UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR

DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

MÁSTER UNIVERSITARIO EN INVESTIGACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

"SISTEMA DE RECONOCIMIENTO DE ACCIONES MEDIANTE CÁMARAS CON DETECCIÓN DE PROFUNDIDAD"

AUTOR: ANTONIO TEJERO DE PABLOS
TUTOR: DR. MARIO MARTÍNEZ ZARZUELA

Julio de 2013

TITULO:	**"SISTEMA DE RECONOCIMIENTO DE ACCIONES MEDIANTE CÁMARAS CON DETECCIÓN DE PROFUNDIDAD"**
AUTOR:	Antonio Tejero de Pablos
TUTOR:	Dr. Mario Martínez Zarzuela
DEPARTAMENTO:	Teoría de la Señal y Comunicaciones e Ingeniería Telemática

---

Miembros del Tribunal

PRESIDENTE:	Francisco Javier Díaz Pernas
SECRETARIO:	José Fernando Díez Higuera
VOCAL:	Carlos Gómez Peña
SUPLENTE:	Míriam Antón Rodríguez

FECHA DE LECTURA:	23 de Julio de 2013
CALIFICACIÓN:	

AGRADECIMIENTOS

Doy las gracias a los profesores del Máster por ofrecerme los conocimientos que me han ayudado a llevar a cabo este proyecto y me han dado su apoyo.

Agradezco a mi familia su apoyo constante, que me ha permitido llegar hasta aquí, y a mis amigos por ayudarme fuera de la universidad.

Le agradezco a mi tutor Mario su confianza y el contar siempre conmigo para trabajar juntos, brindándome una oportunidad tras otra para demostrar mis capacidades.

RESUMEN

El campo de monitorización y reconocimiento de movimientos es el centro de muchas investigaciones, dadas sus múltiples aplicaciones. Mediante el estudio de los sistemas actuales pueden observarse limitaciones y otros aspectos de mejora. Es posible aplicar nuevas técnicas de forma que se haga uso de tecnología comercial actual para mejorar el rendimiento de dichos sistemas de reconocimiento. De esta forma, este trabajo presenta un estudio del estado del arte en la monitorización y reconocimiento mediante información de profundidad, en el que se destacan las fortalezas y debilidades de los paradigmas actuales. A partir de este análisis, se propone un esquema de reconocimiento con capacidad de mejorar los sistemas actuales. La implementación consiste en un ToolBox de MatLab® con distintas funcionalidades. Sobre este se realizan una serie de experimentos y se discuten los resultados obtenidos, de forma que pueda trazarse un plan para futuros desarrollos que hagan uso de este estudio.

PALABRAS CLAVE

Monitorización, reconocimiento de movimientos, entornos 3D, Kinect.

ABSTRACT

The field of tracking and action recognition systems is the focus of many researchers, due to the variety of their applications. By studying current tracking and action recognition systems, limitations and room for possible improvements can be observed. It is possible to apply new techniques so they make usage of current commercial technology to improve the performance of the aforementioned recognition systems. This way, this work presents a study of the state-of-the-art in tracking and recognition systems that use depth information, highlighting the strengths and weaknesses of current paradigms. From this analysis, we propose a recognition scheme which could be able to improve current systems. The implementation consists of a MatLab® ToolBox with different functionalities. A series of experiments are carried out on this system and the results are discussed, so we can describe a plan for future developments that may start off from this work.

KEY WORDS

Tracking, action recognition, 3D spaces, *Kinect*.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1: Introducción

La forma en la que los seres humanos perciben el espacio y, en concreto, perciben el movimiento es un tema de estudio de complejidad elevada. De la misma forma, dotar a una máquina de la capacidad para realizar esta tarea de forma automática tiene una dificultad equiparable. Es por ello que un gran número de investigaciones actuales hacen referencia a este tipo de sistemas. Puede definirse un movimiento como un conjunto de desplazamientos del cuerpo humano secuenciados en el tiempo. En un movimiento pueden estar involucrados una o más partes del cuerpo de forma concurrente para componer la acción completa. Si se desea analizar las acciones contenidas en una secuencia de *frames* de video, el área de la visión artificial también está involucrada en el proceso. Por lo tanto, desde un punto de vista externo, el reconocimiento de una acción pasa por una serie de fases en la que la observación (p. ej. un vídeo) es comparada con una serie de modelos pre-definidos y es clasificada.

El reconocimiento de movimientos ha sido objeto de investigación y usado para propósitos de vigilancia, interacción hombre-máquina, recuperación de video, etc. El problema puede definirse de la siguiente manera: dada una secuencia de movimiento, el ordenador debe identificar las acciones que son realizadas por el sujeto. Dependiendo de su complejidad una acción humana puede ser categorizada en cuatro niveles diferentes: gestos, acciones, interacciones y actividades de grupo (Aggarwal & Ryoo, 2011). Este trabajo tiene en consideración el movimiento humano en general. El sistema que lleva a cabo el reconocimiento está compuesto por una serie de componentes: extracción de características, aprendizaje de movimientos y su clasificación, y segmentación. Sin embargo, en algunos casos se omiten pasos para simplificar el proceso. Un sistema simple consiste en tres pasos: detección del cuerpo humano en los *frames* de video, monitorización del movimiento corporal, y usar la información recogida para el reconocimiento. A pesar de esta simplicidad, este paradigma depende en gran medida de la precisión del subsistema de monitorización. El proceso de monitorización consiste en llevar a cabo un registro de las características del movimiento a lo largo del tiempo, y puede llevarse a cabo sobre diferentes atributos del cuerpo humano: articulaciones, flujo óptico de las partes en movimiento, etc.

1.1. Aplicaciones

Existen muchos ámbitos en los que se aplica reconocimiento de movimientos, para muchos propósitos diferentes: analizar la trayectoria de un objeto, observar comportamientos, o monitorizar y detectar acciones humanas (Poppe, 2007).

Mediante esta monitorización es posible realizar una gran cantidad de tareas, entre las que destaca el reconocimiento de las acciones que realizan los individuos que se encuentran en el entorno. Este último caso engloba muchas aplicaciones interesantes y es el campo de estudio de este trabajo. Las aplicaciones de los sistemas de monitorización y reconocimiento de este tipo de movimientos son muchas y muy variadas. Los siguientes apartados contienen las más relevantes.

1.1.1. Vigilancia

Poder discernir qué es lo que ocurre en la escena monitorizada. Presencia de personas, obstáculos, etc. donde no debería haberlos, para prevenir situaciones indeseadas o peligrosas.

1.1.2. Monitorización de individuos

Actualmente existen un gran número de proyectos que tienen en cuenta la monitorización de personas mayores o con alguna discapacidad con el objetivo de garantizar su seguridad, ya sea mediante la detección de acciones potencialmente peligrosas, alarma ante caídas y accidentes o localización dentro de un recinto. Un ejemplo de ello es el estudiado en (Han, Lee & Peña-Mora, 2012), donde a partir de grabaciones de acciones inseguras se realiza reconocimiento de movimientos. El estudio utiliza modelos 3D y métodos para reducir la dimensión de los datos, los cuales también se tratarán posteriormente en este trabajo.

1.1.3. Interfaces hombre-máquina

Las personas con discapacidades motrices sufren especialmente a la hora de interactuar con ciertos dispositivos, como puede ser un ordenador o una casa domótica. Mediante el reconocimiento de gestos sencillos de realizar, con las manos o con la cabeza, la persona discapacitada es capaz de realizar operaciones que de otro modo serían imposibles para él o ella. Otra aplicación de gran complejidad se encuentra en el área de la robótica, donde ya hay trabajos en los que se intentan generar secuencias que imitan movimientos capturados (Ariki, Hyon & Morimoto, 2013). Para ello, se extraen primitivas del movimiento original, mediante las cuales se construye el comportamiento deseado. Estas primitivas se obtienen a partir de las trayectorias de los ángulos de las articulaciones del cuerpo y su velocidad. Esto es aplicado al movimiento de robots.

1.1.4. Rehabilitación

Monitorizar un ejercicio realizado por una persona para determinar si lo está realizando bien o mal. No sólo eso, además es posible aportar un valor o puntuación sobre cómo de parecido es el movimiento con el ejercicio ideal,

llevado a cabo por un profesional como referencia. Además, los entornos virtuales también son aplicables a la rehabilitación de personas con discapacidades de tipo mental, debido a la mayor inmersión a la hora de realizar los ejercicios.

Por otra parte, al analizar los movimientos de una persona puede extraerse información sobre cómo lo está realizando (estado de ánimo, velocidad, etc.). Esto aporta un nuevo nivel de información de contexto a la hora de presentar los resultados al paciente.

1.2. Motivación y relevancia del problema

Mediante el análisis del estado del arte, puede observarse que los sistemas de monitorización y reconocimiento actuales sufren de diversas limitaciones:

- Los dispositivos de captura (cámaras) 2D ofrecen información limitada al sistema sobre lo que ocurre en la escena. Actualmente existe hardware aún por explotar para poder mejorar dichos sistemas de monitorización. Además, los sistemas tradicionales son muy vulnerables a las oclusiones.
- Los sistemas no son capaces de reconocer una cantidad muy variada de movimientos. Muchos de ellos se ven limitados a gestos predeterminados.
- La precisión de los sistemas es mejorable. Pese a conseguir tasas de reconocimiento altas para ciertos movimientos, los sistemas se confunden a la hora de reconocer movimientos parecidos como avanzar saltando con ambos pies o sólo uno.
- No todos los sistemas funcionan en tiempo real. El tiempo real supone un coste adicional de procesamiento, a la hora de extraer y clasificar los movimientos realizados por la persona.
- Existe un número reducido de *datasets* de movimientos que estén basados en información de profundidad. Esto es un inconveniente a la hora de desarrollar y probar un nuevo sistema, e incluso para comparar los resultados con otros sistemas que hagan uso del mismo *dataset*.

En la actualidad existen dispositivos de captura comerciales que ofrecen información 3D de la escena grabada. Entre ellos, destaca el hardware Kinect de Microsoft®, al que a partir de ahora nos referiremos simplemente como Kinect. Asimismo, existen tecnologías hardware con múltiples procesadores que permiten la ejecución de sistemas con gran carga computacional de forma más rápida y eficiente. Estas tecnologías van acompañadas de sus respectivos lenguajes de programación que permiten utilizar dicha información, y que también requieren de un estudio particular para usarse correctamente.

Este trabajo está enfocado en el ámbito del reconocimiento de movimientos usando un modelo de esqueleto 3D. Este esqueleto se calcula monitorizando las articulaciones de la persona que realiza el movimiento. En este documento, nos referiremos a estos puntos de articulación del modelo de esqueleto del cuerpo humano como *joints*. La aparición de técnicas novedosas, como la invención de Microsoft® Kinect, permite la extracción de esqueletos humanos de forma cómoda y fiable a partir de videos de movimiento. La motivación de este Trabajo de Fin de Máster es el estudio de distintas técnicas que permitan analizar y procesar la trayectoria de las articulaciones o *joints* de dichos esqueletos para poder clasificar el movimiento realizado por el usuario en una u otra acción, de forma que se mejore el rendimiento en cuanto a precisión. También se consideran técnicas de procesado en GPU para mejorar el rendimiento en cuanto a tiempo de ejecución.

1.3. Estructura del documento

El resto del documento se estructura como sigue. La Sección 2 revisa los paradigmas actuales sobre los sistemas de reconocimiento, y más concretamente, los basados en tecnologías de monitorización en 3D. Además se estudian ejemplos ya existentes del estado del arte, y se ofrece información sobre la utilidad de la computación en GPU para este tipo de sistemas. La Sección 3 plantea el estudio a realizar con sus objetivos, introduciendo la metodología a seguir en la fase de experimentación y técnicas a utilizar. En la Sección 4 se desarrollan los experimentos con el sistema de reconocimiento, siguiendo el plan de experimentación propuesto en la sección anterior. Una vez realizados, se analizan los resultados obtenidos, y se comparan con otros sistemas de enfoque similar. Finalmente, la Sección 5 dibuja una serie de conclusiones del trabajo y posible trabajo a realizar en el futuro.

Capítulo 2: Conocimientos previos

Este apartado tiene como objetivo introducir los trabajos ya existentes en el ámbito del reconocimiento de movimientos. Primero se hablará de los sistemas de reconocimiento en general, y se continuará con una descripción de distintas técnicas de monitorización y reconocimiento basadas en información 3D, más concretamente en modelos de esqueleto. Por último se hablará de los *datasets* existentes para usar en este tipo de sistemas.

2.1. Reconocimiento de movimientos

Existe una gran cantidad de variantes a la hora de presentar una solución para un problema de reconocimiento de movimientos. Si bien cada sistema tiene sus peculiaridades dependiendo de la aplicación concreta, todos ellos deben ser capaces de identificar movimientos independientemente del aspecto y fisiología del usuario final y, a ser posible, en un entorno no restrictivo y cambiante. La Figura 1 descompone este paradigma en distintos tipos de métodos.

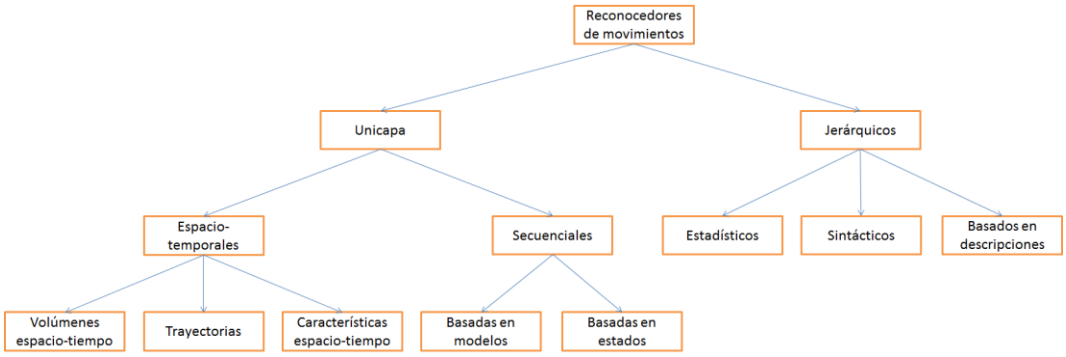

Figura 1: Diagrama sobre los diferentes métodos usados en reconocimiento de movimientos

A continuación se ofrece una breve explicación de cada uno.

2.1.1. Métodos mono-capa

Las acciones son reconocidas procesando los datos de video directamente, mientras que los sistemas jerárquicos analizan sub-acciones primitivas. Estas acciones primitivas son obtenidas a partir de videos o *datasets*; pueden usarse a continuación para identificar acciones más complejas (Sección 2.1.2). Pueden distinguirse dos clases de sistemas en este grupo: los espacio-temporales y los secuenciales. La principal diferencia es el uso de la dimensión temporal:

- Los sistemas espacio-temporales tratan el tiempo (T) como otra dimensión espacial (X-Y) y extraen características del volumen 3D (X-Y-T) que representa el video.
- Los sistemas secuenciales toman las acciones como un conjunto de observaciones ordenadas en el tiempo. Tienen en consideración relaciones secuenciales entre las características globales/locales de un conjunto de *frames*.

Los métodos espacio-temporales se descomponen asimismo en varios tipos:

- Volúmenes espacio-temporales: Las acciones desconocidas se comparan con las existentes usando el volumen 3D completo como plantilla. Sufren la presencia de ruido e información inservible de fondo, por lo que algunas implementaciones usan modelos de personas como siluetas o esqueletos.
- Trayectorias: Se monitorizan únicamente las posiciones de las articulaciones y otros puntos de interés del cuerpo humano. Las trayectorias son clasificadas representándolas en varias formas y usando los algoritmos correspondientes.
- Características espacio-temporales locales: Este método es heredado del reconocimiento de objetos en imágenes. Las características locales se refieren a la descripción de puntos y sus alrededores en el volumen de datos.

Los métodos secuenciales tienen dos subcategorías:

- Basados en modelos: Representan un movimiento como un modelo de secuencias de observaciones, y se centra en definir cómo un nuevo video de entrada puede ser comparado con ese modelo.
- Basadas en modelos de estados: Aprenden un modelo por cada acción, la cual se representa mediante una serie de estados ocultos. Las secuencias de observaciones son generadas y cada secuencia de observaciones es asociada con una instancia de la acción correspondiente. Los modelos ocultos de Markov (HMM) son muy usados.

2.1.2. Métodos jerárquicos

Su objetivo es reconocer eventos interesantes (actividades de alto nivel) basados en sub-acciones más simples, o de bajo nivel. Una acción de alto-nivel puede descomponerse en una secuencia de varias sub-acciones; por ejemplo, dar la mano puede integrarse como la secuencia de dos manos extendiéndose, uniéndose y separándose. Las sub-acciones pueden considerarse también como de alto nivel, hasta descomponer el movimiento en acciones atómicas. Esto tiene la ventaja de

poder modelar la estructura de acciones humanas complejas, incluyendo interacciones entre personas y acciones grupales. Existen tres categorías de métodos jerárquicos:

- Estadísticos: Una Red Bayesiana Dinámica (DBN) representa el estado de un sistema usando un conjunto de variables aleatorias. De esta forma, múltiples niveles de estados ocultos forman una representación jerárquica de una acción.
- Sintácticos: Integran acciones usando una cadena de símbolos, que en este contexto son las denominadas sub-acciones atómicas. Estas sub-acciones pueden reconocerse usando cualquiera de las técnicas descritas en esta Sección 2.1. Sin embargo, la representación en forma de cadena de símbolos tiene limitaciones:
    - No es posible el reconocimiento concurrente.
    - El usuario debe proveer un conjunto de reglas de producción.
- Basadas en descripciones: Pueden expresar explícitamente la estructura espacio-temporal de los movimientos. Pueden reconocerse movimientos secuenciales y concurrentes. Los movimientos son modelados como una ocurrencia de sub-acciones, que deben satisfacer las relaciones temporales, espaciales y lógicas especificadas, propias de la actividad de alto nivel.

2.2. Tecnologías de captura 3D

Sin lugar a dudas, la complejidad de los algoritmos subyacentes en los distintos sistemas puede llegar a depender en gran medida de la forma en que la información de la escena es adquirida. Dejaremos a un lado las técnicas en las que el seguimiento de la persona se realiza mediante marcadores adheridos en distintas partes del cuerpo humano y nos centraremos en sistemas no intrusivos en los que únicamente se emplean cámaras y visión artificial. Dentro de estos últimos podemos generalizar entre sistemas de seguimiento que emplean información 2D y aquellos que emplean información 3D (Hrabar, Corke & Bosse, 2009). En los sistemas 2D, se emplea un dispositivo de captura de imagen o cámara, bien en color RGB o bien en escala de grises. Las imágenes capturadas permiten analizar la variación del movimiento en un único plano de coordenadas frontal al dispositivo, y son afectadas por problemas como las oclusiones o el ruido de fondo. En los sistemas 3D, por contra, se añade información adicional de profundidad, a través de la estimación, para cada píxel de la imagen, de su distancia relativa a la cámara (Plagemann, Ganapathi, Koller & Thrun, 2010).

Existen estudios (Raptis & Kirovski, 2011) (Chen & Kotani, 2012) que confirman que el uso de información 3D resulta beneficioso sobre los métodos basados solamente en imágenes. Los datos capturados son más robustos ante la variabilidad asociada a la escala, punto de vista e iluminación. Además, extrayendo características locales espaciales de las secuencias generadas mediante los esqueletos detectados se consiguen ventajas notables: la manera en la que una acción es reconocida por el ser humano se ve favorecida mediante esta representación, ya que el efecto de las características personales se ve reducido, aislando la acción del usuario que la realiza.

Históricamente, el reconocimiento de movimiento se ha dividido en subtareas como el reconocimiento de gestos en interfaces (Erol, Bebis, Nicolescu, Boyle & Twombly, 2007) o expresiones faciales (Zhao, Chellappa, Phillips & Rosenfeld, 2003). En un sistema para la monitorización de movimientos del cuerpo humano, son necesarias las siguientes tareas:

- Adquisición de imagen (y profundidad en sistemas 3D).
- Segmentación y modelado cinemático del cuerpo humano para seguimiento.
- Segmentación y reconocimiento de acciones.

En los últimos años, la investigación en técnicas de reconocimiento se ha ido apoyando, en dispositivos de captura 3D como las cámaras binoculares (Hrabar, Corke & Bosse, 2009) (Elmezain, Al-Hamadi & Michaelis, 2008) o las ToF (Time of Flight) (Gallo, Manduchi & Rafii, 2008). Ambas tienen asociados inconvenientes; las primeras sufren de falta de precisión y sus algoritmos son costosos, y en el caso de las segundas, pese a ser muy precisas, la resolución que ofrecen es baja (176×44 px) y su precio es alto. Como alternativa, apareció 2010 la cámara basada en infrarrojos Kinect, con una resolución de 640×480 px a 30 fps, lo que la hace particularmente útil en aplicaciones de monitorización de movimientos. Este dispositivo cuenta con gran apoyo en diversas comunidades científicas y del propio Microsoft®, generando un nuevo sector de *consumer depth-cameras*.

Kinect facilita la tarea de adquisición de datos de imagen y profundidad, y de forma paralela se desarrollan que facilitan en gran medida el trabajo en las operaciones de segmentación y modelado. Entre estos últimos destacan los drivers no oficiales modificados de dispositivos compatibles PrimeSense, y los controladores oficiales para Kinect liberados a continuación por Microsoft®. Un detalle importante es que toda esta variedad permite a los desarrolladores

desplegar sus aplicaciones sobre distintos sistemas operativos, como puede verse en la Figura 2.

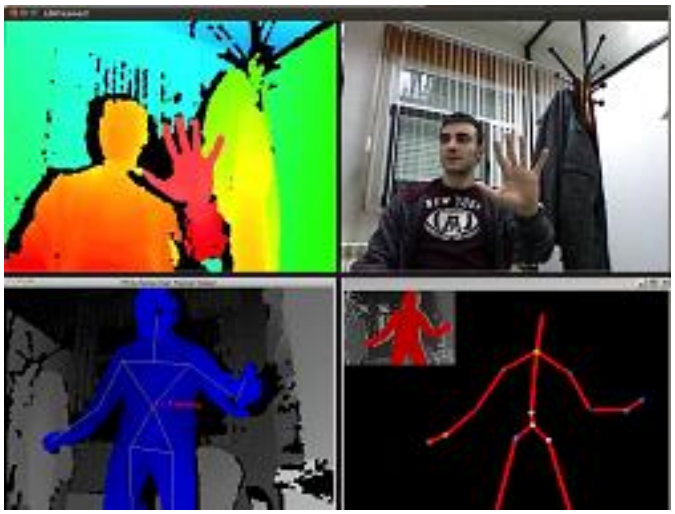

Figura 2: Arriba: Librería OpenKinect funcionando para captura de profundidad sobre Linux Ubuntu. Abajo izquierda: Modelo cinemático de OpenNI funcionando en MacOS. Abajo derecha: Modelo cinemático de Kinect SDK funcionando en Windows.

Adicionalmente, la comunidad de desarrolladores está trabajando en la integración de Kinect con otras herramientas ya existentes. En este sentido caben destacar algunos plugins que permiten la adaptación de Kinect para su utilización con MatLab® y LabView® (OpenKinect, 2011), así como la combinación de información 3D de profundidad con librerías de procesado de imagen de OpenCV.

2.3. Adquisición de la información 3D monitorizada

Dependiendo del nivel de la información representada y la utilidad, el modelado puede realizarse con modelos 2D y 3D (Weinland, Ronfard & Boyer, 2011). Los modelos 2D se basan principalmente en el etiquetado (labeling) y el uso de siluetas, mediante los que encontrar una serie de puntos característicos en una región de interés de la imagen para construir un esqueleto en dos dimensiones (Chen, Chou, Fookes & Sridharan, 2008). Aunque el trabajo con siluetas ofrece robustez frente a los cambios de color, textura y contraste (Aggarwal & Park, 2004), dependen de un procesado anterior de segmentación del fondo de la escena.

El mayor inconveniente que presenta el trabajo con imágenes en 2D es que el procesado es muy sensible a oclusiones. Sin embargo, mediante el uso de tecnología como Kinect, se aporta una capa de información de profundidad a las imágenes, diferenciándose los elementos según su proximidad al objetivo (Zhu,

Dariush & Fujimura, 2010). El sensor de Kinect permite distinguir información de profundidad con un margen de error de aproximadamente 1 centímetro y su altura y anchura con un margen de 3 milímetros. Adicionalmente, Kinect proporciona imágenes de 640x480 píxeles en color a 30 *frames* por segundo. La distancia de funcionamiento recomendada es entre 1 y 3.5 metros, por lo que es adecuada para monitorización de cuerpo entero. La información de profundidad proporcionada por Kinect facilita enormemente la substracción de fondo de la escena, al no requerirse técnicas tradicionales basadas en diferencias de color y textura. De esta forma se facilita la segmentación de la figura humana respecto al fondo. Esta segmentación es muy conveniente como paso previo al modelado del cuerpo humano, que tiene como objetivo obtener una representación espacial que discrimine el movimiento de los datos visuales, mediante características de postura y acción. Adicionalmente, la segmentación es robusta a variaciones en iluminación y gracias al infrarrojo es posible en escenarios sin ningún tipo de iluminación.

En los modelos 3D, para cada *frame* de vídeo se recupera la pose del sujeto en base a la disposición de una serie de puntos característicos del cuerpo humano. Un sistema de estas características suele requerir de un tiempo de calibración en el que el sujeto a monitorizar debe tener una postura inicial definida (Poppe, 2007), si bien esto no es necesario en las últimas librerías de Kinect. Una vez estimado el 'esqueleto' del sujeto, en base a la captura y un modelo cinemático (*kinematic model*) de los *joints* (Moeslund, Hilton & Krüger, 2006), puede realizarse un posterior seguimiento del movimiento de las trayectorias de los puntos representativos. Este tipo de modelados son más robustos a oclusiones parciales o totales del cuerpo humano. En el caso de Kinect, cada punto está marcado como "detectado" o "inferido", dándose este último caso cuando el puto ha sido estimado dada la oclusión de la parte del cuerpo correspondiente. Los puntos inferidos son inherentemente más ruidosos.

Las librerías libres y multiplataforma de OpenNI permiten la obtención de un esqueleto 3D anclado a las articulaciones del cuerpo en 24 puntos, de los cuales 15 están disponibles al trabajar con Kinect como receptor (situados en zonas como caderas, torso, cuello, cabeza, hombros, codo, manos, rodillas y pies). Una vez OpenNI haya detectado y segmentado el sujeto a analizar, se detectan los *joints* del esqueleto mencionados (Figura 2) y se encajan en la nube de puntos resultante. La detección del esqueleto de varios sujetos es posible mientras estos puedan convivir en la escena a una distancia suficiente. Las librerías de Kinect SDK de Microsoft® permiten la obtención de un esqueleto 3D anclado a las articulaciones del cuerpo en un total de 20 puntos. Respecto a los 15 *joints* detectados con

OpenNI empleando Kinect, se detectan 5 *joints* adicionales en muñecas, torso y tobillos. A diferencia de lo que OpenNI, Kinect SDK tiene la restricción de poder utilizarse solamente en entornos Windows (Martínez-Zarzuela, Díaz-Pernas, Tejero de Pablos, Perozo-Rondón, Antón-Rodríguez & González Ortega, 2011).

También han surgido herramientas que combinan el modelado de OpenNI con el reconocimiento de gestos del software NITE para el seguimiento de movimientos, como FAAST (Flexible Action and Articulated Skeleton Toolkit). FAAST ha sido desarrollado dentro de un proyecto de la University of Southern California y consiste en una herramienta para el desarrollo de aplicaciones que varían en el rango desde la rehabilitación hasta el control de juegos (Suma, Lange, Rizzo, Krum & Bolas, 2011). Este middleware facilita la integración de control de cuerpo completo con aplicaciones de realidad virtual y videojuegos usando sensores de profundidad compatibles con OpenNI (actualmente el PrimeSensor y el Microsoft® Kinect). FAAST hace de interfaz para obtener la información de la pose de la persona situada enfrente de la cámara y adicionalmente realiza reconocimiento de alto nivel de gestos básicos para generar eventos basados en las acciones del usuario (control de avatares, acciones de ratón y teclado, etc.). Un ejemplo del uso de FAAST se muestra en la Figura 3.

Figura 3: Ejemplo de funcionamiento del middleware FAAST

A partir de la información de profundidad, puede realizarse un reconocimiento preliminar sobre las posiciones del usuario. Existen trabajos, como en (Xiao, Mengyin, Yi & Ningyi, 2012), sobre reconocimiento de poses en 3D mediante Kinect como dispositivo de captura. En este estudio se monitoriza la persona mediante información y profundidad y se representa en 3D, lo que permite estudiar el movimiento humano. El movimiento capturado es renderizado y representado en base a 20 *joints*, estimados mediante un bosque de decisión. Mediante esto, se consigue mapear secuencias con movimientos como bailar,

correr, y diferentes gestos. Otro trabajo similar se encuentra en (Shotton, Sharp, Kipman, Fitzgibbon, Finocchio, Blake & Moore, 2013), en el que intentan predecir posiciones de *joints* a partir de una imagen de profundidad estática, es decir, sin información temporal. Se busca invarianza a la posición, forma corporal, ropa, etc. del usuario. A continuación se usan de técnicas de reconocimiento de formas, mediante clasificación píxel a píxel, para detectar las zonas del cuerpo. A partir de esta información se estima la posición de los *joints*. Para ello se crea un bosque de decisión. El clasificador puede ejecutarse en paralelo en la GPU para acelerar el sistema y conseguir tiempo real. Para su evaluación, se utilizan datos reales y sintéticos para medir la precisión de la detección de *joints*. Destacan así mismo la importancia de descomponer el esqueleto en distintas partes. La ejecución en GPU permite una aceleración de la ejecución de aproximadamente ×4.

Este procesamiento es necesario para un reconocimiento inicial de las poses. Sin embargo, para realizar reconocimiento de movimientos es necesario construir un modelo cinemático del cuerpo humano y seguir su evolución a lo largo de los *frames*.

2.4. Modelado cinemático del cuerpo humano

Existen investigaciones recientes que contemplan el uso de este esquema basado en esqueletos 3D para reconocer movimientos. En (Sheikh & Sheikh, 2005) se describe la principal dificultad de clasificar acciones viene dada por la variabilidad asociada a la captura mediante una cámara de una persona ejecutando una acción, teniendo en cuenta que el cuerpo humano tiene 244 grados de libertad. Se identifican tres fuentes de variabilidad:

- Punto de vista: Dependiendo de la posición de la cámara con respecto al usuario.
- Tasa de ejecución: Los movimientos naturales son raramente ejecutados a una frecuencia precisa. Los algoritmos deberían ser por tanto robustos a las variaciones temporales, debidas al actor o a la tasa de *frames* del sistema de captura. Un método ampliamente usado en estos casos es *Dynamic Time Warping*, también llamado DTW (ver Sección 3.3.1).
- Antropometría de los actores: El sistema pueda ser usado por gente diferente.

Para intentar reducir esta variabilidad, se representa como una combinación lineal en el dominio espacio-temporal de 13 puntos anatómicos.

En (Raptis & Kirovski, 2011) se propone una metodología para trabajar con el sensor Kinect. Los autores buscan un modelado de forma que el reconocimiento sea más efectivo, en forma de vector de características que siga una distribución tal que pueda aplicarse a distintos clasificadores. Las premisas son:

- Reducción de la redundancia: Expresar *joints* relativos a sus nodos padre. Los autores describen un ejemplo de esta representación en la Figura 4.
- Robustez: Gestionar los errores de los sensores de profundidad y *joints* inferidos.
- Invariancia a la orientación del sensor.
- Continuidad y estabilidad de la señal: Orientar los ejes de coordenadas usados para computar las posiciones relativas.
- Reducción de la dimensión: Reducir la dimensión del espacio de búsqueda para la clasificación, al mismo tiempo que se retiene el carácter del movimiento. Al contrario de otros sistemas como los centrados en animación, los autores no buscan calcular atributos mediante un proceso invertible, sino simplicidad y mejorar el rendimiento.

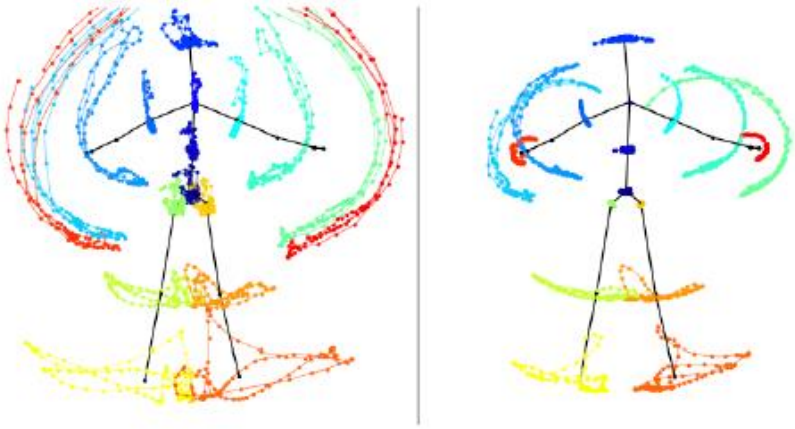

Figura 4: Movimiento absoluto de *joints* VS movimiento relativo de *joints*

Con estas premisas, realizan una distribución del esqueleto en tres clases:

- Torso: Los puntos del torso son considerados como un único bloque, ya que raramente presentan movimientos independientes entre sí. Una vez se estima su posición, es usado para representar el resto de atributos de manera relativa.
- *Joints* de primer orden: Son aquellos adyacentes al torso: codos, rodillas y cabeza. Se representan de forma relativa al *joint* adyacente en el del torso,

en un sistema de coordenadas esférico basado en la inclinación y el acimut.

- *Joints* de segundo orden: Son los extremos del esqueleto: manos y pies. Se representan en el sistema de coordenadas esférico del *joint* de primer orden adyacente (inclinación y acimut).

Este trabajo muestra este tipo de representación en la Figura 5.

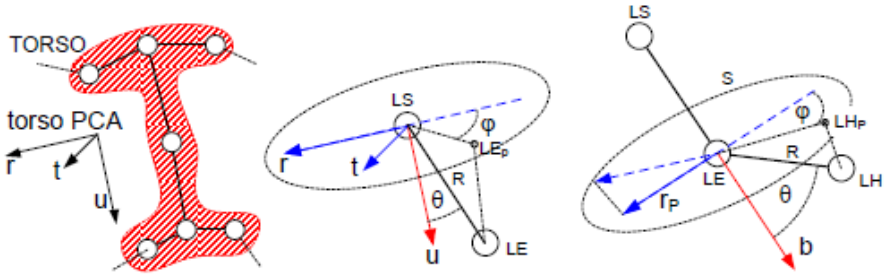

Figura 5: Sistema de coordenadas para el torso y los *joints* de 1er y 2o orden

De esta forma consiguen convertir 45 señales diferentes (3 coordenadas × 15 *joints*) en 19 (2 ángulos × 8 *joints* + 3 yaw-pitch-roll × 1 torso). Pese a la mejor de eficiencia, los autores remarcan la dificultad de definir unas relaciones geométricas robustas y discriminantes para movimientos.

Asimismo, tomando esqueletos como datos de entrada, en (Chen & Kotani, 2012) se propone un enfoque diferente para modelar patrones a partir de una secuencia de movimiento que permitan su reconocimiento. Uno de los problemas que se destaca es las elevadas dimensiones de los datos observados, que esconde las características claves del movimiento. Por ello es necesaria una fase de pre-procesamiento, en la que se reduzca la dimensión de las características. También se remarcan las ventajas de usar características locales de un modelo de esqueleto:

- Apenas se va afectado por atributos personales del usuario, por lo que describe de una forma más "pura" la acción.
- Gracias a la significante mejora de los sistemas de extracción de esqueleto, la información de este tipo es fiable y fácilmente obtenible.

Con estas premisas, los esqueletos 3D son convertidos en histogramas, que son comparables, y estos se usan para resumir los videos 3D en secuencias de *frames*-clave. Normalmente la técnica de "histogramas de forma" se usa en mallados 3D, sin embargo al aplicarlo sobre el esqueleto se eliminan las diferencias debido a la ropa o a la forma del cuerpo, quedándose sólo con las posiciones. Los *frames* clave se hallan mediante la técnica de camino más corto, se extraen de la

secuencia y se agrupan en caso de representar la misma acción, etiquetándose en forma de patrones.

Un patrón construido de esta forma permite una discriminación eficaz si:

- Aparece frecuentemente en el movimiento destino, para ser un patrón que lo represente fielmente.
- Aparece mucho más a menudo en un movimiento que en el resto, por lo que podemos usarlo como pista para identificar la acción.

Los autores representan un histograma tal y como aparece en la Figura 6.

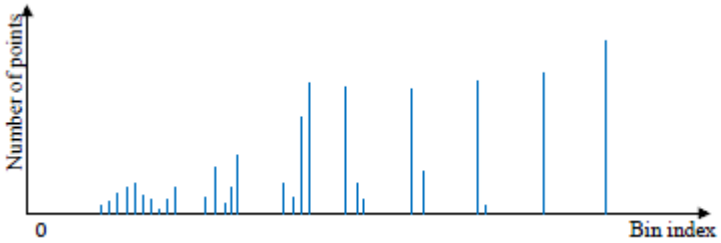

Figura 6: Formación de patrones a partir de un histograma 3D

Los esqueletos 3D son convertidos en histogramas que resumen los videos 3D en una secuencia de *frames*, los cuales se etiquetan como patrones diferentes.

En (Sempena, Maulidevi & Aryan, 2011) se presenta un sistema uni-capa secuencial basado en modelos mediante DTW. Justifican el uso de información de profundidad en 3D para mejorar la tasa de reconocimiento. Construyen su vector de características a partir de la orientación de los *joints* a lo largo de series temporales invariantes al tamaño del cuerpo humano. Utilizan la orientación, en forma de cuaterniones, ya que determinan que es invariante al tamaño del cuerpo humano. Dado que un cuaternión cuenta con 4 componentes, y se monitorizan 15 *joints*, el vector de características se compone de 60 elementos.

En (Chua, Leman & Pham, 2011) se comenta la dificultad inherente en el reconocimiento de movimientos y a la hora de encontrar una representación óptima para dichos movimientos. En este caso, los autores trabajan con videos 2D, pero su metodología está directamente relacionada con el caso de este TFM. Dado que existen varios atributos que caracterizan un movimiento, distintas entradas al clasificador pueden ofrecer información complementaria sobre los patrones a reconocer. Del video se extraen dos tipos distintos de características: forma y movimiento, que se corresponden con los atributos de posición y

velocidad respectivamente. A continuación se representan mediante histogramas, de forma que el movimiento es invariante a la dirección en el eje x. Además, los patrones son calculados mediante la superposición de *frames*. Este modelado puede verse en la Figura 7.

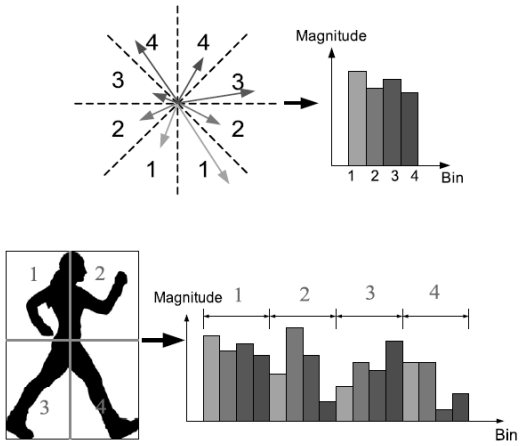

Figura 7: Formación de patrones a partir de la magnitud en cada región del *frame*

El reconocimiento de gestos es esencial para la interacción hombre-máquina. En (Biswas & Basu, 2011) presentan un método de este tipo usando un sensor Kinect. Primero se procede a la substracción del fondo mediante información de profundidad. A continuación se calcula la posición de la mano con respecto al resto del cuerpo, y se representa en un histograma ecualizado según la información de profundidad. El perfil del movimiento es extraído mediante diferencias en la información de profundidad entre dos *frames* consecutivos mediante substracción.

En (Lai, Konrad & Ishwar, 2012) se presenta un sistema de reconocimiento de gestos con las manos. Proponen un sistema computacionalmente sencillo para realizar una interfaz en tiempo real. Destacan que los movimientos usados en este tipo de interfaces no son periódicos, por lo que es necesaria una segmentación desde que el movimiento comienza hasta que termina. En este trabajo, el usuario es avisado para que comience a moverse, y cuando su posición supere un umbral se comienza a monitorizar. Los vectores de características son modelados a partir de los esqueletos ofrecidos con Kinect. Dado que se busca reconocer gestos con las manos, se utilizan sólamente los *joints* asociados a las manos y codos derechos e izquierdos. Estos cuatro *joints* son rectificados en x-y-z restando el *joint* tronco, y normalizados por la distancia vertical entre el cuello y el tronco. Mediante esto se forma un vector de 13 elementos (4 *joints* × 3 coordenadas + 1 timestamp) por cada *frame*, en movimientos de 30 *frames* de duración.

2.5. Reconocimiento de movimientos mediante información de esqueletos 3D

Tal y como se indica en (Sheikh & Sheikh, 2005), el reconocimiento de movimientos implica definir un tipo de agrupamiento y qué hace a un movimiento pertenecer a ese grupo, con el objetivo de encontrar la correspondencia entre una instancia a una acción.

Los patrones construidos en (Raptis & Kirovski, 2011) de la sección anterior son introducidos a un módulo de clasificación, que se comenta a continuación. Ya que el sistema que desarrollan es usado para detectar movimientos de baile, la suposición principal es que la entrada sigue un cierto ritmo periódico. La ventaja de esto es que puede ignorarse el tiempo de entrada y re-muestrear los patrones, en este caso, a una ventana de 8 ritmos.

En el sistema se tiene una serie de instancias de sujetos realizando el baile y un oráculo, o *Ground Truth* de referencia, realizado por un profesional. Para cada atributo, se calcula la trayectoria media de un usuario mediante la correlación cruzada normalizada circular entre el oráculo y cada sujeto. El resultado es, para un gesto específico y un atributo, el modelo genera una tripla con la trayectoria media, el histograma de correlación y el histograma de nivel de energía relativo al oráculo. Para la clasificación, se calcula la representación de atributos de la entrada y se compara con el modelo, obteniendo una puntuación. Estas puntuaciones sirven para hacer un ranking con todas las clases. Con las dos de mayor puntuación se lleva a cabo una clasificación binaria avanzada recursiva, para obtener la decisión final. Además, la señal no sólo se compara con el prototipo medio, sino también con el oráculo ideal.

En el sistema propuesto por (Chen & Kotani, 2012) cada patrón de *frames*-clave es tratado como un documento en los que las palabras son movimientos, y el concepto de clasificación de texto es usado para clasificar una serie de acciones heterogéneas (p. ej. “andar”, “correr”, “boxear”). Se define un peso para evaluar la confianza de que una secuencia sea un movimiento determinado dependiendo de que contenga o no uno u otro de los patrones con cierta frecuencia. El valor de confianza se calcula sumando los pesos de todos los patrones de discriminación. Sin embargo, esta técnica de clasificación no tiene en cuenta la mayor parte de la estructura espacial de los datos.

Para el caso de (Sempena, Maulidevi & Aryan, 2011), se aplica DTW al vector de características resultado, para intentar reconocer una serie de acciones. Este

método es escogido dada su robustez ante la variación en velocidad o estilo al realizar una acción. La puntuación de decisión final se calcula sumando las distancias obtenidas al comparar pares de cuaterniones.

En (Chua, Leman & Pham, 2011), se calculan distancias de Hellinger entre instancias con K-Vecinos Cercanos (K-NN) como clasificador. El resultado de estos clasificadores, también llamado votos suaves, se combina utilizando suma-fuzzy (Figura 8).

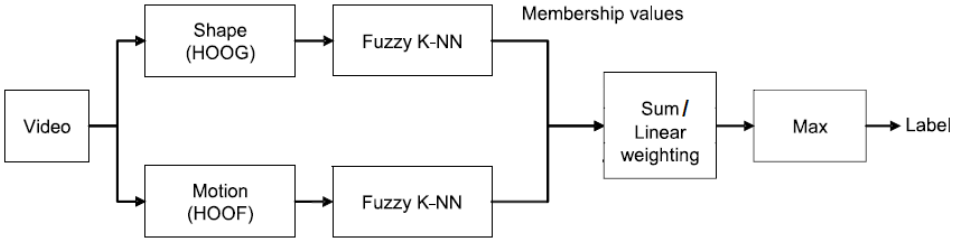

Figura 8: Sistema de reconocimiento con atributos de forma y movimiento

En (Biswas & Basu, 2011) el subsistema de clasificación es un SVM multiclase, en el que cada fila de los datos de entrada es un *frame*, y cada columna es un atributo (10 bins × 196 celdas).

En el trabajo de (Lai, Konrad & Ishwar, 2012) mencionado en la sección anterior, se extraen los parámetros en una "bolsa de parámetros", se calcula su matriz de covarianza empírica, y se transforma al espacio euclídeo tomando el logaritmo de la matriz. Dada esta representación, se calcula K-NN basado en distancia Euclídea.

2.6. Base de datos de movimientos con información de profundidad

Para desarrollar un sistema de reconocimiento es necesario tener un conjunto de movimientos que aprender y clasificar. Esta sección describe brevemente algunos de los *datasets* existentes que usan información de profundidad y se explica la construcción de una base de datos de movimientos propia generada usando una cámara comercial de profundidad.

Tal y como se describe en (Chaquet, Carmona & Fernández-Caballero, 2013), los *datasets* se clasifican de acuerdo a distintos criterios como la complejidad de las acciones, el tipo de problema que abordan o la fuente. Se denominan gestos heterogéneos a aquellos que representan acciones naturales y realistas (p. ej. saltar, andar, saludar…).

Una de las bases de datos más utilizadas para evaluar algoritmos de seguimiento y estimación de movimiento es HumanEva (Sigal, Balan & Black, 2010) (Weinland, Ronfard & Boyer, 2011). En esta base de datos se emplea un sistema de adquisición con marcadores y los vídeos son capturados empleando varias cámaras de alta velocidad. El *dataset* HumanEva-I consiste en 4 sujetos realizando 6 acciones predefinidas un cierto número de veces. Las acciones son: andar, correr despacio (jogging), gesto ("hola" y "adios"), lanzar/coger (diferentes estilos), boxeo y combo (secuencia de varios movimientos). En lugar de información de profundidad, esta base de datos se basa en el uso de marcadores.

Por otro lado, las bases de datos adquiridas con cámaras en tiempo de vuelo sí que incluyen información de profundidad. En la base de datos ARTTS 3D-TOF (ARTTS, 2011), se proporcionan varios *datasets* orientados a evaluar detección de caras, orientación de la cabeza y reconocimiento de movimientos. En este último *dataset* se incluyen 9 acciones diferentes como "empujar" (push) y "redimensionar" (resize), realizadas por diferentes sujetos.

Schwartz et al. emplean una base de datos adquirida con cámaras ToF (Schwarz, Mateus, Castañeda & Navab, 2010). Esta base de datos contiene 10 actividades diferentes realizadas por 10 sujetos. Cada uno de los 6 movimientos fue grabado seis veces. Los movimientos son: clapping, golfing, hurrah, jumping jack, knee, bends, picking something up, punching, scratching head, playing the violin y waving.

Ganapathi et al. construyeron una base de datos compuesta de 28 secuencias con complejidad creciente (Ganapathi, Plagemann, Koller & Thrun, 2010). Los movimientos fueron capturados en situaciones reales con una única cámara ToF y generaron un ground truth mediante el empleo de marcadores 3D comercial. Las secuencias disponibles incluyen desde movimientos breves con una sola extremidad hasta secuencias más largas en las que participan varias partes del cuerpo humano, como fast kicks y swings, y en las que va aumentando la complejidad al introducir auto oclusiones y rotaciones de todo el cuerpo.

Si bien el estudio de estas bases de datos es adecuado para conocer cuáles son los movimientos más comúnmente empleados para evaluar los sistemas de reconocimiento, la utilización de las mismas para evaluar un sistema de monitorización que emplee cámaras de tipo Kinect no es posible. Las bases de datos adquiridas con cámaras ToF presentan el inconveniente de que la resolución de las imágenes es muy baja, debido a la limitación del dispositivo de captura.

Adicionalmente, en estas bases de datos no se proporciona información de imagen en RGB.

Además de las cámaras ToF, como ya hemos venido hablando en este Trabajo de Fin de Máster, otra manera de obtener información 3D es capturando video a través de un dispositivo Kinect comercial, el cual ofrece además una mejor resolución, aunque en este caso los *dataset* son muy escasos. En (Ni, Wang & Moulin, 2011) se ofrece un *dataset* RGBD capturado mediante Kinect, que contiene acciones específicas de la vida cotidiana. Los movimientos del *dataset* son los siguientes:

- Estar ocioso, sin hacer nada
- Ponerse la chaqueta
- Ponerse la chaqueta (repetición)
- Salir de la habitación por la puerta
- Sentarse
- Beber de un vaso sentado
- Entrar en la habitación por la puerta
- Comer de un plato sentado
- Quitarse la chaqueta
- Fregar el suelo
- Recibir una llamada (móvil)
- Levantarse e irse
- Irse a la cama a dormir
- Despertarse y levantarse de la cama

Otra base de datos de movimientos capturados con Kinect es la de (Fanelli, Weise, Gall & Van Gool, 2011), encaminada a calcular la orientación de la cabeza. Sin embargo, con lo mejor de nuestro conocimiento, no fue posible encontrar al comienzo de este trabajo un *dataset* basado en Kinect que contenga movimientos heterogéneos de cuerpo completo como correr, saltar, etc. Por otra parte, otras implementaciones que usan esqueletos 3D, como algunas de las mencionadas anteriormente, también son evaluadas usando este tipo de movimientos.

2.7. Computación en GPU

La carga de trabajo asociada a este tipo de sistemas es muy elevada. Estos sistemas tienen que tratar con gran cantidad de información: la imagen grabada, los movimientos capturados, etc. En particular, las tareas de reconocimiento son especialmente costosas. Si además se requiere que el sistema funcione a Tiempo Real, se necesita una capacidad de procesamiento superior. El uso de Tiempo Real

en este tipo de sistemas es necesario en los casos en los que los movimientos puedan ocurrir de manera espontánea, como es el caso de los accidentes.

En los últimos años, con el auge de la industria de los videojuegos, la capacidad de los procesadores tradicionales ha sido ampliamente superada por las tarjetas gráficas. El núcleo de estas tarjetas gráficas, llamado GPU (Graphic Processing Unit) es un procesador con una arquitectura altamente paralela, con una gran cantidad de unidades de procesamiento capaces de realizar operaciones gráficas sobre muchos datos simultáneamente. Esto se debe a que muchas aplicaciones gráficas conllevan un alto grado de operaciones simultáneas.

Además, las últimas series de tarjetas gráficas permiten un estilo de programación basado no únicamente en elementos gráficos como vértices y píxeles, sino orientado a la programación en paralelo de propósito general o GPGPU (General-Purpose computing on Graphics Processing Units), lo que abre las puertas a la implementación de un mayor número de algoritmos. Más específicamente, la GPU se ajusta especialmente bien a problemas que pueden ser expresados como computaciones en paralelo. Muchas aplicaciones que procesan grandes conjuntos de datos como arrays pueden usar un modelo de programación en paralelo para acelerar los cálculos.

De hecho, en la actualidad muchos algoritmos fuera del campo de investigación del procesamiento de imagen son acelerados por procesamiento de datos en paralelo, desde el procesamiento general de señales o simulaciones físicas, hasta finanzas, administración pública, biología o sistemas de control en plantas industriales, permitiendo así operaciones que antes eran imposibles. En concreto, los estudios que se han llevado a cabo sobre la aplicación de técnicas GPGPU en sistemas de reconocimiento han sido altamente exitosas, consiguiendo acelerar los tiempos de ejecución varias decenas de veces (Martínez-Zarzuela, Díaz-Pernas, Tejero de Pablos, Antón-Rodríguez, Díez Higuera, Boto Giralda & González Ortega, 2009).

Sin embargo, estas técnicas requieren traducir los algoritmos convencionales a operaciones en paralelo y de realizar sus respectivas implementaciones usando los lenguajes adecuados (Martínez-Zarzuela, Díaz-Pernas, Tejero de Pablos, Perozo-Rondón, Antón-Rodríguez & González Ortega, 2011). Por ello este proyecto incluye la aplicación de técnicas de paralelización sobre los algoritmos usados en el sistema de reconocimiento para poder ejecutarlos sobre una GPU adecuada y de esta forma mejorar su rendimiento y facilitar su ejecución en Tiempo Real.

Para este propósito, al mismo tiempo aparecen lenguajes de programación sobre GPU adaptados a este nuevo hardware en paralelo, como es CUDA (Compute Unified Device Architecture).

2.7.1. CUDA

CUDA es un lenguaje de programación y una arquitectura desarrollada por NVIDIA para sus GPUs. Este sistema permite la ejecución en paralelo del código de las aplicaciones, lo cual permite que se procese gran cantidad de información al mismo tiempo. Conocida la arquitectura y recursos disponibles de la máquina, CUDA permite aprovechar todo esto para acelerar la ejecución de los programas. CUDA tiene una gran comunidad de desarrolladores a nivel internacional, lo que la convierte en una plataforma confiable para desarrollar en ella.

Existen trabajos en los que se hace uso de CUDA para tareas relacionadas con la monitorización y reconocimiento de movimientos del cuerpo, como el caso de (Zhang, Seah, Quah & Sun, 2013). El estudio consiste en programar un kernel en GPU capaz de monitorizar de forma precisa (comparable a sistemas con marcadores) movimientos de cuerpo completo en 3D en tiempo real. El algoritmo que mapea multi-vistas 2D a un espacio 3D es paralelizable, y permite conseguir 9 fps en GPU.

Capítulo 3: Planteamiento del estudio

3.1. Presentación del problema

La monitorización de entornos y, más concretamente, de individuos utilizando cámaras 3D es un problema al cual aún no se ha dado una solución definitiva, dado que la tecnología y las necesidades cambian a medida que progresan las investigaciones. El uso de nueva información y su procesamiento permite mejorar el rendimiento de los sistemas actuales, con el objetivo de paliar las limitaciones de estos. En la siguiente subsección, se busca analizar el problema desde su raíz, para decidir cómo abordar el problema y poder definir más delante una serie de objetivos y metodología para ello.

3.1.1. Cómo perciben los humanos los movimientos

La percepción del movimiento es el proceso de inferir la velocidad y dirección de elementos en una escena basándose en entradas visuales, vestibulares y propioceptivas (Cavanagh & Mather, 1989):

- Entradas visuales se refieren a las que percibe el ser humano procesando la información contenida en la luz visible, como es el cambio de posición, etc.
- El sistema vestibular, es un sistema biológico situado en el oído interno que contribuye a la percepción del movimiento, equilibrio y sentido de la orientación espacial
- La propiocepción (Proske & Gandevia, 2009) es el sentido que informa al organismo de la posición de los músculos, es la capacidad de sentir la posición relativa de partes corporales contiguas.

La zona del cerebro encargada de esta tarea viene dada por el flujo dorsal (Figura 9).

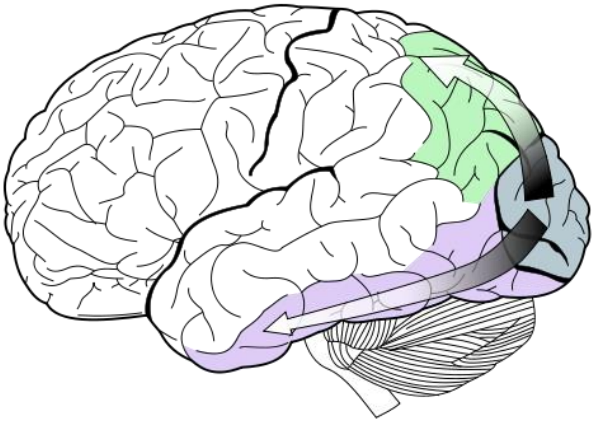

Figura 9: El flujo dorsal (verde) es el encargado de detectar posición y movimiento

La percepción del movimiento es estudiada en muchas disciplinas, incluyendo psicología, neurología, neurofisiología y, como en este caso, ingeniería. A continuación se describen los tipos de percepción existente, así como la importancia de la profundidad a la hora de percibir el movimiento.

Percepción de movimiento de primer orden

Dos o más estímulos que son activados de forma alterna pueden producir dos percepciones de movimiento diferentes (Ball & Sekuler, 1982). Esto se produce por ejemplo al presentar una serie de imágenes estacionarias, o *frames*, como en un video; a mayor velocidad de alternancia, la percepción de movimiento es más clara. Un ejemplo es el que se muestra en la Figura 10, donde el punto iluminado varía con cierta frecuencia. Este fenómeno es un ejemplo de detección de movimiento pura, o de primer-orden, y es llevada a cabo por sensores de movimiento en el sistema visual.

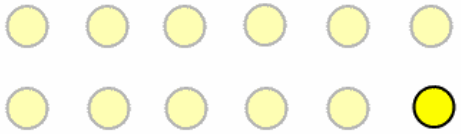

Figura 10: Ejemplo de percepción de movimiento de primer orden

Percepción de movimiento de segundo orden

En este otro caso, el contorno del movimiento lo determinan factores como el contraste, textura o parpadeo. Estos mecanismos tienen una peor resolución temporal y por si solos presentan un efecto de movimiento más débil.

Movimiento en profundidad

Como en otros aspectos de la visión, la entrada visual del observador es generalmente insuficiente para determinar la verdadera naturaleza de las fuentes del estímulo. En la visión monocular, por ejemplo, la entrada visual será una proyección 2D de una escena en 3D. La información presente en la proyección plana es insuficiente para reconstruir el movimiento presente en una escena en tridimensional. Por decirlo de otra forma, existen múltiples escenas en 3D compatibles con una única proyección en 2D.

3.1.2. ¿Puede entrenarse a un humano para aprender movimientos?

Existe una discapacidad para percibir el movimiento (Hadad, Maurer & Lewis, 2001), llamada acinetopsia, causada por una lesión en el área cortical del cerebro. La detección y discriminación de movimiento puede ser mejorada mediante entrenamiento. Los pacientes que son entrenados para detectar movimientos de una serie de puntos en una dirección mejoran a la hora de detectar movimientos en direcciones próximas a las que han sido entrenados.

3.1.3. Fundamentos del trabajo

Para nuestro sistema de reconocimiento de movimientos, se desea basar el estudio del problema desde su raíz. El sistema a desarrollar se plantea con una serie de fundamentos:

- Se busca representar el movimiento con características propias e intrínsecas, las llamadas "de primer orden" en la descripción anterior.
- Se desea analizar las diferencias existentes entre distintos movimientos, es decir, ver qué hace un movimiento distinto de otro, en términos de percepción de movimiento.
- Al mismo tiempo, se busca seguir las recomendaciones de los trabajos del estado del arte estudiados, y tener en cuenta ciertos aspectos de su metodología.
- Asimismo, se intenta diseñar un sistema que permita recuperar la señal original a partir de la procesada, procurando conservar los atributos del movimiento original en la medida de lo posible. Esto permitiría usar la información para un posterior procesamiento o reconstruir lo ocurrido en la escena.

Para conseguirlo, se utiliza monitorización de individuos y captura de información en 3D. Una vez capturado el movimiento, se representa en base a las trayectorias

de los *joints* del modelo de esqueleto a lo largo de los *frames* de duración. Además se toman las acciones completas como atómicas, sin dividirlas en subacciones más sencillas. En la siguiente sección se detallan los objetivos del proyecto.

3.2. Objetivos

El objetivo global que persigue el trabajo puede expresarse a través de la pregunta de investigación que se intenta resolver:

¿Es posible aplicar tecnología moderna y nuevas técnicas para mejorar el rendimiento de los sistemas de monitorización 3D y reconocimiento de movimientos?

De esta forma, pueden destacarse una serie de objetivos parciales derivados de este objetivo general, los cuales se presentan a continuación.

3.2.1. Monitorización de individuos

En este proyecto se plantea un sistema que contenga una representación 3D de un espacio monitorizado, sobre el cual se estudiará el comportamiento de los sujetos presentes en él. El sistema podrá ser entrenado para reconocer situaciones potencialmente peligrosas como caídas, ataques, y reaccionar ante ellos.

3.2.2. Reconocimiento de movimientos

Los movimientos detectados durante la monitorización serán introducidos dentro de un sistema inteligente para determinar en tiempo real qué acciones se están realizando. Las acciones capturadas a lo largo del tiempo son tratadas como una señal de movimiento. Estas señales requieren aplicar técnicas de procesado de señal para poder trabajar con ellas de la forma más eficiente posible. Por lo tanto, es necesario analizar el efecto de distintos métodos de procesado de señal. Este proyecto incluye la evaluación de un conjunto de técnicas de procesado de señal (métodos estadísticos, transformadas, etc.).

Otra problemática asociada a la monitorización y reconocimiento de acciones es la gran cantidad de información asociada a un movimiento. Para solucionar esto, es necesario aplicar técnicas de reducción de la dimensión y selección de atributos. Por ello, de la misma forma se estudiarán técnicas de selección de atributos para mejorar el rendimiento del sistema.

Asimismo, una vez el movimiento es procesado, se introduce a una etapa de clasificación que determina qué clase de acción se está realizando. Las tareas de reconocimiento automático requieren utilizar un sistema inteligente adecuado al

problema que se intenta resolver. Para tal efecto, se estudiará el uso de varios tipos de sistemas inteligentes, como redes neuronales o algoritmos de vecinos cercanos, de forma que se concluya cuál es el más eficiente para el reconocimiento de acciones en tiempo real.

Una vez ha sido clasificado, puede llegar a realizarse un análisis posterior para determinar la calidad del movimiento: calidad, estado de ánimo, evolución desde la última ejecución, etc. Esto es especialmente útil en tareas de rehabilitación y trabajo con pacientes.

3.2.3. Base de datos de movimiento

Tras el análisis de las bases de datos de movimiento existentes, se consideró necesario construir una que considerase las posibilidades actuales en cuanto a tecnología de captura 3D, para evaluar el sistema de reconocimiento propuesto y compararlo con otros sistemas del estado del arte. La base de datos debe incluir movimientos genéricos, realizados por personas con diferentes fisionomías y que incluyesen la participación del cuerpo entero y no solamente partes específicas, de forma análoga a la base de datos (Sigal, Balan & Black, 2010).

Esta contribución no sólo resulta útil para la construcción de nuestro sistema, sino que los investigadores que estén realizando estudios similares podrían aprovecharse de la existencia de un *dataset* de acciones variadas que contiene información de profundidad capturado mediante tecnología Kinect.

3.3. Metodología

Una vez realizado el estudio previo, se propone un sistema basado en las utilidades de monitorización 3D y detección de esqueleto de Kinect con OpenNI. El sistema permite analizar los movimientos almacenados en los archivos de la base de datos, pero también secuencias de movimiento capturadas y analizadas online. La Figura 11 muestra las etapas involucradas en el reconocimiento de acciones en el sistema propuesto, las cuales se detallan a continuación.

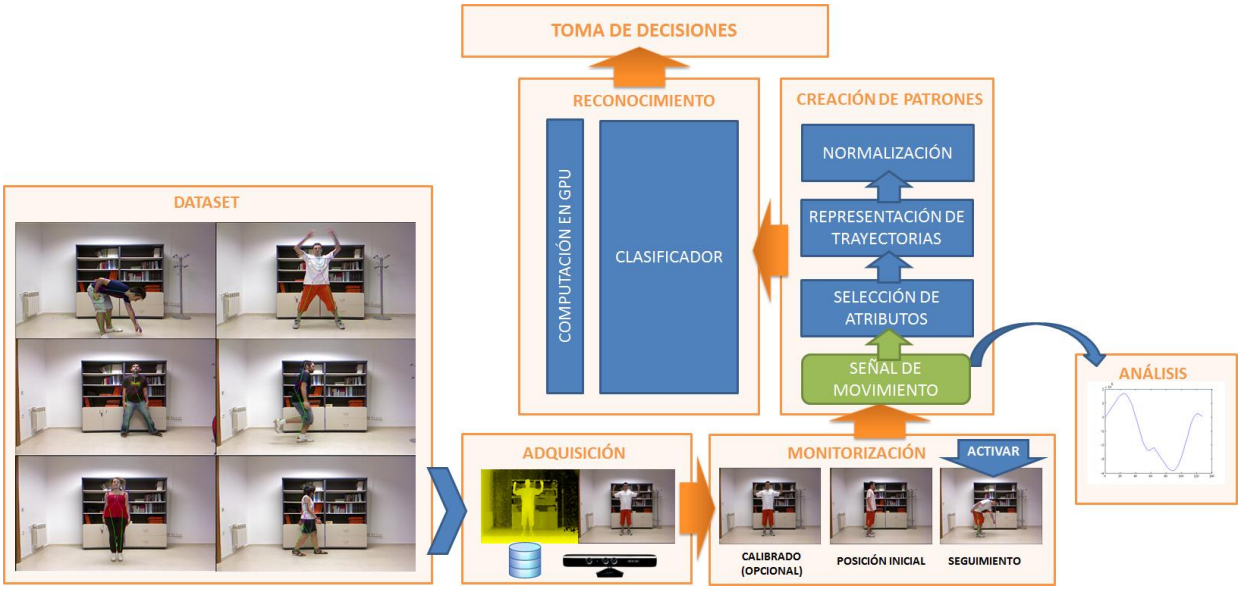

Figura 11: Esquema del sistema propuesto

3.3.1. Características a analizar

Monitorización y adquisición de movimientos

El sistema monitoriza la posición del esqueleto del usuario a lo largo de una secuencia de *frames*. El modelo del esqueleto usado se compone de 15 articulaciones, o *joints*, numeradas como aparece en la Figura 12. Una vez que la información postural a lo largo de todos los cuadros de una secuencia es obtenida vía OpenNI, es necesario descomponerla en información sobre movimiento representativa de la acción realizada (Guo, Xu & Tsuji, 1994). Las coordenadas de un determinado *joint* se expresan en un sistema de coordenadas cuyo origen se sitúa en el centro de la imagen y en el que en el eje z representa la distancia respecto a la cámara. Las posiciones relativas entre los distintos puntos de interés se miden en milímetros.

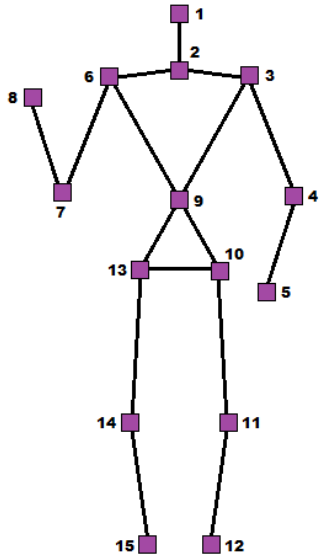

Figura 12: Modelo usado en nuestro sistema

Por lo tanto, el resultado de la monitorización de la escena durante un periodo de t cuadros, es una matriz de movimiento formada por las trayectorias de cada *joint j*: $\vec{p}_j(t) = (x_j, y_j, z_j),\ j \in [1,15]$.

Análisis de las señales de movimiento

Antes de modificar de alguna forma las señales de movimiento obtenidas, se desea analizar las características que poseen, para así poder determinar más robustamente qué técnicas aplicar para su reconocimiento. A continuación se muestran las técnicas a utilizar.

Diagrama de bigotes

Un diagrama de caja o de bigotes es un gráfico basado en cuartiles, mediante el cual se visualiza la distribución de un conjunto de datos en base a sus valores numéricos (McGill, Tukey & Larsen, 1978). Está compuesto por un rectángulo, la "caja", y dos brazos, los "bigotes", de forma que se representa la variabilidad respecto a los cuartiles superior e inferior (Figura 13). Pertenece a las herramientas de la estadística descriptiva.

Su utilidad radica en:

- Proporcionan una visión general de la simetría de la distribución de los datos; si la mediana no está en el centro del rectángulo, la distribución no es simétrica.

- Son útiles para ver la presencia de valores atípicos, que en ocasiones se representan en forma de puntos individuales.
- Permite ver como es la dispersión de los puntos con la mediana, los percentiles 25 y 75 y los valores máximos y mínimos.

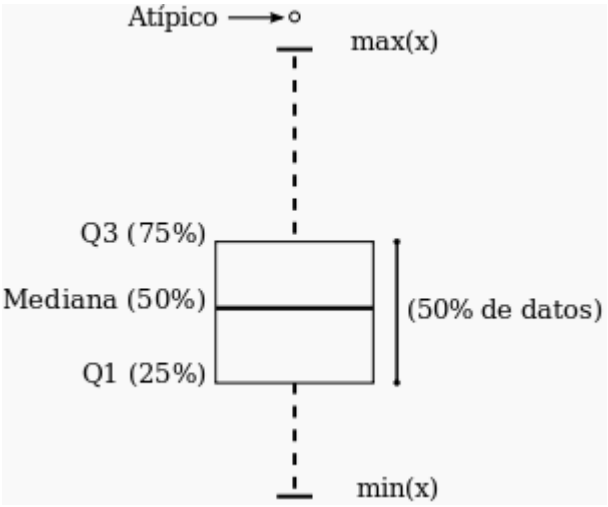

Figura 13: Modelo de un diagrama de bigotes

Test de Wilcoxon (test de rango con signo)

La prueba de los rangos con signo de Wilcoxon es una prueba no paramétrica para comparar la media de dos muestras relacionadas, muestras emparejadas, o medidas repetidas para una misma muestra, y determinar si existen diferencias entre ellas (Wilcoxon, 1945). Se utiliza como alternativa al test-t de Student para muestras dependientes entre sí, cuando no se puede suponer que las muestras siguen una distribución normal. Es decir, cuando la variable subyacente es continua pero no se presupone ningún tipo de distribución particular.

Para un conjunto de n pares de observaciones, denominadas (x_i, y_i), el objetivo del test es comprobar si puede dictaminarse que los valores x_i e y_i son o no iguales.

Se siguen tres suposiciones:

- Los datos están emparejados y provienen de la misma población.
- Cada par es escogido de forma aleatoria e independiente.
- Los datos son medidos al menos en una escala ordinal, pero no tienen por qué ser normales.

En ocasiones, esta prueba se usa para comparar las diferencias entre dos muestras de datos de pacientes tomados antes y después del tratamiento, cuyo valor central se espera que sea cero. En nuestro caso, los vectores de muestras a comparar

pertenecen a los mismos pacientes/usuarios en ambos movimientos, razón por la que se utiliza un test de Wilcoxon de rango con signo en lugar de otra de sus variantes.

Correlación cruzada

En procesamiento de señales, la correlación cruzada (o a veces denominada "covarianza cruzada") es una medida de la similitud entre dos señales, frecuentemente usada para encontrar características relevantes en una señal desconocida por medio de la comparación con otra que sí se conoce. Es función del tiempo relativo entre las señales, a veces también se la llama producto escalar desplazado, y tiene aplicaciones en el reconocimiento de patrones y en criptoanálisis. La correlación cruzada tiene una naturaleza similar a la convolución de dos funciones. Difiere en que la correlación no involucra una inversión de señal como ocurre en la convolución.

El resultado de una correlación entre dos señales de tamaño M y N es otra señal de tamaño M+N-1 cuyos picos indican puntos de máxima coincidencia. Un ejemplo puede verse en la Figura 14.

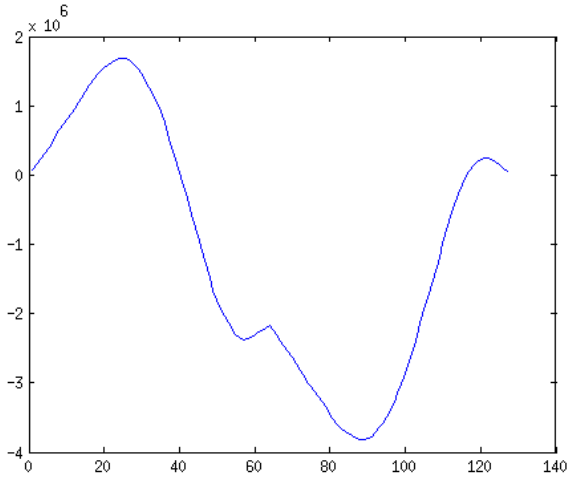

Figura 14: Correlación cruzada entre dos señales

Creación de patrones de movimiento

Una vez extraída la información de la acción realizada por el usuario en forma de señal de movimiento, es necesario procesar dicha información de forma eficiente para que pueda ser clasificada por un sistema de reconocimiento con la mayor precisión posible. Cada secuencia temporal se transforma en un patrón, que seguirán el resto de secuencias para poder ser comparables entre sí. En esta

sección trataremos los temas relativos a la selección de atributos que conforman el patrón, y en posibles transformaciones aplicables para construir el patrón de forma óptima para facilitar su clasificación.

Selección de atributos

Uno de los principales problemas que existen en el reconocimiento de movimientos está relacionado con la enorme dimensión de los datos involucrados en el proceso (p. ej. ruidos en los videos 2D, número de *joints* de los esqueletos 3D…) (Chen & Kotani, 2012). Esto deriva en un incremento de la complejidad computacional y hace más difícil la extracción de las características principales del movimiento. Existen distintos métodos de compresión para reducir la información de las trayectorias de los *joints* capturada a lo largo de todos los cuadros que dura la secuencia de movimiento (Kulic, Takano, & Nakamura, 2007) (Gu, Peng, Deng & 2009).

Representación geométrica de trayectorias

Este tipo de preprocesado tiene el fin de que el reconocimiento de la acción sea independiente respecto a la ubicación del sujeto en la escena. Dado que se utilizan características posicionales para la construcción del patrón, se aplican transformaciones geométricas, para situar el centro geométrico del movimiento en el origen de coordenadas. Asimismo, el "tamaño" del movimiento realizado es dependiente de la proximidad del usuario a la cámara. Esto reduce en gran manera la variabilidad de los patrones de movimiento.

Para el caso de las trayectorias del cuerpo humano, este problema puede solucionarse mediante un rectificado de las posiciones de los *joints* para expresarlas respecto a un punto local del esqueleto. Esto elimina la variabilidad debida a la posición en la imagen (coordenadas x, y) como de proximidad respecto de la cámara (coordenada z). Las representaciones consideradas son las siguientes:

- Torso como centro de masas: Si para todas las señales tomamos el torso como centro del esqueleto, puede realizarse una rectificación de las trayectorias de los *joints j* en cada *frame* m respecto de las coordenadas iniciales del torso en el primer *frame* capturado (Ecuación 1).
  $$\left(x_{j_m}, y_{j_m}, z_{j_m}\right) = \left(x_{j_m}, y_{j_m}, z_{j_m}\right) - \left(x_{torso_0}, y_{torso_0} z_{torso_0}\right) \qquad (1)$$
- Jerarquía de esqueleto: Se plantea una jerarquía de esqueleto basada en la de la Figura 15. Sigue el mismo método que el centro de masas, pero en lugar de usar siempre el punto del torso, cada *joint* tiene asignado un padre p en la jerarquía. Mediante este método se busca eliminar redundancia en la información de las coordenadas (Ecuación 2).

$$(x_{j_m}, y_{j_m}, z_{j_m}) = (x_{j_m}, y_{j_m}, z_{j_m}) - (x_{p_0}, y_{p_0} z_{p_0}) \quad (2)$$

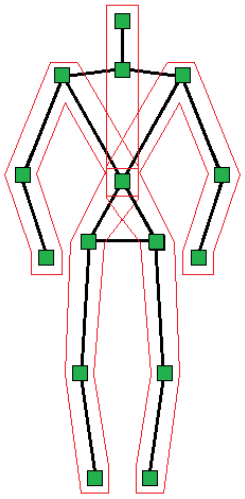

Figura 15: Jerarquía de esqueleto basada en tren superior izdo y dcho, tren inferior izdo y dcho y línea central

Deltas

Otra forma de representar una trayectoria es no solamente atendiendo a su posición exacta en el espacio, sino en cómo varía esa posición en el tiempo. Una forma de analizar esto es mediante el uso de deltas de movimiento. Para calcularlas, es necesario representar cada punto de la trayectoria respecto a la posición instantánea del *frame* anterior (Ecuación 3).

$$(x_{j_m}, y_{j_m}, z_{j_m}) = (x_{j_m}, y_{j_m}, z_{j_m}) - (x_{j_{m-1}}, y_{j_{m-1}} z_{j_{m-1}}) \quad (3)$$

Esta forma de representar un movimiento está directamente relacionada con la teoría descrita en la Sección 3.1.1, sobre cómo los humanos perciben el movimiento. Se calculan las variaciones entre los puntos intermedios que componen la trayectoria de un cuerpo que se desplaza.

Normalización

El objetivo es obtener vectores de características cuyas componentes tengan valores similares entre sí, es decir, coherentes dentro del mismo rango de valores. No pueden introducirse datos de distinta naturaleza en un clasificador. Su uso beneficia al rendimiento de los clasificadores. Dado un conjunto de *n* características capturadas en *m* instantes de tiempo, se representa según la Ecuación 4.

$$v_i = (v_{i1}, v_{i2}, \dots, v_{in}), i = 1 \dots m \quad (4)$$

Existen diversas técnicas; normalmente se normalizan por columnas de características individuales, pero existen aplicaciones que normalizan valores por filas de características de una muestra dada. Tres de las más utilizadas son:

- Z-norm: Se normaliza las características del vector de forma individual, es decir, por columnas (Ecuación 5). Al final se consigue que las características tengan una media (μ) 0 y desviación (σ) 1.

  $$v_{ij} = \frac{v_{ij} - \mu_j}{\sigma_j}, \mu_j = \frac{\sum_{i=1}^{m} v_{ij}}{m} \quad (5)$$

- Max-norm: Se divide cada valor por el máximo de su columna (Ecuación 6). Con esto se consigue llevar a todos los elementos de la matriz de características al rango [-1, 1].

  $$v_{ij} = \frac{v_{ij}}{abs[\max_j(v_{ij})]} \quad (6)$$

- MinMax-norm: Traslada todos los elementos de la matriz al rango [0,1] (Ecuación 7). Es una técnica usada en redes neuronales.

  $$v_{ij} = \frac{v_{ij} - \min_j(v_{ij})}{\max_j(v_{ij}) - \min_j(v_{ij})} \quad (7)$$

Transformada de Fourier

En procesamiento de señales, la Transformada de Fourier (FT) suele considerarse como la descomposición de una señal en componentes de frecuencias diferentes (Boashash, 2003). Es decir, la transformada de Fourier es básicamente el espectro de frecuencias de una función. Se transforma una función matemática en otra, obteniendo una representación en el dominio de la frecuencia, siendo la función original una función en el dominio del tiempo. Asimismo, la señal original puede recuperarse gracias que la transformada es invertible.

En el caso de que la función de entrada sea una secuencia discreta y de duración finita, se usa la denominada DFT (Discrete Fourier Transform). Dichas secuencias se suelen generar a partir del muestreo de una función continua, como puede ser la voz humana o, como en este caso, una trayectoria variante con el tiempo. Por estas razones, se dice que la DFT es una transformada de Fourier para análisis de señales de tiempo discreto y dominio finito.

La entrada de la DFT es una secuencia finita de números reales o complejos, de modo que es ideal para procesar información almacenada en soportes digitales. En particular, la DFT se utiliza comúnmente en procesado digital de señales y otros campos relacionados dedicados a analizar las frecuencias que contiene una señal muestreada. Un factor muy importante para este tipo de aplicaciones es que la DFT y su inversa pueden ser calculadas de forma eficiente en la práctica utilizando el algoritmo de la transformada rápida de Fourier o FFT (Fast Fourier

Transform). Este algoritmo es usado frecuentemente en diversas aplicaciones (Brigham, 1988), dado que reduce el orden de complejidad de la DFT, de $O(N^2)$ a O(N *log* N). Sin embargo, el algoritmo pone algunas limitaciones en la señal y en el espectro resultantes. Por ejemplo: la señal de la que se tomaron muestras y que se va a transformar debe consistir de un número de muestras igual a una potencia de dos.

Una ventaja de esta técnica es que es posible truncar la señal transformada a las primeras componentes manteniendo información relevante acerca del movimiento (Naftel & Khalid, 2006). Esto además realiza una función de suavizado en la señal que permite eliminar ruido motriz generado por el usuario en la fase de captura.

Elección del clasificador

La última etapa del sistema de reconocimiento se corresponde con un sistema inteligente capaz de aprender y reconocer los patrones generados mediante las técnicas anteriores.

Técnicas de Matching: DTW

Dynamic Time Warping (DTW) es un método que permite a una máquina encontrar el camino óptimo entre dos secuencias dadas con ciertas restricciones (Keogh & Ratanamahatana, 2005). Las secuencias son "deformadas" no linealmente en la dimensión temporal para determinar una medida de su similitud independientemente de ciertas variaciones no lineales en la dimensión temporal. La diferencia entre una distancia Euclídea simple y DTW aparece representada de forma visual en la Figura 16. Este método de alineamiento de secuencias es usado normalmente para la clasificación de series temporales.

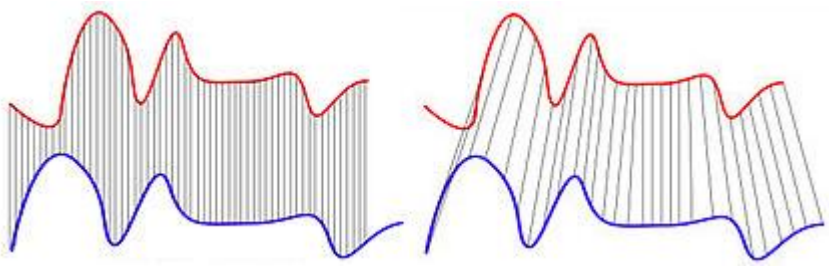

Figura 16: Distancia Euclídea (izda) vs DTW (dcha)

En general, DTW es un algoritmo para medir la similitud entre dos secuencias que podrían variar en tiempo o velocidad. Por ejemplo, podrían detectarse similitudes en patrones al caminar, incluso si en una señal la persona estuviese caminando despacio y en otra estuviese caminando más rápidamente, o incluso si hubiese aceleraciones y deceleraciones durante el curso de la observación. DTW ha sido

aplicado a video, audio y gráficos; de hecho, cualquier conjunto de datos que pueda expresarse mediante una representación lineal puede ser analizado mediante DTW. Una aplicación altamente conocida es reconocimiento de voz, para tratar con distintas velocidades del habla. Otras aplicaciones incluyen reconocimiento del hablante y reconocimiento de firma. También puede observarse su uso en aplicaciones de *matching* de formas y figuras.

El método DTW es, por tanto, muy utilizado para comparar cadenas de distinta longitud; encuentra una alineación no linear entre los puntos en las dos cadenas, de forma que la suma de las diferencias entre cada par de puntos alineados sea mínima. La distancia entre la firma de referencia y de test es el valor acumulado en la esquina superior derecha de la matriz DTW (Figura 17). La menor distancia se correspondería con la línea diagonal de una matriz cuadrada. Dado que para encontrar la diferencia mínima todas las posibles alineaciones deben de ser investigadas, este algoritmo es computacionalmente costoso; una solución tradicional es utilizar programación dinámica. De esta etapa se obtienen una serie de valores de disimilitud. Las distancias obtenidas son combinadas usando distintas técnicas (mínimo, media, mediana). Se ha demostrado que su rendimiento supera al de métodos estadísticos en ciertas tareas de reconocimiento (Pascual-Gaspar, Cardeñoso-Payo, Vivaracho-Pascual, 2009).

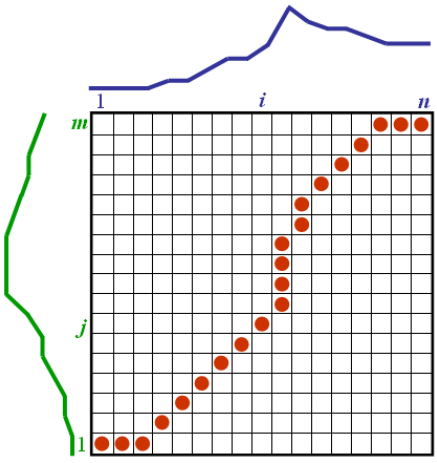

Figura 17: DTW aplicado a dos señales de longitud *m* y *n*

DTW no deja de tener restricciones (Ratanamahatana & Keogh, 2005). Actualmente existen versiones mejoradas de DTW, en complejidad computacional, que además han sido utilizadas de forma exitosa para otros problemas de clasificación de movimientos (Zhou & De la Torre, 2012).

Entropía

En teoría de la información, la entropía cruzada (De Boer, Kroese, Mannor & Rubinstein, 2005) entre dos distribuciones sobre el mismo espacio de probabilidad mide la media de bits necesarios para identificar un evento de un conjunto de posibilidades, si un esquema de codificación está basado en una distribución de probabilidad dada *q*, más que en la distribución "verdadera" *p*. Al ser aplicada sobre los componentes de una señal, permite determinar la existencia de información redundante. En el caso de señales de movimiento, valores de entropía más pequeños indican que la información es más parecida.

La entropía aproximada (*ApEn*) es una familia estadística introducida como una cuantificación de la regularidad en series de datos temporales. Fue construida por Pincus (Pincus, 1991), motivada inicialmente por aplicaciones para conjuntos de datos relativamente cortos y ruidosos, como la mayoría de señales biomédicas. De hecho, *ApEn* puede aplicarse a señales con al menos 50 puntos de datos y una amplia cantidad de clases de modelos. Es invariante a la escala e independiente del modelo, evalúa patrones de datos tanto dominantes como subordinados, y discrimina clases para las cuales el reconocimiento de características es complicado. Para computar *ApEn* es necesario especificar dos parámetros: una longitud *m* y una ventana de tolerancia *r*. En resumen, *ApEn* mide la probabilidad de que ejecuciones de patrones que son cercanos (dentro de *r*) para *m* observaciones contiguas siguen siendo cercanas (dentro de la misma anchura de tolerancia *r*) en comparaciones incrementales subsiguientes. Se denomina *N* al número de puntos de las series temporales.

La Entropía Cruzada Aproximada (*Cross-ApEn*) fue propuesta para comparar secuencias correladas, sugiriendo su aplicación a señales fisiológicas. *Cross-ApEn* es temática y algorítmicamente similar a *ApEn*, con una diferencia crítica en el enfoque: es aplicada a dos señales en lugar de a una única serie, y por lo tanto mide la disimilitud estadística de dichas series temporales. Extendiendo la definición de *ApEn*, para calcular *Cross-ApEn* es necesario especificar los valores de *m* y *r*. Dadas dos series temporales *u* y *v*, mide (dentro de la tolerancia *r*), la regularidad o frecuencia de patrones de *v* similares a un patrón dado *u* para una longitud de ventana *m*.

Técnicas de combinación de resultados

En el caso de utilizar medidas de distancia u otras métricas asociados a un único usuario o *joint*, surge un escenario en el que se desea reducir dicho conjunto de puntuaciones a un valor único (Pascual-Gaspar, Faundez-Zanuy & Vivaracho-

Pascual, 2011). Para combinar un número *S* de distancias, las siguientes técnicas son comúnmente utilizadas:

- Valor mínimo (Ecuación 8).
  $dist_p = \min(d_1, \dots, d_S)$ (8)
- Valor máximo (Ecuación 9).
  $dist_p = \max(d_1, \dots, d_S)$ (9)
- Suma (Ecuación 10): Para un número fijo de *S*, es equivalente a la función media, *mean()*.
  $dist_p = \sum_{i=1}^{S} d_i$ (10)
- Producto (Ecuación 11): Se usa el logaritmo para evitar los valores muy bajos.
  $dist_p = \log\left(\prod_{i=1}^{S} d_i\right) = \sum_{i=1}^{S} \log(d_i)$ (11)
- Suma de los valores extremos (SEV) (Ecuación 12).
  $dist_p = \min(d_1, \dots, d_S) + \max(d_1, \dots, d_S)$ (12)
- Suma ponderada (Ecuación 13): Los c_i pueden ser independientes para cada movimiento.
  $dist_p = \sum_{i=1}^{S} c_i d_i$ (13)
  Existen variantes en el cálculo de pesos:
  - Basados en las estadísticas del entrenamiento: Una vez se ha entrenado el modelo dado, las distancias de cada sección son calculadas, y de estas se obtienen la media y la desviación estándar. Existen distintas técnicas:
    - Suma ponderada basada en la desviación: Los pesos son inversamente proporcionales a la desviación estándar.
    - Suma ponderada basada en medias elevadas: Se refuerzan las secciones con puntuaciones de distancia elevadas.
    - Suma ponderada basada en medias bajas: Se refuerzan las secciones con puntuaciones de distancia bajas.
  - Basadas en errores de reconocimiento: Se refuerzan las secciones con mejor rendimiento. Implementaciones:
    - Suma ponderada basada en errores dependientes del usuario: Tomando el set de entrenamiento el del propio usuario contra los set del resto, se calculan los pesos igual que los casos anteriores.
    - Suma ponderada basada en umbrales EER dependientes del usuario: Igual que el anterior, pero tomando umbrales en lugar del EER.

Redes Neuronales: Fuzzy ARTMAP

Esta red está basada en la teoría ART (Adaptive Resonance Theory) (Carpenter, Grossberg, Markuzon, Reynolds & Rosen, 1992), por lo que incluye características como computación difusa y aprendizaje incremental. Debido a la naturaleza de estas redes, los patrones de entrada han de ser normalizados de forma conjunta, ajustando sus valores en el rango [0,1] antes de ser introducidos a la red para su aprendizaje o evaluación. De la misma forma, han de ser codificados-complementados doblando su tamaño, con lo que la red trabaja con patrones de elevada información.

La red neuronal se compone de una capa de entrada L1, una de salida L2 y una etapa de etiquetado. L1 y L2 se conectan mediante un filtro de pesos adaptativos w_j. De forma similar, L2 se conecta a la etapa de clasificación mediante pesos L_{jl}. Los patrones I activan L1 mediante el patrón complementado I_c=(I,I-1). Cada entrada activa un nodo j en L2 con una intensidad T_j a través de pesos adaptativos w_j. La Ecuación 14 es de activación, donde | | es la norma del vector, ^ es el operador difuso AND ($(p \wedge q)_i = \min(p_i, q_i)$).

$$T_j = \frac{\left| I \wedge w_j \right|}{\left(0.05 + \left| w_j \right| \right)} \qquad (14)$$

Una diferencia entre la red neuronal propuesta y la red neuronal ARTMAP es la inclusión de un mecanismo que permite seleccionar de forma rápida los nodos de L2 que deben activarse ante un patrón de movimiento de entrada. Esto acelera el aprendizaje de movimientos y hace que el algoritmo sea paralelizable. En L2 se lleva a cabo una competición en la que se selecciona un nodo J tal que $T_J=\max(T_j)$. Entonces se mide la semejanza entre el nodo w_J y el patrón de entrada I mediante la Ecuación 15.

$$\frac{\left| I_I \wedge w_J \right|}{\left| I_I \right|} \geq threshold \qquad (15)$$

Si no se satisface el criterio, se promueve un nuevo nodo en L2. En caso contrario, la red entra en resonancia y aprende el vector de entrada (Ecuación 16).

$$w_J = \beta(I_I \wedge w_J) + (1-\beta) w_J \qquad (16)$$

Durante la fase de clasificación, los pesos no cambian y el nodo ganador en la capa L2 determina la clasificación.

Mejora de rendimiento: Computación en GPU

Como se ha comentado anteriormente en la Sección 2.5, los sistemas de clasificación pueden llegar a tener un coste computacional muy alto; en este caso es especialmente notable dada la gran dimensión de los patrones de movimiento. Las técnicas de computación en GPU permiten llamar a funciones especiales denominadas *kernel*, las cuales trabajan sobre un conjunto de datos en paralelo. Esto es gracias a la potencia de procesamiento multi-hilo de las tarjetas gráficas. Como contrapartida, es necesario gestionar de forma eficiente los accesos a memoria para no ralentizar el programa. La Figura 18 muestra un esquema de la ejecución de un programa que posee fragmentos de código paralelizados.

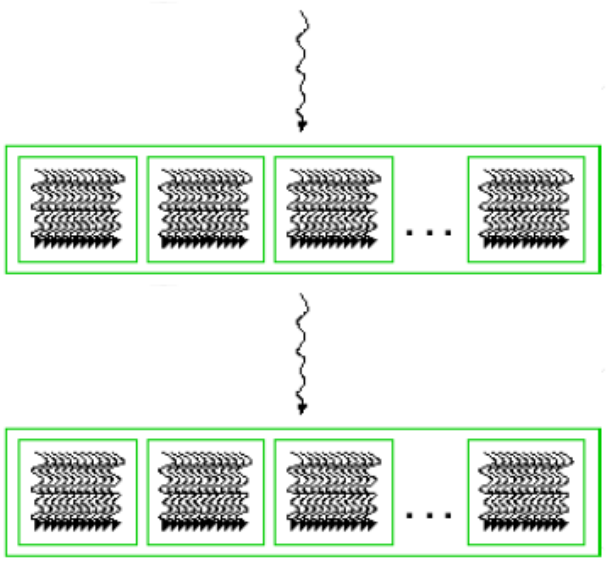

Figura 18: Ejecución en paralelo de un programa gracias a las operaciones multi-hilo en GPU

El objetivo de aplicar técnicas de este tipo es por tanto conseguir una mejora adicional del rendimiento del sistema, esta vez centrada en la velocidad de ejecución, paralelizando los cuyos algoritmos tengan un mayor coste computacional.

3.3.2. Diseño de experimentos

Esta sección plantea una serie de experimentos a llevar a cabo utilizando el conjunto de técnicas descritas en la sección anterior. El objetivo final es determinar qué combinación de técnicas son más efectivas a la hora de reconocer movimientos extraídos de un esqueleto 3D capturado mediante Kinect. Esto permitirá definir una configuración optimizada para el sistema de reconocimiento de movimientos planteado.

Fase 1: Análisis de movimientos

El primer conjunto de experimentos planteados tiene en cuenta las características de las señales extraídas para determinar su naturaleza. Esto resulta de gran ayuda a la hora de determinar qué técnicas son útiles para procesar y clasificar los movimientos del sistema.

Método 1

Se plantea primero un análisis de la distribución de los valores que toman las trayectorias de los *joints* de las señales mediante el uso de diagramas de bigotes sobre las secuencias temporales de cada coordenada.

Se realizan comparaciones para analizar un mismo movimiento en base a distintos criterios (Figura 19). El objetivo es analizar la variabilidad de cada acción en base a:

- Comparación entre usuarios: Se visualiza cómo varían el movimiento para los *joints* que intervienen en él. La idea es determinar diferencias entre ejecuciones realizadas por distintos usuarios.
- Comparación entre *joints*: Se visualiza cómo varía el movimiento para los usuarios que lo realizan. La idea es determinar qué *joints* son más determinantes para definir el movimiento.

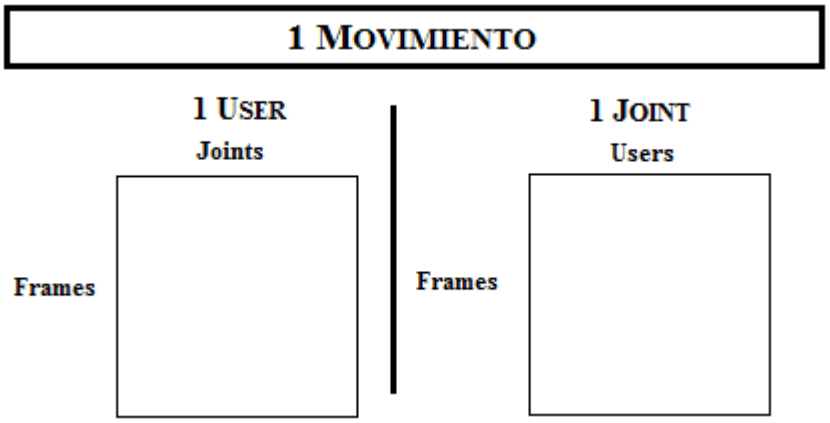

Figura 19: Distribución de usuarios y *joints* a la hora de analizar las secuencias de un movimiento mediante el diagrama de bigotes

Método 2

Utilizar el test de rango con signo para cada matriz de movimiento, realizando comprobaciones entre pares de movimientos. Se toman como pacientes los usuarios que realizan un movimiento dado, y se crea una matriz de tres

dimensiones (45 trayectorias × 64 *frames* × 9 usuarios). Se comparan los movimientos de dos en dos; esto resulta en una matriz 45 × 64 que representa la probabilidad de similitud entre dichos movimientos. Cada punto vale entre 0 y 1, y un valor menor de 0.05 indica que son estadísticamente diferentes.

Por tanto, una vez se tienen los conjuntos de datos emparejados, se plantea una hipótesis nula: "Ambas muestras no son sustancialmente diferentes". Una vez escogido un valor para α (normalmente el 5%), el resultado del test de Wilcoxon puede representarse de forma binaria de la siguiente manera:

- No se puede rechazar la hipótesis (valor mayor que 0.05): Los datos de entrada son parecidos.
- Puede rechazarse la hipótesis (valor menor que 0.05): Los datos de entrada son sustancialmente diferentes.

La Figura 20 muestra un ejemplo en el cual el movimiento 0 se parecería al movimiento 1 en la coordenada x del *joint* 2, a partir del *frame* 24 (la zona gris representaría el no rechazo de la hipótesis).

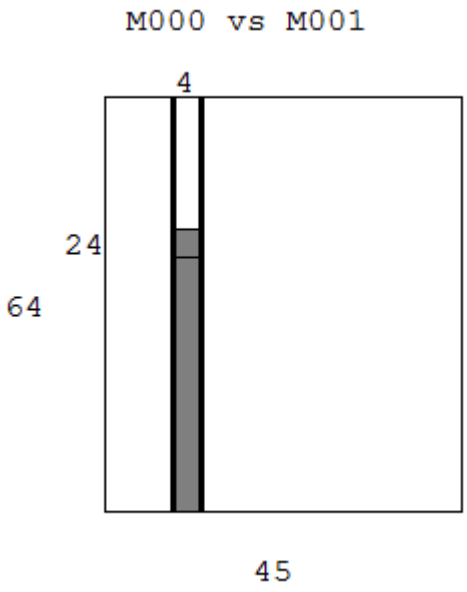

Figura 20: Resultado de realizar un test de signo sobre las trayectorias de dos movimientos

Además de la utilidad ya mencionada, esta técnica puede servir para determinar cuáles son los *joints* más característicos de un movimiento, es decir, los que rechazan la hipótesis siempre al ser comparados con el resto. Esta información puede ser útil a la hora de asignar un peso a cada trayectoria dentro de la acción.

Método 3

Otra forma de analizar las señales en base a una comparación entre ellas es mediante la correlación cruzada. Se propone por tanto realizar comparaciones entre señales conflictivas:

- Trayectorias del mismo *joint* que el sistema considere diferentes para entradas pertenecientes al mismo movimiento.
- Trayectorias del mismo *joint* que el sistema considere similares para entradas pertenecientes a movimientos distintos.

Fase 2: Construcción de patrones

Método 4

Se plantea el uso de técnicas de selección de atributos con el objetivo de reducir la dimensión de los patrones de movimientos. Una posible técnica es prescindir de las trayectorias o *joints* sin capacidad discriminatoria, es decir, que no difieran entre distintos movimientos.

Método 5

Se probarán distintos tipos de transformaciones geométricas mencionadas anteriormente sobre las trayectorias de los patrones de entrada: torso como centro de masas, jerarquía de esqueleto. La Figura 21 muestra la estructura de la jerarquía creada para tal propósito.

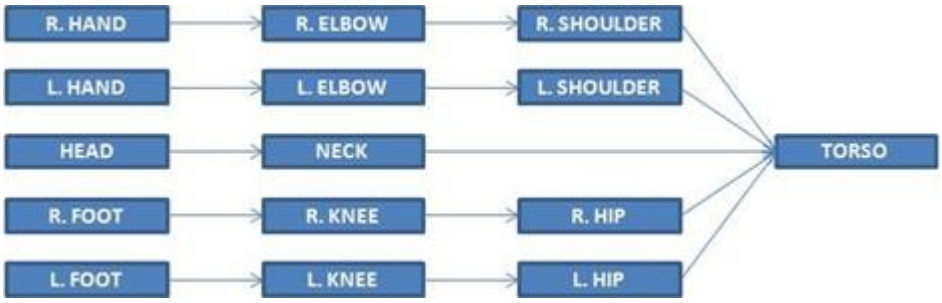

Figura 21: Esquema de jerarquía en los *joints* del esqueleto dado

Estas técnicas serán aplicadas en los distintos experimentos para determinar su efectividad bajo distintas configuraciones.

Método 6

Se aplicarán deltas de movimiento a las trayectorias para determinar el efecto de este tipo de representación a la hora de reconocer acciones. Al igual que las anteriores, esta transformación se aplica en combinación con cualquiera de las anteriores.

Método 7

Se propone el uso de distintos esquemas de normalización que se combinarán con el resto de parámetros para encontrar la precisión más alta del sistema.

- Normalización Z para simplificar la distribución de valores.
- Normalización MinMax para dejar los valores entre 0 y 1. Esta normalización es imprescindible a la hora de utilizar redes neuronales Fuzzy como la propuesta con anterioridad.

Método 8

Se utilizará la transformada de Fourier de los patrones construidos tras la aplicación de los experimentos 4, 5 y 6. El objetivo es aprovechar las funcionalidades de suavizado de la FFT y aislamiento temporal que ofrece el dominio de la frecuencia. Utilizando esta técnica en nuestras trayectorias, podemos construir un patrón transformando las coordenadas de los *joints* para generar un vector $\tilde{P}_j = FFT(\vec{p}_j(t)) = (\tilde{X}_j, \tilde{Y}_j, \tilde{Z}_j)$. De esta forma se generan patrones truncados de características empleando la componente continua y las primeras *f* componentes más significativas de las partes real e imaginaria de la Transformada de Fourier, para cada uno de los *j joints* seleccionados en la fase de selección de características (Ecuación 17). Esto forma un patrón de $3\times j\times(1+2\times f)$ componentes.

$$I = \bigcup_{k=1}^{j} \begin{pmatrix} \tilde{X}_{k_0}, \mathrm{Re}\{\tilde{X}_{k_1}\}, \mathrm{Im}\{\tilde{X}_{k_1}\}, \ldots, \mathrm{Im}\{\tilde{X}_{k_f}\}, \\ \tilde{Y}_{k_0}, \mathrm{Re}\{\tilde{Y}_{k_1}\}, \mathrm{Im}\{\tilde{Y}_{k_1}\}, \ldots, \mathrm{Im}\{\tilde{Y}_{k_f}\}, \\ \tilde{Z}_{k_0}, \mathrm{Re}\{\tilde{Z}_{k_1}\}, \mathrm{Im}\{\tilde{Z}_{k_1}\}, \ldots, \mathrm{Im}\{\tilde{Z}_{k_f}\} \end{pmatrix} \qquad (17)$$

Una vez aplicadas estas fases, se introducirían los patrones construidos a uno de los sistemas inteligentes que se presentan a continuación.

Fase 3: Selección del clasificador

Método 9

Se plantea el uso de DTW sobre los patrones de movimientos, de forma que se calculen las distancias entre la señal desconocida y los modelos aprendidos. Para cada patrón de entrada, el modelo de cada movimiento sería una matriz tridimensional con los distintos usuarios realizando el mismo. Para tomar la decisión final se usarán las técnicas de combinación de distancias descritas en la Sección 4.4.3.

Método 10

Se utiliza una función de entropía sobre las secuencias temporales de cada *joint*, de forma que se comparen todos los movimientos entre sí. El modelo se forma igual que con la función de DTW. Esto nos ofrece una serie de valores de entropía, que habrán de ser combinados, como se explica en el siguiente apartado. La función de entropía aproximada cruzada permite también modificar el valor del parámetro de tolerancia r, con lo que se probará el efecto de variar este valor.

Método 11

Las técnicas de clasificación que tienen como salida una serie de distancias asociadas, requieren de alguna técnica para combinar dichos valores. Normalmente, las técnicas se usan para combinar los resultados de comparar la entrada con los usuarios de cada modelo, y los resultados finales de los modelos. Esta metodología está representada en la Figura 22.

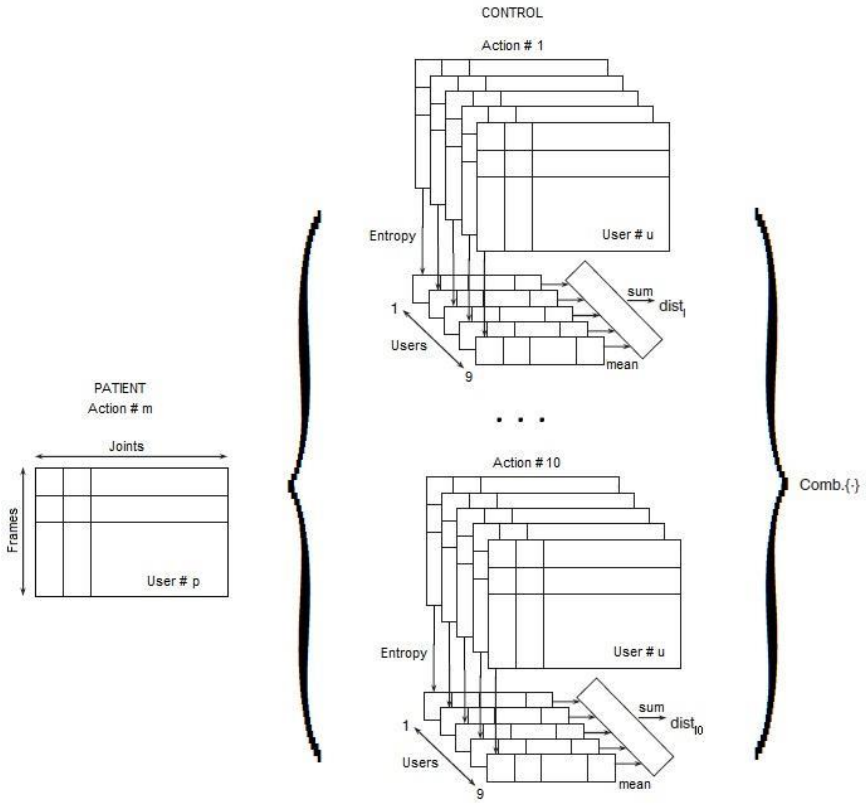

Figura 22: Esquema de combinación de los resultados al utilizar un clasificador basado en distancias

Se plantea el uso de las técnicas descritas en la Sección 4.4.3. Asimismo, al utilizar la suma ponderada basada en el set de entrenamiento, pueden utilizarse los

pesos mencionados en la Sección 5.1.2 como criterio para dar más importancia a un *joint* que a otro dependiendo de su capacidad para discriminar el movimiento.

Cabe mencionar, que al uso de estas técnicas viene asociada al cálculo de un valor de umbral para decidir la pertenencia o no a un modelo.

Método 12
Se propone el uso de una red neuronal basada en Fuzzy ARTMAP para clasificar los movimientos (Figura 23). Normalmente los patrones de esta red se introducen en forma de un único vector, por lo que habría que concatenar las secuencias temporales de una acción dada generando un vector de dimensiones elevadas. Además, esta red requiere aplicar a los patrones uno de los tipos de normalización mencionados en la Sección 4.3.3.

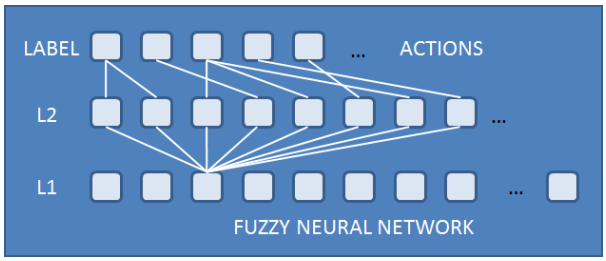

Figura 23: Estructura de una red neuronal Fuzzy ARTMAP

La salida de los clasificadores (fase 3) se analizaría y reutilizaría para realizar análisis de señales concretas (fase 1) y diseñar nuevas configuraciones del sistema (fase 2), realimentando así el proceso de búsqueda de la solución óptima al problema.

Fase 4: Computación en GPU
En el caso de detectar una configuración cuyo coste computacional excediese el admisible en un sistema de estas características, se estudiaría el uso de técnicas de paralelización del algoritmo o algoritmos que emplea. Es de destacar, que el mayor coste en este tipo de sistemas normalmente viene dado por el algoritmo de clasificación.

Implementación de un Toolbox de procesamiento de movimientos en MatLab®
Para terminar de completar este trabajo, se ofrecerá una implementación de las utilidades desarrolladas en MatLab® para llevar a cabo los experimentos. Éstas se proveerán en forma de Toolbox para mayor facilidad en su manejo y configuración. Asimismo, vendrán acompañadas de un manual de usuario que

explique su uso, la declaración de las funciones, la estructura de directorios, los formatos de los ficheros, dependencias, etc.

3.3.3. Elección de *datasets*

Los experimentos descritos en las secciones anteriores se realizan sobre un conjunto de datos de movimiento obtenidos mediante una cámara 3D, concretamente Kinect. Nuestra base de datos consiste en 8 movimientos independientes realizados por 9 usuarios diferentes. Los diferentes tipos de acciones son: *bending down* (agacharse), *jumping-jack* (saltos de tijera), *jumping* (saltos de canguro), *jumping in place* (saltos en el sitio), *galloping sideways* (trotar de lado), *walking* (andar), *waving one hand* (saludar con una mano) y *waving both hands* (saludar con ambas manos). El escenario donde se realizan los movimientos incluye un fondo complejo de oficina. Estos movimientos representan acciones en las que se usan múltiples partes del cuerpo, ya que se desea testear la eficacia del sistema al intentar reconocer movimientos genéricos de cualquier tipo. Esto es debido a que se trata del diseño inicial del reconocedor, pudiéndose en un futuro probar acciones para aplicaciones más concretas.

En el *dataset* construido, se comienza con el usuario mirando de frente a la cámara y colocándose en la posición inicial durante unos segundos. A continuación, el movimiento comienza y se continúa durante aproximadamente 5 segundos.

Una vez grabados los movimientos mediante la herramienta que proporciona OpenNI, se dispone de un conjunto de videos en formato .oni con información RGBD (imagen de color + profundidad). De la misma forma, OpenNI puede procesar estos videos para proporcionar *frame* a *frame* las coordenadas del "mundo real" de las articulaciones que conforman un esqueleto que sigue el movimiento del usuario. Estas coordenadas 3D en x-y-z son las que se desea extraer y analizar para llevar a cabo el reconocimiento de movimientos. En la siguiente sección sobre experimentación se detalla el procesamiento realizado con estos datos.

Capítulo 4: Experimentación y resultados

4.1. Experimentos realizados

Los experimentos se realizaron con un set de 8 movimientos:

- Agacharse
- Saltos de tijera
- Saltar hacia adelante
- Saltar en el sitio
- Galopar de lado
- Andar
- Saludar con una mano
- Saludar con las dos manos

Como se indica en la Sección 3.3.3, estos movimientos han sido elegidos para testear el sistema ya que representan acciones heterogéneas, bien conocidas y de cuerpo completo. Estos movimientos son realizados por 9 usuarios diferentes, con una única repetición.

Una vez el usuario reconocido está en posición, el sistema se activa mediante un *trigger* cuando el usuario empieza a moverse. En ese momento, el sistema almacena las coordenadas x-y-z de los *joints* monitorizados en cada *frame*, obtenidas mediante OpenNI. Por lo tanto, un movimiento queda representado mediante la trayectoria de los *joints* monitorizados, en sus tres coordenadas. Los valores que toman x-y-z son en milímetros reales, y toman como origen de coordenadas el centro de la imagen, que a su vez depende de la posición de la cámara (Figura 23). Por ejemplo, si el usuario se encuentra a 2 metros de la imagen, la coordenada z tendrá un valor de 2000.

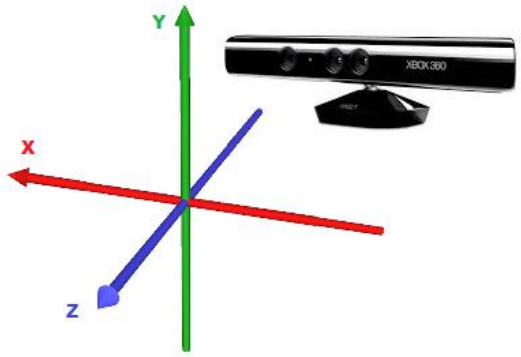

Figura 24: Sistema de coordenadas extraídas de Kinect mediante OpenNI

De esta forma, en este proyecto se ha definido como unidad de procesamiento la denominada matriz de movimiento (*motion matrix*). Como se muestra en la Figura 24, se compone de tantas filas como *frames* dura la acción, y cada columna es una de las coordenadas tres de los *joints*. La tercera dimensión de la matriz sirve para almacenar más repeticiones de usuarios realizando el mismo movimiento.

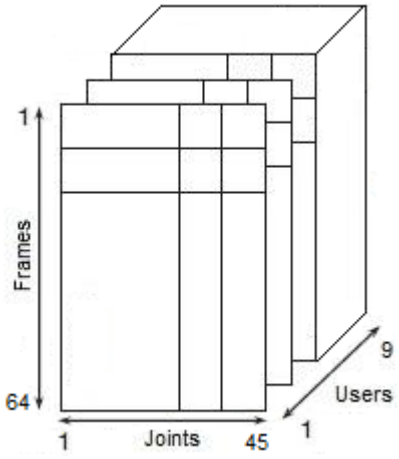

Figura 25: Matriz de movimiento

La Tabla 1 muestra en orden la lista de *joints* monitorizados en este trabajo.

***Joint* ID**	**Parte del cuerpo**	**Posición en la matriz de movimiento**
1	Cabeza	(1:3)
2	Cuello	(4:6)
3	Hombro izquierdo	(7:9)
4	Codo izquierdo	(10:12)
5	Mano izquierda	(13:15)
6	Hombro derecho	(16:18)
7	Codo derecho	(19:21)
8	Mano derecha	(22:24)
9	Torso	(25:27)
10	Cadera izquierda	(28:30)
11	Rodilla izquierda	(31:33)
12	Pie izquierdo	(34:36)
13	Cadera derecha	(37:39)
14	Rodilla derecha	(40:42)
15	Pie derecho	(43:45)

Tabla 1: Correspondencia entre partes del cuerpo monitorizadas y *joints* del sistema

En resumen, la dimensión del problema es la siguiente:

- 9 usuarios. Por cada usuario:
    - 8 movimientos. Por cada movimiento:
        - 15 *joints*. Por cada *joint*:
            - 3 trayectorias (x-y-z). Por cada trayectoria:
                - 64 *frames* de duración.

En este trabajo se han utilizado dos tipos distintos de clasificador, cada uno con unas perspectivas distintas (Sección 3.3.2). Mientras que los resultados de las pruebas se detallan separadamente, ambos clasificadores admiten los mismos parámetros de preprocesamiento, que siguen el siguiente esquema con una o varias opciones:

- Rectificación: Centro de masas / Jerarquía
- *Joints* seleccionados: No / Sí (1 4 5 7 8 11 12 14 15)
- Deltas: No / Sí
- FFT: No / Sí (tamaño 64)
- Construcción de patrones: No / Sí
- Normalización: ZNorm, MinMaxNorm, MaxNorm

La evaluación es realizada mediante la técnica de Left One Out (LOO). Por lo tanto, se realiza una iteración de clasificación por cada usuario del sistema y la precisión final se calcula como la media de las tasas de acierto de cada iteración. Por lo tanto, en cada iteración la fase de entrenamiento se realiza con todos los movimientos excepto los de un usuario, y la fase de clasificación se realiza con los movimientos de dicho usuario.

Primero se explican los experimentos realizados usando una red neuronal Fuzzy ARTMAP, y a continuación los relativos a distancias. En cada subsección se explican las configuraciones aplicadas y los resultados obtenidos, de forma que los experimentos sean reproducibles. Los casos en los que el resultado sea satisfactorio (la precisión supere el 90%), se incluirá además la matriz de confusión asociada y un gráfico con las tasas de acierto de cada movimiento. Sin embargo, previamente se realizó una prueba del sistema de monitorización sobre un *dataset* distinto para comprobar su efectividad

4.1.1. Monitorización de entornos

Antes de comenzar a probar el sistema de pre-procesamiento y clasificación, y aprovechando la existencia de otro *dataset* público basado en Kinect (Ni, Wang &

Moulin, 2011), se quiso probar el sistema de monitorización con él. Se descargó uno de los usuarios, y se probaron los movimientos. Las características de los videos .oni son:

- Fondo: Oficina
- Punto de vista: Cenital
- Actor: Varón joven de raza asiática

Los movimientos son los descritos en la Sección 2.6. Nuestro sistema es capaz de detectar y comenzar a monitorizar al usuario poco después de entrar en la escena. El usuario es capturado de forma correcta en la práctica totalidad de los movimientos (Figura 25), salvo en los que desaparece total o parcialmente de la escena. Por ejemplo al salir de la habitación, el sistema pierde al usuario. De la misma forma, cuando el usuario se tumba y se tapa con una manta, no se consigue recuperar el modelo de esqueleto de la imagen de profundidad. Sin embargo, el objetivo de este trabajo no se centra en movimientos de este tipo, sino en acciones heterogéneas. Una vez probada la efectividad del sistema de captura, se procede a realizar las pruebas de clasificación con los movimientos de nuestro *dataset*.

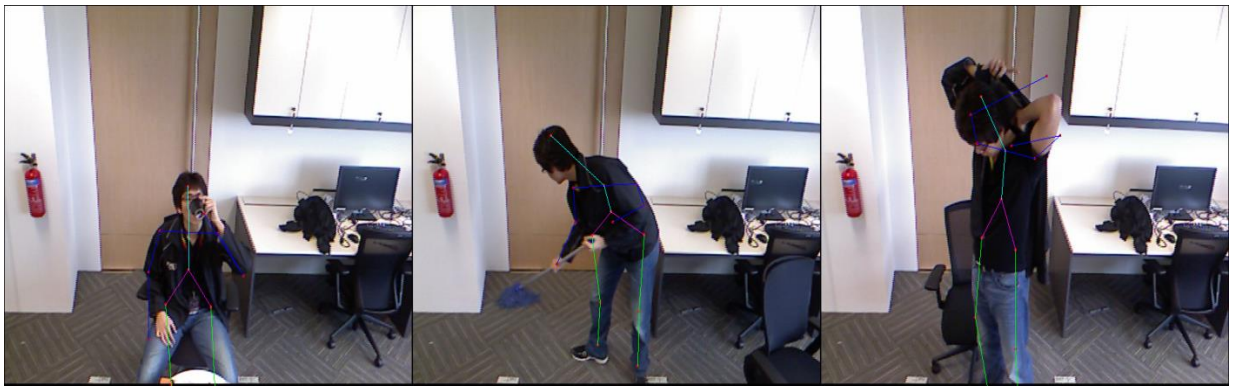

Figura 26: Sistema de monitorización con un *dataset* de acciones genéricas para Kinect

4.1.2. Reconocimiento mediante Red Neuronal

Como se ha indicado en la Sección 3.3.2, la red neuronal utilizada es una Fuzzy ARTMAP.

Consideraciones iniciales

Sin embargo, antes de su uso es necesario tener en cuenta ciertas consideraciones asociadas a este tipo de red neuronal (Henniges, Granger & Sabourin, 2005). Mientras que otros clasificadores sencillos como kNN no tienen este problema, se sabe que las redes neuronales ARTMAP sufren de sobre-entrenamiento, especialmente en *datasets* con distribuciones de clases solapadas. Este sobre-

entrenamiento está relacionado con el problema de la proliferación de categorías. Tampoco cabe olvidar que Fuzzy ARTMAP es sensible al orden en el que se presentan los datos de entrenamiento. En este trabajo se busca la metodología más adecuada al usar Fuzzy ARTMAP, en cuanto a la estructura de los datos, la estrategia de entrenamiento, el número de *epoch*, la normalización de los datos y el tamaño del *dataset* de entrenamiento.

Un número elevado de instancias por clase, hay sobre-entrenamiento si estas se superponen. Asimismo, el sobre-entrenamiento aumenta al aumentar el número de *epoch*. El uso de una técnica de normalización u otra no afecta. La principal conclusión que obtienen es que las estrategias en las que se entrena la red hasta la convergencia producen un mayor sobre-entrenamiento que las basadas en validación cruzada. Nuestras clases de movimientos se superponen en cierto modo, ya que muchos de los *joints* comparten parcialmente la trayectoria en algunos movimientos. También se tiene en cuenta que no se dispone de un número elevado de movimientos capturados, con lo que no es factible destinar muchos movimientos para la fase de entrenamiento. La estrategia de evaluación seleccionada es por tanto Validación Cruzada mediante Left-One-Out.

Configuración de los experimentos
Los parámetros utilizados para las pruebas realizadas con este clasificador son:

- Vigilancia (ρ): 0.9 (entrenamiento) y 0.001 (test)
- Bias (α): 0.005
- Tasa de aprendizaje (β): 0.5
- Número de *epochs*: 100

Al aplicar un valor de vigilancia casi nulo en el test, se obliga al sistema a seleccionar una de las categorías creadas.

Creación de patrones
A continuación se detallan los experimentos realizados. El uso de esta red neuronal implica una serie de medidas a seguir en la creación de patrones.

En primer lugar, todos los datos asociados a un movimiento han de aparecer en un único vector. Por lo tanto, habrá un vector por cada usuario realizando un movimiento, con lo que es necesario pasar cada "loncha" de una matriz de movimiento a un único vector. En este caso nos encontramos con uno de las dificultades mencionadas en la Sección 2.4, sobre la elevada dimensión de este problema. Por ello, en el preprocesamiento de los datos se ha optado por utilizar

técnicas de reducción de atributos. Tras estudiar diferentes esquemas de representación (Kulic, Takano & Nakamura, 2007), se dedujo que usando las coordenadas directamente en una secuencia temporal resulta en vectores de tamaño enorme, para modelar un patrón de movimiento. Para transformar una trayectoria en crudo en una representación más manejable se estudiaron distintas técnicas de compresión (Gu, Peng & Deng, 2009).

La forma mediante la cual se consiguió reducir la dimensión de las trayectorias de los *joints* es aplicando la Transformada de Fourier. Tal y como se especifica en (Naftel & Khalid, 2006), esta técnica es usada en muchos campos, y permite representar señales que varían en el tiempo de duración variable. También, al mantener solamente las primeras componentes es posible representar la información sin perder información altamente relevante, suavizando la señal original. Esto también permite eliminar algún posible ruido existente debido a la realización del usuario durante la fase de captura. También se indica en (Naftel & Khalid, 2006) que seleccionando las 5 primeras componentes de la Transformada de Fourier, la principal información siempre se conserva.

Al contrario que en el paper anteriormente referenciado, este sistema no solamente sigue la trayectoria de un único punto, sino de un esqueleto completo compuesto por 15 *joints* (Ecuación 18).

$$\vec{p}_j(t) = (x_j, y_j, z_j),\ j \in [1,9] \tag{18}$$

Por lo tanto, se calcula la FFT de cada secuencia de (18) para dar lugar a una nueva matriz de triplas de *joints* (Ecuación 19).

$$\tilde{\vec{P}}_j = FFT(\vec{p}_j(t)) = (\tilde{X}_j, \tilde{Y}_j, \tilde{Z}_j) \tag{19}$$

Para terminar de construir el patrón de movimiento, las trayectorias FFT de cada punto son concatenadas para componer un único vector Para reducir incluso más la dimensión del patrón de movimiento final, solamente los *joints* principales son introducidos al sistema final. Se ha determinado que los *joints* a seleccionar son cabeza y extremidades, descartando el cuello, hombros, caderas y torso. Esto es debido a que el rango de movimiento de estos *joints* es bastante limitado. Por último, un subvector de 9 elementos es construido usando las partes reales e imaginarias de las primeras 5 componentes de la FFT. El patrón de movimiento se representa por lo tanto uniendo los sub-vectores de Fourier de cada coordenada de cada *joint*, es decir, 9 elementos de la FFT (0…4) × 3 coordenadas (x-y-z) × 9 *joints* (k) = 243 (Ecuación 20).

$$P = \bigcup_{k=1}^{9} \begin{pmatrix} \tilde{X}_{k_0}, \mathrm{Re}\{\tilde{X}_{k_1}\}, \mathrm{Im}\{\tilde{X}_{k_1}\}, \ldots, \mathrm{Im}\{\tilde{X}_{k_4}\}, \\ \tilde{Y}_{k_0}, \mathrm{Re}\{\tilde{Y}_{k_1}\}, \mathrm{Im}\{\tilde{Y}_{k_1}\}, \ldots, \mathrm{Im}\{\tilde{Y}_{k_4}\}, \\ \tilde{Z}_{k_0}, \mathrm{Re}\{\tilde{Z}_{k_1}\}, \mathrm{Im}\{\tilde{Z}_{k_1}\}, \ldots, \mathrm{Im}\{\tilde{Z}_{k_4}\} \end{pmatrix} \quad (20)$$

Otra restricción es que el valor de los datos de entrada ha de encontrarse entre 0 y 1 en conjunto, por lo tanto se aplicará una normalización min-max final, sobre todos los componentes del conjunto de patrones (no sólo por columnas).

Por último, para poder ser introducidos a la red neuronal los patrones han de complementarse doblando su tamaño original (Ecuación 21), y como consecuencia la red trabaja con vectores de 243×2=486 componentes. Esto es otro motivo por el que la reducción de las dimensiones se torna necesaria, y las opciones de configuración quedan fijas a excepción de la rectificación.

$$P' = [P, 1 - P] \quad (21)$$

Experimento 1

El primer experimento se realiza con la siguiente configuración (el número de los *joints* se corresponde con el asignado en la Figura 12):

- Rectificación: Centro de masas
- *Joints* seleccionados: Sí (1 4 5 7 8 11 12 14 15)
- Deltas: No
- FFT: Sí (tamaño 64)
- Construcción de patrones: Sí
- Normalización: MinMax (sobre todas las componentes)

La tasa de acierto media para el experimento es: 93,05%, y la matriz de confusión se muestra en la Tabla 2.

	Agacharse	Jumping Jack	Saltar adelante	Saltar en el sitio	Galopar de lado	Andar	Saludar 1 mano	Saludar 2 manos
Agacharse	9							
Jumping Jack		9						
Saltar adelante			8			1		
Saltar en el sitio				9				
Galopar de lado			1		8			
Andar			3			6		
Saludar 1 mano							9	
Saludar 2 manos								9

Tabla 2: Matriz de confusión para reconocimiento mediante red neuronal - Experimento 1

También pueden visualizarse de forma gráfica estos resultados en la Figura 26.

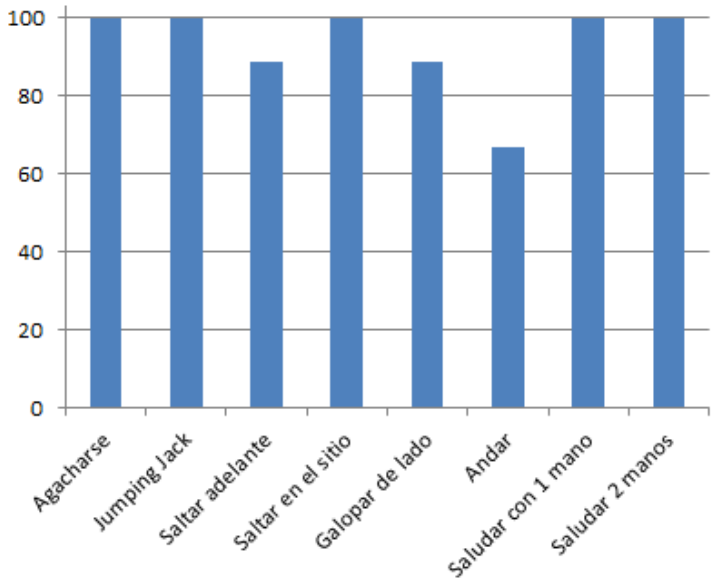

Figura 27: Tasas de acierto para reconocimiento mediante red neuronal - Experimento 1

Las principales conclusiones que pueden observarse es que los errores se producen en movimientos similares en los que existen transiciones rápidas del tronco inferior: saltar adelante, galopar de lado y andar.

Experimento 2

El segundo experimento se realiza con la siguiente configuración:

- Rectificación: Jerarquía
- *Joints* seleccionados: Sí (1 4 5 7 8 11 12 14 15)
- Deltas: No
- FFT: Sí (tamaño 64)
- Construcción de patrones: Sí
- Normalización: MinMax (sobre todas las componentes)

La tasa de acierto media para el experimento es: 93,05%. La matriz de confusión se muestra en la Tabla 3, y las tasas de acierto en la Figura 27.

	Agacharse	Jumping Jack	Saltar adelante	Saltar en el sitio	Galopar de lado	Andar	Saludar 1 mano	Saludar 2 manos
Agacharse	9							
Jumping Jack		9						
Saltar adelante			7		1	1		
Saltar en el sitio				9				
Galopar de lado			1		8			
Andar			1		1	7		
Saludar 1 mano							9	
Saludar 2 manos								9

Tabla 3: Matriz de confusión para reconocimiento mediante red neuronal - Experimento 2

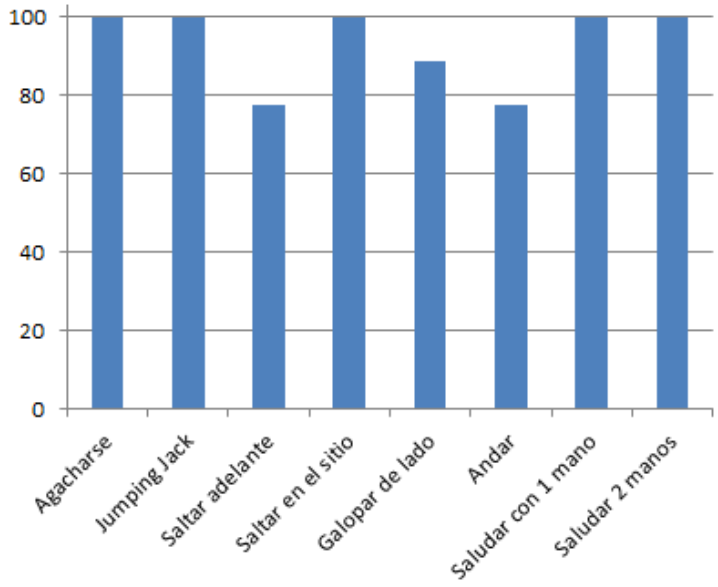

Figura 28: Tasas de acierto para reconocimiento mediante red neuronal - Experimento 2

Al realizar rectificación, la tasa de acierto permanece constante, aunque los resultados varían mínimamente. Los errores siguen estando en los movimientos mencionados en el experimento 1, salvo que algo más distribuidos.

Experimento 3

En este experimento se toma la configuración anterior, y se usan las deltas de movimiento en lugar de las trayectorias rectificadas originales:

- Rectifiación: Jerarquía
- *Joints* seleccionados: Sí (1 4 5 7 8 11 12 14 15)
- Deltas: Sí
- FFT: Sí (tamaño 64)

- Construcción de patrones: Sí
- Normalización: MinMax (sobre todas las componentes)

Como conclusión, los resultados de aplicar las deltas presentan una tasa de acierto inferior a las anteriores, del 75%.

Experimento 4
Dado que no hay más combinaciones de opciones de configuración relevantes, se realiza un último experimento con un *superset* del *dataset* original, añadiendo dos nuevos movimientos "conflictivos": correr y saltar a la pata coja. Igualmente, se utilizan todos los usuarios para evaluar el sistema.

Los parámetros de la red neuronal son los mismos, salvo que el número de *epoch* es 1.

Se utilizaron las configuraciones de los experimentos 1 y 2 (las más altas), obteniendo los siguientes resultados (Tabla 4):

- Centro de masas: 68,89%
- Jerarquía: 66,67%

	Agacharse	Jumping Jack	Saltar adelante	Saltar en el sitio	Correr	Galopar de lado	Pata coja	Andar	Saludar 1 mano	Saludar 2 manos
Agacharse	9									
Jumping Jack		9								
Saltar adelante			9							
Saltar en el sitio				9						
Correr			3		6					
Galopar de lado			4		3	2				
Pata coja			8		1					
Andar			8		1					
Saludar 1 mano									9	
Saludar 2 manos										9

Tabla 4: Matriz de confusión para reconocimiento mediante red neuronal - Experimento 4

Las conclusiones principales es que los nuevos movimientos entran en conflicto con los que producían errores ya en los experimentos anteriores, provocando nuevos errores de clasificación.

4.1.3. Reconocimiento mediante distancias

El segundo esquema de reconocimiento tiene en cuenta el cálculo de distancias para clasificar los movimientos. Este sistema es más flexible, ya que se computará

el algoritmo DTW sobre trayectorias individuales, combinando los resultados a posteriori.

Esquema de reconocimiento

Basándonos en el diseño propuesto en la Figura 22, se presenta una metodología a seguir para los experimentos realizados mediante el uso de distancias. Este esquema puede verse descrito en la Figura 28.

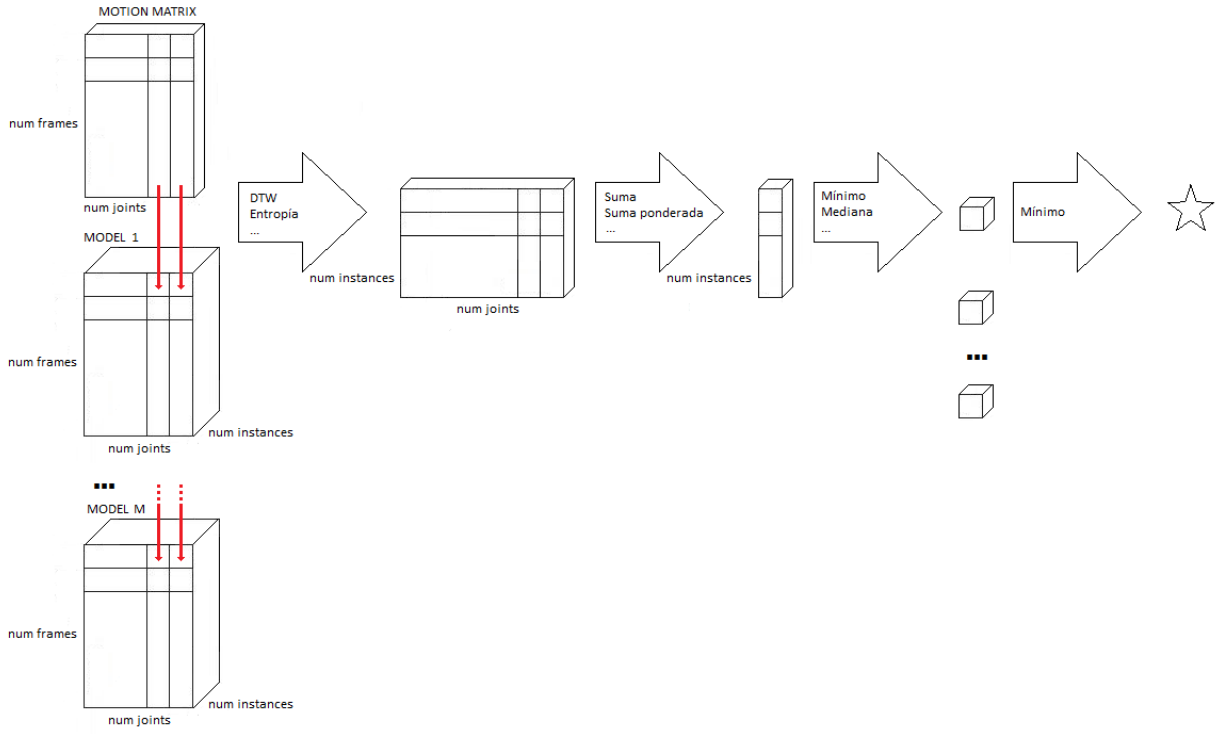

Figura 29: Esquema usando en el reconocedor por distancias

1. Primero, se construyen una serie de modelos de cada movimiento, almacenando las instancias de entrenamiento del sistema deseadas. Ante la entrada de un movimiento desconocido, se realizan una serie de operaciones.
2. Se calcula un valor de distancia entre cada par de trayectorias (*num joints*), de la matriz de entrada y cada instancia del modelo (*num instances*). Este proceso puede hacerse mediante el uso de la técnica DTW, entropía.
3. El siguiente paso es combinar las distancias de las trayectorias calculadas (*num joints*) para cada instancia (*num instances*). Para ello se proponen las técnicas de suma y suma ponderada.

4. A continuación se tiene un único valor de distancia para la entrada y cada instancia del modelo (*num instances*). Para combinar estos valores en un único valor, los métodos más utilizados son el mínimo, media, mediana. En los experimentos realizados, se utiliza la función mínimo, ya que es la que mejores resultados ofrece en la mayor parte de las veces (Pascual-Gaspar, Faundez-Zanuy & Vivaracho-Pascual, 2011).

Una vez se tiene el valor final de cada modelo, se tomará la decisión en base al que consiga una puntuación menor, es decir, el mínimo.

Suma ponderada: Cálculo de pesos

En el esquema descrito en la sección anterior, aparece una técnica de combinación mediante suma ponderada. Es similar a la suma normal, salvo que cada elemento es multiplicado por un peso antes de sumarlo. La complejidad viene a la hora de decidir qué peso asignar a cada atributo, en este caso, a la distancia entre las trayectorias de dos *joints* (en una de sus coordenadas). La idea básica sería dar más importancia a las trayectorias que permiten distinguir mejor un movimiento.

Puede realizarse un análisis de cómo se diferencian las trayectorias entre sí mediante el test de Wilcoxon de signo (Sección 3.3.1). Podemos comparar pares de matrices de movimiento usando todos los usuarios, y obteniendo para cada componente si es sustancialmente diferente o no. Por ejemplo, para el movimiento 0 con el movimiento 1 se obtiene la distribución de hipóstesis de la Figura 29. Como puede verse, las trayectorias asociadas con la cabeza y las manos son claramente diferentes (zona en blanco).

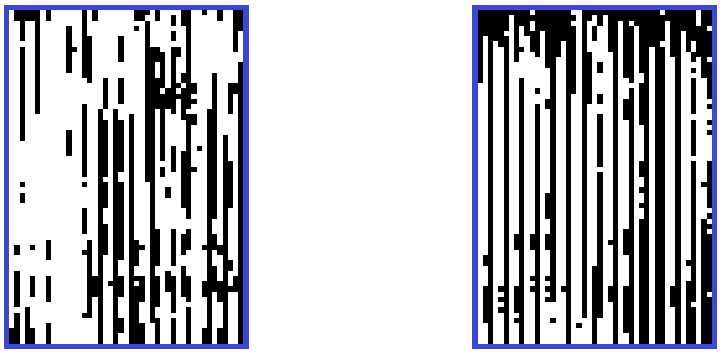

Figura 30: Test de signo para agacharse frente a jumping jack (izda) y agacharse frente a saltar hacia adelante (dcha)

Si se comprueba qué trayectorias permanecen siempre en blanco al comparar un movimiento con el resto, puede determinarse qué trayectorias son las más representativas. Para ello, se realiza una operación de AND lógico para cada punto, entre los tests de signo de un movimiento con los demás (Ecuación 22).

$$h_{ij}^{m} = s_{ij}^{m} AND s_{ij}^{n}, n \in [1,8], i \in [1,45], j \in [1,64] \qquad (22)$$

Con ello se consiguen las matrices que se muestran en la Figura 30.

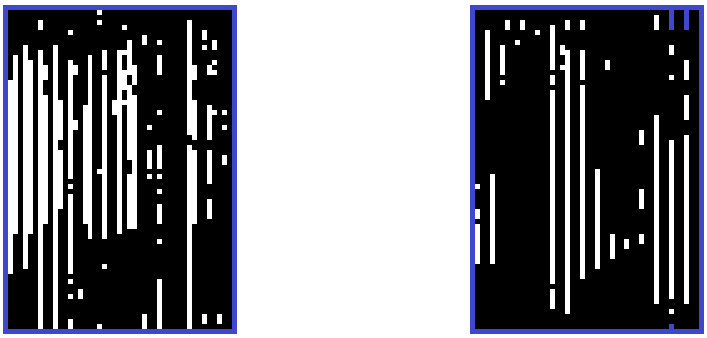

Figura 31: Test de signo combinado para agacharse (izda) y saltar hacia adelante (dcha)

A continuación es necesario transformar estos valores en una puntuación única para cada trayectoria (columna). Es necesario tener en cuenta que hay trayectorias que son solo "parcialmente" distintas. Además, dado que se trata de ponderar distancias, la teoría sería crear un peso que disminuyese el valor de estas. Con esto en mente, se ha diseñado la fórmula que aparece descrita en la Ecuación 23, usando como técnica de combinación la media.

$$w_i^m = 1 - mean(h_i^m) \qquad (23)$$

Existirán por tanto un conjunto de pesos por cada movimiento contemplado en el sistema. Concatenando estos valores uno debajo de otro, puede visualizarse el resultado en la Figura 31. Los puntos más oscuros indican que se les asigna más importancia (menor valor) ya que discriminan mejor una característica.

Figura 32: Matriz (8×45) con los 45 pesos para cada uno de los 8 movimientos

Hay que aclarar, que antes de realizar el test de signo es necesario aplicar un procesamiento a las matrices de movimiento de entrada igual al que se usará luego en el sistema de reconocimiento. En este caso, como pre-procesamiento se aplicó una simple rectificación por centro de masas.

Experimento 1

El primer experimento se realiza con la siguiente configuración:

- Rectificación: Centro de masas
- *Joints* seleccionados: No (todos)
- Deltas: No
- FFT: No
- Construcción de patrones: No
- Normalización: No

La configuración de distancias es:

- Algoritmo de cálculo de distancias: DTW
- Combinación de resultados: Suma ponderada

La tasa de acierto media para el experimento es: 91,67%. La matriz de confusión se muestra en la Tabla 5, y las tasas de acierto en la Figura 32.

	Agacharse	Jumping Jack	Saltar adelante	Saltar en el sitio	Galopar de lado	Andar	Saludar 1 mano	Saludar 2 manos
Agacharse	9							
Jumping Jack		9						
Saltar adelante			6			3		
Saltar en el sitio				9				
Galopar de lado					8	1		
Andar			1		1	7		
Saludar 1 mano							9	
Saludar 2 manos								9

Tabla 5: Matriz de confusión para reconocimiento mediante distancias - Experimento 1

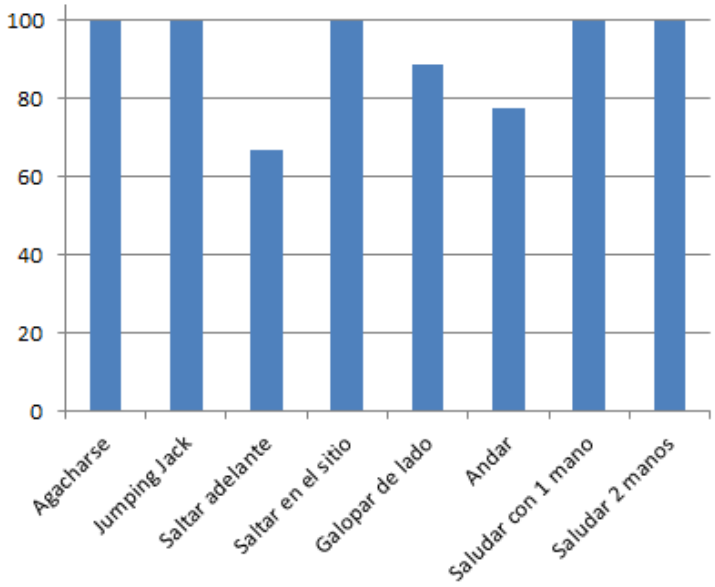

Figura 33: Tasas de acierto para reconocimiento mediante distancias - Experimento 1

Las principales conclusiones que pueden observarse es que, al igual que en el caso de la red neuronal, los errores se producen en movimientos en los que existen transiciones rápidas y del tronco inferior: saltar adelante, galopar de lado y andar.

Experimento 2
El segundo experimento se realiza modificando el tipo de rectificación:

- Rectificación: Jerarquía
- *Joints* seleccionados: No (todos)
- Deltas: No
- FFT: No
- Construcción de patrones: No
- Normalización: No

La configuración de distancias es:

- Algoritmo de cálculo de distancias: DTW
- Combinación de resultados: Suma ponderada

La tasa de acierto media para el experimento es: 88,89%. En este caso, los movimientos en conflicto siguen siendo saltar adelante, galopar de lado y andar, además de que la jerarquía empeora la clasificación.

Experimento 3
En este experimento se analiza el efecto de no usar suma ponderada:

- Rectificación: Centro de masas
- *Joints* seleccionados: No (todos)
- Deltas: No
- FFT: No
- Construcción de patrones: No
- Normalización: No

La configuración de distancias es:

- Algoritmo de cálculo de distancias: DTW
- Combinación de resultados: Suma

La tasa de acierto media para el experimento es: 91,67%. Al analizar los errores, se observa que los errores son los mismos que los del experimento 1, siendo la

matriz de confusión igual que la que aparece en la Tabla 5. Por lo tanto, se deduce que la suma ponderada no tiene ningún efecto en la prueba.

Experimento 4
A continuación se aplicará normalización a los vectores de características. En el caso de usar DTW es muy común utilizar la normalización Z en las entradas al sistema:

- Rectificación: Centro de masas
- *Joints* seleccionados: No (todos)
- Deltas: No
- FFT: No
- Construcción de patrones: No
- Normalización: Z-Norm

La configuración de distancias es:

- Algoritmo de cálculo de distancias: DTW
- Combinación de resultados: Suma ponderada

La tasa de acierto media para el experimento es: 70,83%. En este caso la normalización no ayuda a mejorar la precisión del sistema.

Continuando con esta tanda de experimentos, se llevan a cabo dos pruebas más, modificando una opción distinta en cada uno:

- Rectificación: Centro de masas
- Combinación de resultados: Suma

Como en casos anteriores, la aplicación de estas técnicas no afecta a la precisión del sistema, que sigue siendo 70,83%. Se determina por tanto que la normalización Z no es beneficiosa para el esquema de reconocimiento planteado. Asimismo, visto que no afecta al resultado de las pruebas, se desestima el uso de jerarquía y suma ponderada en los experimentos posteriores, usándose las opciones de centro de masas y suma por ser más sencillas.

Experimento 5
En estas pruebas se aplica la Transformada de Fourier a las secuencias de entrada, para determinar si son más sencillas de discriminar en el dominio frecuencial:

- Rectificación: Centro de masas
- *Joints* seleccionados: No (todos)

- Deltas: No
- FFT: Sí (tamaño 64)
- Construcción de patrones: No
- Normalización: No

La configuración de distancias es:

- Algoritmo de cálculo de distancias: DTW
- Combinación de resultados: Suma

La tasa de acierto media para el experimento es: 43,05%. Al ser una tasa de acierto demasiado baja, se plantea que el formato de los datos de entrada tras aplicar Fourier no sea el más adecuado. Se añade normalización para intentar buscar una mejor combinación de opciones:

- Normalización: Z-Norm

En este caso, la tasa de acierto media para el experimento es: 23,61%. Se deduce por tanto que, mientras que la FFT es de gran ayuda a la hora de condensar información, por sí sola no ofrece ninguna mejora al sistema.

Experimento 6
Estos experimentos se llevan a cabo aplicando deltas de movimiento a las trayectorias. La configuración es la siguiente:

- Rectificación: Centro de masas
- *Joints* seleccionados: No (todos)
- Deltas: Sí
- FFT: No
- Construcción de patrones: No
- Normalización: No

La configuración de distancias es:

- Algoritmo de cálculo de distancias: DTW
- Combinación de resultados: Suma

La tasa de acierto media para el experimento es: 94,44%. La matriz de confusión se muestra en la Tabla 6, y las tasas de acierto en la Figura 33.

	Agacharse	Jumping Jack	Saltar adelante	Saltar en el sitio	Galopar de lado	Andar	Saludar 1 mano	Saludar 2 manos
Agacharse	9							
Jumping Jack		8						1
Saltar adelante			8		1			
Saltar en el sitio				9				
Galopar de lado					9			
Andar			1		1	7		
Saludar 1 mano							9	
Saludar 2 manos								9

Tabla 6: Matriz de confusión para reconocimiento mediante distancias - Experimento 6

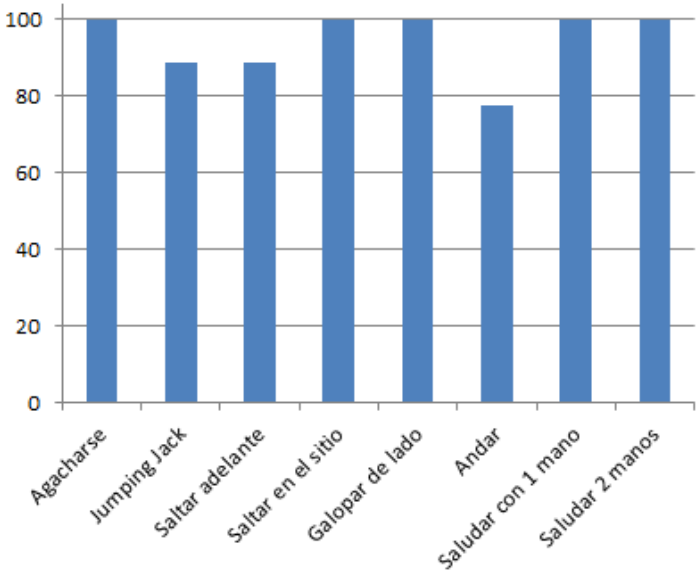

Figura 34: Tasas de acierto para reconocimiento mediante distancias - Experimento 6

Esta configuración es altamente beneficiosa para el esquema de reconocimiento de movimientos basado en distancias. Se consigue la tasa de acierto más alta hasta el momento, superando incluso a la de la red neuronal. Se detectan errores mínimos en los movimientos conflictivos anteriormente mencionados, además de una confusión espontánea entre el movimiento de *jumping jack* y saludar con las dos manos.

Dado el éxito de las deltas de movimiento, se prueba esta técnica combinada con otras opciones de configuración:

- FFT: Sí
- Normalización: Z-Norm

Al aplicar deltas junto con FFT, la precisión es del 40,28%. Cuando se aplica normalización, la tasa es del 73,61%. Si se aplican ambas técnicas junto con las deltas, la precisión baja a 20,83%. Se concluye que las deltas de movimiento tienen un efecto muy positivo en el sistema, aplicadas de forma independiente.

Experimento 7

Una posible configuración aplicable al uso de distancias es introducir los patrones creados para la red neuronal, comparando con el modelo un único patrón, en lugar de 45 trayectorias diferentes. La configuración para tal experimento es:

- Rectificación: Centro de masas
- *Joints* seleccionados: Sí (1 4 5 7 8 11 12 14 15)
- Deltas: No
- FFT: Sí (tamaño 64)
- Construcción de patrones: Sí
- Normalización: MinMax

La configuración de distancias es:

- Algoritmo de cálculo de distancias: DTW
- Combinación de resultados: Suma

La tasa de acierto media para el experimento es: 93,05%, y la matriz de confusión se muestra en la Tabla 7. Las tasas de precisión individuales aparecen en la Figura 34.

	Agacharse	Jumping Jack	Saltar adelante	Saltar en el sitio	Galopar de lado	Andar	Saludar 1 mano	Saludar 2 manos
Agacharse	9							
Jumping Jack		9						
Saltar adelante			6			3		
Saltar en el sitio				9				
Galopar de lado					9			
Andar			2			7		
Saludar 1 mano							9	
Saludar 2 manos								9

Tabla 7: Matriz de confusión para reconocimiento mediante distancias - Experimento 7

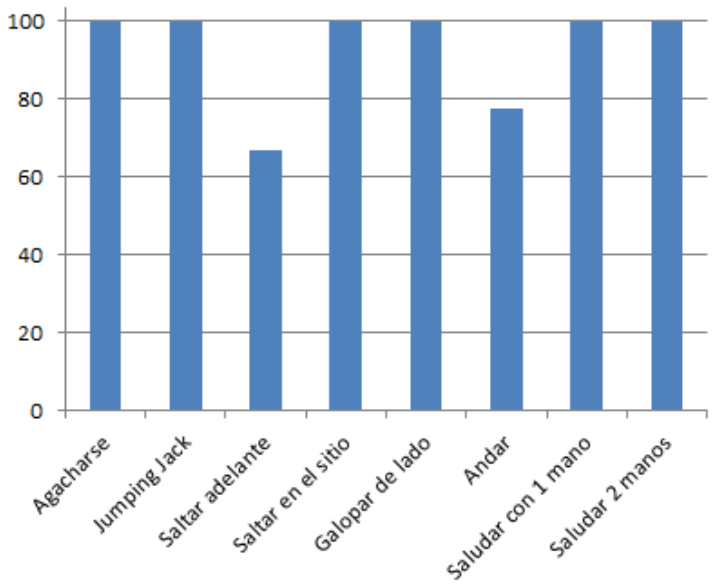

Figura 35: Tasas de acierto para reconocimiento mediante distancias - Experimento 7

La precisión del sistema es la misma que en el experimento 1 de la red neuronal, aunque con una distribución de errores diferente. En este caso no hay error en el movimiento de galopar. Sin embargo, y como veremos más adelante, el cálculo de distancias es computacionalmente más costoso que la red. Se concluye por tanto que los patrones creados en la Sección 4.1.2 son efectivos en diversos tipos de clasificadores, y permiten una clasificación precisa de movimientos heterogéneos.

Experimento 8

Además del cálculo de distancias mediante DTW, se ha propuesto el uso de la medida de entropía, más concretamente, Entropía Aproximada Cruzada. Para realizar este experimento, se utiliza la configuración que mayor tasa de acierto ha conseguido hasta el momento (experimento 6):

- Rectificación: Centro de masas
- *Joints* seleccionados: No (todos)
- Deltas: Sí
- FFT: No
- Construcción de patrones: No
- Normalización: No

La configuración de distancias es:

- Algoritmo de cálculo de distancias: Entropía Aproximada Cruzada
- Combinación de resultados: Suma

La tasa de acierto media para el experimento es: 26,39%, demasiado baja como para considerar el uso de esta técnica. Aun así, también fueron probadas otras configuraciones de entropía, pero sin demasiado éxito. Por lo tanto se descarta el uso de Entropía Aproximada Cruzada en el esquema de clasificación basado en distancias.

Experimento 9

Por último, y al igual que se ha realizado con la red neuronal, se propone un experimento utilizando un *superset* del *dataset* original que incluye las acciones de correr y saltar a la pata coja. Se utilizan todos los usuarios para evaluar el sistema.

La configuración elegida es la del experimento 6, por ser la más alta. La tasa de acierto conseguida es del 80%. La matriz de confusión aparece en la Tabla 8, y las tasas de acierto correspondientes a cada movimiento en la Figura 35.

	Agacharse	Jumping Jack	Saltar adelante	Saltar en el sitio	Correr	Galopar de lado	Pata coja	Andar	Saludar 1 mano	Saludar 2 manos
Agacharse	9									
Jumping Jack		8								1
Saltar adelante			8			1				
Saltar en el sitio				9						
Correr			2		2	2		3		
Galopar de lado						9				
Pata coja			5			2	2			
Andar			1			1		7		
Saludar 1 mano									9	
Saludar 2 manos										9

Tabla 8: Matriz de confusión para reconocimiento mediante distancias - Experimento 9

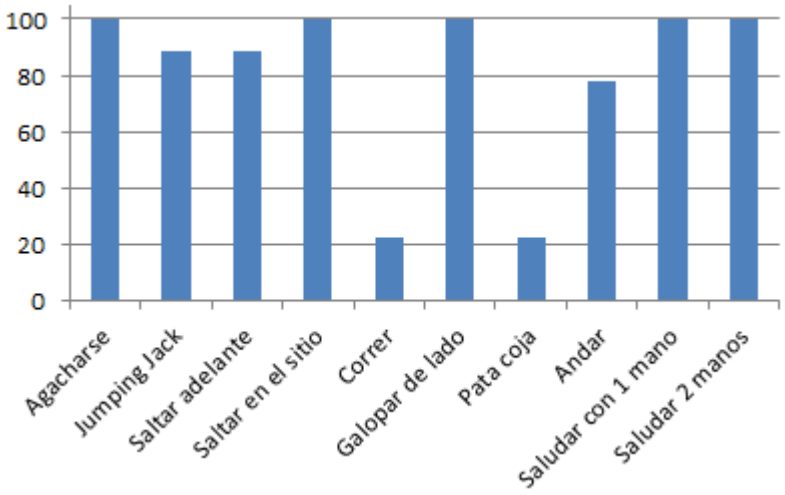

Figura 36: Tasas de acierto para reconocimiento mediante distancias - Experimento 9

Al igual que en el caso de la red neuronal, los nuevos movimientos perjudican al reconocimiento de los movimientos que ya anteriormente resultaban problemáticos, estos son: saltar adelante, correr, galopar de lado, pata coja y andar. Es especialmente notable la confusión entre pata coja y saltar hacia adelante, o cómo correr apenas tiene aciertos. Aun así, se supera la tasa de acierto de la red neuronal.

4.1.4. Comparación de tiempos

Durante la realización de los experimentos, se ha medido lo que tarda la ejecución del sistema de reconocimiento. Dado que los patrones se clasifican de forma secuencial, se ha tomado el tiempo necesario para reconocer un único patrón de entrada. La Tabla 9 recoge estos valores; se recuerda que se tienen 8 movimientos, 9 usuarios y se utiliza Left One Out para entrenar al sistema.

Sistema	**Tiempo (segundos)**
Red Neuronal	0,05
DTW	23
DTW + Patrones de Red Neuronal	9

Tabla 9: Tiempos de reconocimiento para un único movimiento

A la vista de estos tiempos, puede determinarse que la red neuronal se ejecuta mucho más rápidamente que el algoritmo de DTW basado en modelos, incluso cuando utilizan el mismo tipo de patrones.

4.2. Discusión de resultados

Tras la realización de todos los experimentos, pueden extraerse una serie de conclusiones. En primer lugar, se observa que los reconocedores son capaces de ofrecer una precisión elevada para una cantidad variada de movimientos ejecutados por usuarios de diversas características (hombres, mujeres, altos, bajos…).

En el uso de una red neuronal, el esquema de patrones utilizado es capaz de ofrecer una precisión elevada de forma eficiente. Se consigue reducir la dimensión en gran parte, mediante el uso de técnicas de selección de *joints*, FFT y concatenación. Las siguientes fórmulas muestran la reducción de tamaño

- Matriz de movimiento:
  - 64 *frames* × 15 *joints* × 3 coordenadas = 2880 componentes
- Patrón de red neuronal:
  - 9 componentes FFT × 9 *joints* × 3 coordenadas = 243 componentes
- Reducción conseguida:
  - 2880 ÷ 243 = 11,85 veces
  - 100 – 100/11,851 = 91,50% de reducción

Por lo tanto, se reduce el tamaño del problema en algo más de un orden de magnitud, sacrificando en la menor medida posible la precisión.

El sistema de reconocimiento mediante distancias, ha probado ser el más preciso, debido al uso de trayectorias completas, y a la comparación de las mismas mediante la técnica de DTW. DTW ofrece una mayor precisión, así como una mayor maleabilidad a la hora de realizar los modelos que representen los movimientos del sistema, permitiendo almacenar instancias específicas que representen un movimiento sin deformar las demás, como ocurre con las categorías de la red neuronal. La configuración más efectiva en este caso ha sido el uso de deltas de movimiento en las trayectorias. Es posible realizar una analogía entre la forma en la que el cerebro humano percibe el movimiento, y el uso de esta técnica. Es acertado, por tanto, pensar que la mejor forma de representar un movimiento sea mediante las variaciones de sus trayectorias en el espacio, relativas al punto inicial del movimiento. O al menos, esto ha demostrado ser cierto en este caso, por lo que se deduce este resultado.

Por otra parte, tanto la red neuronal como el método por distancias sufren fallos de clasificación en los mismos movimientos, que son aquellos que conllevan variaciones similares y muy rápidas en el tronco inferior: saltar adelante, galopar

de lado y andar. Para analizar más en detalle estos movimientos, se utiliza el método de diagrama de bigotes (Sección 3.3.1). La Figura 36 muestra los diagramas de bigotes asociados a los movimientos de saltar hacia adelante, andar y agacharse para poder compararles. Puede observarse cómo los *joints* de los dos primeros comparten muchas características, con lo que no es de extrañar que se produzcan los errores de clasificación mencionados, dado que las trayectorias siguen caminos similares. Por otra parte, el movimiento de agacharse es siempre reconocido, y puede verse que las trayectorias de los *joints* tienen unos valores distintivos, al compararles con los dos anteriores.

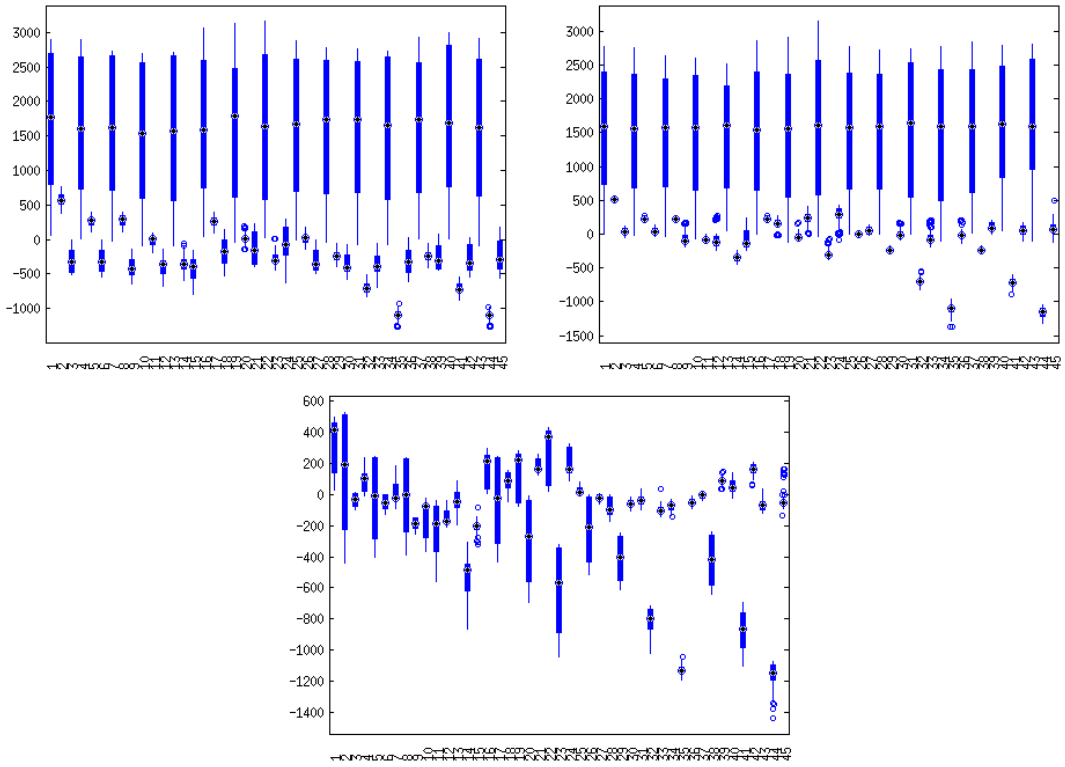

Figura 37: Diagrama de bigotes para saltar hacia adelante (arriba izda), andar (arriba dcha) y agacharse (abajo)

La Figura 37 muestra dos ejecuciones del movimiento correr, en las que la distribución de valores es algo diferente. En el primer caso, el movimiento es muy rápido al principio y más lento al final, mientras que en el segundo la distribución es uniforme. Este es un ejemplo de variabilidad dentro de una misma clase.

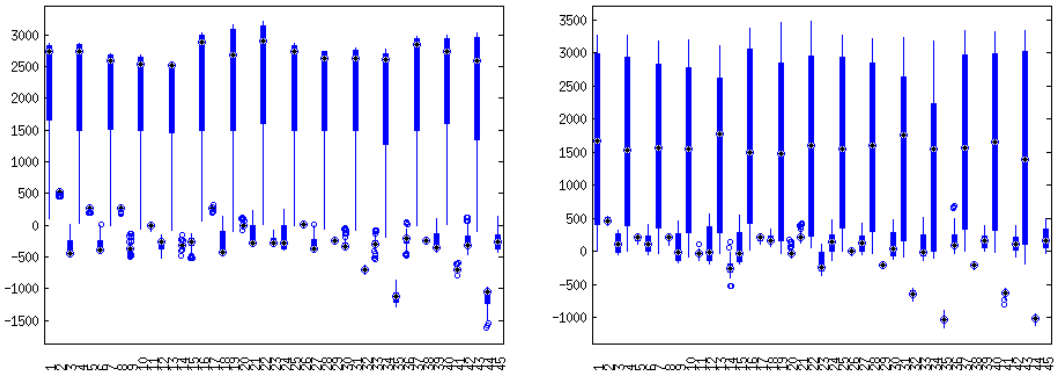

Figura 38: Diagrama de bigotes para correr, con dos usuarios diferentes

Esto implica que hay acciones que, no solamente son muy variables en cada ejecución dentro de la misma clase (intra-movimiento), sino que además puede que se parezcan más a las ejecuciones de otros movimientos (inter-movimiento). Estos errores de clasificación también pueden atribuirse al sistema de captura. No hay que olvidar que la arquitectura de Kinect utilizada (Sensor Kinect + OpenNI + Wrapper MatLab®) no ofrece una tasa fps óptima. Más concretamente, de los 30 fps que ofrece Kinect, se consiguen unos 64 *frames* / 5 seg ≈ 13 fps, algo inferior a la mitad. Por tanto, movimientos pequeños y de repeticiones rápidas, aunque detectados, pueden no ser capturados con la precisión deseada.

Otra conclusión relevante es que, mientras que una rectificación del esqueleto se considera imprescindible en el pre-procesamiento de los patrones de entrada, no se han notado mejoras aparentes entre utilizar centro de masas y jerarquía.

La suma ponderada no ha tenido ningún efecto, ni positivo, ni negativo, sobre los experimentos realizados. Es muy posible a que se deba a que los valores de los pesos sean muy bajos, y apenas modifiquen los resultados. Podría considerarse aumentar su influencia aumentando la magnitud de los valores, o eliminando trayectorias enteras, dejando solamente las más significativas.

En cuanto al coste de cálculo de las técnicas utilizadas, DTW muestra unos tiempos de ejecución muy elevados, aunque ya es bien conocido su complejidad computacional (Pascual-Gaspar, Cardeñoso-Payo, Vivaracho-Pascual, 2009). Sin embargo, y al igual que ya se ha hecho con el algoritmo de una red neuronal Fuzzy ARTMAP (Martínez-Zarzuela, Díaz-Pernas, Tejero de Pablos, Perozo-Rondón, Antón-Rodríguez, González Ortega, 2011), es posible paralelizar el algoritmo de DTW para su ejecución en GPU. Usando las mismas técnicas, se optimizarían los parámetros de la GPU, y se buscaría ejecutar el mayor número posible de operaciones independientes en paralelo. En una primera aproximación,

se observa que para identificar un movimiento, el esquema requiere realizar un cálculo de distancias por cada modelo de movimiento, por cada instancia del modelo, y por cada trayectoria dentro de cada instancia:

9 modelos × 8 instancias/modelo × 45 trayectorias/instancia = 3240 distancias

Estas distancias es posible calcularlas en paralelo, ya que son independientes, y luego realizar las técnicas de combinación con los valores obtenidos. Por tanto, no se descarta el uso de DTW pese a su coste computacional. Otra forma de aprovechar la velocidad de la red neuronal con la precisión de DTW, sería computar DTW entre cada instancia y los modelos de las 2 o 3 clases que obtuvieran una mayor puntuación en la red neuronal.

A continuación, en la Tabla 10 se presenta un ranking de los resultados más relevantes obtenidos en los experimentos, ordenados de mayor a menor precisión. En este ranking no aparecen los experimentos realizados con el *superset* de 10 movimientos.

Experimento	**Precisión**
Distancias #6 (Deltas)	94,44%
Red Neuronal #1&2 (Básico)	93,05%
Distancias #7 (Patrones)	93,05%
Distancias #1&3 (Básico)	91,67%
Distancias #2 (Jerarquía)	88,89%
Red Neuronal #3 (Deltas)	75%
Distancias #4 (Z-Norm)	70,83%
Distancias #5 (FFT)	43,05%
Distancias #8 (Entropía)	26,39%

Tabla 10: Ranking de los experimentos realizados de acuerdo a la precisión del sistema

En la siguiente sección se realiza una comparación de los sistemas desarrollados con otros sistemas de reconocimiento de movimiento similares del estado del arte.

4.3. Comparación con sistemas del estado-del-arte

A continuación se comparan los resultados obtenidos y analizados en las secciones anteriores con los de otros sistemas similares.

El experimento realizado en (Sheikh & Sheikh, 2005) consta de cuatro movimientos: sentarse, levantarse, caer, andar, bailar y correr, realizados por varios usuarios en 51 instancias. Su método consigue clasificar correctamente la mayor parte de las acciones, sin embargo levantarse y sentarse son mejor clasificadas que andar y correr, ya que estas dos se parecen más entre sí. Nuestro caso puede verse reflejado en este ejemplo, en el que dos acciones parecidas son

más difíciles de reconocer. Sin embargo, nuestro sistema apuesta por más variedad de movimientos en el *dataset* utilizado.

En (Raptis & Kirovski, 2011), utilizaron 28 movimientos de baile diferentes distribuidos en 11361 videos. Las instancias fueron obtenidas mediante Kinect a 30 fps sobre distintos entornos, y se normalizaron para obtener 120 muestras en 8 ritmos (4 segundos). Dado la gran cantidad de instancias a su disposición, la precisión se realiza mediante 4-validación-cruzada, con una división de entrenamiento/test del 70/30. Pese a que muchos de los bailes comparten movimientos "primarios", se consigue una tasa de acierto del 96,90% de media. Aunque nuestro sistema no llega a esa tasa de precisión, no tiene tantos requerimientos, acepta prácticamente cualquier movimiento. También sería necesario contar con más videos en nuestro *dataset*, añadiendo movimientos y usuarios, para poder realizar distribuciones de entrenamiento distintas de Leave One Out. También, en el paper los autores reconocen que su sistema utiliza movimientos periódicos que siguen un ritmo específico, además de utilizar un bailarín profesional como oráculo.

En (Chen & Kotani, 2012) se dispone de 7 acciones diferentes: subir y bajar escaleras, saltar hacia adelante, saltar hacia arriba, boxear, swing de golf, correr y andar. Cada acción tiene 15 videos en 3D, de los cuales 2/3 se usan para el entrenamiento y 1/3 restante se usa para el test. El experimento se repite 10 veces y se calcula la media. La precisión media final, del 88,70%, resulta reducida ya que tres acciones (subir/bajar escaleras, saltar arriba y saltar adelante) sufren de solapamiento en las posiciones de los patrones. Una de las soluciones que proponen es investigar las diferencias mutuas entre pares de movimientos similares. Este trabajo también reconoce la dificultad de clasificar movimientos similares (con solapamiento), sin embargo nuestro sistema consigue mayor precisión en condiciones similares.

En (Sempena, Maulidevi & Aryan, 2011) las acciones utilizadas son saludar, aplaudir, golpe de tenis, boxear, patada lateral y footing. Cada acción se repite 5 veces, con el actor mirando a cámara, y grabando a 5 fps. Se dice que el método es aplicable para el reconocimiento, sin embargo no aportan ningún tipo de información sobre la precisión del sistema desarrollado. Los resultados al involucrar partes del tren inferior son los más problemáticos para el sistema. También desaconsejan DTW para el reconocimiento de acciones complejas realizadas por mucha gente. En nuestro trabajo, se tienen más movimientos y usuarios, además de que se realiza un uso más avanzado de la técnica DTW, combinándola con pre-procesamiento de las trayectorias.

El trabajo de (Chua, Leman & Pham, 2011) tiene como datos de entrada los *datasets* de:

- Weizmann: 9 usuarios y 10 acciones. *bend*, *jumping-jack*, *jump-forward*, *jump-in-place*, *run*, *gallop-sideways*, *jump-forward-one-leg*, *walk*, *wave-one-hand*, and *wave-two-hands*. Emplean siluetas.
- KTH: 25 usuarios y 6 acciones: *boxing*, *handclapping*, *handwaving*, *jogging*, *running*, and *walking*. Emplean imágenes en gris.

La evaluación se realiza entrenando un clasificador basado en K-Vecinos-Cercanos mediante Leave One Out Cross Validation, para varios tamaños de K. En el caso de ambos *datasets*, la mejor configuración es: K = 3, combinación mediante sum-rule, uso de ambos tipos de características (forma y movimiento). Individualmente, las características de movimiento son más discriminativas que las de forma. Destacan la utilidad de dar varios niveles de importancia a los datos de entrada usando diversas técnicas (suma, producto, min, max, mayoría). Los resultados superan a los obtenidos usando combinación por mayoría de voto. Esto puede deberse a la facilidad que ofrece esta técnica para distinguir ambigüedades debidas a la existencia de poses parecidas en movimientos diferentes.

Pueden verse varias similitudes entre el planteamiento de este trabajo y el nuestro. Mientras que ellos utilizan más movimientos y técnicas en 2D, consiguen aplicar de forma satisfactoria la combinación mediante suma ponderada. Sus resultados de atributos de forma y movimiento pueden relacionarse con los de trayectorias y deltas de movimiento obtenidas en nuestro estudio.

En (Biswas & Basu, 2011) utilizan 8 gestos realizados por 5 sujetos: aplaudir, llamar a alguien, saludar, agitar los brazos, decir que no con la cabeza, decir que sí con la cabeza, manos detrás de la cabeza, barbilla descansando en la mano. El porcentaje de aciertos para cada clase es elevado, siendo los más complicados de reconocer "sí" y "no", dada su similitud y ausencia de movimiento apenas. El *dataset* utilizado en nuestro trabajo es más grande y variado; al no aportar tasas de precisión de su sistema no es posible realizar más comparaciones.

En (Lai, Konrad & Ishwar, 2012) hacen uso de Leave One Out Cross-Validation, consiguiendo una precisión del 97% para reconocer 8 gestos de la mano realizados por 20 sujetos. Destacan las limitaciones del uso de distancia Euclídea, como que requiere que las secuencias sean de la misma longitud y que además es sensible a desalineamientos temporales. La precisión conseguida es elevada, pero los movimientos son demasiado simples, además de seguir realizando Leave One Out pese a tener más cantidad de instancias en el *dataset*. En nuestro sistema, no

se contempla el uso de distancia Euclídea, dado que la gran variabilidad de las ejecuciones de los movimientos realizados dificultaría en gran medida el reconocimiento de los mismos.

Otra de las ventajas a destacar de nuestro sistema, que puede pasar desapercibida, es que las trayectorias originales son recuperables tras su pre-procesamiento, cosa que en otros sistemas no ocurre. Esto es debido a que es posible realizar las operaciones inversas, y permitiría utilizar nuevamente las trayectorias para su reconstrucción, etc. Además, la red neuronal elegida permite aprendizaje incremental de nuevas repeticiones, usuarios y movimientos en el sistema creado.

Capítulo 5: Conclusiones

Este trabajo presenta un estudio sobre la monitorización 3D con reconocimiento de acciones. La gran cantidad de aplicaciones para los sistemas de reconocimiento de acciones hacen que surjan nuevas investigaciones al respecto. Sin embargo, muchos de los sistemas son mejorables en cuanto a su precisión o la información extraída de la escena y el uso que hacen de ella. La aparición del dispositivo Microsoft® Kinect y otras cámaras de profundidad comerciales similares abren nuevas vías de investigación en las técnicas de interacción hombre-máquina mediante seguimiento y, en particular, en el reconocimiento de acciones del cuerpo humano. Sin embargo, la información del uso de estas herramientas, así como la existencia de *dataset* basados en esta tecnología son muy limitadas.

En primer lugar se ha llevado a cabo un análisis de los sistemas y herramientas actuales, enfocado en las implementaciones que usan esqueletos 3D y en el sensor Kinect, dadas sus ventajas específicas. Se ha estudiado la adquisición, modelado y reconocimiento de movimientos en este tipo de sistemas, atendiendo a distintas implementaciones con enfoque similar del estado del arte. Esto ha permitido detectar dificultades como las altas dimensiones o la variabilidad de los datos. Con esto en mente, se ha creado un enfoque propio, basado en la propia naturaleza del reconocimiento de movimientos, a partir del cual se han determinado una metodología y una serie de experimentos a llevar a cabo mediante la implementación de un Toolbox en MatLab®. El sistema propuesto realiza funciones de reconocimiento de acciones de cuerpo completo. Para evaluarlo, se construyó un *dataset* desde cero, con 8 movimientos realizados por 10 usuarios.

Mediante una serie de experimentos, se han encontrado las configuraciones más beneficiosas para conseguir una precisión muy elevada en dos tipos de clasificadores: una red neuronal basada en Adaptive Resonance Theory (ART) (93,05%) y un clasificador basado en Dynamic Time Warping (DTW) (94,44%). Los resultados muestran que es posible realizar una reducción de la dimensión de forma eficaz (~91,50%) sin sacrificar apenas precisión. Por otra parte, se ha visto la efectividad del uso de deltas sobre las trayectorias 3D del cuerpo a la hora de reconocer movimientos. El uso de deltas de movimiento implica utilizar la diferencia entre la posición de cada *joint* en el *frame* actual y el *frame* anterior. Este tipo de representación es afín a la forma en la que el ser humano detecta los movimientos, y permite reconocer variaciones bruscas en el movimiento más fácilmente. Ambos esquemas basados en trayectorias monitorizadas con Kinect han demostrado ser robustos y eficaces al ser aplicados al reconocimiento de movimientos.

Asimismo, se han descartado en este esquema otras técnicas, como son el uso de una jerarquía en la rectificación, además de la suma ponderada, siendo necesaria una revisión de las mismas para su uso. Los principales errores producidos en el sistema de reconocimiento son debidos a movimientos similares con altas variaciones en el tren inferior: saltar hacia adelante, galopar de lado, andar… Sin embargo, en las configuraciones ganadoras los fallos son esporádicos y no afectan al resto de movimientos. Una vez encontrada una aplicación práctica al sistema, es más sencillo determinar qué movimientos se desean incluir en el *dataset*. Atendiendo a los tiempos de ejecución, la red neuronal tiene un coste asumible debido a la reducción de la dimensión, pero trabajar con los modelos de DTW supone mayor carga computacional. Una solución plausible es realizar la implementación de este último sistema en GPU para acelerar su ejecución múltiples decenas de veces y facilitar su ejecución en tiempo real. Al comparar los resultados obtenidos con el resto de sistemas del estado del arte con enfoques similares, se determina que la tasa de acierto conseguida sitúa a los sistemas en una muy buena posición, siendo superado en precisión por sistemas que se limitan a reconocer gestos simples o movimientos periódicos.

Dada la elevada cantidad de aplicaciones asociadas al tema propuesto, el proyecto planteado tiene una implementación práctica directa, en favor de la innovación. Asimismo, dado los estudios implicados en el desarrollo del proyecto, éste puede considerarse beneficioso para otros investigadores que deseen realizar un sistema con una infraestructura similar en futuros trabajos.

5.1. Contribuciones y publicaciones

Mientras que los sistemas de reconocimiento de movimiento y monitorización 3D ganan en popularidad, aún no se ha encontrado el sistema óptimo para resolver este paradigma. El trabajo realizado supone una contribución más, que añade nuevos e interesantes resultados con los que comparar el resto de sistemas. Asimismo, la base de datos de movimientos ha sido grabada de forma particular, y una vez avanzado el trabajo podría ponerse a disposición junto con las ya existentes. De esta forma, la realización de este proyecto ha dado lugar a dos artículos enviados a dos revistas diferentes, una en inglés y otra en castellano respectivamente:

- M. Martínez-Zarzuela, F.J. Díaz-Pernas, A. Tejero-de-Pablos, D. González-Ortega, M. Antón-Rodríguez. 'Action recognition system based on human body tracking with depth images'. International Journal of Advanced Robotic Systems (En revisión).

- M. Martínez-Zarzuela, F.J. Díaz-Pernas, A. Tejero-de-Pablos, D. González-Ortega, M. Antón-Rodríguez. 'Diseño de un sistema de reconocimiento de acciones con adquisición 3D para interacción hombre-máquina'. Revista Ibérica de Sistemas y Tecnologías de la Información (En revisión).

5.2. Trabajo Futuro

Una ampliación directa de este trabajo sería mejorar la precisión del sistema. Una de las principales razones de realizar un Toolbox es que pueden realizarse nuevos experimentos de forma sencilla modificando unos pocos parámetros, con lo que pueden probarse nuevas configuraciones con poco esfuerzo. Asimismo, aprovechando que se dispone de dos sistemas de reconocimiento diferentes, podría usarse la salida de ambos para tomar la decisión final, introduciendo en cada uno una parte distinta de la jerarquía de esqueleto, por ejemplo. Existen diversos tipos de técnicas de voto para combinar varias salidas (Cho & Kim, 1995), como mayoría o Borda. Además existen técnicas avanzadas como la usada en (Okada & Hasegawa, 2008). En este trabajo se presenta una técnica avanzada para reconocimiento de movimientos 2D basada en la combinación de una red neuronal y el algoritmo DTW. La red neuronal consta de dos capas con aprendizaje no-supervisado, y es usada para eliminar ruido y representar su distribución. Los experimentos son realizados con un *dataset* manual que contiene 7 movimientos distintos (no-naturales) realizados por 5 usuarios. Se extrae del video el cambio de dirección del centroide corporal, y se calcula la diferencia entre *frames*. Mediante validación cruzada se consigue en los experimentos una tasa de acierto del 98%, y consigue superar a sistemas tradicionales como DTW y HMM.

En 2014 aparecerá en el mercado la nueva versión del sensor Kinect disponible para PC. Este nuevo hardware permitirá monitorizar un mayor número de *joints*, obtener imágenes de profundidad de mayor resolución, detectar rotaciones en los *joints* (uso de cuaterniones) y otro tipo de información mejorada. Todo ello será accesible mediante el SDK de Microsoft®, aunque posiblemente emerjan iniciativas independientes como las de OpenNI con la actual Kinect.

En cuanto a los sistemas actuales, además de las limitaciones mencionadas es posible detectar otras debilidades:

- Muchos de ellos no dan la suficiente importancia a la interfaz, y a cómo facilitar el uso al usuario, así como la presentación de la información.

- Una vez procesado el movimiento, éste no es usado para otras tareas como análisis de la calidad de la acción o reconstrucción del movimiento original.
- La información de profundidad de la escena permite reconocer mucho más que el movimiento, también obstáculos o utilizarse para generar entornos de realidad aumentada.

Por tanto, el proyecto planteado puede verse mejorado mediante nuevas ampliaciones.

5.2.1. Reconocimiento en tiempo real

Uno de los puntos donde cabe poner más énfasis es en la implementación de código ejecutable en GPU, que lleve a cabo tareas costosas como la clasificación mediante DTW o red neuronal. Esta aceleración permitiría encontrar y analizar acciones dentro de un *stream de frames*, lo cual evitaría la necesidad de segmentar el movimiento. Otra utilidad de paralelizar el algoritmo sería la posibilidad de utilizar varios clasificadores sobre la jerarquía de esqueleto diseñada, combinando los resultados a la salida de cada clasificador.

5.2.2. Reconstrucción de espacios 3D

Además de los movimientos detectados, es de gran utilidad tener en cuenta el resto de información capturada mediante los sensores 3D. Mientras que una cámara convencional ofrece una imagen plana en 2D, las cámaras con sensores 3D son capaces de obtener una representación tridimensional de los elementos presentes en la imagen. Esto es de gran utilidad ya que permite recrear en un ordenador lo ocurrido en un escenario real que está siendo grabado mediante una cámara que utilice información en profundidad. Además de utilizarse para este tipo de simulaciones, los espacios 3D permiten una mayor inmersión del usuario en un sistema y presentar de forma visual la información capturada, de forma similar a lo que ocurre en los sistemas de telepresencia (Okura, Kanbara & Yokoya, 2012).

Los paradigmas de realidad virtual y realidad aumentada permiten crear un entorno artificial tomando como base contenidos generados a partir de información de la realidad. Esto también es clave a la hora de presentar la información al usuario y la realización de interfaces hombre-máquina. Se contempla por tanto la inclusión de un sistema de gráficos por ordenador, o motor gráfico (p. ej. Unity) con el que realizar tareas como reconstruir escenas y movimientos monitorizados previamente, generación de escenarios y avatares

virtuales, etc. a partir de la integración de información del mundo real y contenido generado en el mundo virtual.

5.2.3. Interacción con el usuario

El sistema ha de ser capaz de comunicarse con el usuario, de forma cómoda y fácil de usar y entender para éste. Se plantea la implementación de interfaces intuitivas, con reconocimiento de voz, etc. La presentación de la información también es otra de los puntos clave a la hora de desarrollar la interfaz. Además, el sistema podría ser capaz de reconocer de forma biométrica quién es la persona delante de la máquina dada su complexión física y utilizar un perfil con información útil capturada anteriormente. Esto permitiría añadir *feedback* al poder realizar un análisis añadido del movimiento teniendo en cuenta la información conocida del usuario. Además, actualmente existen técnicas muy avanzadas que permiten este tipo de interacciones (Billinghurst, Kato & Myojin, 2009). Además de identificar movimientos, puede ser posible aprender nuevos ejes dentro de cada uno de ellos. Esos gestos pueden corresponderse con atributos como la emoción o energía; trabajos anteriores en gráficos por ordenador (Rose, Cohen & Bodenheimer, 1998) han mostrado formas de sintetizar animaciones 3D de estos atributos.

5.2.4. Ampliación de la base de datos

La base de datos creada puede ser extendida para incluir más usuarios y acciones/gestos. De esta forma es posible adaptarla para distintas situaciones, mediante acciones genéricas y variadas, entrenando con gestos para interfaces, caídas y otros accidentes, acciones de la vida cotidiana, etc. Además pueden realizarse divisiones y subsets de todas estas clases, dependiendo de la aplicación:

- Acciones heterogéneas: Andar, correr, saltar, etc.
- Gestos: Interacción con las manos, con la cabeza, etc.
- Otros: Ejercicios físicos específicos, ejemplos de caídas o accidentes, etc.

Referencias bibliográficas

Aggarwal, J.K., Park, S. (2004). Human Motion: Modeling and Recognition of Actions and Interactions. *3D Data Processing, Visualization and Transmission*. 640-647.

Aggarwal, J.K., Ryoo, M.S. (2011). Human activity analysis: A review. *ACM Computing Surveys (CSUR), 43*(3), 16.

Ariki, Y., Hyon, S. H., Morimoto, J. (2013). Extraction of primitive representation from captured human movements and measured ground reaction force to generate physically consistent imitated behaviors. *Neural Networks, 40*, 32-43.

ARTTS 3D-TOF Database http://www.artts.eu/publications/3d_tof_db, Última visita Julio de 2013.

Ball, K., Sekuler, R. (1982). A specific and enduring improvement in visual motion discrimination. *Science, 219*, 697-698.

Billinghurst, M., Kato, H., Myojin, S. (2009). Advanced interaction techniques for augmented reality applications. *Virtual and Mixed Reality*, 13-22. Springer Berlin Heidelberg.

Biswas, K. K., Basu, S. K. (2011). Gesture Recognition using Microsoft Kinect®. Automation, Robotics and Applications (ICARA), 2011 5th International Conference on, 100-103.

Boashash, B. (2003) Time frequency analysis. *Elsevier Science*.

Brigham, E.O. (1988) The Fast Fourier Transform and its applications. Prentice Hall.

Carpenter, G., Grossberg, S., Markuzon, N., Reynolds, J., Rosen, D. (1992). Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks, 3*(5), 698-713.

Cavanagh, P., Mather, G. (1989). Motion: the long and short of it. *Spatial vision 4* (2–3), 103-129.

Chaquet, J.M., Carmona, E.J., Fernández-Caballero, A. (2013). A survey of video datasets for human action and activity recognition. *Computer Vision and Image Understanding, 117*(6), 633-659.

Chen, D., Chou, P., Fookes, C., Sridharan, S. (2008). Multi-View Human Pose Estimation using Modified Five-point Skeleton Model. Signal Processing and Communication Systems International Conference on, 17-19.

Chen, H.L.F., Kotani, K. (2012). Extraction of Discriminative Patterns from Skeleton Sequences for Human Action Recognition. IEEE RIVF International Conference 2012. IEEE, 1–6.

Cho, S. B., Kim, J. H. (1995). Combining multiple neural networks by fuzzy integral for robust classification. *Systems, Man and Cybernetics, IEEE Transactions on, 25*(2), 380-384.

Chua, T. W., Leman, K., Pham, N. T. (2011). Human action recognition via sum-rule fusion of fuzzy K-nearest neighbor classifiers. Fuzzy Systems (FUZZ), 2011 IEEE International Conference on, 484-489.

CUDA, Plataforma de programación y computación paralela, http://www.nvidia.com/object/cuda_home_new.html, última visita Mayo 2013.

De Boer, P.T., Kroese, D.P, Mannor, S., Rubinstein, R.Y. (2005) A Tutorial on the Cross-Entropy Method. *Annals of Operations Research, 134*(1), 19-67.

Elmezain, M., Al-Hamadi, A., Michaelis, B. (2008). Real-Time Capable System for Hand Gesture Recognition Using Hidden Markov Models in Stereo Color Image Sequences, *Journal of WSCG 16*:1, 65-72, February 4-7. Plzen, CZ.

Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D., Twombly, X. (2007). Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding, 108*(1), 52-73.

FAAST, Middleware para facilitar la integración del control mediante el cuerpo en juegos y aplicaciones de realidad virtual, http://projects.ict.usc.edu/mxr/faast/, última visita Julio 2013.

Fanelli, G., Weise, T., Gall, J., Van Gool, L. (2011). Real Time Head Pose Estimation from Consumer Depth Cameras, 33rdAnnyal Symposium of the German Association for Pattern Recognition (DAGM'11).

Ganapathi, V., Plagemann, C., Koller, D., Thrun, S. (2010). Real Time Motion Capture Using a Single Time-Of-Flight Camera. *Computer Vision and Pattern Recognition*, 755 - 762.

Gallo, O., Manduchi, R., Rafii, A. (2008). Robust curb and ramp detection for safe parking using the canesta ToF camera. Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on, 1-8.

Gu, Q., Peng, J., Deng, Z. (2009). Compression of Human Motion Capture Data Using Motion Pattern Indexing. *Computer Graphics Forum, 28*(1), 1-12.

Guo, Y., Xu, G., Tsuji, S. (1994). Understanding Human Motion Patterns. Proceedings of International Conference on Pattern Recognition, Track B:325-329.

Hadad, B., Maurer, D., Lewis, T. L. (2001). Long trajectory for the development of sensitivity to global and biological motion. *Developmental Science, 14*(6), 1330-1339.

Henniges, P., Granger, E., Sabourin, R. (2005). Factors of overtraining with fuzzy ARTMAP neural networks. Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on, 2, 1075-1080.

Hrabar, S., Corke, P., Bosse, M. (2009). High dynamic range stereo vision for outdoor mobile robotics, *Robotics and Automation*, 430-435.

Keogh, E., Ratanamahatana, C.A. (2005) Exact indexing of dynamic time warping. *Knowledge and Information Systems 7*(3), 358–386.

Kulic, D., Takano, W., Nakamura, Y. (2007). Towards Lifelong Learning and Organization of Whole Body Motion Patterns. *International Symposium of Robotics Research*, 113-124.

Kinect SDK, Kinect for Windows SDK. http://www.microsoft.com/en-us/kinectforwindows/, última visita Julio 2013.

Lai, K., Konrad, J., Ishwar, P. (2012). A gesture-driven computer interface using Kinect. *Image Analysis and Interpretation (SSIAI), 2012 IEEE Southwest Symposium on*, 185-188.

Martínez-Zarzuela, M., Díaz-Pernas, F. J., Tejero de Pablos, A., Perozo-Rondón, F., Antón-Rodríguez M., González Ortega, D. (2011) Monitorización del cuerpo humano en 3D mediante tecnología Kinect. *SAAEI XVIII Edition*, 747-752.

Martínez-Zarzuela, M., Díaz-Pernas, F. J., Tejero de Pablos, A., Perozo-Rondón, F., Antón-Rodríguez, M., González Ortega, D. (2011) Fuzzy ARTMAP Based Neural Networks on the GPU for High-Performance Pattern Recognition. *J. M. Ferrández et al. (eds.) IWINAC 2011, Part II, Lecture notes in Computer Science* , 6687, 343-352.

Martínez-Zarzuela, M., Díaz-Pernas, F. J., Tejero de Pablos, A., Antón-Rodríguez, M., Díez Higuera, J.F., Boto Giralda, D., González Ortega, D. (2009) Adaptive Resonance Theory Fuzzy Networks Parallel Computation Using CUDA. *J. Cabestany, M.G.F.S. Hernández, A.P. and (eds.) IWANN, Lecture notes in Computer Science, 5517*, 150-157.

McGill, R., Tukey, J.W., Larsen, W.A. (1978) Variations of Box Plots. *The American Statistician 32*(1), 12–16.

Moeslund, T. B., Hilton, A., Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, *104*(2), 90-126.

Naftel, A., Khalid, S. (2006). Motion Trajectory Learning in the DFT-Coefficient Feature Space. IEEE International Conference on Computer Vision Systems, 47-54.

Ni, B., Wang, G., Moulin, P. (2011). RGBD-HuDaAct: A color-depth video database for human daily activity recognition. International Conference on Computer Vision Workshops (ICCV Workshops), 2011 IEEE, 1147-1153.

NITE, Infraestructura algorítmica para identificación de usuarios, detección de características y reconocimiento de gestos, http://www.primesense.com/solutions/nite-middleware/. Última visita Julio 2013.

Okada, S., Hasegawa, O. (2008). Motion recognition based on dynamic-time warping method with self-organizing incremental neural network. *Pattern Recognition, 2008.* ICPR 2008. 19th International Conference on, 1-4.

Okura, F., Kanbara, M., Yokoya, N. (2012). Fly-through heijo palace site: historical tourism system using augmented telepresence. *Proceedings of the 20th ACM international conference on Multimedia*, 1283-1284.

OpenKinect, Comunidad interesada en hacer uso del harware de Kinect en PCs y otros dispositivos, http://openkinect.org/wiki/Main_Page, última visita Julio 2013.

OpenNI, Plataforma para promover la interoperabilidad entre dispositivos, aplicaciones y middleware de Interacción Natural (NI), http://www.openni.org, última visita Julio 2013.

Pascual-Gaspar, J.M., Cardeñoso-Payo, V., Vivaracho-Pascual, C.E. (2009) Practical on-line signature verification. *Lecture Notes in Computer Science, 5558*, 1180-1189.

Pascual-Gaspar, J.M., Faundez-Zanuy, M., Vivaracho-Pascual, C.E. (2011) Fast on-line signature recognition based on VQ with time modeling. *Engineering Applications of Artificial Intelligence, 24*, 368-377.

Han, S., Lee, S., Peña-Mora, F. (2012). Vision-based motion detection for safety behavior analysis in construction. In 2012 Construction Research Congress (CRC), 21-23.

Pincus, S. M. (1991) Approximate entropy as a measure of system complexity. *Proceedings of the National Academy of Sciences, 88*(6), 2297-2301.

Plagemann, C., Ganapathi, V., Koller, D., Thrun, S. (2010). Real-time identification and Localization of Body Parts from Depth Images. IEEE International Conference on Robotics and Automation, 3108-3113.

Poppe, R. (2007). Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding, 108*, 4–18.

Proske, U., Gandevia, S.C. (2009). The kinaesthetic senses. *The Journal of Physiology 587*(17), 4139-4146.

Raptis, H. H. M., Kirovski, D. (2011). Real-Time Classification of Dance Gestures from Skeleton Animation, *Proceedings of the 2011 ACM SIGGRAPH. ACM Press*, 147-156.

Ratanamahatana, C. A., Keogh, E. (2005) Three myths about dynamic time warping data mining. Proceedings of SIAM International Conference on Data Mining (SDM'05), 506-510.

Rose, C., Cohen, M. F., Bodenheimer, B. (1998). Verbs and adverbs: Multidimensional motion interpolation. *Computer Graphics and Applications, IEEE, 18*(5), 32-40.

Schwarz, L.A., Mateus, D., Castañeda, V., Navab, N. (2010). Manifold Learning for ToF-based Human Body Tracking and Activity Recognition. Proceedings of the British Machine Vision Conference, 80, 1–11.

Sempena, S., Maulidevi, N. U., Aryan, P. R. (2011). Human action recognition using dynamic time warping. In Electrical Engineering and Informatics (ICEEI), 2011 International Conference on, 1-5.

Sheikh, M.S.Y., Sheikh, M. (2005). Exploring the Space of a Human Action. *IEEE International Conference on Computer Vision 2005, 1*. 144–149.

Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1), 116-124.

Sigal, L., Balan, A.O., Black, M.J. (2010). HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision, 87*(1-2), 4-27.

Suma, E.A., Lange, B., Rizzo, A., Krum, D.M., Bolas, M. (2011). FAAST: The Flexible Action and Articulated Skeleton Toolkit. *Virtual Reality Conference*. 247-248.

Weinland, D., Ronfard, R., Boyer, E. (2011). A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding, 115*, 224–241.

Wilcoxon, F. (1945) Individual Comparisons by Ranking Methods. *Biometrics 1*, 80-83.

Xiao, Z., Mengyin, F., Yi, Y., Ningyi, L. (2012). 3D Human Postures Recognition Using Kinect. *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference on, 1*, 344-347.

Zhang, Z., Seah, H., Quah, C., Sun, J. (2013). GPU-accelerated Real-time Tracking of Full-body Motion with Multi-layer Search. *IEEE Transactions on Multimedia, 15*(1), 106-119.

Zhao, W., Chellappa, R., Phillips, P. J., Rosenfeld, A. (2003). Face recognition: A literature survey. *Acm Computing Surveys (CSUR), 35*(4), 399-458.

Zhou, F., De la Torre, F. (2012) Generalized time warping for multi-modal alignment of human motion. *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 1282-1289.

Zhu, Y., Dariush, B., Fujimura, K. (2010). Kinematic self retargeting: A framework for human pose estimation. *Computer Vision and Image Understanding, 114*, 1362-1375.

ANEXO: UGOKI3D Toolbox – Manual de Usuario

Tras seleccionar las herramientas con las que se deseaba trabajar, se pasó a la implementación de las mismas. MatLab® es un entorno de desarrollo que utiliza un lenguaje de alto nivel. Ofrece una gran de herramientas ya implementadas que facilitan las tareas de prototipado; es por ello que es utilizado por gran cantidad de profesionales para sus investigaciones.

Además, la idea para la implementación es ofrecer el código en forma de ToolBox, con varios ejemplos y experimentos. Mediante esto, se facilita la realización de nuevos experimentos en futuras investigaciones, así como la ampliación del ToolBox con nuevas herramientas.

En esta sección se ofrece una descripción del uso y funcionamiento de las funciones realizadas en forma de Manual de Usuario. Asimismo se realiza un comentario general de las entradas y salidas de las funciones, así como una descripción gráfica.

A.1. Estructura

El ToolBox se compone de varios elementos según su utilidad:

- Captura
    - Capturador de esqueleto
- Análisis
    - Diagrama de bigotes
    - Test de signo
    - Correlación cruzada
- Preprocesamiento
    - Pre-procesador
    - Rectificación de esqueleto
        - Centro de masas
        - Jerarquía
    - Selector de *joints*
    - Deltas
    - Transformada de Fourier
    - Construcción de patrones
        - Construcción de patrones
        - Construcción de patrones para red neuronal

    - Normalización
        - Normalización Z
        - Normalización Min-max
        - Normalización Max
- Clasificadores
    - Red Neuronal
        - Complementar código
        - Crear red
        - Aprendizaje
        - Clasificación
        - Set test
    - DTW
        - Calcular DTW
        - DTW
    - Entropías
        - Calcular entropía
        - Entropía cruzada aproximada
    - Combinación de resultados
        - Suma ponderada
- Herramientas
    - Entrada y salida
        - Leer/escribir matriz de movimiento
        - Leer/escribir modelo para red neuronal/distancias
        - Leer/escribir configuración para red neuronal/distancias
        - Leer/escribir pesos
    - Tasa de equi-error
        - Calcular EER
- Movimientos
    - Matrices de movimiento: M000, M001...
- Configuración
    - Modelos
    - Configuraciones
    - Pesos
- Resultados
- Scripts (ejemplos)
    - Añadir el PATH del Toolbox
    - Captura de un movimiento
    - Análisis mediante diagrama de bigotes
    - Análisis mediante correlación cruzada

- Análisis mediante test de signo
- Creación de experimentos mediante red neuronal/distancias
- Creación de experimentos para distancias en base a la EER
- Creación de modelos para red neuronal/distancias
- Reconocimiento de movimientos mediante red neuronal/distancias
- Calculador de pesos para distancias

Dentro de los scripts, cabe destacar SETPATH.m. Su utilidad es añadir al path de MatLab® las rutas a los distintos elementos del ToolBox, de forma que puedan llamarse a las funciones directamente desde el directorio raíz. Es esencial ejecutarlo antes de comenzar a utilizar el ToolBox.

Asimismo definiremos la unidad de procesamiento de este ToolBox, la matriz de movimiento (o *motion matrix*). Esta matriz contiene un movimiento representado por su duración en número de *frames* (numF), el número de *joints* que lo ejecutan en sus 3 coordenadas (numJ) y el número de distintas ejecuciones del movimiento, normalmente relacionado con el número de usuarios (numU). Dado el *dataset* de entrada utilizado, los valores típicos serán:

- numF = 64
- numJ = 45
- numU = 9

De esta forma, puede visualizarse como aparece en la Figura 38.

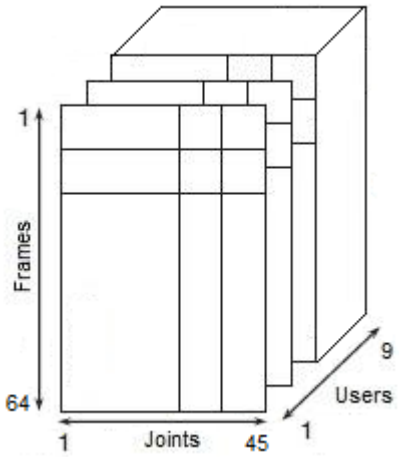

Figura 39: Ugoki3D Toolbox – Matriz de movimiento como unidad de procesamiento

A.2. Captura: [0] Capturing

Esta parte consta de una función: SkeletonCapture.

A.2.1. SkeletonCapture

Esta función obtiene una matriz de movimiento a partir de un esqueleto monitorizado. La definición de la función es como sigue:

function trackM = SkeletonCapture(online, action, user, numF, numJ)

- online es la opción de captura
  - online = 0: Usa un video en formato .oni
  - online = 1: Usa el stream de Kinect
- action es la acción que se monitoriza cuando se usa un video como entrada (p. ej. 'M000').
- user es el usuario que se monitoriza cuando se usa un video como entrada (p. ej. 'U000').
- numF es el número de *frames* que dura la acción.
- numJ es el número de *joints* que se monitorizan.
- trackM es la matriz de movimiento del usuario monitorizado (Figura 39).

Para poder funcionar utiliza un *wrapper* para MatLab® de OpenNI, que se encuentra compilado en la máquina. Mediante este consigue obtener la información de Kinect o el video de profundidad .oni.

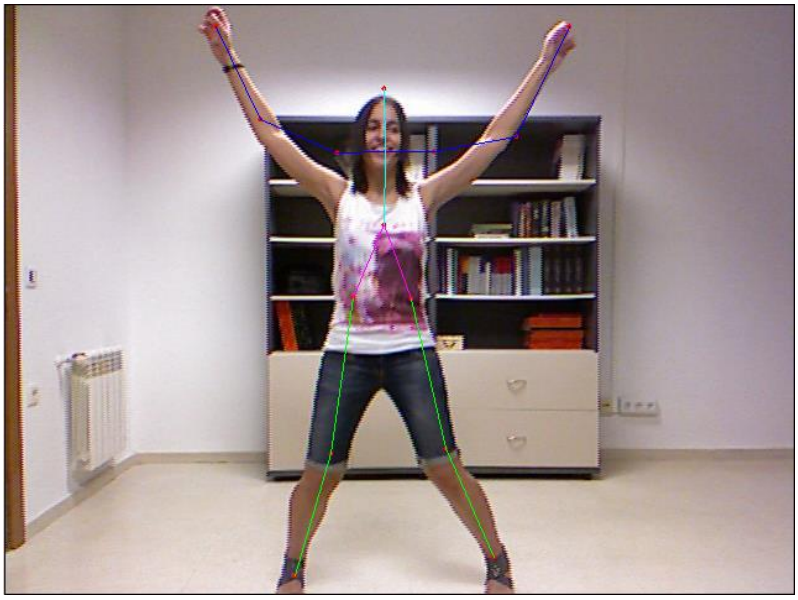

Figura 40: Ugoki3D Toolbox – Captura del esqueleto de usuario monitorizado

A.3. Análisis: [1] Analysis

Se compone de tres herramientas: BoxWhiskersPlot, SignRankTest, CrossCorrelation.

A.3.1. BoxWhiskersPlot

Esta función dibuja un diagrama de bigotes a partir de una matriz de movimiento, para visualizar la varianza de las trayectorias. La definición de la función es como sigue:

function BoxWhiskersPlot(M, option)

- M es la matriz de movimiento a analizar, realizado por uno o varios usuarios
- option indica el tipo de análisis:
  - option = 0: Se realiza un análisis basado en usuario, es decir, para un mismo movimiento se observan los distintos *joints* para cada ejecución de los usuarios (Figura 40.a).
  - option = 1: Se realiza un análisis basado en *joints*, es decir, para un mismo movimiento se observan las ejecuciones de los usuarios para cada *joint* (Figura 40.b).

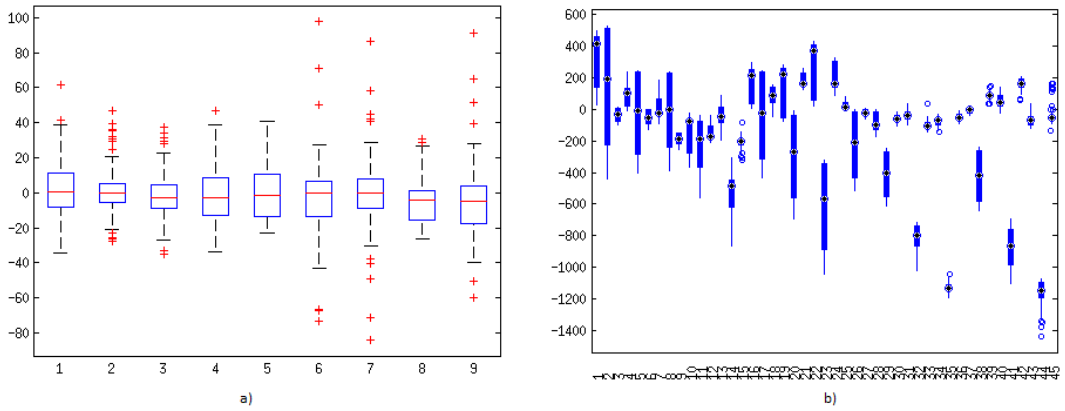

Figura 41: Ugoki3D Toolbox – Diagrama de bigotes basado en usuario (a) y basado en *joints* (b)

A.3.2. SignRankTest

Esta función realiza un test de signo de Wilcoxon sobre dos matrices de movimientos realizados por los mismos usuarios. La definición de la función es como sigue:

function [pAB hAB] = SignRankTest(A, B)

- A es la primera matriz de movimiento a comparar.
- B es la segunda matriz de movimiento a comparar.
- pAB es la matriz de diferencia de acuerdo a las probabilidades.
- hAB es la matriz de diferencia de acuerdo al criterio de hipótesis (Figura 41).

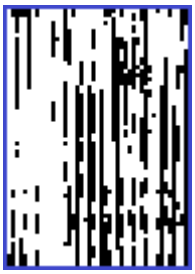

Figura 42: Ugoki3D Toolbox – Matriz de hipótesis del test de signo entre dos movimientos

A.3.3. CrossCorrelation

Esta función realiza una correlación cruzada entre un usuario 'paciente' que realiza un movimiento y uno o varios 'controles' que realizan el mismo u otro movimiento. La definición de la función es como sigue:

function XCA = CrossCorrelation(P, C)

- P es la matriz de movimiento paciente.
- C es la matriz de movimiento control.
- XCA contiene las correlaciones promedio entre P y los usuarios de C.

La señal de correlación puede visualizarse, de forma que la existencia de picos indicaría puntos donde el paciente y los controles son similares (Figura 42). La Figura 43 describe la estructura de la matriz de salida.

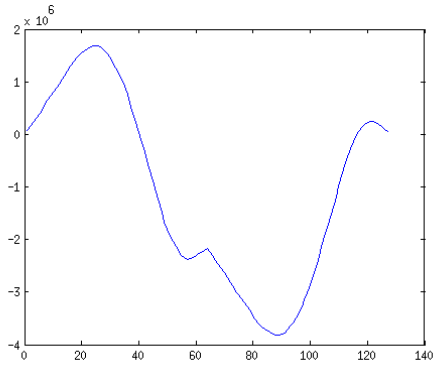

Figura 43: Ugoki3D Toolbox – Señal de correlación entre dos trayectorias de movimiento

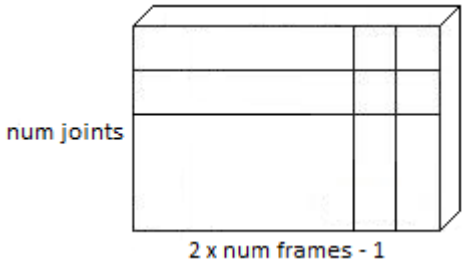

Figura 44: Ugoki3D Toolbox – Matriz de salida de CrossCorrelation

A.4. Preprocesamiento: [2] Preprocessing

Se compone de varios elementos: PreProcessor, GeometricRectificationToolbox, JointSelector, Deltas, FourierTransform, PatternCreator, NormalizationToolbox.

A.4.1. PreProcessor

Esta función pre-procesa una matriz de movimiento siguiendo la configuración indicada. Su definición es:

function M = PreProcessor(M, configID)

- M es la matriz de movimiento antes de ser preprocesada
- configID es el ID de la configuración usada en el preprocesamiento
- P es la matriz de movimiento tras el preprocesamiento

A.4.2. GeometricRectificationToolbox

Se compone de varias herramientas: CenterOfMass, Hierarchy.

CenterOfMass

Esta función representa todas las coordenadas en crudo de una matriz de movimiento respecto al punto central del torso. La definición es la siguiente:

function R = CenterOfMass(M)

- M es la matriz de movimiento antes de la rectificación
- R es la matriz rectificada

Para ello se va restando *frame* a *frame* la posición de cada *joint* con la posición del tronco en el primer *frame*.

Hierarchy

Esta función rectifica las coordenadas en crudo de una matriz de movimiento de acuerdo a una jerarquía de esqueleto (Sección 3.3.2). La definición es la siguiente:

function R = Hierarchy(M)

- M es la matriz de movimiento antes de la rectificación
- R es la matriz rectificada

Para ello se va restando *frame* a *frame* la posición de cada *joint* con la posición en el primer *frame* de su padre en la jerarquía.

A.4.3. JointSelector

Esta función extrae un conjunto de *joints* de una matriz de movimiento. Su definición es:

function S = JointSelector(M, JointSet)

- M es la matriz de movimiento de entrada
- JointSet es el vector que contiene los IDs de los *joints* deseados
- S es la matriz tras aplicar la selección de *joints*

La modificación de la matriz se muestra en la Figura 44:

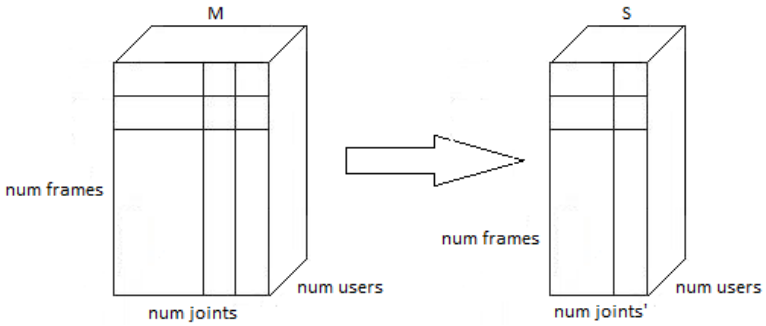

Figura 45: Ugoki3D Toolbox – Procesamiento de JointSelector

A.4.4. Deltas

Esta función calcula la diferencia entre posiciones de trayectorias entre *frames* consecutivos. Su definición es:

function D = Deltas(M)

- M es la matriz de movimiento antes de aplicar deltas
- D es la matriz de movimiento tras aplicar deltas

A.4.5. FourierTransform

Esta función transforma las trayectorias de una matriz de movimiento en series de Fourier. Su definición es:

function F = FourierTransform(M, sizeF)

- M es la matriz de movimiento
- sizeF es el número de elementos de la operación FFT
- F es la matriz de movimiento que contiene Fourier

La modificación de la matriz se muestra en la Figura 45:

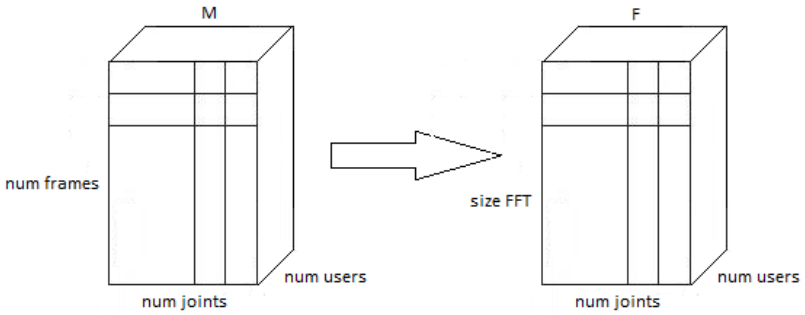

Figura 46: Ugoki3D Toolbox – Procesamiento de FourierTransform

A.4.6. PatternCreationToolbox

Se compone de varias herramientas: PatternCreatorNN, PatternCreator.

PatternCreator

Esta función concatena un patrón a partir de una matriz de movimiento y lo cambia de forma. Su definición es:

function P = PatternCreator(F)

- F es la matriz de movimiento
- P es la matriz de patrones resultado

La modificación de la matriz se muestra en la Figura 46:

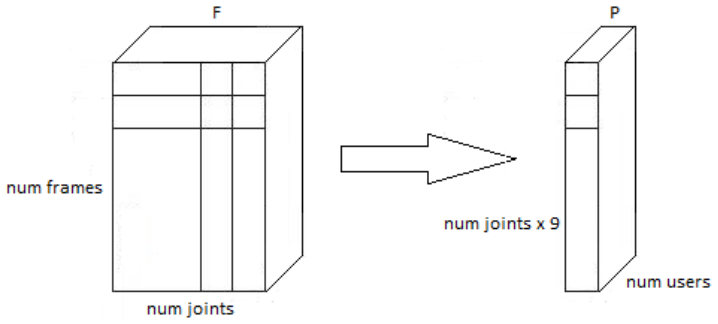

Figura 47: Ugoki3D Toolbox – Procesamiento de PatternCreator

PatternCreatorNN

Esta función concatena un patrón a partir de una matriz de movimiento para introducirlo en una red neuronal. Su definición es:

function P = PatternCreator(F)

- F es la matriz de movimiento
- P es la matriz de patrones resultado

La modificación de la matriz se muestra en la Figura 47:

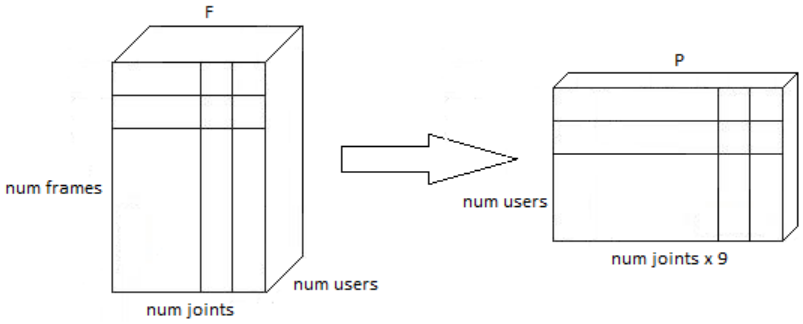

Figura 48: Ugoki3D Toolbox – Procesamiento de PatternCreatorNN

A.4.7. NormalizationToolbox

Se compone de varias herramientas: ZNorm, MinMaxNorm, MaxNorm.

ZNorm

Esta función expresa los valores de cada columna de una matriz de movimiento siguiendo una distribución normal. La definición es:

function data_s = ZNorm(data_e)

- data_e es la matriz de movimiento antes de la normalización
- data_s es la matriz de movimiento normalizada

MinMaxNorm

Esta función expresa todos los valores de cada movimiento de una matriz de movimiento en el rango [0,1]. La definición es:

function [minimo maximo data_s] = MinMaxNorm(data_e)

- data_e es la matriz de movimiento antes de la normalización
- data_s es la matriz de movimiento normalizada
- minimo es el parámetro min usado en la normalización
- maximo es el parámetro max usado en la normalización

MaxNorm

Esta función expresa todos los valores de cada columna de una matriz de movimiento en el rango [-1 1]. La definición es:

function data_s = MaxNorm(data_e)

- data_e es la matriz de movimiento antes de la normalización
- data_s es la matriz de movimiento normalizada

A.5. Clasificadores: [3]Classifiers

Se compone de varios elementos: NeuralNetworkToolbox, DTWToolbox, EntropyToolbox, CombinationToolbox.

A.5.1. NeuralNetworkToolbox

Se compone de varias herramientas: ART_Complement_Code, ARTMAP_Create_Network, ARTMAP_Learn, ARTMAP_Classify, SetTest.

ART_Complement_Code

Esta función codifica de forma complementada los datos para el uso en una red de tipo ART. Definición:

complementCodedData = ART_Complement_Code(data)

- data es la matriz de patrones antes de ser complementada, cada columna es un patrón.
- complementCodedData es la matriz de patrones complementada.

La modificación de la matriz se muestra en la Figura 48:

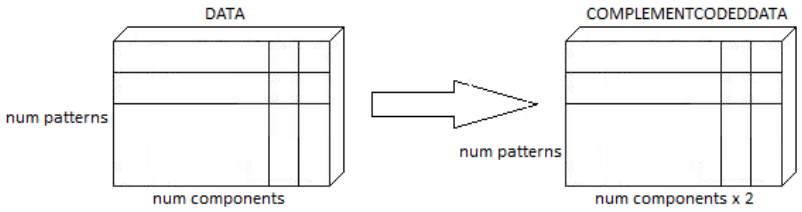

Figura 49: Ugoki3D Toolbox – Procesamiento de ART_Complement_Code

ARTMAP_Create_Network

Esta función crea una nueva red ARTMAP con el número especificado de características y clases. Definición:

function artmap_network = ARTMAP_Create_Network(numFeatures, numClasses)

- numFeatures es el número de características que la red espera de los datos de entrada. Debe ser un valor entero positivo.
- numClasses es el número de clases que existe para la señal supervisora. El valor debe ser un entero positivo mayor que 1.
- artmap_network es la estructura que contiene toda la información de la red.

Los valores utilizados por defecto son se muestran en la Tabla 11.

Característica	Valor
Vigilancia (ρ)	0,9
Bias (α)	0,005
Tasa de aprendizaje (β)	0,5
Número de *epoch*	100

Tabla 11: Valores por defecto de la red ARTMAP del Toolbox

ARTMAP_Learn

Esta función entrena una red ARTMAP para los datos de entrada dados. Definición:

function new_artmap_network = ARTMAP_Learn(artmap_network, data, supervisor)

- artmap_network es la red ARTMAP a entrenar (creada con la función ARTMAP_Create_Network).
- data son los datos de entrenamiento. Es una matriz cuyo tamaño coincide con el especificado al crear la red.
- supervisor es la clasificación correcta para cada patrón de entrada.
- new_artmap_network es una nueva red ARTMAP que ha aprendido los datos de entrada.

ARTMAP_Classify

Esta función usa una red ARTMAP para clasificar los datos de entrada dados para el parámetro de vigilancia dado, dando valor -1 a las clases desconocidas. Definición:

function classification = ARTMAP_Classify(artmap_network, data)

- artmap_network es la red ARTMAP entrenada con las funciones anteriores.

- data son los datos a clasificar, cuyo tamaño es el dado a la red al crearla.
- classification es un vector de tamaño igual al número de muestras que contiene la clase en la que la red ARTMAP ha colocado cada muestra.

SetTest
Esta función divide un conjunto de muestras P en sets de entrenamiento y test. Definición:

function [P1 P2 sup truth] = SetTest(P, numM, numU, setT)

- P es el número de muestras
- numM es el número de clases (acciones) contenido en las muestras.
- numU es el número de usuarios que realiza cada acción.
- setT es el número de muestras de cada clase (acción) para el entrenamiento.
- P1 es el set de entrenamiento.
- P2 es el set de test.
- sup es el vector de supervisión para el entrenamiento.
- truth es el vector de supervisión para el test.

A.5.2. DTWToolbox

Se compone de varias herramientas: CalculateDTW, DTW.

CalculateDTW
Esta función devuelve las distancias entre las trayectorias de los *joints* de la matriz de movimiento de prueba "P" al modelo "M". Definición:

function distancias = CalculateDTW(M, P)

- M es la matriz de movimiento del modelo.
- P es la matriz de movimiento a testear.
- distancias es la matriz con las distancias entre las trayectorias del movimiento de los *joints*.

La estructura de las matrices se muestra en la Figura 49:

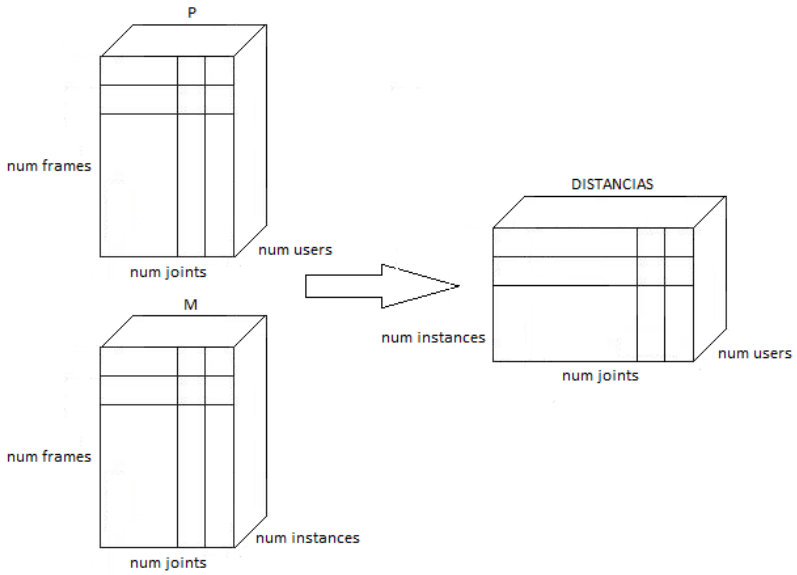

Figura 50: Ugoki3D Toolbox – Procesamiento de CalculateDTW

DTW

Esta función implementa el algoritmo de *Dynamic Time Warping* entre dos secuencias temporales. La definición es:

function [Dist,D,k,w]=DTW(t,r)

- t es el vector contra el que se calcula la distancia
- r es el vector que se testea
- Dist es la distancia sin normalizar entre t y r.
- D es la matriz de distancia acumulada.
- k es el factor de normalización
- w es el camino óptimo

Normalmente, tras recibir el valor de distancia Dist este se divide entre k, para obtener un valor independiente de la longitud de las secuencias.

A.5.3. EntropyToolbox

Se compone de varias herramientas: CalculateEntropy, CrossApEn.

CalculateEntropy

Esta función devuelve los valores de entropía entre las trayectorias de los *joints* de la matriz de movimiento de prueba "P" al modelo "M". Su definición es la siguiente:

function distancias = CalculateEntropy(M, P)

- M es la matriz de movimiento del modelo.
- P es la matriz de movimiento a testear.
- distancias es la matriz con las entropías entre las trayectorias del movimiento de los *joints*.

La estructura de las matrices es equivalente a la mostrada en la Figura 38.

CrossApEn
Esta función calcula los valores de entropía entre dos matrices de trayectorias (un único usuario en cada una). Su definición es:

function ResultsCrossApEn = CrossApEn(A1, B2)

- A1 es la matriz de movimiento del modelo
- B2 es la matriz de movimiento a testear
- ResultsCrossApEn es el vector con las entropías entre las trayectorias de las matrices de entrada.

La estructura de las matrices se muestra en la Figura 50.

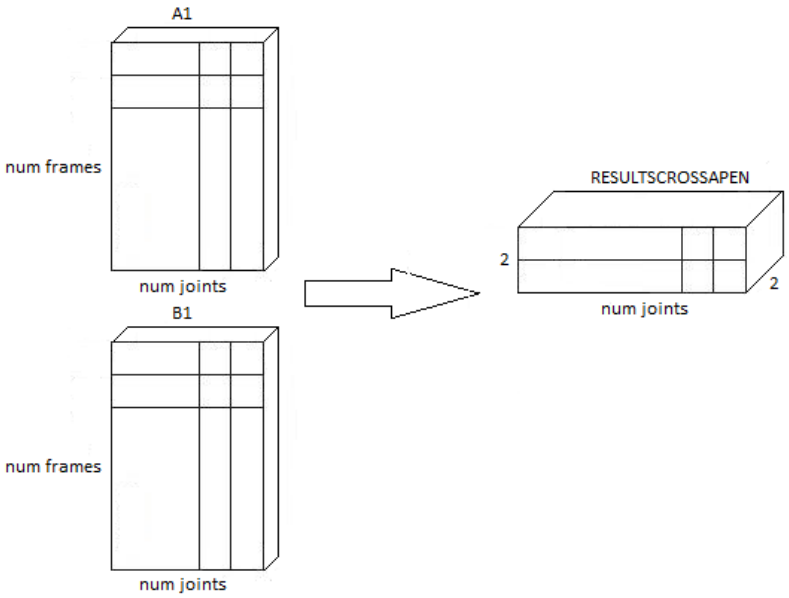

Figura 51: Ugoki3D Toolbox – Procesamiento de CrossApEn

A.5.4. CombinationToolbox

Como se explica en la Sección 3.3.1, existen diversas técnicas de combinación de resultados, sin embargo MatLab® ya implementa directamente muchas de ellas

(mínimo, máximo, media, mediana, suma, producto...). Para nuestro propósito, se ha desarrollado una herramienta de suma ponderada: WeightedSum.

WeightedSum
Esta función suma un conjunto de valores ponderados mediante pesos.

function sumVal = WeightedSum(setVal, configID, weightID)

- setVal es el conjunto de valores.
- configID es el ID de la configuración usada para obtener dichos valores.
- weightID es el ID del conjunto de pesos que representa una acción.
- sumVal contiene la suma ponderada.

La operación realizada se muestra en la Figura 51.

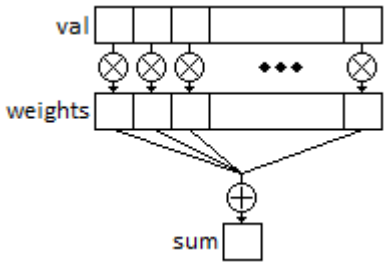

Figura 52: Ugoki3D Toolbox – Operación de Weighted Sum

A.6. Herramientas: [4]Tools

Se compone de varios elementos: EERToolbox, IOToolbox.

A.6.1. IOToolbox

Se compone de varias herramientas: ReadMotionMatrix, WriteMotionMatrix, ReadModelNN, WriteModelNN, ReadModelDist, WriteModelDist, ReadConfDist, WriteConfDist, ReadWeightsDist, WriteWeightsDist.

ReadMotionMatrix
Esta función lee de un conjunto de archivos de usuarios el movimiento indicado. Su definición es:

function [M numF numJ] = ReadMotionMatrix(action, users)

- action es una cadena con el ID del movimiento que se desea leer.
- users es una celda (cell) de IDs de los usuarios que ejecutan la acción.
- M es la matriz de movimiento de salida que contiene uno o más usuarios.
- numF es el número de *frames* de la matriz de movimiento.

- numJ es el número de trayectorias de la matriz de movimiento.

La matriz de movimiento es como la mostrada en la figura 38.

WriteMotionMatrix
Esta función escribe una matriz de movimiento en una serie de archivos. Definición:

function WriteMotionMatrix(M, action, users, rep)

- M es la matriz de movimiento.
- action es una cadena con el ID del movimiento de la matriz.
- users es una celda (cell) con los IDs de los usuarios que realizan el movimiento.
- rep es el número de repetición del movimiento, en caso de que los usuarios ya hubieran realizado una ejecución previa en el *dataset*.

La matriz de movimiento es como la mostrada en la figura 38.

ReadModelNN
Esta función lee de un archivo un modelo correspondiente a una red neuronal. Definición:

function [newNN minimo maximo] = ReadModelNN(modelID, sizeC)

- modelID es el ID del modelo a leer.
- siceC es el número de componentes de cada peso del modelo.
- newNN es la red neuronal creada a partir del modelo.
- minimo es el parámetro min usado en la normalización.
- maximo es el parámetro max usado en la normalización.

WriteModelNN
Esta función escribe en un archivo un modelo correspondiente a una red neuronal. Definición:

function WriteModelNN(NN, minimo, maximo, modelID)

- NN es la red neuronal de la que obtener el modelo.
- minimo es el parámetro min usado en la normalización.
- maximo es el parámetro max usado en la normalización.
- modelID es el ID del modelo a escribir.

ReadModelDist
Esta función lee de un archivo un modelo correspondiente a un reconocedor de distancias. Dado que este reconocedor se compone de varios modelos, uno por cada movimiento, es necesario especificar el subíndice del modelo. Definición:

function M = ReadModelDist(modelID, sub)

- modelID es el ID del modelo a leer.
- sub es el subcomponente (movimiento) del modelo a leer.
- M es el modelo de salida.

El modelo es equivalente a una matriz de movimiento (Figura 38).

WriteModelDist
Esta función escribe en un fichero un modelo de un reconocedor de distancias. Definición:

function WriteModelDist(M, modelID, sub)

- M es el modelo de entrada.
- modelID es el ID del modelo a escribir.
- sub es el subcomponente (movimiento) del modelo a escribir.

ReadConfigDist
Esta función lee un vector de configuración de un fichero. La definición es la siguiente:

function config = ReadConfigDist(configID)

- configID es el identificador asociado a la configuración.
- config es el vector de configuración de salida, que se compone de 6 elementos.

WriteConfigDist
Esta función escribe un vector de configuración en un fichero. La definición es la siguiente:

function WriteConfigDist(config, configID)

- config es el vector de configuración de entrada, que se compone de 6 elementos.
- configID es el identificador asociado a la configuración.

ReadConfigNN
Esta función lee un vector de configuración de un fichero. La definición es la siguiente:

function config = ReadConfigDist(configID)

- configID es el identificador asociado a la configuración.
- config es el vector de configuración de salida, que se compone de 6 elementos.

WriteConfigNN
Esta función escribe un vector de configuración en un fichero. La definición es la siguiente:

function WriteConfigDist(config, configID)

- config es el vector de configuración de entrada, que se compone de 6 elementos.
- configID es el identificador asociado a la configuración.

ReadWeightsDist
Esta función lee un conjunto de pesos de un fichero. Su definición es la siguiente:

- function weights = ReadWeights(configID)
- configID es la configuración usada para crear los pesos.
- weights contiene los pesos calculados para cada movimiento.

La figura 52 muestra la estructura de la matriz de pesos.

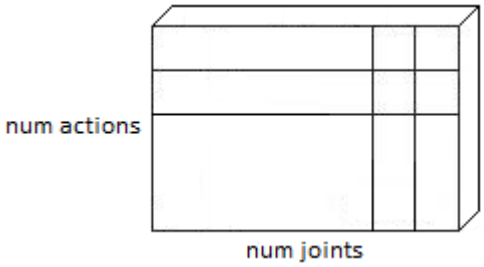

Figura 53: Ugoki3D Toolbox – Matriz de salida de ReadWeightsDist

WriteWeightsDist
Esta función escribe un conjunto de pesos en un fichero. Su definición es la siguiente:

function WriteWeights(weights, configID)

- weights contiene los pesos calculados para cada movimiento.
- configID es la configuración usada para crear los pesos.

La estructura de la matriz de pesos se muestra en la Figura 52.

A.6.2. EERToolbox

Se compone de varias herramientas, de las cuales la principal es CalculateEER.

CalculateEER

Esta función calcula la tasa de equierror, o EER, y dibuja una gráfica DET para los valores de distancias de entrada. Definición:

function EER = CalculateEER(TAR, NON)

- TAR es el vector con las distancias entre un modelo de movimiento y otras instancias del mismo movimiento.
- NON es el vector con las distancias entre un modelo de movimiento y otras instancias de un movimiento diferente.
- EER es el valor de la tasa de equierror.

A.7. Movimientos: Movimientos

Esta carpeta se ha habilitado con el objetivo de almacenar los ficheros de movimiento de las acciones capturadas mediante Kinect. Cada fichero se corresponde con un movimiento realizado por un usuario. Asimismo, el mismo usuario puede realizar varias repeticiones del mismo movimiento. Existe un directorio por cada movimiento, siguiendo la siguiente estructura:

- M000
    - M000_U000_1.txt
    - M000_U001_1.txt
    - …
    - M000_U008_1.txt
- M001
    - M000_U000_1.txt
    - …
    - M000_U008_1.txt

  …
- M009
    - M000_U000_1.txt

- …
- M000_U008_1.txt

MXXX es el identificador del movimiento, UXXX es el identificador de la acción, y el subíndice indica el número de repetición. El contenido de cada fichero está formado por

- Una cabecera:

```
#M000 - U000 (13-Feb-2013): Kinect (640x480), POV frontal

#OpenNI (15 puntos * 3 coordenadas) x MatLab (64 frames en 4.487899 seg)
```

- El número de puntos monitorizado: `45`
- El número de *frames* monitorizados: `64`
- Las trayectorias que componen el movimiento (cada fila es un *frame*):

```
-285.081085    773.291138    ...
```

A.8. Configuración: Config

La función de este directorio es almacenar los ficheros de configuración y otras opciones utilizadas para el reconocimiento de movimientos. Se compone de varios directorios: Models, ConfigDist, WeightsDist.

A.8.1. Models

Esta carpeta almacena los modelos generados al entrenar uno de los reconocedores del Toolbox. Se distinguen dos tipos de modelos según el reconocedor: ModelNN y ModelDist.

ModelNN

Este es un modelo creado a partir de una red neuronal. La nomenclatura es ModelNN_X.txt, siendo X la ID del modelo, que coincide con la de la configuración utilizada en su creación. La estructura de este fichero es como sigue:

- El valor mínimo usado para normalizar en la creación del modelo: `-92607.528622`
- El valor máximo usado para normalizar en la creación del modelo: `263889.915986`
- El número de clases usadas en el modelo: `8`
- El número de categorías generadas por la red neuronal: `10`
- El vector de MapField: `0 1 2 3 4 4 5 5 6 7`
- Los pesos:

```
0.407561    0.570237    0.325356    ...
```

ModelDist

Este es un modelo creado a partir del reconocedor de distancias. La nomenclatura es ModelDist_X_Y.txt. donde X es la ID del modelo (que coincide con el de la configuración) e Y es el subíndice dentro del modelo, ya que en el método de distancias, cada modelo tiene un submodelo por cada movimiento del sistema. La estructura del fichero es la siguiente:

- El número de instancias del movimiento que contiene el modelo: `3`
- El número de trayectorias de *joints* usados en el modelo: `45`
- El número de *frames* usados en el modelo: `64`
- El modelo:

```
-1.614428    1.095545    0.943008    …
```

A.8.2. ConfigDist

Esta carpeta almacena las configuraciones utilizadas en los reconocedores y experimentos realizados con distancias. Estas configuraciones dan también el ID a los modelos y pesos que se crean con ellas. La nomenclatura utilizada es ConfigDist_X.txt, donde X es dicho ID. El contenido del fichero es una línea con 6 elementos, uno para cada opción de configuración.

```
2   0   0   1   0   1
```

A.8.3. WeightsDist

En esta carpeta se almacenan los pesos creados para utilizar la combinación por suma ponderada para los movimientos. La nomenclatura es WeightsDist_X.txt, donde X representa el ID de la configuración usada para crear los pesos. El contenido del fichero es el siguiente:

- El número de vectores de pesos generados (uno por cada movimiento): `8`
- El número de pesos en cada vector (uno por cada trayectoria): `45`
- Los pesos:

```
0.062230    0.105460    0.461897    0.064052    …
```

A.9. Resultados: Results

Esta carpeta se ha habilitado para el almacenamiento de imágenes, variables, etc. obtenidos durante la realización de experimentos con el *dataset*.

A.10. Ejemplos: Scripts

En el directorio raíz del Toolbox se encuentran los scripts y ejemplos utilizados para llevar a cabo experimentos. Declaran ciertas variables de configuración y llaman a las funciones definidas en el Toolbox. Se compone de varias utilidades: SETPATH, ActionCapturer, ActionAnalyzerBoxPlot, ActionAnalyzerCrossCorr, ActionAnalyzerSignRank, ActionExperimenterNN, ActionExperimenterDist, ActionExperimenterDistEER, ActionModelerNN, ActionModelerDist, ActionRecognizerNN, ActionRecognizerDist, ActionWeightsDist.

A.10.1. SETPATH

Este script añade las herramientas del Toolbox al PATH de MatLab® de forma que puedan llamarse desde cualquier script. Esto incluye carpetas y subcarpetas.

A.10.2. ActionCapturer

Este script captura un movimiento y lo escribe en un fichero de texto. Las variables que utiliza son:

- Número de *frames* a capturar: `numF = 64`
- Número de trayectorias de *joints* que se monitorizan: `numJ = 15*3`
- Acción a capturar: `action = 'M000'`
- Usuario que lleva a cabo la acción: `user = cell([1 1])`
- Repetición en el *dataset*: `rep = 1`
- Origen del movimiento:
  - `online = 0` (usa un video .oni)
  - `online = 1` (usa el stream de Kinect)

A.10.3. ActionAnalyzerBoxPlot

Este script analiza un movimiento mediante un diagrama de bigotes. Las variables que utiliza son:

- Acción a analizar: `action = 'M000'`
- Usuarios que realizan la acción: `users=cell([1 9])`
- Configuración que usarán los *joints*: `configID = 0`
- Tipo de análisis:
  - `option = 0` (basado en usuario)
  - `option = 1` (basado en *joints*)

A.10.4. ActionAnalyzerCrossCorr

Este script analiza un par de movimientos usando correlación cruzada. Variables:

- Acción que será comparada: `actionP = 'M000'`
- Acción contra la que comparar: `actionC = 'M000'`
- Usuario que será comparado: `patient=cell([1 1])`
- Usuarios contra los que comparar: `control=cell([1 8])`
- Configuración para el experimento: `configID = 0`
- *Joint Of Interest* (el que será mostrado al final de la prueba): `JOI = [2 3 4]`

A.10.5. ActionAnalyzerSignRank

Analiza un par de movimientos usando el test estadístico de Wilcoxon. Las variables son:

- Acción del paciente: `actionP = 'M000'`
- Acción del control: `actionC = 'M001'`
- Cell de usuarios que realizan las acciones: `users=cell([1 9])`
- Configuración del procesamiento usado en el experimento: `configID = 0`

A.10.6. ActionExperimenterNN

Crea experimentos y obtiene la tasa de aciertos para un sistema de reconocimiento de movimientos basado en red neuronal. Variables:

- IDs de las acciones que toman parte en el experimento: `actions=cell([1 8])`
- IDs de los usuarios que realizan la acción: `users=cell([1 9])`
- Conjunto de *joints* que serán usados: `J = [1 4 5 7 8 11 12 14 15]`
- Tamaño de la transformada de Fourier: `sizeF = 64`
- Número de usuarios de cada acción para entrenar el sistema: `setT = 8`
    - Para realizar Left-One_out, dar valor del número de usuarios menos uno.

A.10.7. ActionExperimenterDist

Crea experimentos y obtiene la tasa de aciertos para un sistema de reconocimiento de movimientos usando distancias. Variables:

- IDs de las acciones que toman parte en el experimento: `actions=cell([1 8])`
- IDs de los usuarios que realizan la acción: `users=cell([1 9])`
- Número de usuarios para crear el modelo de distancias: `numT=8`
- ID de la configuración del procesamiento usado en el experimento: `configID = 0`

A.10.8. ActionExperimenterDistEER

Crea experimentos en base a la tasa de equierror obtenida para cada modelo de un sistema de reconocimiento de movimientos basado en distancias. Variables:

- IDs de las acciones que toman parte en el experimento: `actions=cell([1 8])`
- IDs de los usuarios que realizan la acción: `users=cell([1 9])`
- Número de usuarios para crear el modelo de distancias: `numT=8`
- ID de la última configuración que se desea procesar (se ejecutará desde la ID 0 hasta la indicada): `maxConfigID=5`

A.10.9. ActionModelerNN

Crea un modelo para un reconocedor de movimientos basado en red neuronal y lo almacena en un fichero, así como la configuración utilizada. Variables:

- IDs de las acciones que toman parte en el experimento: `actions=cell([1 8])`
- IDs de los usuarios que realizan la acción: `users=cell([1 9])`
- Conjunto de *joints* que serán usados: `J = [1 4 5 7 8 11 12 14 15]`
- Tamaño de la Transformada de Fourier: `sizeF = 64`
- Número de usuarios de cada acción para entrenar el sistema: `setT = 9`
- Identificador para el modelo y su configuración: `configID = 0`
- Vector de configuración, para poder utilizar la misma en otros modelos ([rectification, selection, deltas, FFT, patterns, normalization]): `config = [1 1 0 1 1 2]`

A.10.10. ActionModelerDist

Crea un modelo para un reconocedor de movimientos basado en distancias y lo almacena en un fichero. Variables:

- IDs de las acciones que toman parte en el experimento: `actions=cell([1 8])`
- IDs de los usuarios que realizan la acción: `users=cell([1 9])`
- Identificador para el modelo y la configuración a utilizar: `configID = 0`

A.10.11. ActionRecognizerNN

Lee una acción y un modelo de red neuronal y lleva a cabo su reconocimiento. Variables:

- Modelo y configuración usados para el reconocimiento: `configID = 0`
- Origen del movimiento:
  - `online = -1` (obtiene el movimiento de un fichero de texto)
  - `online = 0` (obtiene el movimiento de un video)
  - `online = 1` (obtiene el movimiento de Kinect)
- Acción realizada (caso no online): `actionI = 'M000'`
- Usuario que realiza la acción (caso no online): `userI = cell([1 1])`

- Duración en *frames*: `numF = 64`
- Número de *joints*: `numJ = 15*3`
- Selección de *joints*: `J = [1 4 5 7 8 11 12 14 15]`
- Tamaño de la FFT: `sizeF = 64`

A.10.12. ActionRecognizerDist

Lee una acción y un modelo basado en distancias y lleva a cabo su reconocimiento. Variables:

- Modelo que será usado para reconocimiento: `modelID = 0`
- Número de acciones en el modelo (sub-modelos): `numM = 8`
- Origen del movimiento:
  - `online = -1` (obtiene el movimiento de un fichero de texto)
  - `online = 0` (obtiene el movimiento de un video)
  - `online = 1` (obtiene el movimiento de Kinect)
- Acción realizada (caso no online): `actionI = 'M000'`
- Usuario que realiza la acción (caso no online): `userI = cell([1 1])`
- Duración en *frames*: `numF = 64`
- Número de *joints*: `numJ = 15*3`

A.10.13. ActionWeightsDist

Computa los pesos que serán utilizados en un reconocedor basado en distancias. Variables:

- Número de *frames*: `numF = 64`
- Número de *joints* (3 coordenadas, x-y-z): `numJ = 15*3`
- IDs de los movimientos a analizar: `actions=cell([1 8])`
- IDs de los usuarios a analizar: `users=cell([1 9])`
- ID de la configuración del pre-procesamiento: `configID = 0`

Referencias del manual de MatLab®

Average or mean value of array – MATLAB mean – MathWorks España, http://www.mathworks.es/es/help/matlab/ref/mean.html, última visita Julio 2013.

Box plot – MATLAB boxplot – MathWorks España, http://www.mathworks.es/es/help/stats/boxplot.html, última visita Julio 2013.

Continuous Dynamic Time Warping – File exchange – MATLAB Central, http://www.mathworks.es/matlabcentral/fileexchange/16350-continuous-dynamic-time-warping/content/dtw.m, última visita Julio 2013.

Cross-correlation – MATLAB xcorr – MathWorks España, http://www.mathworks.es/es/help/signal/ref/xcorr.html, última visita Julio 2013.

One-way analysis of variance – MATLAB anova1 – MathWorks España, http://www.mathworks.es/es/help/stats/anova1.html, última visita Julio 2013.

Wilcoxon rank sum test – MATLAB ranksum – MathWorks España, http://www.mathworks.es/es/help/stats/ranksum.html, última visita Julio 2013.

Wilcoxon signed rank test – MATLAB signrank – MathWorks España, http://www.mathworks.es/es/help/stats/signrank.html, última visita Julio 2013.