Clasificación de actividades humanas en tiempo real a partir de representaciones en esqueleto

Asier Aguado

Directores:
Elena Lazkano
Basilio Sierra

14 de septiembre de 2016

Índice general

Índice de figuras

Índice de cuadros

Resumen

En este trabajo, se ha desarrollado un sistema de reconocimiento de acciones y actividades humanas mediante técnicas aprendizaje supervisado. Se ha realizado un especial enfoque en la selección de características, donde se propone una representación del esqueleto en coordenadas esféricas, añadiendo información del movimiento a cada pose para la clasificación instantánea de actividades. Dicha clasificación instantánea la realiza un clasificador entrenado, como un *random forest*. Para clasificar una serie temporal completa, se realizan clasificaciones instantáneas de cada fotograma (una pose con información temporal), y se elige la actividad de la serie temporal mediante el voto de la mayoría. Este método de clasificación es suficientemente rápido para funcionar en tiempo real, y ha mostrado resultados prometedores clasificando actividades *online* en ventanas de tiempo de 1 a 5 segundos. Los experimentos de clasificación *offline* con conjuntos de datos conocidos han alcanzado tasas de acierto del 90 %. Esta Tesis de Máster es una continuación del Trabajo de Fin de Grado "Reconocimiento de posturas mediante Kinect en ROS".

Capítulo 1

Introducción

La detección, clasificación y predicción automática de actividades humanas es una tarea de interés en diferentes áreas de las ciencias de la computación, tanto por su dificultad, como por sus aplicaciones. El reconocimiento de actividades puede ser la base para sistemas que realicen tareas complejas, por ejemplo, sistemas de monitorización de pacientes, cuidado de ancianos, sistemas de rehabilitación, entrenamiento físico, vigilancia inteligente, o robots inteligentes.

Típicamente, este proceso se realizaba mediante sistemas basados en visión, como cámaras (mono o estéreo), pero la aparición de sensores asequibles que combinan visión con una noción de profundidad, como Micorosft Kinect (Figura 1.1) o Asus Xtion, ha provocado que, en la actualidad, gran parte de la investigación se centre en estos sensores. En esta Tesis de Máster se aborda el problema de clasificación de actividades a partir de representaciones en esqueleto, que hoy en día pueden obtenerse fácilmente con estos sensores. También se tiene en cuenta su posible aplicación en un sistema de rehabilitación: el reconocimiento de actividades serviría como base para orientar a los pacientes para realizar sus ejercicios en casa, además de servir de ayuda para asistencia médica remota.

Este trabajo aborda el problema de clasificación de actividades mediante técnicas de aprendizaje automático (*machine learning*), en concreto, clasificación supervisada. El enfoque, más que en los algoritmos y técnicas utilizados, se realiza en extracción de características y representación de datos que se utilizarán para el proceso de entrenamiento y aprendizaje. Esta representación que proponemos, además, podría utilizarse en sistemas que no estén basados en aprendizaje automático: donde sea más interesante disponer de una *grabación* del movimiento realizado.

Esta Tesis de Máster tiene como precedente el Trabajo de Fin de Grado realizado en 2015 [1], que también realiza clasificación supervisada a partir del esqueleto humano. El trabajo anterior se centra en clasificar posturas estáticas (como estar de pie, sentado o tumbado) a partir

Figura 1.1: Sensor Microsoft Kinect (v1)

de la información del esqueleto en un instante de tiempo. También contempla un método para reconocer movimientos simples: consiste en comparar posturas estáticas con poses clave del movimiento (obtenidas mediante *clustering* k-means), y sólo tiene en cuenta la velocidad del movimiento para distinguir los estados de "parado" y "en movimiento". El objetivo es distinguir si un movimiento previamente definido y procesado es válido o no válido. Se observaron varios problemas, como que puede reconocer movimientos muy diferentes como válidos, siempre que estos pasen por las posturas clave. No disponer del orden en el que aparecen estas posturas también es una limitación. Este enfoque es aún menos viable para clasificación supervisada multi-clase con varios movimientos, ya que el número de posturas clave aumenta con cada movimiento a reconocer. Además, encontrar las posturas clave en conjuntos de datos muy grandes mediante k-means es una tarea con mucho coste computacional. Debido a estos motivos, en el trabajo de este documento proponemos un método diferente para tratar movimientos, enfocado en la clasificación supervisada de varios movimientos. En este método proponemos una nueva representación geométrica del esqueleto, además de añadir información del movimiento a las posturas estáticas para facilitar el trabajo de los algoritmos de aprendizaje automático.

El 19 octubre de 2016, en el congreso *International Conference on Neurorehabilitation* (ICNR 2016), se expondrá un artículo que recoge los resultados de clasificación de posturas estáticas obtenidos durante Trabajo de Fin de Grado [2].

La estructura de la memoria es la siguiente: en primer lugar, presentamos una descripción y análisis del estado del arte, donde exponemos trabajos relacionados que se pueden encontrar en la bibliografía. Se analizan diferentes enfoques y soluciones para la clasificación de activida-

des humanas, y se comparan con las soluciones que se mostrarán en este trabajo. Más adelante (capítulo 3) se detallan las herramientas y técnicas utilizadas, incluyendo sensor, fuentes de datos, bibliotecas de software y entorno de desarrollo. En el capítulo 4, describimos la representación de datos propuesta para reconocer actividades humanas en tiempo real. El capítulo 5 está dedicado a los algoritmos de aprendizaje automático con los que se han realizado experimentos de entrenamiento y clasificación. El capítulo 6 describe el protocolo utilizado para los experimentos de clasificación, y muestra resultados para clasificación *online* y *offline*. El documento finaliza con un capítulo (7) de conclusiones y propuestas de trabajo futuro.

Capítulo 2

Estado del arte

En los últimos años, se han publicado muchos trabajos de clasificación de actividades humanas en base a imágenes con profundidad. Algunos de estos trabajos utilizan mapas de profundidad, que consisten en una imagen 2D con información de profundidad (e.g. [3]). Otros, que pertenecerían a la misma categoría del trabajo que presentamos aquí, utilizan representaciones en esqueleto: secuencias de posiciones 3D de las articulaciones humanas (e.g. [4]). Es habitual el uso del aprendizaje automático (*machine learning*) en muchos de estos trabajos. El propio esqueleto humano también se obtiene a partir de un algoritmo de aprendizaje automático: el algoritmo de Microsoft para Kinect utiliza árboles de decisión aleatorios (*random forests*), previamente entrenados, para generar el esqueleto humano a partir de un solo mapa de profundidad [5].

Existen diferentes enfoques dentro de los trabajos que realizan clasificación de actividades humanas con información del esqueleto. En [6], se centran en los diferentes enfoques de representación de las características del esqueleto, más que en los algoritmos de entrenamiento y clasificación. Clasifican tres grandes categorías para posibles formas de representar acciones 3D: representaciones basadas en articulaciones, representaciones basadas en articulaciones seleccionadas, y representaciones basadas en dinámicas.

En la primera categoría, se incluyen aquellos métodos que extraen características alternativas a partir de las posiciones del esqueleto. Estas características alternativas pueden ser distancias entre las partes del cuerpo, matrices de covarianza de dichas distancias, o ángulos entre las articulaciones.

En la segunda categoría, se engloban los métodos que seleccionan qué partes del cuerpo son relevantes o útiles para distinguir movimientos. Por ejemplo, en [7], utilizan información relativa basada en la entropía para distinguir aquellas articulaciones más relevantes.

Los métodos de la última categoría serían aquellos que tratan las secuencias del esqueleto

como trayectorias 3D y modelan las dinámicas de la serie temporal. Diferentes enfoques incluyen sistemas lineales dinámicos o modelos ocultos de Markov (*hidden Markov models*).

En este trabajo, se ha propuesto una representación de las acciones teniendo en cuenta tanto el enfoque de la primera categoría (representación basada en articulaciones, concretamente, coordenadas esféricas de las articulaciones), como el de la última (dinámicas de la serie temporal, concretamente, velocidades angulares medias de las coordenadas).

Algunos trabajos anteriores también recogieron y publicaron conjuntos de datos con información del esqueleto, que pueden utilizarse para realizar experimentos y validar nuevos métodos. En particular, los conjuntos de datos utilizados aquí se han recogido en los trabajos [3] y [4].

Un trabajo con un enfoque similar al expuesto aquí, en cuanto a representación de datos, es el de L. Miranda y otros [8]. En este trabajo, representan el esqueleto mediante coordenadas esféricas. Después, reconocen diferentes gestos utilizando poses clave (*key poses*) predefinidas, las cuales se clasifican mediante máquinas de soporte vectorial (SVM) multi-clase. Los autores definen un kernel propio para estos SVM, utilizando métricas esféricas. Un bosque de decisión (grupo de árboles de decisión) se utiliza finalmente para clasificar gestos a partir de poses clave. Este método obtiene buenos resultados, pero una de sus limitaciones es que está limitado a un conjunto de poses clave. Los movimientos a clasificar deben poder definirse con poses clave distintivas, y, además, es necesario definirlas previamente. Podemos encontrar otro ejemplo de representación en coordenadas esféricas en [9], donde además utilizan modelos ocultos de Markov para el reconocimiento de acciones.

En [10], presentan este mismo problema de reconocimiento de acciones enfocado en la robótica móvil. Su objetivo es construir un robot con las habilidades cognitivas necesarias para ofrecer asistencia a personas en entornos de interior. Estas habilidades cognitivas consisten en diferenciar actividades humanas, con el objetivo de que el robot sea capaz de distinguir rutina diaria y situaciones de riesgo. Para ello, utilizan el esqueleto proporcionado por el sensor Kinect y la biblioteca OpenNI. El reconocimiento de actividades se realiza entrenando un modelo de clasificación compuesto de varios clasificadores probabilísticos: estos clasificadores se combinan en una mixtura de bayesianas con pesos dinámicos, resultando en una distribución de probabilidades final que proporciona la actividad más probable. Tanto para entrenamiento como para test, utilizan su propio conjunto de datos que consiste en las siguientes actividades: caminando, de pie, trabajando en el ordenador, hablando por teléfono, corriendo, saltando, cayendo al suelo y sentado. En este conjunto de datos en particular, algunas de estas actividades podrían ser reconocidas desde un punto de vista estático (comprobando una sola postura instantánea), por lo que las dinámicas no afectan mucho al conjunto de datos. Aún así, los propios autores señalan que una de las mayores

dificultades del conjunto de datos era la variabilidad de una sola clase (por ejemplo, sujetar el teléfono con la mano izquierda o derecha), y también las diferentes perspectivas. Al recoger los datos desde un robot móvil, las posturas no siempre eran captadas de frente o desde el mismo ángulo.

Por último, un posible enfoque sin utilizar coordenadas del esqueleto es utilizar mapas de movimiento en profundidad (*depth motion maps*), como en [11]. Estos mapas consisten en una matriz de distancias acumuladas, proyectadas sobre una imagen que representa la profundidad en cada punto.

Se ha observado que muchos de los trabajos recientes obtienen unas tasas de acierto de entre el 80 % y el 90 % clasificando actividades; algunos incluso superiores al 90 %. Aún así, es difícil comparar estas tasas de acierto, debido a que los protocolos de validación suelen ser diferentes en cada trabajo de investigación. En general, se puede decir que las tasas de acierto son menores (o menos optimistas) cuando se realiza validación cruzada entre sujetos, es decir, los datos pertenecientes a unos sujetos son utilizados para entrenamiento y los otros para test. El estudio de Presti y La Cascia [6] incluye un buen resumen de los protocolos de validación más comunes, junto con tablas de tasas de acierto para diferentes métodos de extracción de datos y algoritmos de entrenamiento-clasificación utilizados en varios trabajos. Además, también realiza un análisis basado en latencia: en este análisis mide la influencia de la ventana de tiempo en el rendimiento del método de clasificación.

Los resultados analizados en este último trabajo han influido en la decisión de utilizar validación cruzada entre sujetos para el trabajo presentado aquí. En cuanto al análisis basado en latencia, se ha evaluado el rendimiento del clasificador en línea con una ventana de tiempo fija, clasificando acciones captadas desde el sensor Kinect en tiempo real.

Capítulo 3

Herramientas utilizadas

Se han utilizado conocidas herramientas y técnicas para el trabajo de investigación y software desarrollados. Como entrada de datos para entrenar y evaluar clasificadores, se han utilizado conjuntos de datos con información del esqueleto recogidos por otros investigadores. Este proceso de entrenamiento y evaluación se ha realizado con la ayuda de la biblioteca Scikit-Learn de Python. Para la clasificación en línea, la entrada de datos se ha realizado con el sensor Kinect. La información del esqueleto es proporcionada por la biblioteca OpenNI en tiempo real, y el software desarrollado procesa esta información para obtener la representación de datos deseada. Los datos obtenidos son finalmente consumidos por los clasificadores entrenados con Scikit-Learn. Nuestro método introduce novedades en la fase previa de preprocesado y extracción de características, y está diseñado para clasificar acciones en tiempo real con una ventana de tiempo predefinida, manteniendo una buena generalización entre sujetos. En este capítulo, veremos en detalle cada grupo de herramientas utilizadas, incluyendo datos, software y dispositivos físicos.

3.1. Sensor Kinect

El sensor Kinect (versión 1) de Microsoft se ha utilizado para los experimentos de clasificación en linea. Además, también es el dispositivo del que provienen los conjuntos de datos conocidos que se han utilizado para la fase de entrenamiento, validación y test.

Este dispositivo suele ser clasificado como una cámara RGB-D (o *RGB-Depth*) porque proporciona dos imágenes: una imagen RGB y una de profundidad (Figura 3.1). Una cámara RGB proporciona la primera imagen. La segunda imagen es obtenida con un sensor de infrarrojos, y Kinect tiene, además, un proyector que emite puntos de luz infrarroja en todo el campo de visión. El sensor de infrarrojos recoge estos puntos y obtiene una imagen de profundidad en base a su

Figura 3.1: Ejemplo de imagen de profundidad obtenida con Kinect

posición. De esta forma, se obtiene información de profundidad que no estaría disponible con una cámara RGB normal. Kinect es capaz de recoger imágenes RGB y de profundidad a 30 capturas por segundo (FPS). También incluye un micrófono y un motor para mover la orientación de la cámara, pero ninguno de estos dos componentes se utiliza para el trabajo presentado aquí.

3.2. Conjuntos de datos

Se han utilizado dos principales fuentes de datos para los experimentos: Microsoft Research Action 3D Dataset (MSRA3D; obtenido desde `http://research.microsoft.com/en-us/um/people/zliu/actionrecorsrc/`) y UT-Kinect-Action3D Dataset (UTKA), [4]. El origen de ambos conjuntos de datos es un sensor Kinect 1.

3.2.1. Conjunto de datos MSRA3D

Este conjunto de datos proporciona tanto información del esqueleto como imágenes de profundidad obtenidas con el sensor. En este caso, de entre la proporcionada, sólo hemos utilizado la información del esqueleto. Los datos se han obtenido de 20 acciones realizadas por 7 sujetos, con hasta 3 experimentos realizados por cada acción y sujeto. La representación del esqueleto contiene 20 puntos de articulaciones, que corresponden a los puntos del cuerpo representados en la Figura 3.2. Los datos vienen representados en ficheros de texto con capturas de las posiciones de los 20 puntos (en coordenadas espaciales 3D), ordenadas temporalmente. La lectura secuencial

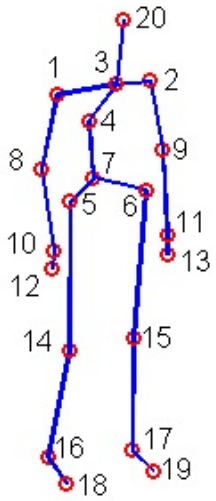


Figura 3.2: Modelo del esqueleto en el conjunto MSRA3D

de un fichero proporciona la serie temporal de un experimento, que es una de las 20 acciones realizada por uno de los 7 sujetos disponibles. Esta serie temporal se compone de 15 capturas por segundo (15 FPS), donde cada una de las capturas contiene 60 valores (20×3) de coordenadas 3D.

Las acciones que están recogidas en este conjunto de datos son las siguientes:

1. Saludo alto (con la mano)
2. Saludo horizontal (con la mano)
3. Mazo
4. Atrapar con la mano
5. Puñetazo frontal
6. Lanzamiento alto
7. Dibujar X
8. Dibujar tick
9. Dibujar círculo
10. Aplaudir
11. Saludar con las dos manos
12. Boxeo lateral
13. Inclinarse
14. Patada frontal
15. Patada lateral
16. Correr
17. Golpe de tenis
18. Saque de tenis
19. Golpe de golf
20. Levantar y arrojar algo

3.2.2. Conjunto de datos UTKA

El conjunto de datos proporcionado aquí es similar al anterior. En este caso, hay 10 acciones diferentes, 10 sujetos y cada uno de los sujetos realiza 2 veces cada acción. También se proporcionan datos del esqueleto e imágenes de profundidad; nosotros sólo utilizamos los primeros. Las series temporales de los experimentos son de "alrededor de 15 FPS" [12]. Los puntos de articulaciones y su correspondiente numérico son exactamente iguales que en el conjunto de datos anterior (Figura 3.2) La presentación de los datos es ligeramente diferente en este conjunto de datos: se presentan diferentes experimentos por cada sujeto, pero a lo largo de un experimento el sujeto está realizando diferentes acciones (o ninguna). Es otro fichero el que contiene las etiquetas de la acción realizada en cada rango de la serie temporal. Para hacer más fácil el tratamiento de estos datos, se han convertido al formato del conjunto de datos anterior, donde cada fichero se asocia a una acción, sujeto y experimento. Las acciones proporcionadas en este conjunto de datos son las siguientes:

1. Caminar
2. Sentarse
3. Levantarse
4. Recoger
5. Llevar
6. Lanzar
7. Empujar
8. Tirar
9. Agitar las manos
10. Aplaudir

3.3. Scikit-learn

Scikit-learn (`http://scikit-learn.org`) es una biblioteca de aprendizaje automático (*Machine Learning*) en Python. Proporciona herramientas para minería de datos y análisis de datos, y se integra con NumPy, SciPy y matplotlib. De esta biblioteca, se han utilizado los algoritmos *naive Bayes*, redes neuronales, máquinas de soporte vectorial y *random forests*. En el capítulo 5 se analizan más en detalle los algoritmos de aprendizaje y clasificación.

3.4. Robot Operating System (ROS)

ROS es un conjunto de software y bibliotecas de software libre para desarrollar aplicaciones para robots. Proporciona, entre otras cosas, abstracción de hardware, controladores de dispositivos, bibliotecas, visualizadores, envío de mensajes y gestión de paquetes.

Los programas de ROS se estructuran en paquetes (*packages*) y nodos (*nodes*). Los paquetes contienen los nodos, que son programas ejecutables, y también los ficheros que estos nodos necesitan para ejecutarse. Los nodos se comunican entre ellos mediante mensajes, a través de dos canales diferentes: *topics* y *services*.

Entre las bibliotecas proporcionadas por ROS se encuentran algunas que han sido útiles en este trabajo: un controlador de Kinect y un software que proporciona información del esqueleto a través de Kinect, la herramienta de visualización RViz y la biblioteca OpenNI (*openni_tracker*). Estos paquetes de software han formado parte del entorno de desarrollo para un software de reconocimiento de acciones. Este es el software que se ha utilizado para los posteriores experimentos de clasificación en línea. El clasificador puede ser ejecutado de forma independiente como un nodo de ROS, por lo que puede reutilizarse en futuros proyectos de robótica. A pesar de utilizar ROS, el trabajo de desarrollo se ha centrado en desarrollar un módulo de reconocimiento de actividades y acciones, y no en una aplicación completa para un robot.

3.4.1. Biblioteca *openni_tracker*

Este paquete de ROS es el proveedor de los puntos del esqueleto en tiempo real. Los puntos se proporcionan a través de la estructura *tf* de ROS, que se utiliza para representar transformaciones (habitualmente poses de robots). ROS incluye herramientas para interactuar con la estructura *tf*, lo que permite obtener con facilidad las coordenadas de los puntos del esqueleto en el sistema de referencia del sensor (o, incluso, en el sistema de referencia de otro punto).

A diferencia de los conjuntos de datos vistos en el apartado 3.2, el número de puntos que proporciona es 15 (en lugar de 20), tal y como se representa en la Figura 3.3. No sería posible entrenar con modelos de clasificación con 20 puntos y utilizarlos para clasificar esqueletos de 15, por lo que se han eliminado 5 puntos de los conjuntos de datos y asumido que los puntos seleccionados son equivalentes en la articulación correspondiente (e.g. el punto 20 de la Figura 3.2 y el punto 1 de la Figura 3.3 se asumen como el mismo punto, debido a que ambos representan la cabeza).

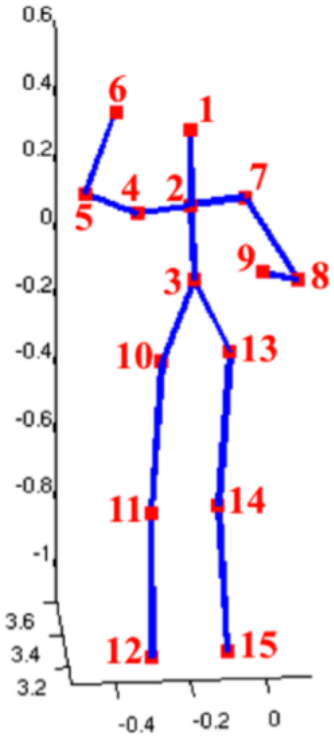


Figura 3.3: Modelo del esqueleto proporcionado por OpenNI.

Capítulo 4

Extracción de características y preprocesado

La única fuente de datos de la que se han extraído características ha sido la información del esqueleto proporcionada por los conjuntos de datos de fuentes externas (MSRA3D y UTKA), y el esqueleto proporcionado por OpenNI. La información del esqueleto, además, contiene cierta información temporal implícita: no se incluye un tiempo de recogida de los datos, pero se sabe la frecuencia de datos de un *experimento* o serie temporal (en las fuentes externas, 15 FPS), y también que los datos están ordenados cronológicamente. En este capítulo, analizamos cómo se ha realizado la extracción de características relevantes a partir de estos datos. También se describen los filtros de preprocesado aplicados a estas características, como paso previo al entrenamiento y test-validación.

4.1. Visualización de datos

Debido a que se han obtenido fuentes de datos externas, la visualización de datos es importante para detectar, a simple vista, las características de los datos y los posibles problemas que habrá al procesarlos. En este caso, los datos pueden ser vistos como series temporales de varios puntos, donde los puntos son las articulaciones del esqueleto. Una buena forma de visualizar estas series temporales es representarlas como un esqueleto en movimiento, de forma que se pueda obtener una idea clara de qué secuencias capturó el sensor, para después procesarlas.

Un *frame* de esta serie, o una sola postura, puede visualizarse como un *scatter plot* o gráfico de dispersión de puntos. Como ejemplo, en la Figura 4.1, podemos observar un gráfico de dis-

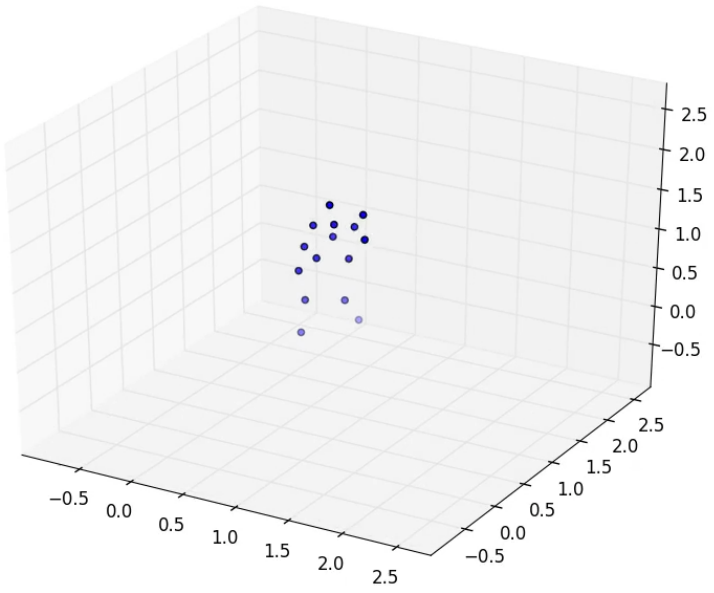


Figura 4.1: Gráfico de dispersión de los puntos del esqueleto, para la visualización de una pose.

persión generado con PyPlot en Python. Para visualizar toda una serie temporal, se genera una imagen por cada *frame* o pose dentro del movimiento. Estas imágenes pueden combinarse en una animación con una herramienta como FFMpeg, obteniendo como resultado un fichero de vídeo donde el movimiento captado por el esqueleto es fácilmente observable.

4.2. Información espacial

La información espacial del conjunto de datos sin tratar viene dada por la representación en esqueleto de una *pose*. Definimos como *pose métrica* una posición estática del cuerpo humano representada en un vector ordenado de puntos del espacio euclídeo 3D que llamaremos *articulaciones*. Sea una pose E, su representación en k articulaciones viene dada por un el vector $E = \{J_0, ..., J_k\}$, donde para cada articulación i, $0 \leq i \leq k$, J_i es un vector de coordenadas $J_i = \{j_{ix}, j_{iy}, j_{iz}\}$. Esta *pose métrica* es simple y contiene suficiente información para representar una postura del cuerpo. Además, el sistema de referencia podría estar anclado en una articulación del esqueleto, eliminando la influencia de la distancia al sensor en los datos.

Pero la representación en *pose métrica* provoca un problema: las distancias entre las articulaciones son diferentes en cada sujeto. En un conjunto de datos con muestras suficientemente amplias para cada sujeto, esto se podría solucionar con la normalización de los datos (es suficiente con obtener los parámetros μ, σ, a partir de la media y la desviación estándar, respectivamente). En cambio, si queremos que esta representación también sirva para clasificación en línea, estos parámetros no se pueden obtener de una muestra suficientemente grande. En este caso,

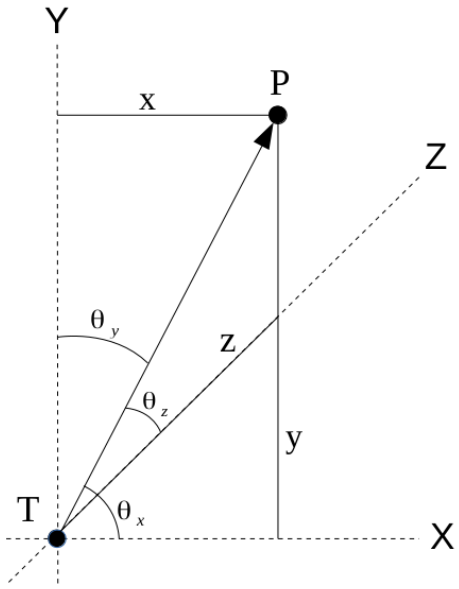


Figura 4.2: Ángulos de un punto del esqueleto relativos al punto T del torso.

necesitamos otro tipo de representación que varíe menos entre diferentes sujetos.

4.2.1. Representación en ángulos de los ejes

Una forma de representar diferentes poses sin que influya el tamaño del cuerpo humano en la representación es tener en cuenta sólo las orientaciones entre unas y otras articulaciones. Manteniendo un sistema de coordenadas euclídeo y el sistema de referencia centrado en una articulación (supongamos que es un punto central como el *torso* T), representamos el resto de articulaciones como los ángulos respecto a los ejes X, Y y Z del espacio euclídeo (Figura 4.2). Sabiendo las coordenadas (x, y, z) de un punto, estos ángulos pueden calcularse como:

$$\theta_x = arctan(\frac{y}{x})$$

$$\theta_y = arctan(\frac{x}{y})$$

$$\theta_z = arctan(\frac{x}{z})$$

Esta representación ha dado buenos resultados para la clasificación de posturas estáticas en un enfoque de aprendizaje supervisado [1], pero tiene una alta dimensionalidad y se reduce su eficacia al clasificar posturas que dependen de muchas articulaciones. Además, contiene información redundante: representa una orientación en 3 valores, exactamente los mismos necesarios

para una coordenada en el espacio, mientras que una orientación podría ser representada con tan solo dos coordenadas (lo veremos en la sección 4.2.2). En el esqueleto proporcionado por OpenNI, la representación en coordenadas 3D o ángulos respecto a los ejes tendría un total de 42 características (14 puntos relativos y 3 valores o características por punto).

4.2.2. Representación en coordenadas esféricas

El sistema de coordenadas esféricas representa un punto en el espacio con tres valores: la distancia radial desde el punto de origen (r), el ángulo polar (θ) medido desde una dirección cénit, y el ángulo acimutal (φ) de su proyección en un plano de referencia ortogonal al cénit, medido desde una dirección de referencia en ese plano. [13]

El sistema de coordenadas esféricas se utiliza en muchos ámbitos y con diferentes convenciones. Por ejemplo, las coordenadas terrestres se representan en latitud, longitud y altura. Las estrellas en la bóveda celeste también se representan en un sistema de coordenadas esférico. En estos dos casos particulares, el ángulo acimutal se calcula en sentido horario, pero la convención más habitual es el sentido antihorario. También es común observar sistemas de coordenadas esféricas que reemplazan el ángulo polar por el ángulo de elevación.

La notación (r, θ, φ) que utilizamos aquí, donde θ y φ son los ángulos polar y acimutal respectivamente, es la más común en física, y es también la especificada por el estándar ISO 80000-2. En matemáticas, la convención es representar las coordenadas como (r, θ, φ) o (r, θ, ϕ), pero se invierte el significado de θ y φ: θ es el ángulo acimutal y φ es el ángulo polar. [14]

Para representar puntos del esqueleto, utilizamos sólo la orientación – ángulos (θ, φ) – en coordenadas esféricas, suprimiendo la distancia radial. Para ello, seguiremos la convención más habitual: el ángulo acimutal φ obtenido en sentido antihorario desde el eje X, y el ángulo polar θ obtenido a partir del cénit, como en la Figura 4.3. En el modelo de esqueleto de OpenNI, estos dos valores por cada una de las articulaciones forman un total de 28 características, proporcionando la misma información que las 42 características de la representación en tres ángulos.

En el contexto de las articulaciones del esqueleto, llamaremos $P_t = (p_{tx}, p_{ty}, p_{tz})$ al punto donde se sitúa la articulación de referencia: en nuestro modelo, el torso. Para cada uno del resto de puntos $P_i = (p_{ix}, p_{iy}, p_{iz})$, las coordenadas esféricas (θ_i, φ_i) (omitiendo el radio) se obtienen con las siguientes ecuaciones:

$$\theta_i = \arctan(\frac{p_{iy} - p_{ty}}{p_{ix} - p_{tx}})$$

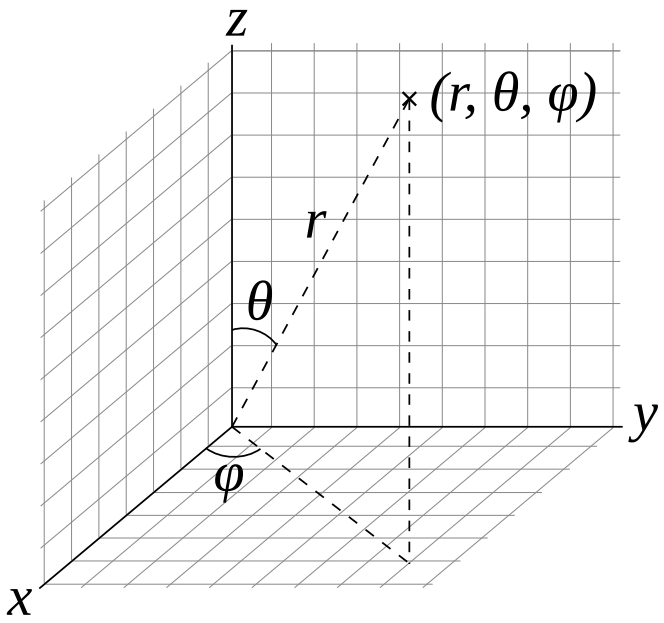


Figura 4.3: Coordenadas esféricas (r, θ, φ) de un punto: distancia radial r, ángulo polar θ, ángulo acimutal φ.

$$\varphi_i = \arctan(\frac{p_{iz} - p_{tz}}{r})$$

donde r es el radio de la esfera, o la distancia euclídea entre los dos puntos,

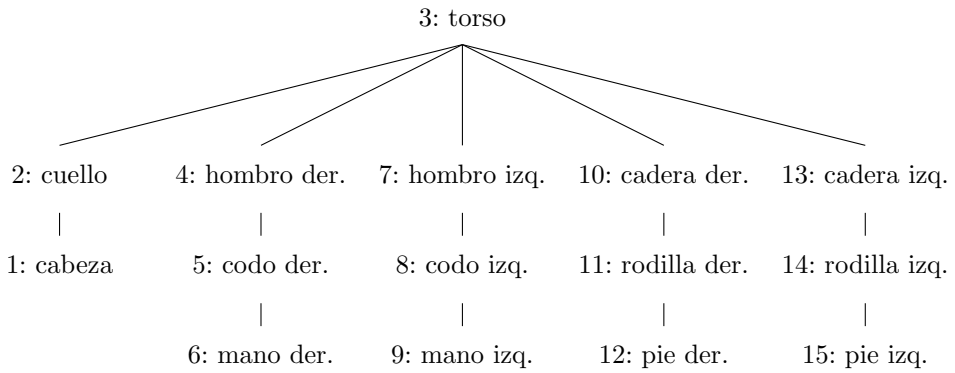
$$r = \sqrt{(p_{ix} - p_{tx})^2 + (p_{iy} - p_{ty})^2 + (p_{iz} - p_{tz})^2}$$

El radio obtenido en la última ecuación también formaría parte del sistema de coordenadas esféricas, pero, en nuestro caso, lo omitimos debido a que sólo queremos obtener información de la orientación y omitir la distancia.

4.2.3. Sistemas de referencia jerárquicos

Una posible forma de representar el esqueleto es utilizar varios sistemas de referencia para las coordenadas relativas, en lugar de utilizar sólo un punto central como referencia. El enfoque propuesto ha sido definir un punto central (*torso*) en el que estén representadas las articulaciones contiguas (cuello, hombros y caderas). Los siguientes puntos de las extremidades serían *descendientes* de sus respectivas articulaciones contiguas, y estarían representados en el sistema de referencia de la articulación *raíz*. Por ejemplo, la rodilla izquierda sería descendiente de la cadera izquierda, es decir, el punto de origen de su sistema de coordenadas se encuentra en la cadera izquierda. Siguiendo este esquema, y usando como referencia el esqueleto de OpenNI (Figura

3.3), se ha propuesto el siguiente árbol jerárquico de articulaciones:



Hemos realizado experimentos de clasificación supervisada con esta representación y la representación que utiliza el *torso* (articulación 3) como único sistema de referencia, utilizando algoritmos Naive Bayes, SVM y Random Forest. Tras observar los resultados, no se ha podido concluir que ninguna de las dos representaciones ofrezca una mejora significativa, debido a que los resultados han sido muy similares. Siguiendo el criterio de utilizar el sistema más simple, se ha descartado este esquema a favor del sistema de referencia único, como hemos visto en la sección 4.2.2.

4.3. Información espacial y temporal

Mientras que la representación de un solo esqueleto es suficiente para reconocer una pose estática, para reconocer una acción, actividad o movimiento es necesario observar una serie de poses en una ventana de tiempo. Si se recogiera toda la información disponible, a 15 FPS, y en representación en coordenadas esféricas, una ventana de tiempo contendría 420 características por segundo. En secuencias de 5 o 10 segundos, este incremento en características se vuelve inviable, por lo que es necesario obtener las nuevas características de otro modo.

La alternativa propuesta es definir una ventana de tiempo de tamaño v (v es el número de fotogramas). En un instante de tiempo t, se obtienen las velocidades angulares medias de θ, φ en cada una de las articulaciones en el tiempo entre $[t - v, t]$. Estas velocidades angulares se añaden a las características de la pose, creando una *pose extendida*. Esta *pose extendida* contiene la posición de las articulaciones en el instante t, junto con las velocidades angulares de cada articulación en el instante de tiempo t y la ventana de tiempo v. Como se muestra en la ecuación 4.1, la velocidad angular media es la diferencia entre el ángulo inicial ϕ_0 y el ángulo final ϕ_1

dividida entre la ventana de tiempo v.

$$\omega = \frac{|\phi_0 - \phi_1|}{v} rad/s \tag{4.1}$$

Para añadir más información temporal a la *pose extendida*, se ha realizado lo mismo para la ventana de tiempo $v/2$. De esta forma, se añaden 4 valores de velocidad angular en total por cada articulación. Tras realizar experimentos con diferentes combinaciones y valores de ventanas de tiempo – se realizaron experimentos con hasta 3 y 4 ventanas de tiempo, y también añadiendo velocidades acumuladas como nueva característica. Ninguno de los experimentos mostró una mejora significativa –, se decidió utilizar las dos ventanas de tiempo v y $v/2$ que hemos definido, con $v = 8$ fotogramas ($\sim 0,53$ segundos). Este tamaño de ventana de tiempo es razonable tanto para capturar movimientos rápidos como para obtener valores de velocidad coherentes.

4.4. Preprocesado

El paso de preprocesado es fundamental para obtener datos útiles para la clasificación de los conjuntos de datos originales. En primer lugar, los datos se han representado en un formato estándar. Después, se han realizado diferentes operaciones para suavizar los efectos de lecturas erróneas y eliminar información poco relevante y redundante.

4.4.1. Estandarización de los datos

La representación en coordenadas esféricas es la forma empleada de estandarizar el conjunto de datos. Se han normalizado las coordenadas para que estén representadas en el rango $[-\pi, \pi]$. Ocurre lo mismo con las velocidades angulares: al estar obtenidas de distancias angulares absolutas, sus escalas son estándares y no varían en diferentes individuos ni experimentos del conjunto de datos.

El único problema derivado de este formato de datos es que, en un ángulo, el valor $-\pi$ es cercano al valor π. Esto puede ocasionar problemas con clasificadores que entrenen distribuciones de probabilidad agrupadas en un valor medio (por ejemplo, un clasificador *naive bayes* que entrene distribuciones de probabilidad gaussianas). Deberá tenerse en cuenta esta particularidad al elegir diferentes clasificadores y algoritmos de entrenamiento.

La otra magnitud que se selecciona para el conjunto de datos es la velocidad angular. Esta es una magnitud diferente a la anterior, con una escala y unidades diferentes. Esto causará problemas en clasificadores sensibles a la escala, como las redes neuronales. Por tanto, se deberán

escalar los datos, pero sólo si el clasificador a utilizar lo exige.

4.4.2. Media móvil

Los conjuntos de datos capturados y obtenidos de fuentes externas se presentan en series temporales de poses en intervalos de tiempo fijos. Dichas series tienen habitualmente pequeños picos de lectura, provocados por lecturas erróneas o baja precisión en el sensor. Con el objetivo de suavizar estos picos de lectura, se ha aplicado un filtro de media móvil de tamaño 3. Se han realizado también experimentos de entrenamiento y validación con medias móviles de tamaños 5 y 10, pero con este tamaño se han obtenido mejores resultados. La media móvil de tamaño 3, en concreto, se aplica reemplazando cada uno de los valores de la serie temporal en el instante de tiempo t por la media entre los tres valores desde $t-1$ hasta $t+1$.

4.4.3. Eliminación de información irrelevante

Al igual que en el filtro anterior, asumimos que la fuente de datos consiste en series temporales de poses en intervalos de tiempo fijos. Mediante la visualización de las secuencias de datos por experimentos, se ha podido observar que al inicio de cada experimento el sujeto permanece habitualmente inmóvil durante un tiempo (desde menos de 1 segundo hasta varios segundos). Al final del experimento, también es posible que haya un tiempo de inactividad. Para evitar procesar estos intervalos de inactividad como movimientos asociados a una acción, aplicamos un filtro que elimina todos los *frames* iniciales y finales con poses muy similares entre sí.

Para poder aplicar el filtro, primero es necesario saber qué son dos poses *similares*. En primer lugar, se puede obtener una medida de proximidad entre dos puntos en coordenadas esféricas. Esta medida, aplicada a la misma articulación en dos poses diferentes, proporcionaría la similitud en un punto de las poses. En el espacio euclídeo, se podría utilizar la distancia euclídea para medir la proximidad entre dos puntos. Pero como nuestros puntos están situados en una misma esfera, una métrica más adecuada para este propósito es la distancia geodésica. La distancia geodésica entre dos coordenadas $A = (a_\theta, a_\varphi)$ y $B = (b_\theta, b_\varphi)$ es:

$$|\arccos(\sin a_\theta \cdot \sin b_\theta + \cos a_\theta \cdot \cos b_\varphi \cdot \cos d_\phi(a_\varphi, b_\varphi))|$$

donde $d_\phi(a_\varphi, b_\varphi)$ es la distancia angular entre a_φ y b_φ.

$$d_\phi(a, b) = (a - b) \text{ mód } \pi$$

Tras calcular las distancias geodésicas entre los pares correspondientes de articulaciones, se compara la distancia máxima con un valor límite predefinido. Si la distancia máxima supera este valor, una de las dos poses se descarta del conjunto de datos. En concreto, el algoritmo del filtro es el siguiente:

```
1  k := número de articulaciones;
2  pose := primer (o último) elemento de la serie;
3  límite := distancia geodésica máxima entre dos puntos de poses demasiado cercanas;
4  while longitud(serie) > 1 and not parar do
5  |  siguiente = siguiente pose de la serie;
6  |  dist[1...k] := d_φ(pose[1...k], siguiente[1...k]);
7  |  if max(dist) < límite then
8  |  |  eliminar pose de la serie;
9  |  |  pose := siguiente;
10 |  else
11 |  |  parar := true;
12 |  end
13 end
```

Algoritmo 1: Algoritmo de filtrado de secuencias inactivas.

Tras aplicar este filtro, se eliminan muchas de las poses que no contienen ninguna información acerca del movimiento que se ha capturado, evitando futuros errores en el entrenamiento y test de clasificadores.

Capítulo 5

Modelos de clasificación

Aunque en este trabajo nos hemos centrado en proponer una selección de características para clasificar movimientos en tiempo real, hemos tenido en cuenta diferentes modelos para la tarea de clasificación supervisada. Se han propuesto tres modelos de aprendizaje supervisado: máquinas de soporte vectorial, *random forests* y redes neuronales. Además, también se ha experimentado con el clasificador *naive Bayes* para comparar resultados con un clasificador que, en teoría, es menos adecuado para nuestra representación, porque utiliza distribuciones gaussianas de probabilidad.

También se ha propuesto un *histograma de clasificaciones instantáneas*, que toma como entrada las clasificaciones en varios instantes de tiempo para generar el resultado de clasificación de una serie temporal.

En este capítulo se describen los modelos de clasificación utilizados y el histograma de clasificaciones instantáneas empleado para clasificar ventanas de tiempo. En el siguiente capítulo (6), se describen los experimentos realizados, los protocolos de validación de los clasificadores y los resultados obtenidos con cada uno de los modelos de clasificación.

5.1. Máquinas de soporte vectorial

Las máquinas de soporte vectorial (*support vector machines*, SVMs) [15] son modelos de aprendizaje supervisado, asociados con algoritmos de entrenamiento que generan un clasificador binario lineal no probabilístico.

Un SVM construye un hiperplano en un espacio de alta dimensión, situado entre el espacio ocupado por los miembros de las dos diferentes clases a clasificar. Este hiperplano puede ser utilizado para regresión, o para clasificación de nuevas instancias como en nuestro problema. Intuitivamente, el hiperplano debería estar situado a la mayor distancia del punto de cada clase

más cercano: a mayor distancia, menor será el error de generalización del clasificador.

La limitación del hiperplano es que, mientras que las instancias pueden estar en un espacio de dimensiones finitas, es posible que las instancias de diferentes clases no sean linealmente separables. Para poder computar un SVM en estos casos, se puede aplicar una función kernel $k(x, y)$ seleccionada para el problema. Esta función kernel transforma el espacio en un espacio de más dimensiones, donde un hiperplano sería más adecuado para separar las clases del problema. Típicamente, esta función kernel podría ser, por ejemplo, lineal (equivalente al SVM lineal), polinomial o sigmoide. También puede ser cualquier otra función adecuada para el problema.

El clasificador SVM utilizado en nuestros experimentos para clasificar movimientos utiliza un kernel polinomial de grado 2. Los experimentos han mostrado mejores resultados con este tipo de kernel que con el lineal, sigmoide o de función tangente hiperbólica. Debería destacarse, sin embargo, que en el trabajo de [8] utilizan un kernel específico para coordenadas esféricas con resultados interesantes en clasificación de posturas estáticas.

5.2. Random forests

Los *random forests* o bosques aleatorios son conjuntos de árboles de decisión generados de forma no determinista. Habitualmente, cada uno de estos árboles se genera a partir de un subconjunto aleatorio de todas las características disponibles en el conjunto de entrenamiento. El tamaño de este subconjunto, en nuestro caso es de $\sqrt{n}$, donde n es el número de características. El resultado de la clasificación se obtiene con el criterio de la mayoría o el resultado medio de todos los árboles. Para entrenar los árboles de decisión, el algoritmo de entrenamiento selecciona iterativamente variables del subconjunto del árbol utilizando un criterio estadístico: dos comúnmente usados son la impureza Gini (utilizado por el algoritmo CART) y la ganancia de información (utlizado por los algoritmos ID3 [16], C4.5 [17] y C5.0 [18]). La motivación para los *random forests* es reducir el error de generalización (evitar el sobreajuste u *overfitting*), ya que el error de generalización converge a un límite al aumentar el número de árboles del bosque [19].

La impureza Gini se obtiene sumando, para cada elemento i, la probabilidad f_i de que el elemento sea seleccionado multiplicado por la probabilidad de que no sea seleccionado:

$$I_G(f) = \sum_{i=1}^{J} f_i(1 - f_i)$$

La ganancia de información $IG(A, B)$ se obtiene a partir de la entropía $H(A)$, $H(A|B)$:

$$IG(A, B) = H(A) - H(A|B)$$

donde la entropía $H(f)$ se define:

$$H(f) = -\sum_{i=1}^{J} f_i log_2 f_i$$

En este trabajo, se han realizado experimentos entrenado bosques de decisión con el criterio de impureza Gini y con el criterio de ganancia de información, y se han obtenido resultados ligeramente mejores con el primero de los criterios. Por este motivo, los resultados presentados en el próximo capítulo se han obtenido de a partir de bosques entrenados con el criterio de impureza Gini. Los bosques aleatorios han sido entrenados con 50 árboles de decisión aleatorios.

5.3. Redes neuronales

Recientemente, las redes neuronales han generado más interés debido a los avances en investigación con nuevas arquitecturas de redes neuronales y la extensión de las técnicas de *deep learning*, que engloban las diferentes técnicas de aprendizaje y arquitecturas de redes neuronales profundas [20]. En este trabajo, hemos realizado experimentos con un modelo clásico de red neuronal profunda: el perceptrón multicapa o *multilayer perceptron* (MLP).

Este tipo de red neuronal se organiza por capas de *neuronas* o unidades de procesamiento. La primera de las capas, la capa de entrada, consiste en un conjunto de neuronas $\{x_i|x_1, x_2, ..., x_m\}$ que representa las características de entrada. Cada una de las capas ocultas, situadas entre la capa de entrada y la de salida, transforma los valores la capa anterior en una suma lineal ponderada $w_1x_1 + w_2x_2 + ... + w_mx_m$ seguida por una función de activación no lineal, como una tangente hiperbólica. La capa de salida recibe los valores de la última capa oculta y los transforma en valores de salida. Estas redes neuronales son capaces de modelar funciones no lineales. Además, los pesos de las neuronas pueden ser actualizados en tiempo real, ofreciendo la posibilidad de adaptar el modelo.

Dados los ejemplos de entrenamiento $(x_1, y_2), (x_2, y_2), ..., (x_n, y_n)$ donde $x_i \in \mathbb{R}$ e $y_i \in \{0, 1\}$, una red neuronal de una capa oculta aprende la función $f(x) = W_2 g(W_1^T x + b_1) + b_2$ donde los parámetros del modelo son $W_1 \in \mathbb{R}^m$ y $W_2, b_1, b_2 \in \mathbb{R}$. W_1 y W_2 son los pesos de la capa de entrada y la capa oculta, respectivamente; b_1 y b_2 son los sesgos añadidos a la capa de entrada y capa oculta; y $g(\cdot)$ es la función de activación de la red neuronal.

La red neuronal se entrena, en términos generales, minimizando el error cuadrático de ca-

da una de las capas mediante propagación hacia atrás (*backpropagation*) [21]. Estos errores cuadráticos pueden ser minimizados con el algoritmo de descenso del gradiente estocástico. El error minimizado durante el entrenamiento suele estar condicionado a varios parámetros, que penalizan aspectos como la complejidad de la red o el error de generalización. Estos parámetros deben ser seleccionados cuidadosamente para entrenar una red adecuada para nuestro problema de clasificación.

La red neuronal entrenada para nuestro problema tiene una capa oculta de 30 neuronas y su función de activación es la función logística:

$$f(x) = \frac{1}{1 + e^{-x}}$$

La red neuronal se ha entrenado con el algoritmo de descenso del gradiente estocástico (SGD), con una fracción de validación de 33 % de la partición de entrenamiento. Se ha utilizado una razón de aprendizaje adaptativa: la razón de aprendizaje se inicializa a 0,3. Si dos iteraciones consecutivas fracasan en reducir el error en, al menos, un valor de tolerancia (fijado en 0,0001), esta razón de aprendizaje se divide entre 5. El error de generalización se penaliza mediante una regularización L2, fijando el valor del parámetro α a 0,001. Se ha fijado un máximo de 10000 iteraciones para el proceso de optimización.

Debido a que las redes neuronales son sensibles a la escala de los datos de entrada, este es el único caso donde los datos han sido escalados en la etapa de preprocesado. Todas las variables han sido centradas alrededor del 0 y normalizadas a varianza 1.

5.4. Naive Bayes

El clasificador bayesiano ingenuo (o *naive Bayes classifier*) se basa en la regla de bayes para entrenar una distribución de probabilidad. La regla de bayes, para la distribución de probabilidad de cada una de las clases C_k condicionada a las características $x_1, x_2, ..., x_n$ sería:

$$p(C_k|x_1, x_2, ..., x_n) = \frac{p(C_k)p(x_1, x_2, ..., x_n|C_k)}{p(x_1, x_2, ..., x_n)}$$

El numerador equivale a la probabilidad conjunta $p(C_k, x_1, x_2, ..., x_n)$, que puede reescribirse, siguiendo la regla de la cadena a $p(x_1|x_2, ..., x_n, C_k)p(x_2|x_3, ..., x_n, C_k)...p(x_{n-1}|x_n, C_k)p(x_n|C_k)p(C_k)$. Aquí realiza la suposición *ingenua* de este clasificador: cada una de las características es independiente de todas las demás dada la clase C. Esto significa que:

$$p(x_i|x_{i+1}, ..., x_n, C_k) = p(x_i|C_k)$$

Por tanto, todo el modelo probabilístico puede expresarse como:

$$p(C_k|x_1, ..., x_n) = \eta \; p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{5.1}$$

donde η es un factor de normalización sólo dependiente de las variables $x_1, ..., x_n$.

El clasificador *naive Bayes* calcula la distribución de probabilidad de cada clase dadas las características en base a la ecuación 5.1. El producto de distribuciones de probabilidad suele obtenerse como un producto de distribuciones gaussianas, cuyos únicos parámetros son la media y la varianza (μ, σ^2). Finalmente, los únicos parámetros que entrena el clasificador son los (μ, σ^2) de la distribución de probabilidad gaussiana para $p(C_k|x_1, ..., x_n)$.

Este clasificador ha sido elegido para comparar su rendimiento al de otros clasificadores porque, hipotéticamente, debería ser problemático y dar peores resultados. Las características que se utilizan para entrenar el modelo son ángulos en el intervalo $[-\pi, \pi]$ y velocidades angulares absolutas. Mientras que una distribución gaussiana podría ser adecuada para velocidades angulares que sólo son positivas, no lo es para los ángulos de las coordenadas esféricas.

Supongamos un ejemplo simplificado, donde queremos entrenar un modelo condicionado a una sola variable angular x (por ejemplo, el ángulo acimutal de uno de los puntos del esqueleto). Alrededor del origen todos los valores son positivos y negativos cercanos al 0, por lo que si la distribución de probabilidades de una clase está centrada en el origen no habría muchos problemas. La situación es diferente con los valores extremos π y $-\pi$. En coordenadas angulares, ambos son equivalentes, y los valores próximos a π también lo son a $-\pi$. Si muchos casos de una clase contienen ángulos cercanos a π, algunos de ellos serán extremos positivos y otros extremos negativos. Esto daría lugar a una distribución gaussiana con una varianza muy grande, entrenando un modelo que aporta poca información para la clasificación. Otros métodos, como la estimación de funciones de densidad para cada variable [22], podrían haber ayudado en este problema. En este trabajo no los hemos considerado: no estaban disponibles en la biblioteca *Scikit-learn* y se tomó la decisión de no realizar una implementación propia.

5.5. Histograma de clasificaciones instantáneas

Las características seleccionadas para los clasificadores vistos hasta ahora se centran en instantes de tiempo concretos (salvo por la característica de la velocidad, que tiene en cuenta un

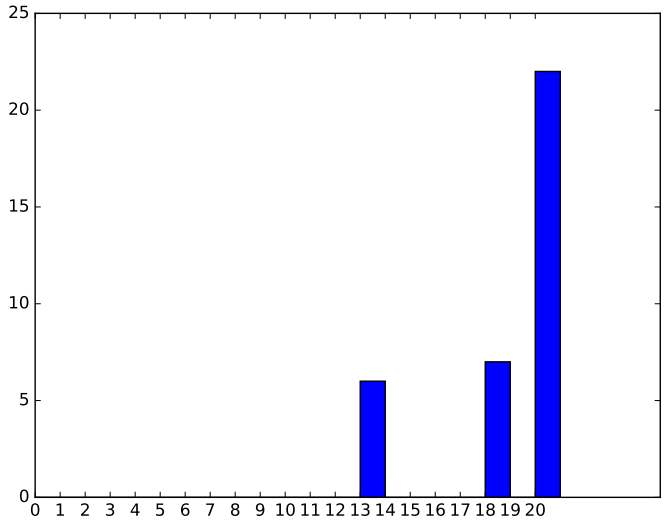


Figura 5.1: Histograma de clasificaciones instantáneas que daría como resultado la clase 20 (*levantar y arrojar*).

intervalo de poco más de medio segundo) para clasificar actividades que pueden durar varios segundos. Algunos de los movimientos a clasificar podrían tener situaciones instantáneas muy similares (podemos pensar, por ejemplo, en un instante donde se levanta el brazo para *atrapar con la mano*, y también para *saludar*). Mientras que la clasificación instantánea puede proporcionar cierta inmediatez, normalmente será necesario tener en cuenta una ventana de tiempo de unos segundos para reconocer un movimiento con mayor fiabilidad.

La solución propuesta para considerar una ventana de tiempo más larga con los anteriores clasificadores es utilizar un *histograma de clasificaciones instantáneas*. Este histograma es un vector $H = \{h_1, .., h_n\}$, donde h_i es el número de veces que una clasificación instantánea ha obtenido la clase $i = 1, .., n$ en la ventana de tiempo w. Hemos construido un meta-clasificador sencillo basado en este vector, donde la salida de la clasificación es $arg\ max_i H(i)$. En la figura 5.1 podemos encontrar un ejemplo de cómo queda este histograma tras acumular clasificaciones en una ventana de tiempo de 35 instantes.

Capítulo 6

Resultados experimentales

Se han realizado dos tipos de experimentos con el objetivo de probar la eficacia de los métodos de clasificación propuestos. El primero de ellos es la clasificación *offline*: se ha elegido el conjunto de datos MSRA3D con los conjuntos de clases propuestos en el artículo [3] (AS1, AS2, AS3), y también un subconjunto en el que eliminamos las clases del 3 al 9 (*mazo*, *atrapar con la mano*, *puñetazo frontal*, *lanzamiento alto*, *dibujar X*, *dibujar tick*, *dibujar círculo*). Llamaremos a este subconjunto AS4. También se han obtenido resultados para el conjunto de datos UTKA, con todas las clases del conjunto.

En este experimento, se ha entrenado el modelo de clasificación con un subconjunto de sujetos (sujetos de entrenamiento), y se han obtenido resultados con el subconjunto restante (sujetos de prueba). Este tipo de validación es referida a veces como validación cruzada entre sujetos o *cross-subject validation*. En concreto, realizamos un entrenamiento-validación del modelo para cada uno de los 10 sujetos del conjunto de entrenamiento, con un sujeto de prueba y los restantes 9 de entrenamiento. En otras palabras, nuestro protocolo de validación consiste en dejar un sujeto fuera del conjunto de entrenamiento (*leave one out*). Este tipo de validación penaliza mucho el error de generalización: obtendrán mejores resultados aquellos modelos que funcionen mejor en sujetos que no hayan sido *vistos* durante el entrenamiento. Los modelos que se ajusten demasiado a los sujetos del conjunto de entrenamiento obtendrán malos resultados, a pesar de que algunos obtendrían buenos resultados con otro tipo de protocolo (como una partición entrenamiento/test independiente de los sujetos). Un buen clasificador debería ser capaz de generalizar entre sujetos, por lo que la validación cruzada entre sujetos es más realista que otro tipo de protocolos que pueden encontrarse en trabajos similares.

El segundo experimento ha consistido en entrenar un clasificador con el conjunto de datos MSRA3D y probarlo en tiempo real, con información recogida por el sensor Kinect. El entorno

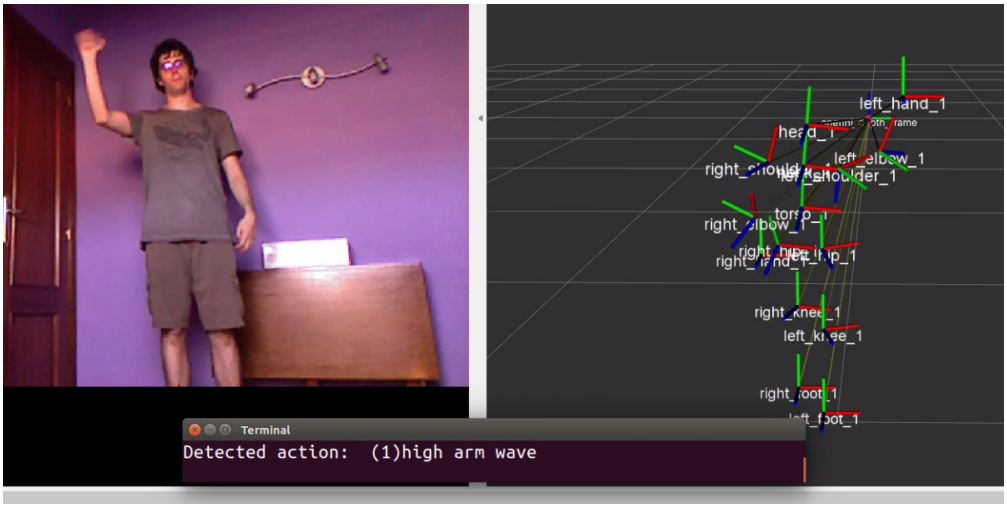


Figura 6.1: Ejemplo de reconocimiento de acciones en tiempo real, donde se reconoce un saludo.

de captura es el que se puede observar en la Figura 6.1. Al igual que en el experimento anterior, se puede decir que los sujetos de entrenamiento y test son diferentes: el sujeto de pruebas no estuvo presente en los experimentos del conjunto de datos original. Incluso el equipo de captura de datos y la biblioteca de software utilizada para generar el esqueleto es diferente. Aún así, los resultados obtenidos son interesantes, ya que el modelo ha podido generalizar lo suficiente como para reconocer algunas posturas con tasas de acierto cercanas a la clasificación *offline*.

En las tablas de este capítulo se muestran como resultados de clasificación *instantánea* aquellos que se han obtenido directamente de los clasificadores SVM, RF, MLP y NB. En general, estos resultados son malos: su tasa de acierto es mucho menor que la de otros métodos que procesan más información de la serie temporal. Para resolver este problema, la clasificación de una serie temporal (o una ventana de tiempo) que representa un movimiento se realiza con el histograma de clasificaciones instantáneas visto en la sección 5.5. Este histograma mejora las tasas de acierto de los clasificadores hasta en un 30 %.

6.1. Clasificación *offline*

En el Cuadro 6.1 se muestran las tasas de acierto obtenidas con el protocolo *leave one out cross-subject validation* en el conjunto de datos MSRA3D. Comparamos estas tasas de acierto por conjunto de datos y método de clasificación (SVM: Support Vector Machines, RF: Random Forest, MLP: Multilayer Perceptron, NB: Naive Bayes). Se incluyen resultados de clasificación instantánea y clasificación con histograma, donde el histograma clasifica cada una de las series temporales completas del sujeto de prueba. El resultado mostrado es la media de los resultados de cada uno de los sujetos de prueba.

Cuadro 6.1: Tasas de acierto *offline* para el conjunto de datos MSRA3D

Datos: MSRA3D	AS1	AS2	AS3	AS4
SVM, instantánea	50.06 %	54.81 %	64.05 %	62.13 %
SVM + Hist.	68.31 %	61.43 %	81.44 %	83.02 %
RF, instantánea	55.45 %	55.94 %	72.31 %	72.54 %
RF + Hist.	**70.81 %**	60.83 %	**86.58 %**	**90.89 %**
MLP, instantánea	49.43 %	55 %	65.11 %	63.45 %
MLP + Hist.	64.38 %	**62.46 %**	80.48 %	81.82 %
NB, instantánea	33.94 %	39.07 %	56.01 %	39.64 %
NB + Hist.	44.16 %	39.54 %	68.19 %	46.86 %

Del mismo modo, también mostramos resultados para el conjunto de datos UTKA en el Cuadro 6.2. En este caso, no se ha dividido el conjunto de datos en grupos de acciones, sino que se han obtenido resultados experimentales para todas las acciones. El protocolo empleado para obtener estos resultados ha sido el mismo que para la tabla anterior.

Se puede comprobar (Cuadro 6.1) que los conjuntos de datos AS1 y AS2 obtuvieron peores resultados que el resto. En el caso de AS2, fueron inferiores en casi 10 puntos porcentuales a los presentados en el artículo para el que se recogieron los datos (71.9 % entre sujetos frente a nuestro 62.46 %). El conjunto de datos AS3, en cambio, mejoró del 79.2 % al 86.58 %. Debemos tener en cuenta, aún así, que los resultados se han obtenido a partir de protocolos de validación diferentes. En el artículo original, realizan una partición de la mitad de sujetos para entrenamiento, y la otra mitad para test. En nuestro caso, se utiliza el protocolo descrito al principio de este capítulo, que tiene en cuenta todos los sujetos para test. En el artículo original también se comentan limitaciones del conjunto de datos: los sujetos podían elegir libremente el estilo con el que realizaban las acciones. Por ejemplo, algunos aplaudieron con los brazos estirados, mientras que otros no. Esto, sumado a que el conjunto de muestras no es muy grande, provoca que haya grandes variaciones en las tasas de acierto de cada sujeto de prueba, como se puede comprobar en el Cuadro 6.3.

Para explicar por qué los resultados del conjunto de acciones AS2 son peores que los del resto, hemos observado las matrices de confusión de clasificaciones instantáneas en cada sujeto de prueba, como en el ejemplo del Cuadro 6.4. Hemos observado que las clases 1 (saludo alto), 4 (atrapar con la mano), 7 (dibujar X) y 9 (dibujar círculo) tienen muchas clasificaciones incorrectas. La clase 8 (dibujar tick) tiene una tasa de acierto elevada, pero también muchos falsos positivos. Podemos intuir, basándonos en estos resultados, que nuestra selección de características no es adecuada para gestos precisos con la mano (como dibujar una X o un círculo), mientras que reconoce de forma sólida acciones con movimientos pronunciados como 11 (saludo con las dos manos) o 14 (patada hacia delante).

Cuadro 6.2: Tasas de acierto *offline* para el conjunto de datos UTKA (completo)

Datos: UTKA	Instantánea	Histograma
SVM	62.29 %	77.39 %
RF	80.71 %	**88.95 %**
MLP	73.75 %	87.17 %
NB	59.47 %	70.43 %

Cuadro 6.3: Acciones clasificadas correctamente para cada sujeto con el conjunto de datos MSRA3D-AS4 y el clasificador RF

Datos: MSRA3D-AS4	Acciones clasificadas correctamente	Tasa de acierto
Sujeto 1	32/36	88.89 %
Sujeto 2	22/33	66.67 %
Sujeto 3	27/35	77.14 %
Sujeto 4	27/27	**100 %**
Sujeto 5	21/21	**100 %**
Sujeto 6	36/39	92.31 %
Sujeto 7	32/33	96.97 %
Sujeto 8	35/35	**100 %**
Sujeto 9	32/33	96.97 %
Sujeto 10	27/30	90 %
Total	291/322	90.37 %
Tasa de acierto media	-	90.89 %

Cuadro 6.4: Matriz de confusión del Sujeto 3 en la clasificación instantánea de AS2 realizada por el clasificador MLP

Acciones Clasificada como → Clase verdadera ↓	1	4	7	8	9	11	12	14
1	20	0	0	18	1	39	0	6
4	26	0	10	35	8	0	0	1
7	21	6	12	24	16	2	0	2
8	16	0	5	42	2	0	0	1
9	31	0	10	22	6	15	0	5
11	2	0	0	2	0	60	2	0
12	2	0	4	0	0	3	54	0
14	3	1	0	0	7	5	0	41

Cuadro 6.5: Resultados de clasificación *online*

Método de clasificación	Tasa de acierto
Offline	96.57 %
Online 1 seg.	79.09 %
Online 5 seg.	80 %

6.2. Clasificación *online*

Debido a la dificultad para observar la clasificación de acciones breves en tiempo real, se ha elegido un nuevo subconjunto de acciones del conjunto de datos MSRA3D para la clasificación *online*. Este subconjunto son acciones con una duración indeterminada, o que pueden realizarse ininterrumpidamente durante al menos 5 segundos. Algunas de estas acciones también podrían considerarse *actividades*, como, por ejemplo, correr. En determinados contextos (como observación de pacientes) clasificar estas actividades es más interesante que clasificar acciones de unos pocos segundos.

El conjunto de acciones elegido es el siguiente:

- (1) Saludo alto
- (2) Saludo horizontal
- (10) Aplaudir
- (11) Saludar con las dos manos
- (16) Correr

Se ha entrenado un clasificador Random Forest a partir de los datos de MSRA3D, con los mismos parámetros que en la clasificación *offline*, pero sólo para las acciones seleccionadas. En primer lugar, se ha evaluado este clasificador con el protocolo de validación *offline* (validación cruzada entre sujetos). Después, se han realizado experimentos clasificando directamente el esqueleto de un usuario recogido por Kinect. Se aplica el mismo proceso de selección de características y preprocesado que en la clasificación *offline*, pero no se calcula la media móvil de tamaño 3: las coordenadas esféricas y velocidades angulares son introducidas directamente en el clasificador. Las salidas del Random Forest se procesan en el histograma de clasificaciones instantáneas. Hemos capturado resultados (Cuadro 6.5) para una ventana de tiempo de 1 segundo (15 *frames*) y 5 segundos (75 *frames*). Por cada *frame*, el Random Forest obtuvo una clasificación instantánea para el histograma, y la salida del histograma es un resultado de clasificación.

Los resultados de clasificación *online* en 1 y 5 segundos se ven de forma mucho más clara en una matriz de confusión (Cuadros 6.6 y 6.7). Debido a la duración necesaria, se tomaron

Cuadro 6.6: Matriz de confusión para clasificación *online* con una ventana de tiempo de 1 segundo

Acciones Clasificada como → Clase verdadera ↓	1	2	10	11	16
1	18	2	0	0	0
2	12	13	0	0	0
10	0	0	26	0	0
11	0	0	0	21	0
16	4	2	0	3	11

Cuadro 6.7: Matriz de confusión para clasificación *online* con una ventana de tiempo de 5 segundos

Acciones Clasificada como → Clase verdadera ↓	1	2	10	11	16
1	6	0	0	0	0
2	6	0	0	0	0
10	0	0	6	0	0
11	0	0	0	6	0
16	0	0	0	0	6

sólo 6 muestras por cada acción para la ventana de tiempo de 5 segundos. Se utilizó un solo sujeto de prueba en ambos casos. Se puede observar que todas las acciones fueron reconocidas correctamente, excepto las de la clase 2 (saludo horizontal), que fueron clasificadas como saludo alto. En comparación a la ventana de tiempo de 1 segundo, el resto de clases vieron aumentada su tasa de acierto, mientras que la de la clase 2 se redujo al 0 %.

Como conclusión, podemos decir que los resultados de clasificación *online* son coherentes con los obtenidos con clasificación *offline*. Debemos tener en cuenta que el sujeto, el entorno de captura, el equipo y el software utilizado para generar el esqueleto han sido diferentes en el conjunto de datos original y el entorno de pruebas *online*. A pesar del carácter limitado de las pruebas, se han obtenido tasas de acierto cercanas al 80 %, un resultado prometedor si lo comparamos con otros experimentos de clasificación de actividades.

Capítulo 7

Conclusiones y trabajo futuro

Los resultados vistos en este trabajo son prometedores, y el enfoque propuesto parece muy eficaz para algunos casos concretos de clasificación de actividades. El sistema ha demostrado funcionar bien para clasificación de actividades prolongadas en tiempo real, aunque también es capaz de reconocer acciones breves de poco más de un segundo. Varios clasificadores, donde destacan *random forest* y SVM, han obtenido buenos resultados a partir de la representación de datos propuesta. En base a estos resultados, sugerimos la representación en coordenadas esféricas como una posibilidad para desarrollar nuevos sistemas de clasificación, sobre todo con actividades donde muchas partes del cuerpo estén involucradas y el objetivo no sea diferenciar acciones breves y detalladas. De entre los clasificadores, recomendamos utilizar *random forest* para obtener el menor error de generalización posible, y SVM cuando la velocidad de procesamiento sea limitada y afecte a la latencia de los *random forest* para clasificación en línea. Se ha podido comprobar que los clasificadores SVM son mucho más rápidos que los *random forest*.

Este método de clasificación puede ser utilizado de forma independiente al sensor y al equipo donde se ejecute. La única condición inicial es que exista una representación en esqueleto de las articulaciones del cuerpo humano. Esto podría obtenerse, por ejemplo, a partir de un sensor de visión (como el que se ha utilizado en este trabajo), o mediante sensores ponibles (*wearables*). También podría ser interesante ejecutar los mismos experimentos con un sensor más preciso, como la versión más reciente de Kinect.

Como trabajo futuro, se podrían recoger datos de varios sujetos realizando actividades prolongadas. Se realizarían más experimentos de clasificación *online* con estos datos, con el objetivo de resolver las limitaciones de los experimentos que se han realizado y obtener resultados más concluyentes.

La selección de características también podría mejorarse: la única característica relacionada

con dinámicas que se selecciona es la velocidad angular. Se podría añadir también aceleración, velocidad media o velocidad lineal, para una representación de las dinámicas más completa. Estas nuevas características podrían combinarse en un mapa de movimiento esférico, quedando representado todo el movimiento de la serie temporal en coordenadas esféricas. Este mapa de movimiento proporciona nuevas posibilidades, como "grabar" movimientos y compararlos con los movimientos que se ejecuten posteriormente. Un posible uso de este mapa de movimiento podría ser evaluar movimientos de rehabilitación de forma desatendida.

Al estar desarrollado en Python y ROS, el software desarrollado en este trabajo puede integrarse en un robot. Se espera que este clasificador pueda formar parte del sistema perceptivo de un robot de rehabilitación. El objetivo de este robot sería proporcionar ayuda y motivación a pacientes de rehabilitación para que puedan realizar sus ejercicios en casa, de forma desatendida. Con este clasificador de actividades, se podrían reconocer posturas de riesgo del paciente (y actuar en consecuencia), o reconocer qué actividad está realizando para que el robot pueda ayudar en la rutina. Para poder realizar esto, se necesitan recoger datos específicos del dominio de rehabilitación y realizar nuevos experimentos en este contexto.

Bibliografía

[1] Asier Aguado. *Reconocimiento de posturas mediante Kinect en ROS*. ADDI: Repositorio Institucional de la Universidad del País Vasco, 2015.

[2] Asier Aguado, Igor Rodríguez, Elena Lazkano, and Basilio Sierra. Supervised + unsupervised classification for human pose estimation with rgb-d images: a first step towards a rehabilitation system. In *International Conference on Neurorehabilitation (ICNR)*. Springer, 2016.

[3] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, pages 9–14. IEEE, 2010.

[4] L. Xia, C.C. Chen, and JK Aggarwal. View invariant human action recognition using histograms of 3d joints. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 20–27. IEEE, 2012.

[5] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[6] Liliana Lo Presti and Marco La Cascia. 3d skeleton-based human action classification: A survey. *Pattern Recognition*, 53:130–147, 2016.

[7] Ferda Ofli, Rizwan Chaudhry, Gregorij Kurillo, René Vidal, and Ruzena Bajcsy. Sequence of the most informative joints (smij): A new representation for human skeletal action recognition. *Journal of Visual Communication and Image Representation*, 25(1):24–38, 2014.

[8] Leandro Miranda, Thales Vieira, Dimas Martínez, Thomas Lewiner, Antonio W Vieira, and Mario FM Campos. Online gesture recognition from pose kernel learning and decision forests. *Pattern Recognition Letters*, 39:65–73, 2014.

[9] Georgios Th Papadopoulos, Apostolos Axenopoulos, and Petros Daras. Real-time skeleton-tracking-based human action recognition using kinect data. In *International Conference on Multimedia Modeling*, pages 473–483. Springer, 2014.

[10] Mário Vieira, Diego R Faria, and Urbano Nunes. Real-time application for monitoring human daily activity and risk situations in robot-assisted living. In *Robot 2015: Second Iberian Robotics Conference*, pages 449–461. Springer, 2016.

[11] Chen Chen, Kui Liu, and Nasser Kehtarnavaz. Real-time human action recognition based on depth motion maps. *Journal of real-time image processing*, pages 1–9, 2013.

[12] L. Xia, C.C. Chen, and JK Aggarwal. Utkinect-action3d dataset. `http://cvrc.ece.utexas.edu/KinectDatasets/HOJ3D.html`. Consultado: 02-09-2016.

[13] Spherical coordinate system. `https://en.wikipedia.org/wiki/Spherical_coordinate_system`. Consultado: 31-08-2016.

[14] Eric W. Weisstein. Spherical coordinates. `http://mathworld.wolfram.com/SphericalCoordinates.html`. Consultado: 31-08-2016.

[15] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[16] J Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[17] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[18] J Ross Quinlan. C5. 0 machine learning algorithm, 1993.

[19] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[20] Dong Yu Li Deng. Deep learning: Methods and applications. Technical report, May 2014.

[21] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[22] George H John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann, 1995.