTES DE IMPUESTOS	16890,22
IVA (21%)	3546,94
PRESUPUESTO FINAL	**20437,16**

Tabla 4 Presupuesto Final.

El presupuesto final que se pasaría a la cliente por el trabajo profesional de ingeniería de elaboración de un algoritmo de reconocimiento de gestos, validado e incorporado al software de un reloj de pulsera, será de

VEINTE MIL CUATROCIENTOS TREINTA Y SIETE EUROS CON DIECISEIS CÉNTIMOS