Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado en Ingeniería en Electrónica Industrial y Automática

Caracterización de la postura de personas en entorno doméstico mediante Visión Artificial.

Autor:

Martín García, Rubén

Tutores:

Gómez García-Bermejo, Jaime

Duque Domingo, Jaime

Ingeniería de Sistemas y Automática

Valladolid, Julio 2024

Agradecimientos.

Antes de comenzar con el desarrollo de este trabajo fin de carrera, quería agradecer a todos aquellos que me han apoyado a seguir adelante y me han motivado a conseguir mi objetivo. A mi familia que siempre ha confiado en mí y me ha apoyado cuando era necesario. A los grandes profesores que me he encontrado por este paso por la universidad que me han aportado grandes conocimientos y ganas por seguir formándome y aprendiendo. Y cómo olvidarme de mis amigos y compañeros de clase que me han ayudado cuando lo necesitaba.

La universidad me ha aportado grandes cosas, como una buena formación y unos buenos valores, una destacable rutina de trabajo, sin la cual hubiera sido imposible conseguir esto y sobre todo ayudarme a encaminar mi vida tanto en el ámbito profesional como en el personal.

Resumen.

El presente trabajo fin de grado se centra en la detección y clasificación de la postura de personas en entornos domésticos mediante visión artificial, con especial atención a la identificación de caídas. El sistema desarrollado está orientado a personas que viven solas, permitiendo una posible integración en soluciones de asistencia y monitorización remota.

Para ello, se ha desarrollo un programa en Python, empleado técnicas de visión artificial para analizar imágenes en las que aparece una única persona. Se utilizan modelos como MediaPipe y YOLO para la extracción de características relevantes, lo que permite clasificar la postura en distintas categorías. Como complemento, se ha explorado el uso de máquinas de soporte vectorial (SVM) para mejorar la precisión de la clasificación.

Este proyecto sienta las bases para futuras aplicaciones en detección de caídas y vigilancia domiciliaria, contribuyendo a la seguridad de personas mayores o en situación de dependencia.

Palabras clave.

Visión artificial, clasificación de posturas, detección de caídas, aprendizaje automático, MediaPipe, YOLO, SVM.

Abstract.

This final degree project focuses on the detection and classification of human posture in domestic environments using computer vision, with special attention to fall detection. The developed system is designed for people living alone, allowing for potential integration into assistance and remote monitoring solutions.

For this purpose, a Python program has been developed, employing computer vision techniques to analyze images containing a single person. Models such as MediaPipe and YOLO are used for feature extraction, enabling posture classification into different categories. Additionally, the use of *support* vector machines (SVM) has been explored to improve classification *accuracy*.

This project lays the groundwork for future applications in fall detection and home monitoring, contributing to the safety of elderly individuals or people in dependent situations.

Key words.

Computer vision, posture classification, fall detection, machine learning, MediaPipe, YOLO, SVM.

Índice.

Índice de Figuras.

Índice de Tablas.

1. Introducción y objetivos.

1.1. Motivación y justificación del proyecto.

En el grado en Ingeniería Electrónica Industrial y Automática se abarcan numerosas áreas de estudio, de las cuales el presente proyecto se enmarca en el ámbito de la visión artificial (VA).

El presente trabajo aborda el desarrollo de un sistema que, mediante técnicas de VA, sea capaz de identificar la postura o posición de personas en entornos domésticos. Más concretamente, este proyecto surge de la propuesta del grupo de investigación dentro del departamento de Ingeniería de Sistemas y Automática, el cual, está trabajando en la atención a personas mayores en el hogar. Para ello disponen de un robot móvil capaz de navegar por las distintas estancias de la casa, tomar imágenes e incluso realizar una primera detección de personas. El propósito final sería poder caracterizar la posición e identificar posibles caídas dentro del hogar.

A continuación, se exponen algunos datos y cifras que subrayan la importancia de abordar este tema.

1.1.1. Datos, cifras y magnitud del problema.

Uno de los propósitos de este proyecto es tener la capacidad de identificar caídas, las cuales, según el Instituto Nacional de Seguridad y Salud en el Trabajo (INSS) [1], son el tipo de accidente más común en el hogar y representan casi la mitad de los accidentes domésticos y de ocio. En la figura 1 quedan representados estos datos, donde se pueden observar en el eje de ordenadas los tipos de accidentes y en el eje de abscisas el porcentaje en el que se dan.

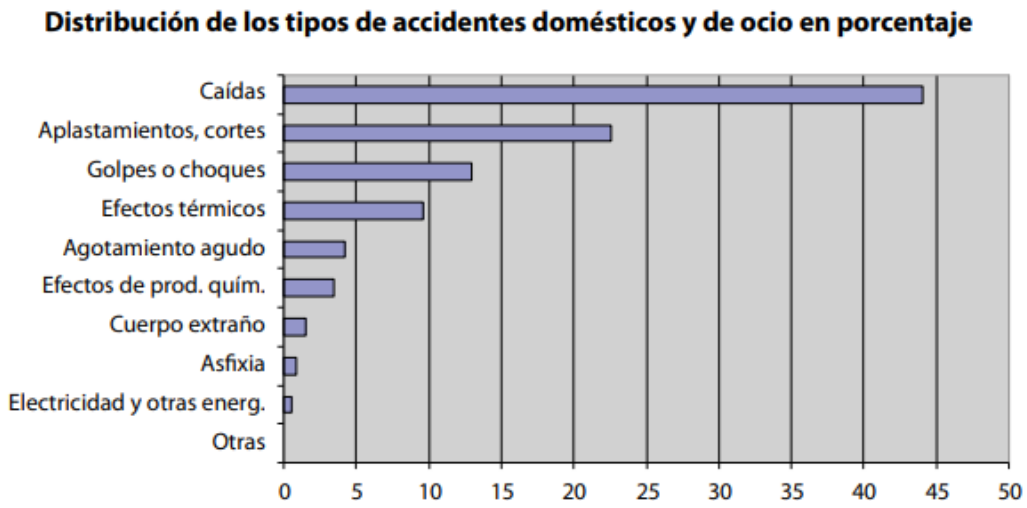


Figura 1: Distribución de los tipos de accidentes domésticos y de ocio en porcentaje. (Fuente: [1])

Según la Organización Mundial de la Salud [2]:

- Cada año se producen 37,3 millones de caídas cuya gravedad requiere atención médica. Un estudio publicado en la revista "Journal of Trauma and Acute Care Surgery" encontró que las personas que fueron atendidas dentro de la primera hora después de una caída tenían un 30% menos de probabilidades de morir que las que no lo fueron.

- Las caídas son la segunda causa mundial de muerte por traumatismos involuntarios. Se calcula que anualmente fallecen en todo el mundo unas 684 000 personas debido a caídas y que más de un 80% de estos accidentes se registran en países de ingresos medianos y bajos.

Otro gran problema de la sociedad actual es el tiempo que se pasa sentados en casa. Según la Pontificia Universidad Católica de Chile [3], estudios internacionales recientes demuestran que los adultos pueden estar sentados entre 7 y 12 horas al día. Este nivel de sedentarismo está relacionado con el trabajo remoto, el ocio digital y la disminución de actividades físicas diarias, como desplazarse a la oficina o realizar tareas domésticas manuales.

Todo esto tiene impactos negativos en la salud, destacando la vulnerabilidad de las personas mayores a sufrir mayor riesgo de enfermedades cardiovasculares, diabetes, pérdida de masa muscular y problemas de salud mental, como ansiedad o depresión.

Para mitigar estos riesgos, se recomienda interrumpir periodos prolongados de estar sentado con pausas activas, como caminar cinco minutos cada hora, usar escritorios ajustables, o realizar ejercicios ligeros en casa.

Tal como se muestra en la figura 2, los hombres pasan la mayor parte del tiempo de su actividad personal sentados, siendo incluso superior el porcentaje al tiempo que pasan de pie. Las mujeres pasan de pie casi la mitad de su jornada, sin embargo, el tiempo que pasan sentadas también es considerablemente elevado, quedando muy cercano al tiempo de pie.

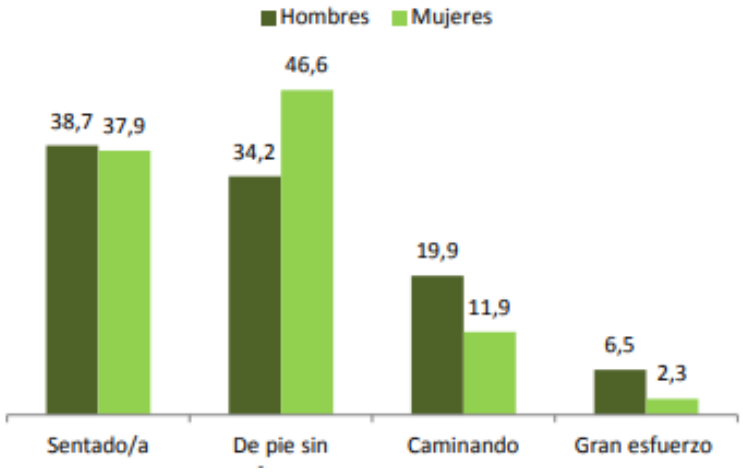


Figura 2: Actividad física en el trabajo o actividad principal en población de 15 y más años ocupada, estudiante o dedicada a labores del hogar, según sexo (%). España 2017. (Fuente: [4])

Sin considerar la población entre 0-4 años, el sedentarismo aumenta con la edad hasta edades medias de la vida, es algo menor en décadas siguientes y remonta nuevamente con la edad. En edades avanzadas a partir de los 74, ambos sexos sufren un fuerte incremento, más acentuado en mujeres. (Figura 3).

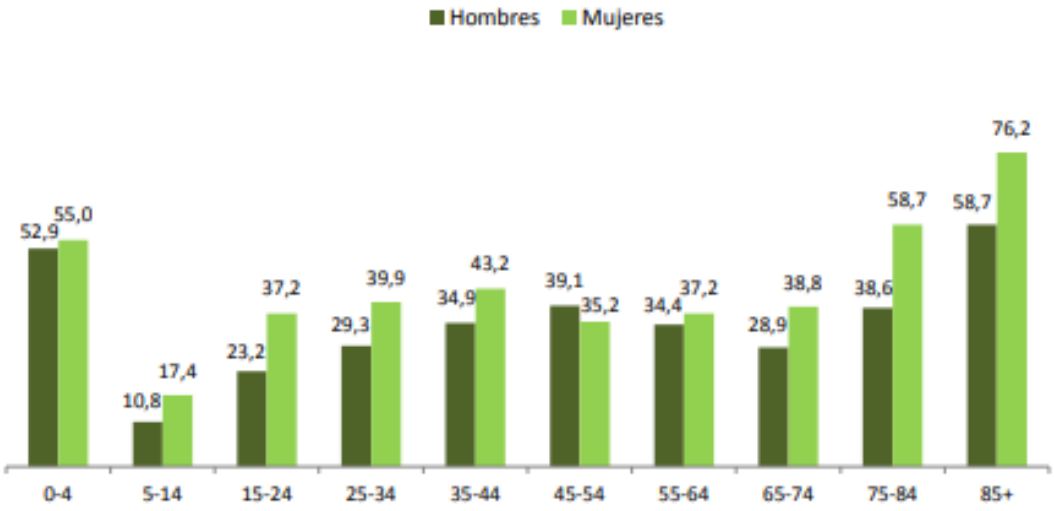


Figura 3: Sedentarismo en tiempo de ocio en población de 0 y más años según sexo y grupo de edad (%). España 2017. (Fuente: [4])

1.1.2. Robot Temi.

Las imágenes desde las que se analizará y clasificará la posición serán tomadas desde un robot asistente de IA personal 100% autónomo. El nombre del modelo es “Temi” y está diseñado para la asistencia e interacción con los humanos.

El robot consta de una pantalla multitáctil capacitiva para una interfaz intuitiva, la cual, se puede inclinar con un rango de movimiento – 15°~+55°. En la parte superior de esta pantalla se encuentra la cámara RGB desde la que está pensado tomar las fotos [5].

El sistema diseñado no solo será válido para el robot sino para cualquier otro tipo de cámara, teniendo en cuenta problemas como que la persona pueda quedar muy pequeña en la imagen.

La figura 4 muestra una escena del robot Temi en el hogar, donde se puede ver también su estructura.

Figura 4: Robot Temi en el hogar. (Fuente: [5])

Temi cuenta con varias funciones más como control remoto, seguimiento, navegación por el hogar y servicios web.

1.2. Objetivos.

El objetivo general del presente trabajo consiste en desarrollar un sistema de visión que, a partir de la imagen de una persona, permita caracterizar su postura corporal. El sistema desarrollado deberá ser capaz, a grandes rasgos, de clasificar en una imagen la pose de una persona (“de pie”, “sentado”, “tumbado” o “caída”) y así poder “monitorizar” cómo se encuentra la persona, detectar estas posibles caídas y poder llegar a controlar la inactividad prolongada.

Se analizarán imágenes que contengan únicamente a una persona, alineándose así con el objetivo principal de diseñar una solución optimizada para aquellos usuarios que viven solos, adecuando el sistema para este escenario particular. Esto permitirá un análisis más preciso y adaptado a las necesidades de este grupo específico.

Se pasará por varias etapas:

- Detección de personas: se hará uso de diferentes técnicas de VA como redes neuronales convolucionales o modelos preentrenados para garantizar una alta precisión y eficiencia.

- Detección puntos clave: utilizando las técnicas anteriores y habiendo filtrado la detección de personas, se pasará a detectar diferentes puntos clave del cuerpo o extraer características que ayuden a poder clasificar la pose presente en la imagen.

- Selección de características relevantes: se seleccionarán y filtrarán que características son más relevantes para utilizar en la clasificación, como ángulos de articulaciones, relaciones alto-ancho del cuerpo, etc. Se analizarán mediante algún método cuáles son las más significativas y distintivas de cada postura y así mejorar la robustez del sistema.

- Métodos de clasificación: por último, se diseñará e implementará un modelo de clasificación mediante uno o varios métodos que, a partir de las características extraídas de la imagen, permitan determinar de manera eficiente la postura (“de pie”, “sentado”, “tumbado”, “caído”, “sin persona”…). Se explotarán enfoques como métodos de clasificación manual a partir de la selección de características relevantes y máquinas de soporte vectorial (SVM).

1.3. Impacto en la mejora de la sociedad.

Los objetivos que se pretenden cumplir en el presente trabajo tienen un impacto significativo en la mejora de la sociedad y están alineados con varios valores de acuerdo con los objetivos presentes en la Agenda 2030.

1.3.1. Salud y bienestar.

La salud y el bienestar son el objetivo con el que se contribuye de manera más directa y sobre el cual se tiene un impacto más representativo.

Este trabajo busca garantizar una mejora en el bienestar para todas las personas, más específicamente mediante la prevención y el monitoreo de situaciones de riesgo. Estos dos aspectos son claves y particularmente sensibles en personas de avanzada edad y que viven solas.

- Detección de caídas, las cuales en una persona mayor o con movilidad reducida pueden tener consecuencias graves. El proyecto pretende abordar directamente este aspecto con una detección temprana ayudando a disminuir los riesgos del accidente incluso llegando a salvar vidas.

- Por otro lado, está la monitorización. Como se ha dicho anteriormente, la inactividad prolongada está asociada con problemas graves de salud. Mediante poder identificar patrones sedentarios, se podrían emitir recomendaciones para fomentar el movimiento, contribuyendo al bienestar del usuario.

- Al detectar posibles accidentes, el sistema puede dar mayor seguridad y tranquilidad tanto al usuario como a familiares. Este aspecto es especialmente relevante en el caso de personas mayores.

1.3.2. Industria, innovación e infraestructura.

Otro objetivo con el que se relaciona estrechamente es con el de industria, innovación e infraestructura, pues integra tecnologías avanzadas y fomenta la innovación en el ámbito de la salud y el bienestar.

- Al emplear herramientas avanzadas de VA para la detección de personas y extracción de características del cuerpo, el sistema que se desarrolla en el presente trabajo aplica tecnologías de última generación a un contexto directamente práctico. Esto refuerza y ayuda a demostrar cómo la innovación tecnológica puede abordar problemas y desafíos reales, impulsando a su vez la investigación y el desarrollo (I+D).

- El proyecto tiene el potencial de ser implementado mediante un coste asumible. Se logra así fomentar una industria inclusiva, haciendo la tecnología y la innovación más accesibles al público general.

- También se promueve la integración tecnológica en los hogares, convirtiéndolos en espacios más seguros y adaptados a las necesidades, dando un paso hacia la creación de infraestructuras domésticas resilientes.

1.3.3. Reducción de desigualdades.

El trabajo contribuye a este objetivo desde varios puntos, eliminando barreras sociales y no discriminando por género ni edad.

- El modelo creado podrá llegar a ser implementado sin un alto costo, estando así disponible para personas con recursos limitados. Esto ayuda a cerrar la brecha tecnológica entre quienes pueden permitirse sistemas de seguridad avanzados y quienes no.

- El sistema asegura que cualquier persona, independientemente de su edad o género, pueda beneficiarse de estas medidas preventivas que se ofrecen.

1.3.4. Ciudades y comunidades sostenibles.

Por último, como se ha citado anteriormente, el proyecto ayuda a la creación de hogares más seguros, inteligentes y adaptados a las necesidades, mejorando la calidad de vida. Además, fomenta la autonomía de las personas, reduciendo la necesidad de atención continua y favoreciendo el seguir formando parte activa de la comunidad sin depender de otros.

1.4. Estructura de la memoria.

El presente trabajo se estructurará de aquí en adelante en diferentes partes, las cuales cada una de ellas abordará los diferentes puntos de estudio y etapas del proyecto.

- **Capítulo 2: Bases teóricas para la caracterización postural.**

En este capítulo se establecen los conceptos clave para comprender las tecnologías del proyecto, comenzando con la VA y sus aplicaciones. Se explorarán los métodos de aprendizaje automático y su evolución hacia el aprendizaje profundo, con énfasis en las redes convolucionales (CNN). Finalmente, se aborda el concepto de pose, la postura corporal y la detección de esta, aspectos fundamentales para la clasificación en el proyecto.

- **Capítulo 3: Panorama actual de la visión artificial en la caracterización de la postura corporal.**

Se analiza el panorama actual de la VA en la caracterización de la postura corporal, los avances recientes, metodologías clave y cómo estas tecnologías mejoran la precisión en la caracterización de posturas, además de discutir los retos y oportunidades en el campo, y cómo el presente proyecto aborda estos desafíos.

- **Capítulo 4: Desarrollo de un sistema para la clasificación de pose.**

El apartado de desarrollo de un sistema para clasificación de pose describe el diseño e implementación del sistema, comenzando con la definición de requisitos y la selección de la mejor estrategia. Se detallan las herramientas utilizadas, como OpenCV, MediaPipe Pose, YOLO y SVM. Luego, se explica el proceso de clasificación de la postura, que incluye la obtención de datos, el uso de MediaPipe y YOLO, el almacenamiento y análisis de características, y el entrenamiento del SVM, finalizando con la selección de características relevantes para optimizar el modelo.

- **Capítulo 5: Resultados obtenidos.**

En este apartado se muestran a los resultados del sistema de clasificación de postura desarrollado, evaluando el rendimiento de MediaPipe y YOLO de manera independiente y comparando ambos métodos. Se explica la implementación de un sistema de decisión que combina las fortalezas de cada método, y se analizan los resultados del SVM utilizando datos combinados y separados por ambos

métodos. Además, se describe la integración mediante la fusión de resultados y el uso del SVM para la clasificación final.

La validación del sistema incluye la evaluación en imágenes de prueba y la comparación entre métodos individuales y la combinación final. Finalmente, se presentan los resultados en el conjunto de test, destacando la efectividad y el buen funcionamiento del sistema.

- **Capítulo 6: Conclusiones y líneas futuras.**

El apartado de conclusiones y líneas futuras resume los hallazgos clave del proyecto y las conclusiones obtenidas. También plantea posibles líneas de trabajo y mejoras futuras, indicando áreas de expansión y optimización del sistema, así como nuevas investigaciones o aplicaciones derivadas de los avances alcanzados.

- **Capítulo 7: Bibliografía.**

La Bibliografía recoge todas las fuentes consultadas y citadas en el trabajo, incluyendo artículos, libros, páginas webs y otros recursos que respaldan la investigación, garantizando la transparencia y validez del contenido.

- **Capítulo 8: Anexos.**

El apartado de Anexos proporciona información adicional que respalda el desarrollo del proyecto, incluyendo un resumen de los códigos desarrollados y la lista de librerías utilizadas.

2. Marco teórico.

2.1. Visión artificial.

La Visión Artificial en sus distintas variantes (visión por computador, visión máquina, procesamiento de imágenes y VA propiamente dicha) es una disciplina en auge por sus innumerables aplicaciones en robótica, automatización, inspección, control de calidad, seguridad, aplicaciones espaciales, medicina, entretenimiento, etc. [6]. Persigue extraer información del mundo físico a partir de imágenes, utilizando un sistema de computación. Desde un punto de vista más ingenieril, un sistema de visión es un sistema autónomo que realiza algunas de las tareas del sistema visual humano. Las operaciones que desarrollan los sistemas de visión pueden ir desde la simple detección de objetos sencillos hasta la interpretación tridimensional de escenas complejas.

El siguiente esquema, (figura 5), muestra cuáles son las etapas comunes que sigue un sistema de visión.

Figura 5: Etapas de un sistema de visión. (Fuente: Elaboración propia)

2.1.1. Progresos de la visión artificial.

En la actualidad, la VA está presente en nuestras vidas en la mayoría de los ámbitos, desde la seguridad, la medicina, la inspección o la navegación automática.

Dentro de la VA, uno de los avances más significativos es la incorporación del aprendizaje profundo. Este enfoque permite a las máquinas u ordenadores aprender de grandes conjuntos de datos y mejorar su capacidad para reconocer patrones y objetos en imágenes. En lugar de decirle al ordenador exactamente qué hacer, esta técnica le permite aprender y mejorar por sí mismo a medida que procesa más información. Esto ha revolucionado aplicaciones como la detección

de objetos, la clasificación de imágenes y la identificación de rostros, mejorando significativamente su precisión.

2.2. Aprendizaje automático.

El *machine learning* (aprendizaje automático) es una rama de la inteligencia artificial que se centra en la creación de sistemas capaces de aprender automáticamente a partir de datos, sin necesidad de ser programados explícitamente. Este aprendizaje se basa en algoritmos que identifican patrones en los datos y los utilizan para realizar predicciones o tomar decisiones.

El aprendizaje automático clásico, o "no profundo", depende más de la intervención humana para aprender. Los expertos humanos determinan el conjunto de características para comprender las diferencias entre las entradas de datos, lo que suele requerir datos más estructurados para aprender.

2.2.1. Métodos de *machine learning*.

Los modelos de *machine learning* se dividen en tres categorías principales. [7]

- ***Machine learning* supervisado:** El aprendizaje supervisado se define por su uso de conjuntos de datos etiquetados para entrenar algoritmos que clasifiquen datos o predigan resultados con precisión. A medida que se introducen datos de entrada en el modelo, este ajusta sus ponderaciones hasta que se han ajustado adecuadamente. Algunos métodos utilizados en el aprendizaje supervisado son las redes neuronales, el clasificador bayesiano ingenuo, la regresión lineal y logística, el bosque aleatorio y la máquina de vectores de soporte.

- ***Machine learning* no supervisado**: El aprendizaje no supervisado, utiliza algoritmos de *machine learning* para analizar y agrupar conjuntos de datos no etiquetados. Estos algoritmos descubren patrones ocultos o agrupaciones de datos sin necesidad de intervención humana. La capacidad de este método para descubrir similitudes y diferencias en la información lo hace ideal para el análisis exploratorio de datos, las estrategias de venta cruzada, la segmentación de clientes y el reconocimiento de imágenes y patrones. El análisis de componentes principales (PCA) y la descomposición en valores singulares (DVE) son dos métodos habituales para ello. Otros algoritmos utilizados en el

aprendizaje no supervisado son las redes neuronales, el *k-means* y los métodos de agrupación probabilística.

- **Aprendizaje semisupervisado:** El aprendizaje semisupervisado ofrece un término medio entre el aprendizaje supervisado y el no supervisado. Durante el entrenamiento, utiliza un conjunto de datos etiquetados más pequeño para guiar la clasificación y la extracción de características a partir de un conjunto de datos más grande sin etiquetar. El aprendizaje semisupervisado puede resolver el problema de no disponer de suficientes datos etiquetados para un algoritmo de aprendizaje supervisado. El *self-training* y el *co-training* son métodos de aprendizaje semisupervisado.

2.2.2. Aprendizaje profundo. *Deep learning*.

El *deep learning* es un subconjunto del *machine learning* que utiliza redes neuronales multicapa, llamadas redes neuronales profundas, para simular el complejo poder de toma de decisiones del cerebro humano. Algunas formas de *deep learning* impulsan la mayoría de las aplicaciones de inteligencia artificial (IA) en nuestra vida actual.

La principal diferencia entre el *deep learning* y el *machine learning* es la estructura de la arquitectura de red neuronal subyacente. Los modelos tradicionales de *machine learning* "no profundos" utilizan redes neuronales simples con una o dos capas computacionales. Los modelos de *deep learning* utilizan tres o más capas, pero normalmente cientos o miles de capas, para entrenar a los modelos. [8]. La arquitectura que siguen estas redes neuronales se muestra en el siguiente gráfico (figura 6).

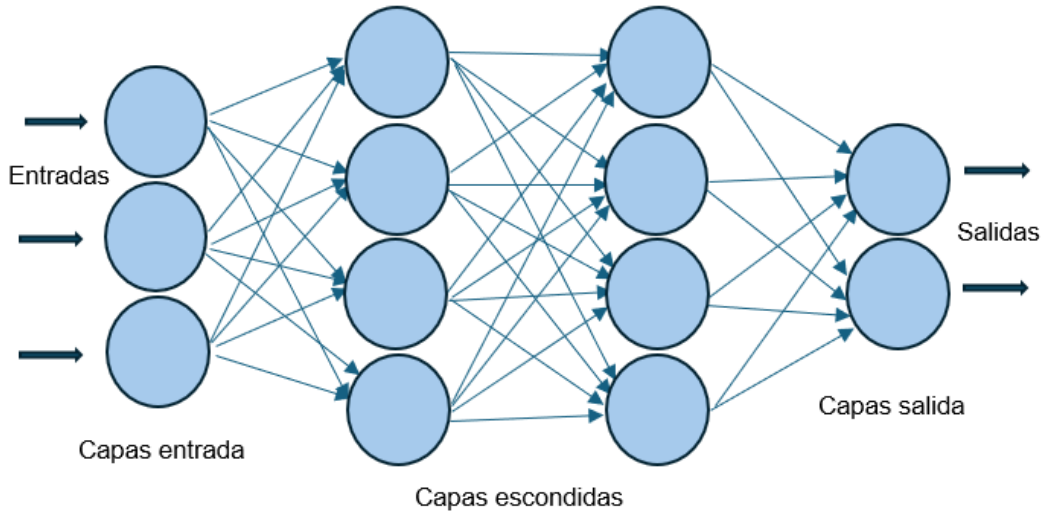


Figura 6: Arquitectura de red neuronal típica. (Fuente: Elaboración propia)

2.2.3. Redes convolucionales (CNN).

Una red neuronal convolucional (CNN) es un tipo de red neuronal diseñada específicamente para procesar datos con estructura espacial, como imágenes. Las CNN son uno de los pilares del *deep learning* en aplicaciones de VA debido a su capacidad para identificar patrones espaciales como bordes, texturas, y objetos en imágenes, y se construyen a partir de una combinación de capas especializadas que procesan las imágenes de forma jerárquica. [9].

- **Capa convolucional**: Es la base de las CNN. Esta capa aplica filtros (*kernel*) a la imagen de entrada para extraer características locales, como bordes o texturas. Cada filtro es una matriz pequeña que se desliza por la imagen (operación de convolución), generando un mapa de características.
- **Capa de activación:** Se aplican después de la convolución para introducir no linealidad al modelo.
- **Capa de pooling:** Reduce las dimensiones de los mapas de características, conservando la información más relevante. Esto disminuye el número de parámetros y evita el sobre ajuste.
- **Capa de normalización**: Ayuda a mejorar la velocidad de convergencia durante el entrenamiento y a evitar problemas como el sobreajuste. Mantiene las activaciones dentro de un rango controlado, mejorando la estabilidad del entrenamiento.
- **Capas completamente conectadas**: Estas capas están al final de la red y conectan todas las neuronas. Su función es combinar las características aprendidas para realizar la clasificación o regresión.
- **Capa de salida:** Genera las predicciones finales, por ejemplo, la probabilidad de que una imagen pertenezca a una clase específica (como "de pie", "sentado", ”tumbado” o "caído").

Estas capas se estructuran típicamente como se muestra en la figura 7.

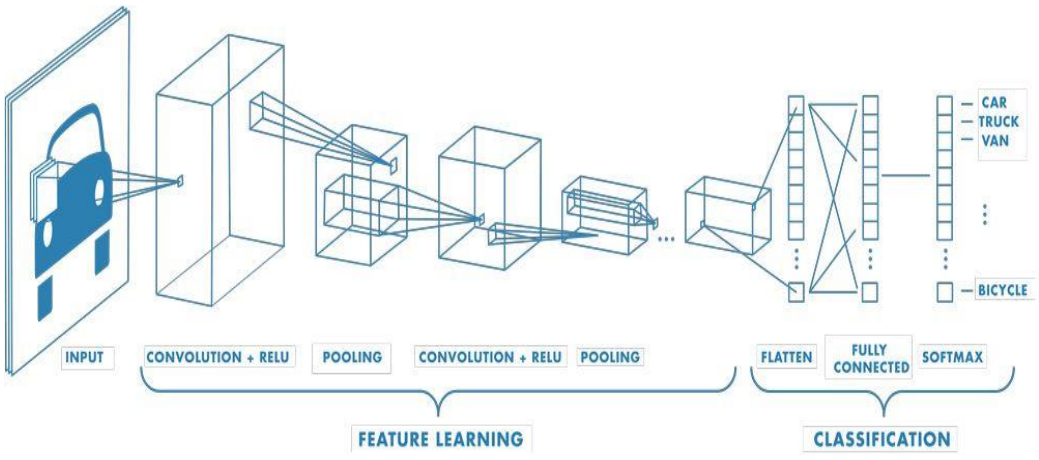


Figura 7: Red con múltiples capas convolucionales. (Fuente: [9])

2.3. Postura corporal.

Concretamente, en este trabajo se hará uso de la VA para detectar y clasificar la postura corporal de una persona.

La postura corporal es el resultado de la posición de todas las articulaciones del cuerpo y cómo estas consiguen situar las extremidades con respecto al tronco en un determinado momento, lo que hace entender cómo se encuentra situada la persona en el espacio que lo rodea [10].

La identificación de la pose humana es una habilidad natural que se desarrolla desde temprana edad. El cerebro humano está capacitado para reconocer y entender las posturas corporales de las personas que nos rodean y ser capaz de identificar si se encuentran “de pie”, “sentado”, “tumbado” o “caída” en una mínima fracción de segundo. Es una combinación de observación visual, experiencia previa, contexto y comprensión de la expresión corporal y gestual.

El problema surge cuando se quiere que esto no lo haga una persona sino un ordenador mediante técnicas de VA. Es aquí donde hay que trasladarse al origen y a las técnicas básicas que se utilizan para determinar cómo se encuentra una persona situada en el espacio.

Algunas de las técnicas básicas que utilizan las personas son:

- **Observación visual**: Nuestros ojos observan la posición de las diferentes partes del cuerpo de una persona.
- **Patrones de movimiento:** Podemos identificar la pose de alguien observando cómo se mueve y posiciona sus partes del cuerpo.
- **Contexto y experiencia previa**: Nuestra experiencia y conocimiento del mundo ayudan a comprender la pose de una persona en un contexto específico.
- **Comparación con nuestro propio cuerpo:** A menudo, nos relacionamos con la pose de una persona comparándola con nuestra propia experiencia corporal. Esto ayuda a comprender mejor la posición y el movimiento de los demás.

2.3.1. Detección de la postura.

La detección de la postura humana es una tarea bastante relevante en el campo de la visión por computador, que consiste en identificar la postura de una figura humana a partir de una imagen. Dicha pose se define a partir de una serie de

puntos clave, que habitualmente serán articulaciones, de forma que el objetivo será encontrar la posición (x, y) de cada uno de esos puntos.

Es un campo emergente en la visión por ordenador en el que se utilizan algoritmos de aprendizaje automático para detectar y rastrear la orientación y disposición de diferentes partes del cuerpo en una imagen. La clasificación de postura puede ser utilizada para una variedad de aplicaciones, desde el análisis de movimiento en deportes y rehabilitación física hasta la interacción hombre-máquina y la realidad aumentada.

Una de las aplicaciones más importantes y prometedoras y en la que se centra en parte este trabajo, es la detección de caídas. Las caídas son un problema grave, especialmente para las personas mayores. La detección temprana y precisa de las caídas puede permitir una respuesta rápida y disminuir las consecuencias del accidente. La VA, combinada con la clasificación de postura, puede desempeñar un papel crucial para resolver este problema.

El uso de la clasificación de postura para la detección de caídas tiene varias ventajas:

- Es poco intrusivo, ya que no requiere que las personas lleven dispositivos de seguimiento.
- Puede proporcionar una detección de caídas más precisa al analizar la postura del cuerpo y los movimientos en detalle.
- También puede ayudar a entender el contexto de la caída, lo que puede ser útil para la prevención de caídas en el futuro.

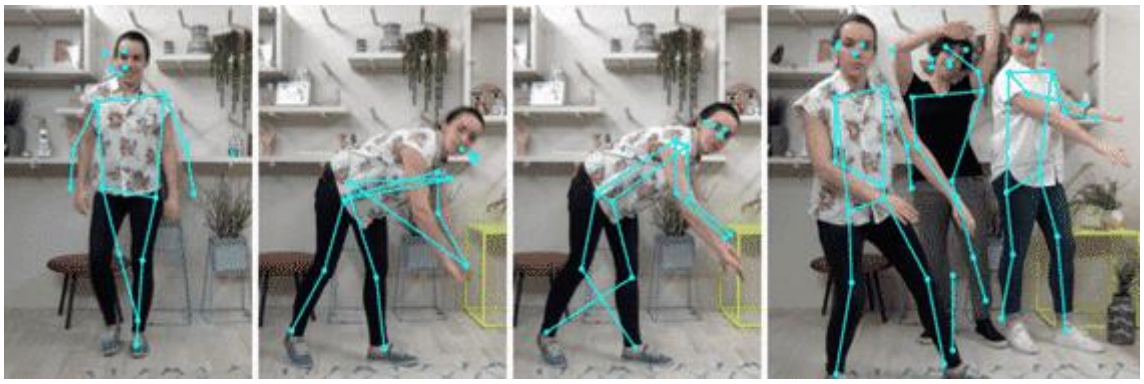


Figura 8: Ejemplo de detección de pose con VA. (Fuente: [11])

La imagen anterior (figura 8) muestra un ejemplo de cómo se visualiza la detección de la pose y los puntos clave del cuerpo mediante VA.

3. Panorama actual de la visión artificial en la caracterización de la postura corporal.

El análisis corporal de personas mediante VA ha sido un área de creciente interés en los últimos años con aplicaciones en muchos campos, por ello, en este apartado se analizarán los enfoques más relevantes de la literatura sobre áreas de estudio similares a las de este proyecto, abarcando desde la extracción de características hasta la clasificación de posturas.

En la actualidad existen numerosas técnicas que permiten reconocer la postura y han surgido muchos proyectos y aplicaciones que utilizan estas herramientas para poder realizar detecciones precisas sobre imágenes en diversas situaciones. La caracterización de la postura se utiliza en ámbitos, desde animación cinematográfica o videojuegos hasta aplicaciones en medicina.

Existen varios factores que complican la correcta caracterización de la postura, como la visibilidad, la iluminación, la obstaculización, la variedad de tamaños del cuerpo o la mala calidad de imagen. Por ello, se han desarrollado técnicas tanto a nivel de software, con algoritmos de aprendizaje profundo, como a nivel de hardware, a través de cámaras y sensores.

3.1. Algoritmos y sistemas para la detección de la postura.

- **MediaPipe Pose.**

En este proyecto se utilizará MediaPipe Pose como modelo para la estimación de la postura humana.

Actualmente, existen otros algoritmos y modelos de detección de posturas como OpenPose o AlphaPose, pero se decide utilizar MediaPipe Pose por su eficiencia y bajos recursos computacionales comparado con otros modelos que, aunque más precisos, requieren más potencia, lo que puede ser un desafío en la detección de caídas la cual, exige procesamiento rápido. Aparte, se cuenta con experiencia previa trabajando con esta herramienta, lo que facilita su uso a lo largo del proyecto.

Esta herramienta forma parte del entorno de trabajo de MediaPipe. Se profundizará más en detalle en el apartado correspondiente 4.4.3.

- **OpenPose.**

Es un sistema de estimación de postura desarrollado por investigadores de la Universidad Carnegie Mellon (CMU) que puede detectar y rastrear el cuerpo humano en tiempo real y determinar con precisión su postura en un espacio 3D. Es conocido por ser el primer sistema de estimación de postura de varias personas en tiempo real que detecta con precisión los puntos clave del cuerpo humano, las manos, el rostro y los pies en imágenes individuales. [12].

La figura 9 muestra un ejemplo de estimación de la pose en 3D utilizando OpenPose y una cámara estéreo ZED que permite capturar imágenes en 3D y estimar la profundidad en tiempo real, siendo ideal para aplicaciones de visión artificial, robótica y realidad aumentada.

Figura 9: Ejemplo de estimación de pose utilizando OpenPose. (Fuente: [13])

OpenPose utiliza una red neuronal convolucional (CNN) para analizar imágenes y extraer "mapas de características", que destacan aspectos como bordes y texturas. A continuación, procesa estos mapas mediante una CNN especializada en varias etapas para generar dos resultados clave: mapas de confianza de las partes (que indican la probabilidad de la ubicación de las partes del cuerpo) y campos de afinidad de las partes (que muestran la orientación y las conexiones entre las partes del cuerpo). Finalmente, un algoritmo toma la opción óptima más inmediata en cada paso con la esperanza de encontrar un óptimo global para la estimación de la pose.

- **AlphaPose.**

AlphaPose es una avanzada herramienta para la estimación de posturas humanas creada por la Academia China de Ciencias. Emplea un algoritmo de aprendizaje profundo para procesar imágenes o videos y determinar la postura

de una o más personas en tiempo real. Esta herramienta está optimizada para operar en diferentes condiciones, incluyendo escenarios con poca iluminación o posturas parcialmente ocultas. [14].

Emplea una red neuronal convolucional (CNN) para calcular la postura de las personas en imágenes o videos. Procesa cada fotograma de la imagen o video, identificando partes del cuerpo humano como la cabeza, torso y extremidades. Luego, determina la posición y orientación de cada parte, generando una estimación global de la postura del cuerpo.

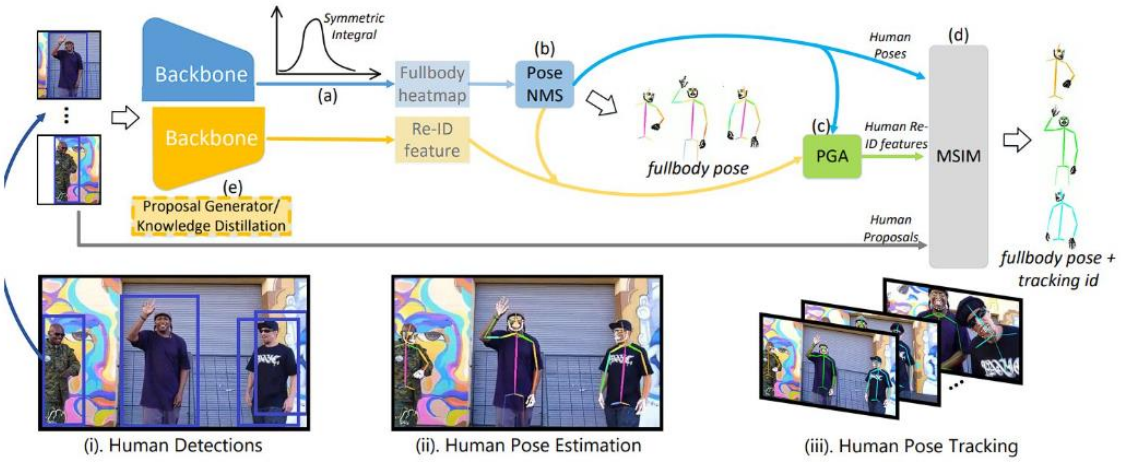


Figura 10: Ejemplo de funcionamiento de AlphaPose. (Fuente: [15])

La figura 10 representa el funcionamiento de AlphaPose. Utilizando un enfoque ascendente, primero detecta las partes individuales del cuerpo antes de estimar la pose general, lo que le permite trabajar con múltiples personas en una misma imagen o video. Además, adopta una técnica basada en mapas de calor, evaluando la probabilidad de que cada píxel corresponda a una parte específica del cuerpo.

3.2. Dispositivos para la detección de la postura.

En el ámbito de la detección de la postura humana, no solo el software juega un papel fundamental, sino que también existen diversas tecnologías de hardware que permiten realizar esta tarea de manera eficiente. Un ejemplo destacado de ello es el sensor Kinect, [16].

Desarrollado por Microsoft, ha sido utilizado no solo en videojuegos, sino también en aplicaciones de visión por computadora y análisis de movimiento. Este dispositivo integra cámaras, sensores de profundidad y micrófonos para ofrecer una captura detallada del cuerpo humano, permitiendo el seguimiento de la

postura y los movimientos en tiempo real. A través de su capacidad para reconocer gestos, detectar la posición de las articulaciones y proporcionar datos tridimensionales, Kinect se convierte en una herramienta esencial para estudios de biomecánica, rehabilitación, robótica y sistemas de interacción sin contacto, demostrando que el hardware y el software deben complementarse para lograr una detección precisa y funcional de la postura humana.

Figura 11: Cámara Kinect en sus versiones 1 y 2. (Fuente: [16])

La figura 11 muestra cómo es la cámara en su versión 1 y 2.

Kinect es capaz de capturar una cantidad increíble de datos. Siempre fijando su objetivo en las cosas que se mueven en su entorno. Gracias al procesamiento de estos datos a través de un algoritmo de inteligencia artificial y a métodos de aprendizaje de máquinas, Kinect puede llegar a mapear los datos visuales que obtiene a través de sus sensores. El objetivo es ser capaz de detectar a los seres humanos y entender en qué posición se encuentra cada persona detectada.

La cámara consta de varias partes:

- Cámara de vídeo de color RGB: Funciona a modo de webcam, capturando las imágenes en vídeo. El sensor Kinect utiliza esta información para obtener detalles sobre objetos y personas en la habitación.
- Emisor IR: El emisor de infrarrojos es capaz de proyectar una luz infrarroja en una habitación. Según la luz infrarroja incide sobre una superficie, el patrón se distorsiona. Esta distorsión es leída gracias a su otro componente, una cámara de profundidad.
- Cámara de profundidad: Analiza los patrones infrarrojos emitidos por el emisor y es capaz de construir un mapa 3D de la habitación y de todos los objetos y personas que se encuentran dentro de ella (figura 12).
- Conjunto de micrófonos: El sensor Kinect tiene incorporado cuatro micrófonos de precisión capaces de determinar de dónde vienen los sonidos y las voces. También es capaz de filtrar el ruido de fondo.
- Motor de inclinación: Este motor tiene la capacidad de ajustar sobre la base, el sensor Kinect. Es capaz de detectar el tamaño de la persona que está delante, para ajustarse arriba y abajo según convenga.

Figura 12: Imagen de profundidad captada por Kinect. (Fuente: [17])

Microsoft liberó el código de Kinect para facilitar su uso en aplicaciones más allá de los videojuegos, lo que permitió a los desarrolladores crear aplicaciones que aprovecharan las capacidades de Kinect, como el reconocimiento de gestos y la captura de movimiento en una variedad de aplicaciones, incluyendo aquellas para ordenadores, robótica, medicina, investigación y más.

3.3. Trabajos relacionados.

En este apartado, se analizarán dos trabajos que están estrechamente relacionados con el enfoque de este proyecto y que exploran aspectos similares en la detección y análisis de la postura humana. A través de la revisión de estos estudios, se podrá observar cómo se abordan desafíos similares a los del presente proyecto y se analizará la relevancia que poseen en él.

3.3.1. Reconocimiento de posturas utilizando una cámara RGB-D.

El trabajo de Elforaici et al. (2018), [18], se centra en la clasificación de posturas humanas utilizando cámaras RGB-D, una tecnología que combina imágenes en color (RGB) y en profundidad (D) para capturar una representación 3D de la escena. El estudio propone un enfoque innovador que integra modelado corporal 3D y aprendizaje profundo para mejorar la precisión en la clasificación de posturas, incluso en escenarios complejos.

Los siguientes esquemas (figura 13 y figura 14), muestran los pasos seguidos para reconocer la postura a partir de imágenes a color y de profundidad respectivamente.

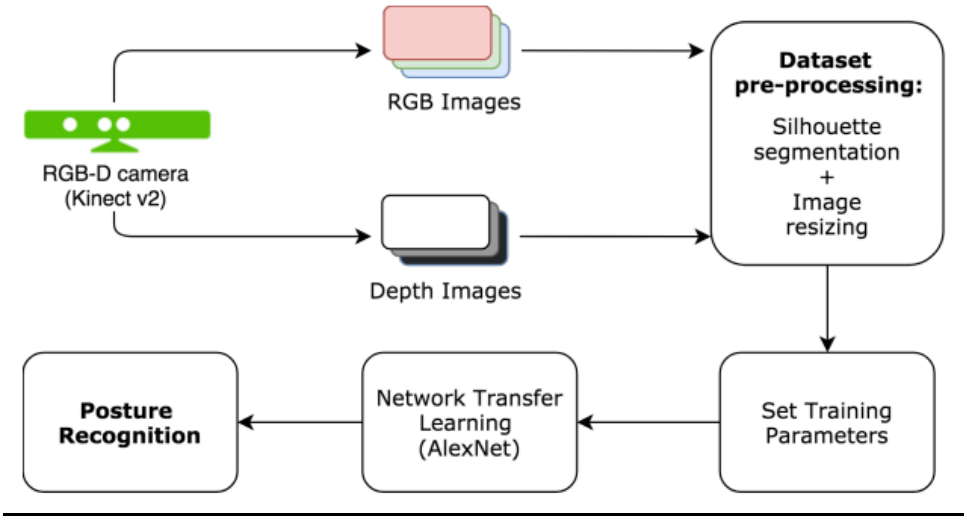


Figura 13: Reconocimiento de la postura utilizando imágenes 2D y CNN. (Fuente [18])

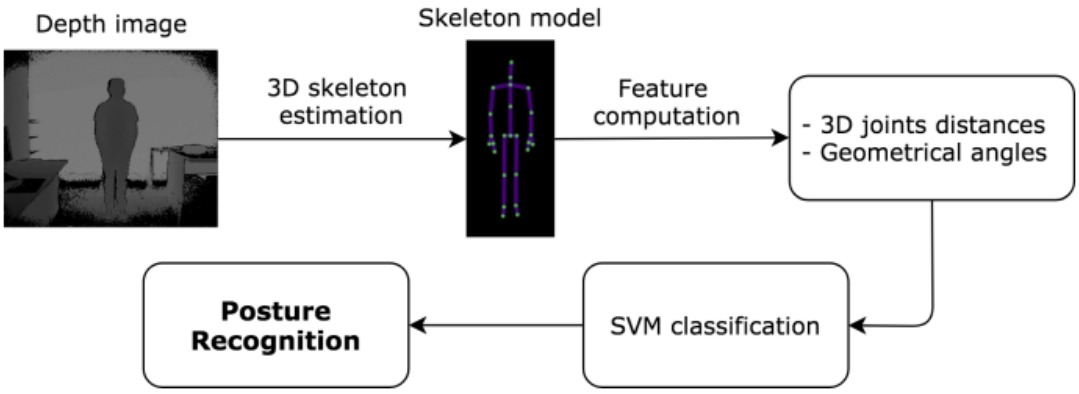


Figura 14: Reconocimiento de la postura utilizando características 3D. (Fuente [18])

Metodología del estudio:

El sistema combina técnicas avanzadas para lograr una clasificación precisa y robusta de posturas en 3D:

- Cámaras RGB-D: Estas cámaras capturan tanto la información visual como la profundidad de la escena, lo que permite una mejor comprensión del contexto espacial y el análisis de las posturas humanas desde diferentes ángulos.
- Redes Neuronales Convolucionales (CNN): Las CNN se utilizan para extraer características visuales relevantes de las imágenes RGB.

- Máquinas de Soporte Vectorial (SVM): Se emplean para clasificar las posturas extraídas por las redes neuronales, utilizando las características visuales y espaciales obtenidas a partir de las imágenes RGB-D.

Resultados y aplicaciones:

- Alta precisión en la clasificación de posturas, alcanzando niveles de exactitud superiores al 90 % en algunos casos.
- Robustez frente a variabilidad, demostrando ser resistente a diferentes condiciones de luz y a cambios en la postura o escala de la persona.
- Aplicaciones potenciales en áreas como la rehabilitación médica, la monitorización de pacientes, y sistemas de interacción hombre-computadora, donde la clasificación de posturas es crucial para la comprensión de las intenciones y el estado físico del usuario.

Relevancia para este trabajo:

Este estudio es particularmente relevante para este trabajo. Se enfoca en la clasificación de posturas humanas y utiliza tecnologías similares a las que se emplean en este proyecto, como las redes neuronales y la captura de características visuales de las personas.

La idea de combinar información de cámaras RGB-D con modelos de aprendizaje profundo puede ser un punto de partida valioso para mejorar la precisión de la detección de posturas y caídas, especialmente al tratar de reconocer y clasificar posturas en un entorno doméstico con diversas condiciones de iluminación y disposición del espacio.

Puntos de mejora y limitaciones:

- El trabajo de Elforaici et al., se enfoca en un entorno controlado utilizando cámaras RGB-D, lo que garantiza condiciones de iluminación y espacios homogéneos. El presente proyecto está orientado a entornos domésticos con condiciones de iluminación y fondos variables y complejos y debe adaptarse de manera más autónoma a estas situaciones.
- El estudio aquí presente no cubre la detección de caídas, sino que se centra únicamente en la clasificación de posturas más estáticas.
- Otra limitación es el uso de redes neuronales convoluciones entrenadas para este propósito, lo que lo hace demasiado complejo para los requisitos que se buscan en este proyecto de simplicidad y robustez.

3.3.2. Extracción de características corporales con MediaPipe y YOLOv5 para la evaluación del rango de movimiento.

Uno de los estudios más relevantes en la extracción de características corporales utilizando visión artificial es el trabajo de Zhu et al. (2024), [19], que presenta un sistema basado en la combinación de MediaPipe y YOLOv5 para evaluar el rango de movimiento en pacientes con afecciones musculo-esqueléticas, como problemas en la columna vertebral o el síndrome de hombro congelado.

Metodología del estudio:

El sistema propuesto combina dos enfoques principales para mejorar la precisión en la detección y análisis del movimiento corporal (figura 15):

- *MediaPipe*: Se utiliza para la estimación de poses, extrayendo los puntos clave del cuerpo humano (*landmarks*) y permitiendo la medición de distancias, ángulos articulares y posiciones relativas de las extremidades.
- *YOLOv5*: Actúa como un sistema de detección de objetos, asegurando la correcta identificación de la persona en la escena y ayudando a reducir el ruido en la segmentación del cuerpo.
- *CBAM (Módulo de Atención de Bloque Convolucional)*: Se incorpora en YOLOv5 para mejorar la extracción de información relevante y reducir la interferencia de elementos de fondo, optimizando la detección de las articulaciones y la precisión del análisis biomecánico.

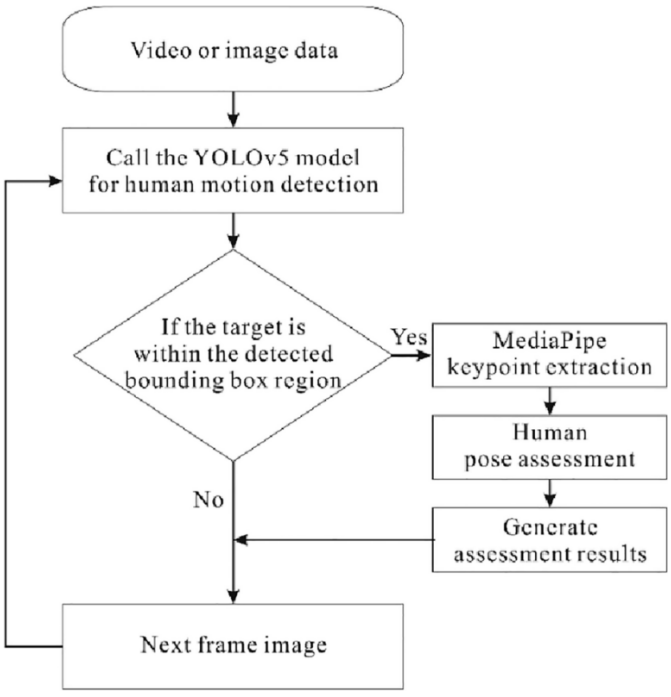


Figura 15: Funcionamiento principal del sistema (Fuente: [19])

Resultados y aplicaciones:

El sistema desarrollado permite realizar una evaluación precisa del rango de movimiento de los pacientes en un entorno clínico sin necesidad de equipos especializados como sensores inerciales o cámaras infrarrojas. Gracias a la combinación de MediaPipe y YOLOv5, se consigue:

- Una extracción robusta de ángulos de articulaciones y distancias entre puntos clave.
- Un modelo capaz de generalizar bien a distintas condiciones de iluminación y variabilidad en las posturas.
- Un método menos invasivo y más accesible que los sistemas tradicionales de análisis del movimiento.

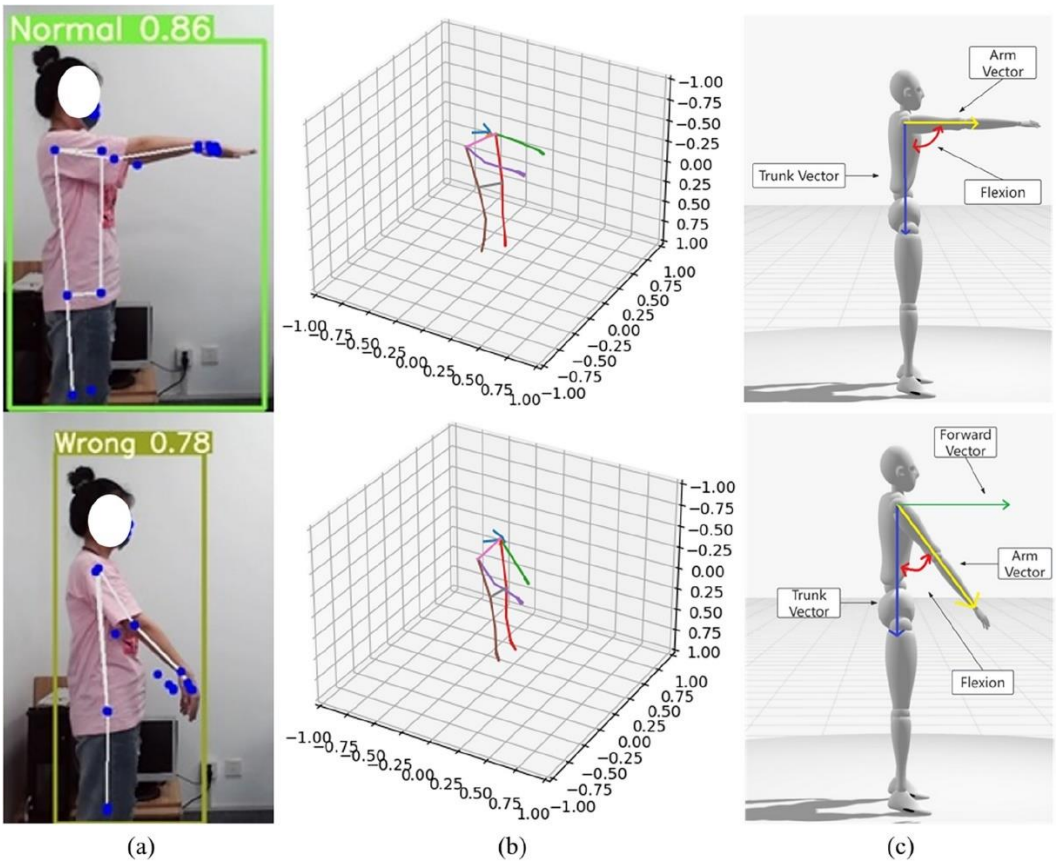


Figura 16: Evaluación de hombro congelado. (Fuente: [19])

La figura 16 muestra: (a) Imágenes de resultado del hombro congelado generadas al combinar YOLOv5 y MediaPipe; (b) Estructura esquelética humana generada a partir de la información de puntos clave extraída por MediaPipe; (c) Cálculo de los ángulos del rango de movimiento del hombro congelado proyectando los ángulos en planos anatómicos definidos.

Relevancia para este trabajo:

Este estudio es particularmente relevante para el desarrollo de sistemas de caracterización de posturas humanas mediante VA, ya que demuestra la viabilidad de combinar MediaPipe y YOLO para extraer características del cuerpo con alta precisión. En el contexto de este trabajo, donde se busca detectar posturas y caídas en entornos domésticos, una metodología similar podría aplicarse para mejorar la detección de posiciones.

Puntos de mejora y limitaciones:

- Mientras que este estudio se centra en evaluar el riesgo de movimientos en pacientes, no cubre las necesidades de con el análisis de esos valores, clasificar las posturas y detectar caídas.
- El estudio se centra en un contexto médico y clínico, pero no está pensado para entornos domésticos con el propósito de asistir a personas que viven solas. Esto implica desafíos adicionales, como condiciones de iluminación variadas, fondos más complejos y la necesidad de un sistema que funcione de manera autónoma sin supervisión médica.

4. Desarrollo de un sistema para la clasificación de pose.

4.1. Alternativas analizadas.

En este apartado se exponen las alternativas evaluadas y las distintas etapas por las que se ha pasado durante el desarrollo del proyecto para alcanzar los objetivos planteados, así como los criterios de selección que llevaron a la configuración final del sistema.

Para el desarrollo se ha seguido una metodología de tipo incremental (figura 17), se ha partido de unas características básicas y según avanzaba el proyecto y se iban obteniendo distintos resultados satisfactorios, se realizaban cada vez versiones un poco más complejas y sofisticadas del código hasta conseguir los requisitos que se exigen. Esto es típico en el desarrollo de un programa, sobre todo aplicado a la VA, pues, se van probando soluciones nuevas y obteniendo distintos resultados que te hacen pensar a su vez en nuevas soluciones para los problemas que van surgiendo o para mejorar la calidad de los resultados obtenidos.

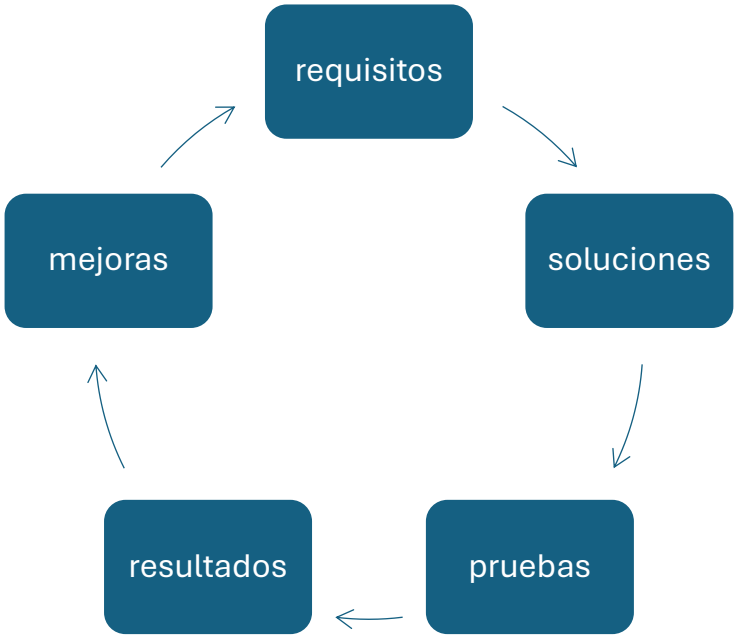


Figura 17: Representación de la metodología aplicada. (Fuente: Elaboración propia).

4.1.1. Definición de requisitos.

El objetivo inicial consiste en clasificar la postura de una persona en cuatro categorías principales: “de pie”, “sentado”, “tumbado” y “caída”. Se buscaba un enfoque simple y eficaz, que además pudiera implementarse en entornos domésticos con recursos computacionales limitados.

Las situaciones elegidas para ser analizadas serán aquellas en las que en la imagen se encuentre una sola persona y principalmente en entornos domésticos. Esto surge por acercarse durante el desarrollo y las pruebas del programa a las situaciones reales en las que será implementado y conseguir con ello un mejor ajuste.

El requisito principal y obligatorio es obtener como respuesta del sistema la postura detectada de la persona con una buena robustez y fiabilidad.

4.1.2. Exploración inicial de soluciones.

Este tipo de proyectos se puede implementar para trabajar con imágenes aisladas o con una secuencia definida de imágenes, lo que más comúnmente se conoce como video. El análisis de video permitiría monitorizar el movimiento continuo de una persona y detectar cambios bruscos de postura, sin embargo, este enfoque se aleja del objetivo inicial del proyecto y requiere un mayor consumo de recursos computacionales, lo que dificultaría su implementación. Dado que el sistema puede operar eficazmente con imágenes estáticas, se opta por esta alternativa para simplificar su ejecución, optimizar el uso de recursos y ajustarse mejor a las condiciones iniciales de las que parte el proyecto.

También es necesario elegir qué clases posturales hay que detectar. Requisito indispensable del proyecto es detectar y clasificar la pose de una persona en cuatro clases diferentes, “de pie”, “sentado”, “tumbado” y “caído”, por lo que diferenciar firmemente las poses principales del proyecto es necesario para cumplir con el propósito.

Por último, definir qué métodos y herramientas se han analizado para conseguir cumplir los requisitos que se han definido.

- **MediaPipe:** Usar MediaPipe Pose para obtener datos clave del esqueleto humano y clasificar posturas basándose en estas coordenadas. Ofrece una estructura bien definida de puntos clave y es relativamente fácil de integrar. Se comenzará a trabajar con esta herramienta como una alternativa prometedora.

- **YOLO:** Emplear YOLO para detectar y clasificar posturas mediante un modelo de detección de objetos. Alta fiabilidad en detección de personas y adaptable para diferenciar entre posiciones generales. También se integrará al flujo de trabajo para evaluar su rendimiento frente y conjunto a MediaPipe.

- **Red neuronal:** Entrenar una red neuronal capaz de clasificar posturas basándose en imágenes etiquetadas. Este enfoque otorgaría resultados mejores que otros métodos, pero implica una alta complejidad, requiere un gran volumen de datos etiquetados y un entrenamiento computacionalmente costoso. Se descarta debido a la complejidad innecesaria en comparación con el objetivo del proyecto.

- **SVM:** Entrenar un clasificador SVM utilizando las características extraídas por MediaPipe y YOLO. Más sencillo y rápido de entrenar en comparación con redes neuronales. Ofrece resultados suficientemente precisos para el propósito del proyecto. Implementar este método logra un balance óptimo entre simplicidad y eficacia para un método de clasificación autónomo.

Estos métodos y herramientas se explicarán y estudiarán más a fondo en los apartados siguientes del proyecto.

4.1.3. Selección final.

Tras analizar los métodos y sus combinaciones, se seleccionó el siguiente enfoque:

- Uso conjunto de MediaPipe Pose y YOLO para obtener características robustas y fiables en diversos escenarios.
- Clasificación mediante un SVM, aprovechando su simplicidad y capacidad de generalización.
- Exclusión de métodos no relevantes (video, red neuronal) para optimizar recursos y alinearse con los objetivos del proyecto.

Esta solución cumple con los requisitos de simplicidad, eficacia y bajo consumo de recursos. Además, se ajusta al contexto específico de analizar imágenes de

personas en entornos domésticos, garantizando un rendimiento adecuado en situaciones reales.

4.2. Herramientas utilizadas.

A continuación, se profundizará más sobre las herramientas y el entorno utilizado para la realización del proyecto.

4.2.1. Entorno utilizado.

Para facilitar la gestión de librerías y la aplicación de Python, el primer paso es realizar la instalación de Anaconda. Anaconda es una distribución de software libre y de código abierto de los lenguajes Python y R, utilizada en ciencia de datos y aprendizaje automático. Está orientado a simplificar el despliegue y administración de los paquetes de software. [20].

En Anaconda se deben instalar todas las librerías necesarias para que el servicio de Python y el programa realizado funcionen correctamente. Estas librerías se pueden comprimir en un archivo de instalación para introducirlas automáticamente, o, por el contrario, realizar la instalación de forma manual una a una. Las diferentes versiones de los paquetes se administran mediante el sistema de gestión de paquetes conda, el cual lo hace bastante sencillo de instalar, ejecutar y actualizar el software. La lista completa de todas las librerías con su correspondiente versión se podrá encontrar en los anexos.

Una vez instalado Anaconda, debemos instalar una versión de Python.

Python es un lenguaje de programación de alto nivel, interpretado, y de propósito general. Entre otras características, Python es conocido por ser fácil de aprender y usar debido a su sintaxis sencilla y legible. Puede utilizarse en diversas áreas de desarrollo y posee una gran comunidad muy activa y una vasta cantidad de bibliotecas. En este caso se ha trabajado en la versión Python 3.9.19.

En Anaconda, para ejecutar Python se hará uso del entorno de desarrollo integrado Spyder, en su versión 5.5.1.

4.2.2. OpenCV.

OpenCV es una biblioteca de visión por computadora de código abierto que permite trabajar con imágenes y videos de una manera muy versátil y eficiente. Es una herramienta ampliamente utilizada tanto en proyectos académicos como en aplicaciones comerciales, ya que ofrece funcionalidades para procesar imágenes, detectar objetos, analizar movimientos, reconocer rostros, entre muchas otras cosas.

Una de las principales ventajas de OpenCV es su capacidad para procesar datos en tiempo real, algo esencial en aplicaciones donde la velocidad es clave, como la detección de personas. Además, es compatible con varios lenguajes de programación como Python, C++ y Java, lo que facilita su integración en distintos proyectos. [21].

En este trabajo se utiliza OpenCV para tareas como el preprocesamiento de imágenes y la visualización de resultados (dibujar cajas delimitadoras o esqueleto sobre las personas detectadas). También actúa como un puente para combinar otras herramientas, como MediaPipe y YOLO, permitiendo trabajar de manera conjunta y eficiente en la caracterización de las posturas de las personas.

Un aspecto por destacar es cómo trabaja OpenCV con las imágenes, pues será útil entenderlo para el futuro tratamiento de las características extraídas. Interpreta cada imagen como una matriz de píxeles. En esta matriz, cada elemento representa un píxel, y su valor puede variar dependiendo del formato de la imagen. Por ejemplo, en imágenes a color cada píxel tiene tres valores (R, G, B), que corresponden a las intensidades de los canales rojo, verde y azul. La intensidad de cada canal se define en un rango [0-255], donde el 0 representa la mínima intensidad y el 255 representa la intensidad máxima para ese canal. Así, un píxel con los siguientes valores en cuanto a color (255,0,0) será plenamente rojo.

Usa un sistema de coordenadas cartesianas adaptado a las imágenes cuyo origen (0,0) se fija en la esquina superior izquierda de la imagen, en consecuencia, el eje X crece hacia la derecha, representando las columnas de la matriz y el eje Y crece hacia abajo, representando las filas de la matriz.

La siguiente ilustración (figura 18) representa el sistema de coordenadas que sigue OpenCV y el valor de color de 3 píxeles presentes en la imagen.

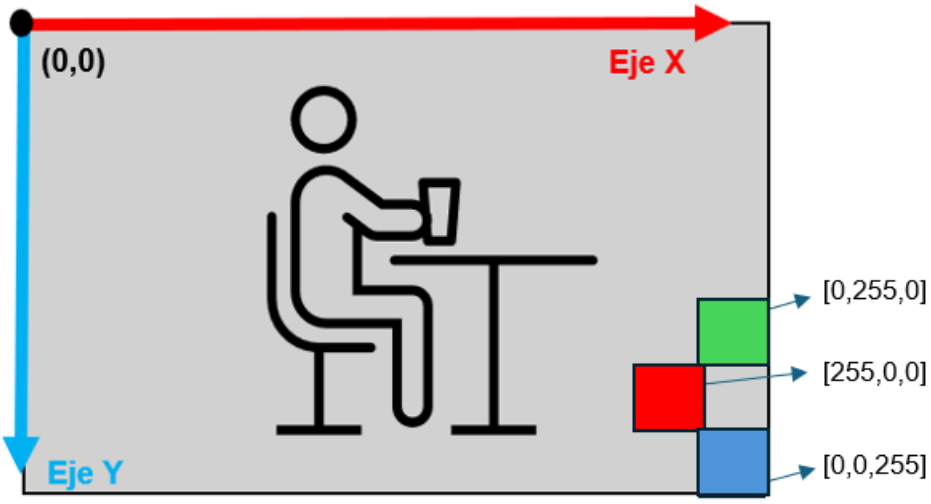


Figura 18: Sistema de coordenadas y matriz de píxeles de OpenCV. (Fuente: Elaboración propia)

4.2.3. MediaPipe y MediaPipe Pose.

MediaPipe es un conjunto de herramientas de código abierto diseñado para facilitar el desarrollo de soluciones de VA en tiempo real. Ofrece una colección de soluciones listas para usar en tareas como el reconocimiento facial, el seguimiento de gestos, la detección de objetos y el seguimiento del cuerpo humano. [22].

Gracias a su funcionalidad modular y a su compatibilidad con diversas plataformas, MediaPipe permite a los desarrolladores crear aplicaciones avanzadas con gran flexibilidad y eficacia. Lo que hace que MediaPipe sea especialmente importante (y útil) es su capacidad para simplificar el proceso de desarrollo de la VA, ofreciendo servicios optimizados para analizar flujos y mejorar la calidad de las soluciones. Al ofrecer soluciones preconfiguradas y optimizadas, se reduce el tiempo y el esfuerzo necesarios para integrar complejas funciones de procesamiento visual en las aplicaciones. Además, su arquitectura modular permite una personalización sencilla y una integración perfecta con otras tecnologías, lo que proporciona una potente plataforma para innovaciones en campos tan diversos como la realidad aumentada, las aplicaciones sanitarias y mucho más.

En la figura 19 se muestra un esquema de cómo funciona MediaPipe y las etapas por las que pasa el procesamiento de la imagen, donde se transforma la imagen original para ajustarla al modelo, luego se convierte en un tensor (estructura numérica multidimensional) para su procesamiento, a continuación, el modelo realiza el análisis sobre el tensor, generando una salida que se traduce en landmarks (puntos clave) que se combinan con la imagen original para generar la imagen procesada.

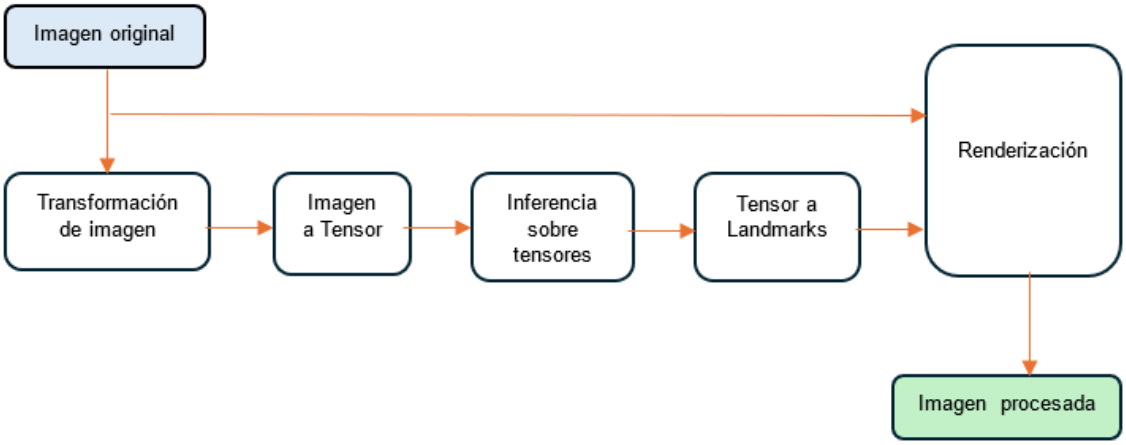


Figura 19: Gráfico que muestra cómo funciona MediaPipe. (Fuente: Elaboración propia)

En este caso se ha hecho uso de MediaPipe Pose. Este es un modelo de visión por computadora específico de MediaPipe diseñado para detectar la pose corporal de una persona en imágenes o videos en tiempo real. El modelo está entrenado para identificar puntos clave del cuerpo (figura 20), proporcionando una representación precisa de la posición y la orientación de una persona. Este modelo es muy útil para aplicaciones como el análisis de la postura, la detección de actividades deportivas, el seguimiento de movimientos y más.

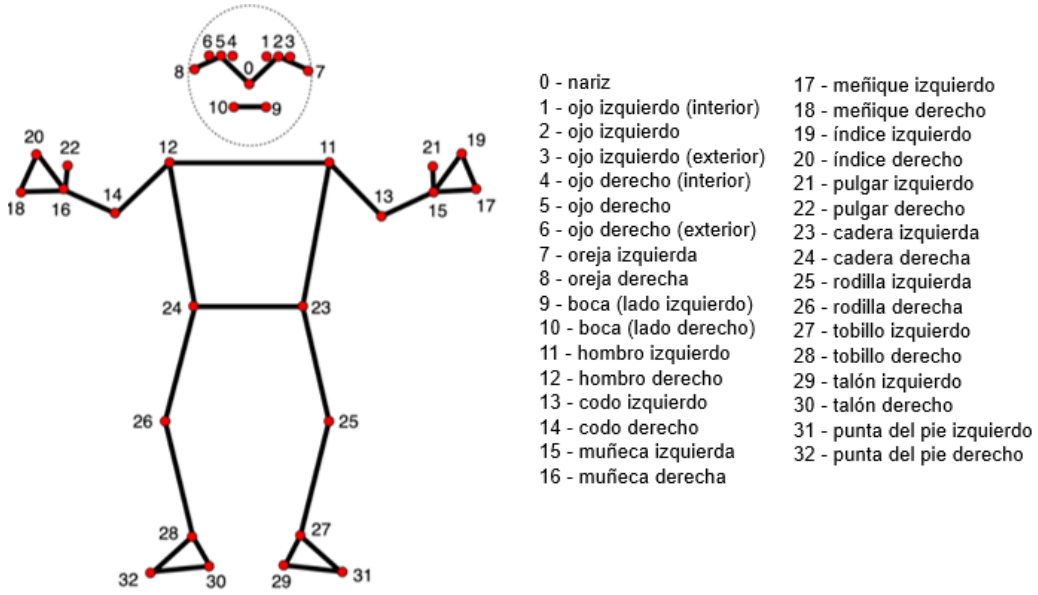


Figura 20: 33 Puntos de referencia corporales detectados por MediaPipe Pose. (Fuente: [23])

MediaPipe Pose recibe una imagen o video en tiempo real y se realiza una serie de transformaciones en la imagen como redimensión y normalización de colores para que sea adecuada para el proceso de detección y reducir así el tiempo de

procesamiento. Se identifica la región de interés (ROI) que contiene a la persona para limitar el espacio de búsqueda, permitiendo enfocarse en el cuerpo humano y no en toda la imagen. Se utiliza una red neuronal especializada para realizar la detección de los 33 puntos clave del cuerpo humano. A estos puntos clave detectados se les denomina "*landmarks*" y son las coordenadas que representan las ubicaciones específicas del cuerpo humano.

Tras la detección de los puntos clave, se realiza un proceso de inferencia para determinar con precisión las posiciones de los puntos, lo que permite que el modelo prediga la posición de las articulaciones y extremidades de una persona en relación con el cuerpo entero, incluso si la persona está parcialmente oculta o en una posición difícil de ver.

Por último, se realiza un posprocesamiento para evitar fluctuaciones o ruido en los resultados utilizando filtros de suavizado que mejoran la estabilidad de los *landmarks* y se dibujan los puntos clave extraídos sobre la imagen original, mostrando visualmente la posición de las articulaciones y las partes del cuerpo.

Con estos resultados obtenidos se puede hacer uso de las coordenadas de cada *landmark* para utilizarlo en diferentes aplicaciones como el análisis de postura.

En la figura 21 se muestran los puntos clave dibujados en una imagen ejemplo.

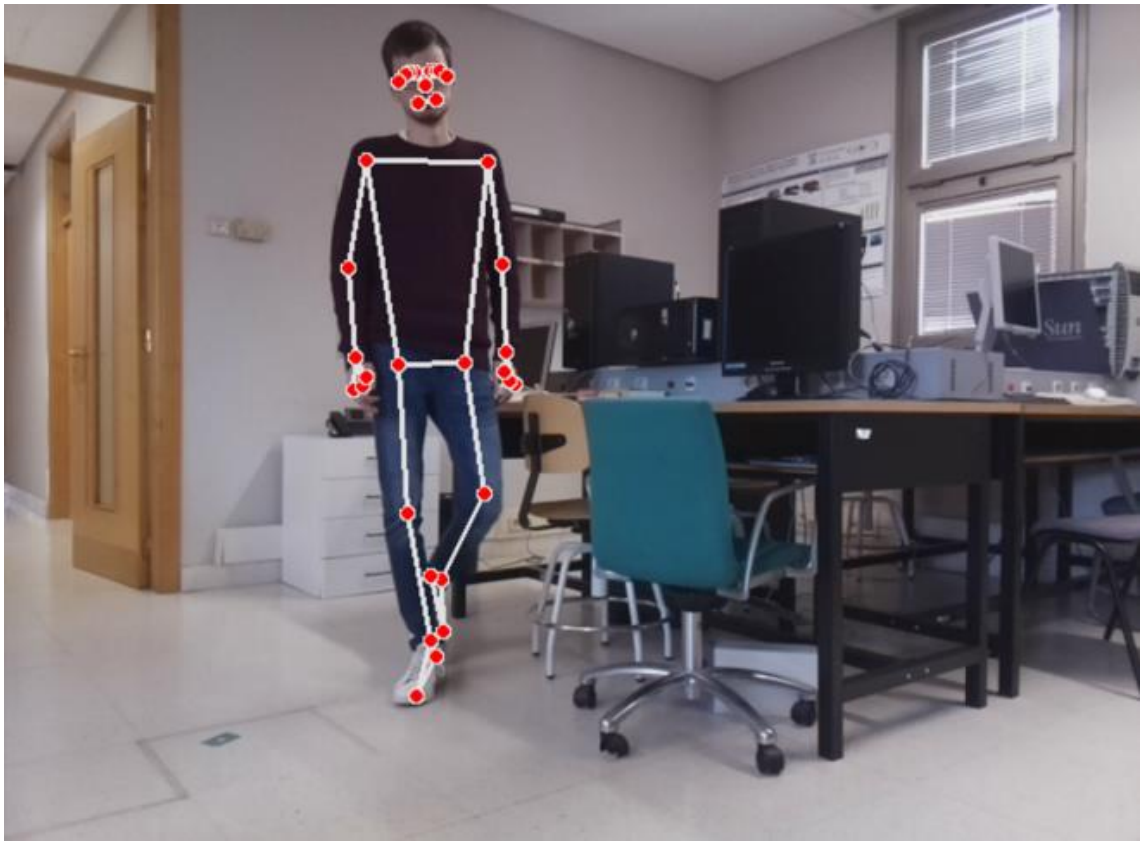


Figura 21: Puntos de referencia de la postura corporal en una imagen. (Fuente: Elaboración propia)

MediaPipe es extremadamente eficiente incluso en dispositivos con recursos limitados gracias a su diseño optimizado. Tiene herramientas y bibliotecas que facilitan la integración en proyectos de VA y es compatible con plataformas como Python. Está entrenando con grandes conjuntos de datos de imágenes de personas, lo que permite obtener resultados precisos sin necesidad de entrenar un modelo desde cero.

La tabla 1 recoge algunas de las ventajas y desventajas de trabajar con MediaPipe Pose:

VENTAJAS	DESVENTAJAS
Utiliza modelos avanzados de aprendizaje profundo que proporcionan estimaciones precisas de las poses humanas, incluso en escenarios desafiantes como posiciones ociosas o dinámicas.	Está diseñado para analizar una sola persona por imagen o cuadro, lo que lo hace inadecuado para escenarios con múltiples sujetos.
MediaPipe Pose está optimizado para funcionar en tiempo real, incluso en dispositivos con recursos limitados como smartphones.	Su precisión puede disminuir en entornos con poca luz o iluminación desigual.
Funciona en diferentes plataformas, incluyendo Android, iOS, Python y JavaScript, lo que facilita la implementación en diversas aplicaciones.	Requiere imágenes o videos de alta resolución para un rendimiento óptimo, lo que puede no ser práctico en dispositivos de baja gama.
Ofrece un modelo listo para usar, reduciendo el esfuerzo de entrenamiento y etiquetado de datos.	Puede fallar al estimar poses cuando partes del cuerpo están ocluidas (por ejemplo, cuando los brazos cruzan el torso).
Incluye una API bien documentada y fácil de implementar, lo que permite a los desarrolladores integrar pose estimation rápidamente en proyectos.	Está entrenado principalmente en datos que reflejan poses humanas generales; puede no ser preciso en poses muy inusuales o extremas.

Tabla 1: Ventajas y desventajas de MediaPipe Pose.

4.2.4. YOLO y YOLOv3.

Los sistemas de clasificación de objetos, utilizados por los programas de inteligencia artificial, tienen como objetivo percibir objetos específicos de una clase como sujetos de interés. Estos sistemas clasifican los objetos de las

imágenes en grupos que colocan juntos a los objetos con características similares y dejan de lado a los demás, a menos que estén programados para hacer lo contrario. Los grupos resultantes ayudan a identificar y categorizar los objetos en función de sus características, lo que contribuye a una comprensión más matizada de la clase prevista a la que pertenece cada objeto.

El algoritmo You Only Look Once (YOLO), es un sistema de código abierto para detección de objetos en tiempo real, el cual hace uso de una única red neuronal convolucional para detectar objetos en imágenes [24]. Para su funcionamiento, la red neuronal divide la imagen en regiones, prediciendo cuadros de identificación y probabilidades por cada región. Las cajas, son ponderadas a partir de las probabilidades predichas. Estas cajas también denominadas “bounding boxes”, encierran al objeto detectado y se definen mediante cuatro parámetros principales, coordenada x del centro de la caja, coordenada y del centro de la caja, ancho de la caja y alto de la caja (figura 22).

El algoritmo aprende representaciones generalizables de los objetos, permitiendo un bajo error de detección para entradas nuevas, diferentes al conjunto de datos de entrenamiento.

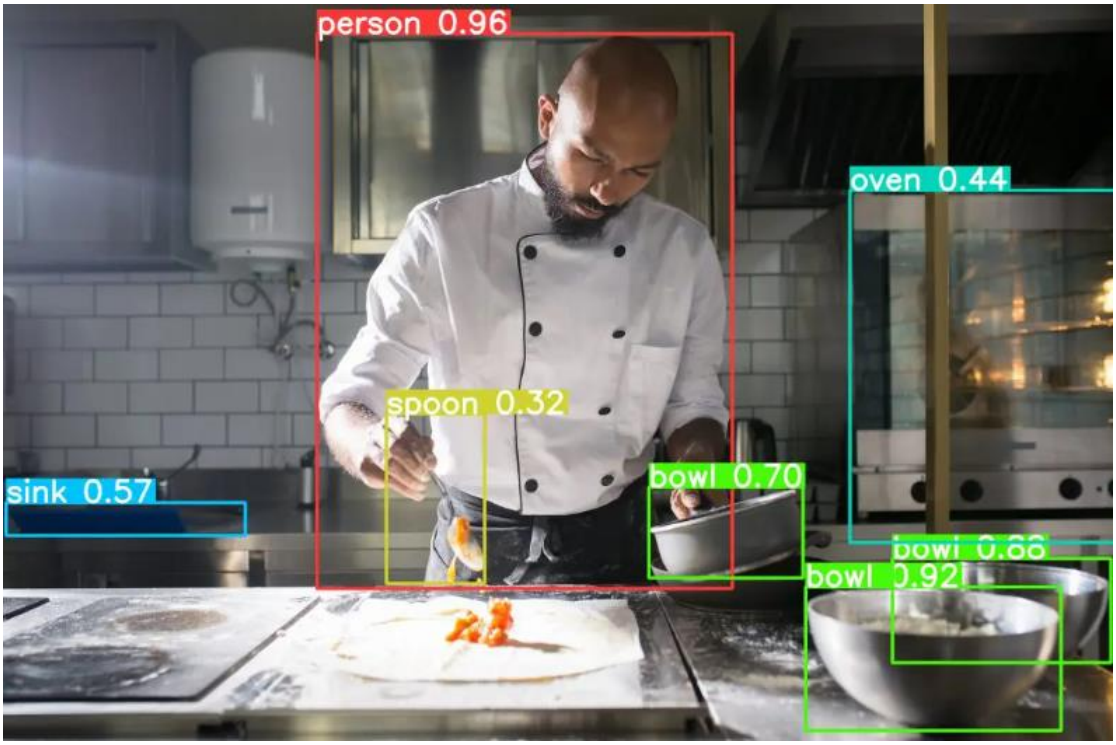


Figura 22: Ejemplo de resultado de detección de YOLO. (Fuente: [25])

Como es habitual en los detectores de objetos, las características aprendidas por las capas convolucionales se pasan a un clasificador que realiza la predicción de detección. YOLO tiene la ventaja de ser mucho más rápida que otras redes y, aun así, mantiene la precisión.

El modelo de detección de objetos puede observar la imagen completa en el momento de la prueba. YOLO y otros algoritmos “puntúan” las regiones en función de sus similitudes con clases predefinidas. Las regiones con puntuaciones altas se consideran detecciones positivas de la clase con la que se identifican más estrechamente. Este mecanismo de puntuación, que implica propuestas regionales, permite la detección precisa y eficiente de objetos en varias escenas.

Una de las versiones de YOLO y la utilizada en este trabajo es YOLOv3. Joseph Redmon y Ali Farhadi crearon la primera versión de los algoritmos YOLO en 2016. Los dos lanzaron la versión 3 dos años después, en 2018. YOLOv3 es una versión mejorada de YOLO y YOLOv2.

YOLOv3 es una versión plenamente probada por científicos y desarrolladores y existen numerosos *datasets* o conjuntos de datos que han sido optimizados para esta versión en concreto. Es una versión estable y madura que ha estado en circulación durante un periodo más prolongado, sometiéndose a extensas pruebas y validaciones en diversas aplicaciones.

Otro punto por destacar es que YOLOv3 es un modelo comparativamente ligero, lo que lo convierte en una opción adecuada para la implementación en dispositivos con recursos limitados, ya que otras versiones posteriores como YOLOv11 requieren más recursos de almacenamiento y computación, aunque sean más precisas.

La figura 23 muestra las etapas principales del algoritmo YOLOv3 aplicado a la detección de un objeto, en este caso un coche.

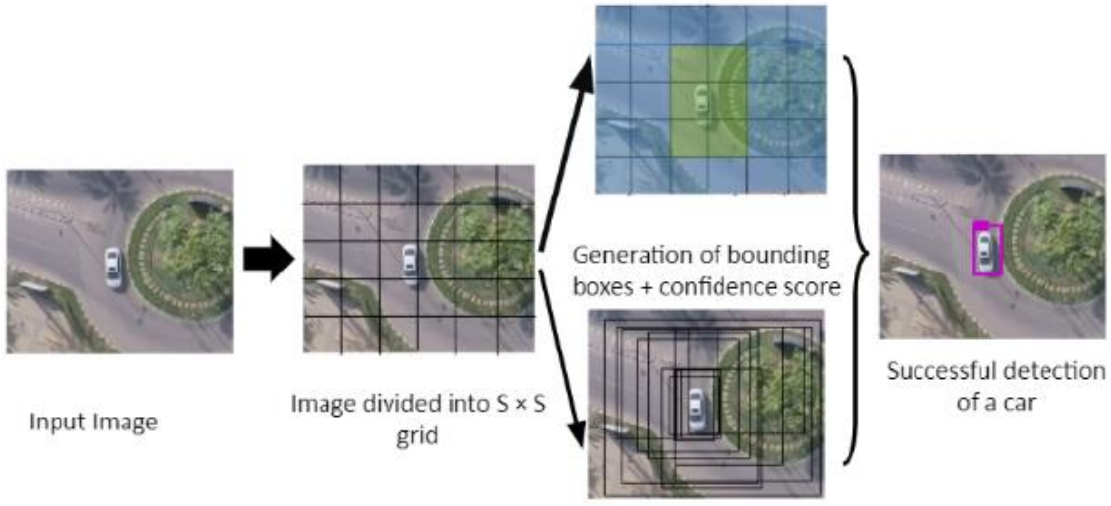


Figura 23: Etapas sucesivas de la detección de YOLOv3. (Fuente: [26])

El algoritmo YOLOv3 primero separa una imagen en una cuadrícula. Cada celda de la cuadrícula predice una cierta cantidad de cuadros delimitadores (a veces denominados cuadros de anclaje) alrededor de los objetos que tienen una puntuación alta en las clases predefinidas mencionadas anteriormente.

YOLOv3 es rápido y preciso en términos de precisión media promedio (mAP) y valores de intersección sobre unión (IOU). Funciona significativamente más rápido que otros métodos de detección con un rendimiento comparable. Además, se puede equilibrar fácilmente la velocidad y la precisión simplemente modificando el tamaño del modelo, sin necesidad de volver a entrenarlo, lo que demuestra la versatilidad de la extracción de características dentro de la arquitectura YOLOv3. [27].

Algunas ventajas y desventajas de utilizar YOLO para el presente trabajo se resumen en la tabla 2:

VENTAJAS	DESVENTAJAS
Alta velocidad de procesamiento debido a su arquitectura basada en redes neuronales convolucionales que procesan toda la imagen en un solo paso.	Tiene dificultades para detectar objetos pequeños en imágenes grandes, ya que estos pueden no ser suficientemente representados en las cuadrículas de la red.
Analiza toda la imagen, lo que ayuda a minimizar errores relacionados con el solapamiento de objetos o detecciones duplicadas.	Aunque es rápido, puede sacrificar precisión en comparación con modelos más lentos.
Puede detectar múltiples objetos en una imagen con alta precisión, haciéndolo útil en aplicaciones como vigilancia, conducción autónoma y robótica.	En imágenes con gran densidad de objetos o ruido, YOLO puede generar detecciones incorrectas.
Es más rápido y menos demandante en recursos que otros métodos como R-CNN o Fast R-CNN, especialmente en versiones más recientes.	Su rendimiento depende mucho de la calidad y cantidad de datos utilizados para entrenarlo, y puede no generalizar bien a dominios diferentes del conjunto de datos original.
La comunidad ofrece soporte extenso, bibliotecas preentrenadas y documentación que facilita su integración en proyectos.	YOLO divide la imagen en cuadrículas para predecir objetos, lo que puede dificultar la detección precisa de objetos cuya posición o escala no se ajusta bien a esta división. Esto puede causar problemas en imágenes donde los objetos están muy cercanos o se solapan.

Tabla 2: Ventajas y desventajas de YOLO.

4.2.5. Máquina se vectores de soporte. SVM.

Durante el desarrollo del proyecto, surge la idea de no solo trabajar la clasificación mediante técnicas de puro análisis de características de manera "manual", sino también, profundizar en alguna técnica de clasificación automática que sea sencilla y fácil de implementar y que ayude a manejar la caracterización desde otro punto de vista.

Se opta por una máquina de vectores de soporte (del inglés *support*-vector-*machine*, SVM). El SVM es un potente algoritmo de aprendizaje automático ampliamente utilizado tanto para la clasificación lineal como para la no lineal, así como para tareas de regresión y detección de valores atípicos. Las SVM son altamente adaptables, lo que las hace adecuadas para diversas aplicaciones.

Las máquinas de vectores de soporte son particularmente eficaces porque se centran en encontrar el hiperplano de separación máximo en un espacio N-dimensional que pueda separar eficazmente los puntos de datos en diferentes clases en el espacio de características destino, lo que las hace robustas tanto para la clasificación binaria como para la multiclase.

El algoritmo garantiza que se maximice el margen entre los puntos más cercanos de diferentes clases, conocidos como vectores de soporte.

La dimensión del hiperplano depende del número de características. Por ejemplo, si hay dos características de entrada, el hiperplano es simplemente una línea, y si hay tres características de entrada, el hiperplano se convierte en un plano 2D. A medida que el número de características aumenta más allá de tres, también aumenta la complejidad de visualización del hiperplano. [28].

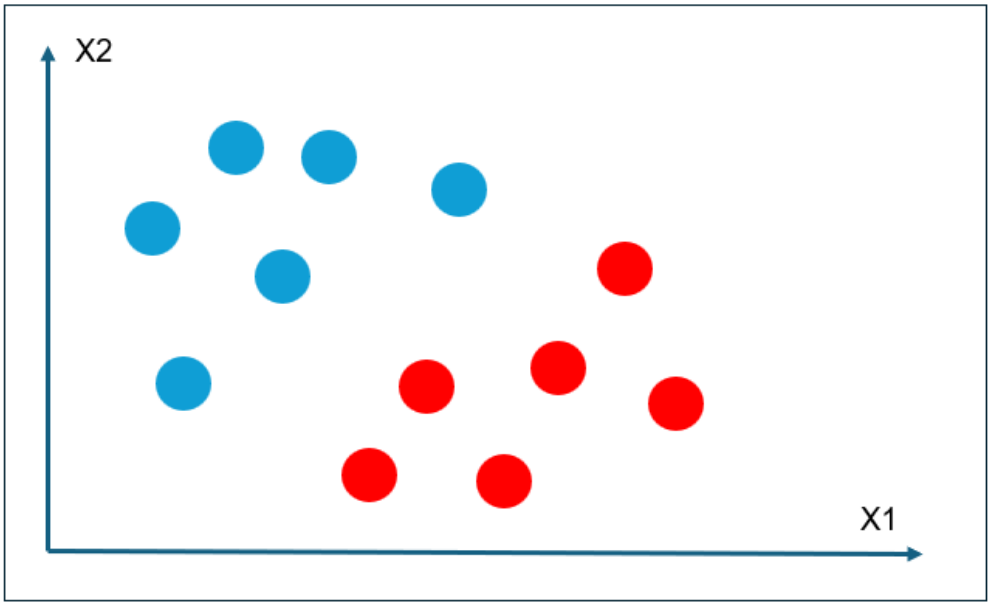


Figura 24: Datos separables linealmente. SVM. (Fuente: Elaboración propia)

Se consideran dos variables independientes, x1 y x2, y una variable dependiente representada como un círculo azul o un círculo rojo (figura 24).

En este escenario, se intenta separar los círculos azules de los rojos, por lo tanto, el problema es linealmente separable. El hiperplano será una línea porque estamos trabajando con dos características (x1 y x2).

De la figura anterior se desprende claramente que existen múltiples líneas que separan los puntos de datos o realizan una clasificación entre círculos rojos y azules. Una opción razonable para el mejor hiperplano en un SVM es aquella que maximiza el margen de separación entre las dos clases. El hiperplano de margen máximo, también conocido como margen duro, se selecciona en función de maximizar la distancia entre el hiperplano y el punto de datos más cercano en cada lado.

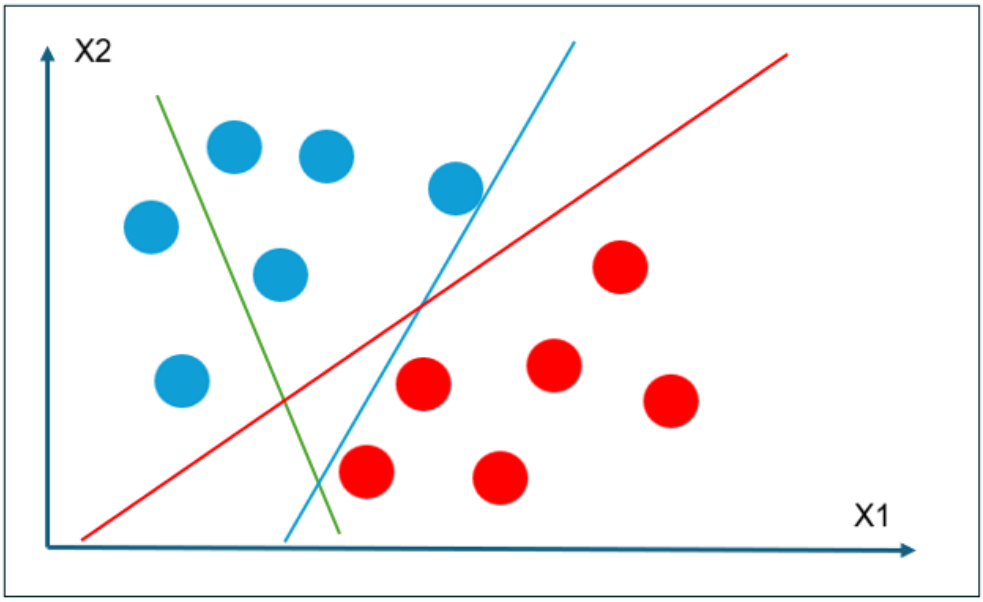


Figura 25: Clasificadores lineales. SVM. (Fuente: Elaboración propia)

En la figura 25, se tienen 3 clasificadores: la línea verde, la línea azul y la línea roja.

El clasificador verde es el peor, pues, teniendo un problema sencillo de separar y una distribución sencilla, no consigue separar las variables. Por otro lado, los clasificadores azul y rojo son perfectos, pero el rojo es mejor que el azul, puesto que va a generalizar mejor. Esto se debe a que deja más espacio entre los elementos de ambos grupos, dando más margen a nuevas instancias para clasificarlas correctamente.

Algunas ventajas y desventajas del uso de SVM se resumen en la siguiente tabla (tabla 3):

VENTAJAS	DESVENTAJAS
Rendimiento de alta dimensión: SVM se destaca en espacios de alta dimensión, lo que lo hace adecuado para la clasificación de imágenes y el análisis de expresión genética.	Entrenamiento lento: SVM puede ser lento para conjuntos de datos grandes, lo que afecta el rendimiento de SVM en tareas de minería de datos.
Capacidad no lineal: al utilizar funciones de *kernel*, SVM maneja exitosamente relaciones no lineales.	Dificultad de ajuste de parámetros: seleccionar el *kernel* correcto y ajustar cuidadosamente parámetros, lo que impacta en los algoritmos SVM.
Resiliencia de valores atípicos: la función de margen suave permite a SVM ignorar los valores atípicos, lo que mejora la solidez en la detección de anomalías.	Sensibilidad al ruido: SVM tiene dificultades con conjuntos de datos ruidosos y clases superpuestas, lo que limita la eficacia en escenarios del mundo real.
Soporte binario y multiclase: SVM es eficaz tanto para la clasificación binaria como para la clasificación multiclase, adecuado para aplicaciones en clasificación de texto.	Interpretabilidad limitada: la complejidad del hiperplano en dimensiones superiores hace que SVM sea menos interpretable que otros modelos.

Tabla 3: Ventajas y desventajas de SVM.

4.3. Clasificación de la pose.

4.3.1. Idea inicial.

1. *Utilización de MediaPipe para la detección y caracterización de posiciones corporales.*

MediaPipe es una herramienta de visión artificial altamente eficiente para la detección de puntos clave del cuerpo humano. Estos puntos clave, representan ubicaciones específicas del cuerpo, como articulaciones o extremidades. En este proyecto, se utilizará MediaPipe para identificar y rastrear estos *landmarks* proporcionando información crítica sobre la postura de una persona.

Esta información se utilizará para calcular ángulos entre extremidades, proporciones del cuerpo, y otras características relevantes que permitan diferenciar posturas como estar “de pie”, “sentado”, “tumbado” o “caído”. MediaPipe también permite evaluar la confianza de cada punto clave, lo que facilita descartar datos poco fiables debido a oclusiones.

2. *YOLO para la detección y caracterización de posiciones corporales.*

YOLO sobresale por su velocidad y precisión. Este modelo se usará en el proyecto para identificar a la persona y objetos presentes en la imagen. Con el bounding box de YOLO también se podrán extraer ciertas características como el alto, ancho o proporciones que ayudarán a proporcionar información sobre la postura. Mediante esta herramienta se llevará a cabo la clasificación de la clase tumbado. Para lograr esta distinción, se tomará en cuenta el contexto del entorno, pues, YOLO puede detectar objetos como sofás o comas y basándose en la presencia de estos objetos y la relación que tienen con la persona detectada, se conseguirá una idea sobre si la persona está tumbada en un sofá o caída en el suelo.

3. *Combinación de YOLO y MediaPipe para mejorar la precisión*

Aunque MediaPipe es extremadamente útil para extraer detalles del cuerpo, tiene limitaciones en casos donde la visión de las extremidades está parcial o totalmente obstruida. Por otro lado, YOLO es más robusto para detectar la presencia de una persona en condiciones complejas, como baja iluminación o posiciones no convencionales.

El sistema combinará ambas herramientas aprovechando sus fortalezas. En situaciones donde MediaPipe no logre identificar todos los *Landmarks* (por ejemplo, en una postura tumbada donde las piernas están parcialmente fuera del marco), el modelo podrá apoyarse en la detección de YOLO para inferir la postura de forma más confiable. En resumen, se utilizarán ambos métodos para cada caso, contando con ambas detecciones y ambos resultados, tomando decisiones basadas en reglas predefinidas para seleccionar el resultado más confiable entre ambos métodos, pues hay situaciones en las que un método presenta fortalezas frente a otro.

4. *Uso de SVM entrenado con datos recopilados.*

A mayores del método de clasificación mediante MediaPipe y YOLO, se utilizará un modelo de clasificación basado en SVM, que se entrenará con las características extraídas de MediaPipe y YOLO. Estas características incluirán posición de puntos clave, distancias entre puntos, ángulos de articulaciones, proporciones corporales, información sobre los bounding boxes... El modelo

SVM permitirá clasificar de forma automática las posturas en las categorías principales.

5. *Resultado final: clasificación manual y automática.*

El sistema generará dos tipos de resultados para cada evaluación.

- Clasificación manual: Una combinación de reglas lógicas basadas en los datos obtenidos de YOLO y MediaPipe. Por ejemplo, si MediaPipe identifica una distribución específica de puntos clave, pero YOLO no detecta a la persona dentro de una región de interés consistente, se podría etiquetar como "caído".

- Clasificación automática: El resultado del modelo SVM entrenado, que tomará como entrada los datos procesados para asignar una de las clases predeterminadas.

Finalmente, se evaluará la precisión y eficiencia de ambos enfoques, identificando fortalezas y áreas de mejora. El resultado será desarrollar un sistema robusto que combine detección manual y automática para aplicaciones en entornos domésticos, brindando apoyo a personas que viven solas.

4.3.2. Desarrollo del sistema.

En este apartado se profundizará en los pasos seguidos en el desarrollo del proyecto, desde la fase inicial hasta la versión final. Este apartado es de gran importancia, pues explica de manera detallada cómo se han tomado las decisiones y cómo se ha llegado a las soluciones elegidas.

4.3.2.1. Obtención de datos.

Se empezará por encontrar y elegir uno o varios conjuntos de datos o "*datasets*" de imágenes que ayuden a realizar el entreno, extraer distintas características de muchas imágenes y realizar las pruebas de test una vez tengamos una versión completa del código.

Los *datasets* contienen imágenes que no se ajustan a las especificaciones del proyecto, como imágenes en las que aparecen varias personas, por ello, se realiza un filtrado del conjunto entero de imágenes. Para conseguir este filtrado

de una manera rápida y eficiente, se ha elaborado un programa (“clasificador de imágenes en carpetas.py”) en Python para clasificar las imágenes en las clases predominantes. La estructura básica de este clasificador se podría resumir con el siguiente diagrama de bloques (figura 26):

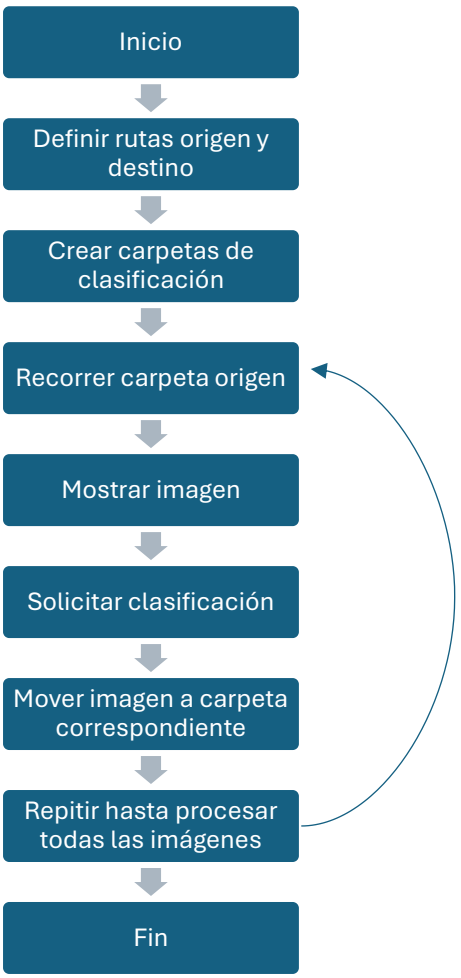
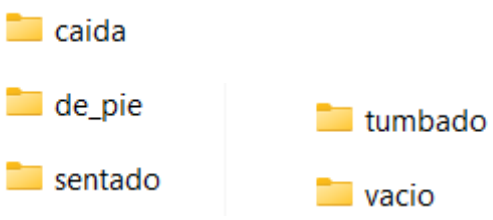


Figura 26: Diagrama de flujo del programa "clasificador de imágenes en carpetas.py". (Fuente: Elaboración propia)

El programa de clasificación básicamente recorre las carpetas origen mostrando al usuario las imágenes y moviéndolas a la subcarpeta correspondiente a las clases predominantes a clasificar (“de pie”, “sentado”, “caída”, “tumbado”, “vacío”) según la entrada del usuario por teclado (“1”, ”2”, ”3”, ”4”, ”5”) correspondientemente.



A las clases básicas se añade también una correspondiente a imágenes vacías, es decir, sin personas, para poder comprobar que el sistema no detecta personas donde no las hay. Este programa ha sido muy útil permitiendo el preprocesamiento de los *datasets*. Con las imágenes ya filtradas y clasificadas se puede comenzar a trabajar con las detecciones de MediaPipe y YOLO.

Entrenamiento y validación:

El *dataset* utilizado para la fase de entrenamiento y validación es el siguiente:

- *Fallen People Detection Capabilities Using Assistive Robot. S. Maldonado-Bascón, C. Iglesias-Iglesias, P. Martín-Martín, S. Lafuente-Arroyo. Electronics 2019.*

Se opta por utilizar este dataset en particular debido a su adecuación para el problema de clasificación de posturas, ya que ha sido desarrollado para tareas de detección de caídas mediante un robot asistente [29] y proporciona datos etiquetados de alta calidad que permiten entrenar y validar el modelo de manera eficiente.

Figura 27: Ejemplo de imágenes del dataset FPDs. (Fuente: [29])

Está formado por 6982 imágenes, con un total de 5023 caídas y 2275 no caídas correspondientes a personas en situaciones convencionales (de pie, sentadas, tumbadas en el sofá o la cama, andando, etc.). Casi todas las imágenes han sido captadas en entornos interiores con situaciones muy diferentes como variación de poses y tamaños, oclusiones, cambios de iluminación, etc. En la imagen superior (figura 27) se muestra un ejemplo de algunas imágenes que contiene este *dataset*.

El *dataset* ha sido optimizado con pesos en YOLOv3. Esta es la razón por la que se utiliza esa versión de YOLO en este proyecto, aparte de ofrecer robustez y fiabilidad al ser una versión testada a fondo por una gran parte de la comunidad y desarrolladores.

Test:

Para la fase de test del proyecto se utiliza un *dataset* completamente distinto al utilizado en las pruebas y entrenamiento. Esto garantiza que el modelo creado generaliza bien y no se cometen errores de sobreajuste o resultados falseados por utilizar las mismas imágenes que en el entreno.

El *dataset* utilizado para el conjunto de test está formado por la unión de varios debido a la escasa diversidad de datos de algunos de ellos, poder cubrir más escenarios con condiciones o contextos diferentes y aumentar en algunos casos la cantidad de ejemplos de algunas de las clases.

- *IASLAB-RGBD (Antonello, M.; Carraro, M.; Pierobon, M.; Menegatti, E. Fast and robust detection of fallen people from a mobile robot. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)).*

Figura 28: Ejemplo de imágenes del dataset IASLAB-RGBD. (Fuente: [30])

Este *dataset* contiene 342 imágenes en las que están presentes las cuatro poses predominantes del proyecto. Al igual que en el *dataset* de entrenamiento, se seleccionan las imágenes según los requisitos del proyecto. En la imagen superior (figura 28) se muestra un ejemplo de algunas imágenes que contiene este *dataset*.

- *Up-Fall Detection (Kwolek, B.; Kepski, M. Human fall detection on embedded platform using depth maps and wireless accelerometer. Computer methods and programs in biomedicine 2014).*

Figura 29: Ejemplo de imágenes del dataset Up-Fall Detection. (Fuente: [31])

En este *dataset* hay presentes 2097 imágenes correspondientes a secuencias de movimientos de una persona, desde estar de pie, andar, caerse, sentarse… Al estar formado por secuencias de movimientos, muchas de las imágenes son demasiado similares, por lo que en el filtrado de este *dataset* se eliminan muchas de ellas, pues, esto podría llegar a falsear en cierta manera los resultados obtenidos. En la imagen superior (figura 29) se muestra un ejemplo de algunas imágenes que contiene este *dataset*.

- *Elderly Set. S. Maldonado-Bascón, C. Iglesias-Iglesias, P. Martín-Martín, S. Lafuente-Arroyo. Electronics 2019.*

Figura 30: Contenido del dataset Elderly Set. (Fuente: [32])

Por último, este *dataset* contiene 413 imágenes, exclusivamente con voluntarios mayores de 65 años y en situaciones domésticas. Este conjunto se ajusta muy bien a las necesidades que pretende cumplir este proyecto. En la imagen superior (figura 30) se muestra un ejemplo de algunas imágenes que contiene este *dataset*.

4.3.2.2. Uso de MediaPipe.

Para empezar a hacer uso de MediaPipe hay que instalarlo en el entorno de trabajo en Anaconda. Esto se puede hacer de dos maneras, a través del sistema de gestión de paquetes conda navegando por él, buscando e instalando MediaPipe, o, a través del prompt de Anaconda.

En este caso se utiliza el segundo método, a través del prompt. También es una tarea muy sencilla, pues solo hay que ejecutar los dos siguientes comandos.

Conda activate ("nombre del entorno creado")

Pip install MediaPipe

Con estos dos comandos ya tenemos en nuestro entorno el paquete de MediaPipe, en este caso la versión 10.8.

MediaPipe es capaz de detectar hasta 33 puntos clave del cuerpo humano. Para poder llevar a cabo estas detecciones en el código, primero hay que inicializar el modelo, configurándolo para que esté listo para detectar las poses y dibujarlas.

```
mp_pose = mp.solutions.pose
pose = mp_pose.Pose()
mp_drawing = mp.solutions.drawing_utils
```

También es importante convertir la imagen a RGB, pues, es el espacio de color que utiliza MediaPipe. Esto es fácil de conseguir utilizando las funciones de OpenCV.

Para detectar los puntos clave del cuerpo humano en la imagen, se hace uso de la función "process" de MediaPipe. Esta función es la que consigue analizar la imagen y devolver un objeto que contiene las coordenadas de los puntos clave detectados.

```
results = pose.process(rgb_frame)
```

Por último, siempre que se quieran visualizar los puntos clave y las conexiones entre ellos, se hace uso de la función de dibujo que posee MediaPipe.

```
        mp_drawing.draw_landmarks(
            frame,
            results.pose_landmarks,
            mp_pose.POSE_CONNECTIONS
        )
```

Estas funciones serán siempre necesarias para poder realizar las detecciones de los puntos clave.

Tras realizar un pequeño script para familiarizarse con el entorno y resultados de MediaPipe se muestran los primeros resultados obtenidos (figura 31).

Figura 31: Resultados primeras pruebas MediaPipe. (Fuente: Elaboración propia)

Analizando las primeras detecciones de MediaPipe se ve cómo detecta con bastante precisión la posición de las articulaciones y puntos clave del cuerpo humano como era previsto.

Este potencial de detección se explotará para poder caracterizar la pose de la persona, pues se puede acceder a las coordenadas de cada *landmark* detectado, por ejemplo:

```
rodilla_izq = np.array([landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].x,
                        landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].y])

cadera_izq = np.array([landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x,
                       landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].y])
```

coordenadas rodilla_izq= [0.45501429 0.74344283]

coordenadas cadera_izq= [0.52214921 0.74060017]

Las coordenadas de las articulaciones, como en este caso, se almacenan en un array de NumPy para facilitar las operaciones y cálculos sobre estos puntos posteriormente. Se intuye, por lo tanto, el potencial que ofrece MediaPipe utilizando estos *landmarks* y sus coordenadas para poder hacer cálculos de ángulos de articulaciones, distancias entre puntos, incluso, relaciones corporales que serán indicativo de una pose u otra.

La extracción de características es un paso fundamental en el análisis de posturas y movimientos humanos mediante VA. El cuerpo humano se modela frecuentemente como un conjunto de puntos clave que representan articulaciones principales, como los hombros, codos, muñecas, caderas, rodillas y tobillos. En el contexto de este proyecto, se utilizan estos puntos clave para caracterizar la postura.

Las siguientes son las articulaciones consideradas: hombros, muñecas, caderas, rodillas y tobillos. Aparte se utilizan también puntos clave como la cabeza.

La selección de estos indicadores permite una representación eficiente del cuerpo humano al minimizar redundancias y capturar información clave para el análisis, ayudando a distinguir las posturas mediante ángulos de flexión, posición respecto a otros indicadores, distancias entre ellos, etc.

A partir de las posiciones de los puntos clave, se han seleccionado varias características relevantes:

1. **Distancias entre articulaciones**:
    - Se calculan distancias entre pares de articulaciones clave, como hombro-cadera, cadera-rodilla y cadera-tobillo.
    - Estas distancias permiten evaluar la extensión de extremidades y detectar posturas específicas.
2. **Ángulos entre segmentos corporales**:
    - Se calculan ángulos utilizando las posiciones de tres articulaciones consecutivas (por ejemplo, hombro-cadera-rodilla o cadera-rodilla-tobillo).
    - Los ángulos proporcionan información sobre la flexión y extensión de las extremidades.
3. **Relaciones espaciales**:
    - Posiciones relativas de los puntos clave.
    - Relación entre partes del cuerpo como el torso y las piernas para caracterizar mejor algunas de las posturas.

A continuación, se muestra un listado completo sobre las características concretas extraídas de cada imagen (tablas 4, 5, 6 y 7).

Distancias y relaciones
Hombro - rodilla derecha en eje y
Hombro – rodilla izquierda en eje y
Hombro – rodilla derecha en eje x
Hombro - rodilla izquierda en eje x
Cadera – rodilla derecha en eje y
Cadera - rodilla izquierda en eje y
Cadera – rodilla derecha en eje x
Cadera - rodilla izquierda en eje x
Hombro – cadera derecha en eje y
Hombro - cadera izquierda en eje y
Hombro - cadera derecha en eje x
Hombro – cadera izquierda en eje x
Entre rodillas
Cabeza – rodilla
Rodilla – mano derecha en eje y
Rodilla – mano izquierda en eje y
Rodilla – mano derecha en eje x
Rodilla – mano izquierda en eje x
Cadera – tobillo derecha en eje y
Cadera – tobillo izquierda en eje y
Cadera – tobillo derecha en eje x
Cadera – tobillo izquierda en eje x
Altura cabeza
Altura cadera
Altura hombro

Tabla 4: Distancias calculadas entre puntos clave.

Coordenadas de puntos clave
Cadera derecha
Cadera izquierda
Rodilla derecha
Rodilla izquierda
Hombro derecho
Hombro izquierdo
Tobillo derecho
Tobillo izquierdo
Cabeza (nariz)
Cadera derecha
Cadera izquierda

Tabla 5: Posiciones extraídas de puntos clave.

Ángulos de articulaciones
Rodilla derecha
Rodilla izquierda
Cadera derecha
Cadera izquierda

Tabla 6: Ángulos calculados de puntos clave.

Relaciones
Torso – pierna derecha
Torso – pierna izquierda

Tabla 7: Relaciones calculadas de puntos clave.

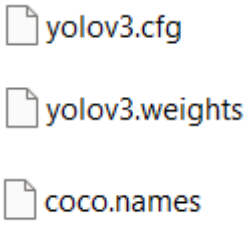
MediaPipe será útil también para detectar la presencia o no de persona en la imagen, pues si no hay detección de *landmarks*, la persona no estará presente en la imagen o, si lo está, no es de una forma evidente.

4.3.2.3. Uso de YOLO.

Para hacer uso de YOLO y detectar personas u objetos en la imagen, se necesita una serie de archivos.

Uno de estos archivos será el de los pesos (.weight), el cual contiene los parámetros entrenados del modelo de YOLO. Otro es el archivo de configuración (.cfg) que contiene la estructura del modelo, es decir, cómo se organiza la red neuronal. Por último, un archivo que puede ser útil es el archivo de clases (.names), el cual contiene los nombres de las clases que el modelo puede detectar (persona, cama, perro, sofá, coche…)

En este proyecto, como se ha citado anteriormente, se hace uso de la versión 3 de YOLO por lo que se necesitarán los archivos de configuración y pesos propios de esta versión.



Al igual que con MediaPipe se realiza un script para familiarizarse con las funciones y resultados.

Figura 32: Resultados primeras pruebas YOLO. (Fuente: Elaboración propia)

Analizando los primeros resultados de las detecciones de YOLO (figura 32), se ve que para algunas detecciones se generan múltiples bounding boxes para el mismo sujeto. Este fenómeno es normal en modelos de detección de objetos y se soluciona mediante una técnica llamada NMS, de las siglas en inglés “Non-Maximum Suppression” (filtro de supresión de no máximos). Esta técnica analiza cuando los bounding boxes se solapan y elimina aquellos con una confianza de detección más baja hasta que solamente queda uno. Esto asegura que solo se conserve la detección más confiable para cada detección.

Una vez aplicado este algoritmo, se consiguen los siguientes resultados (figura 33):

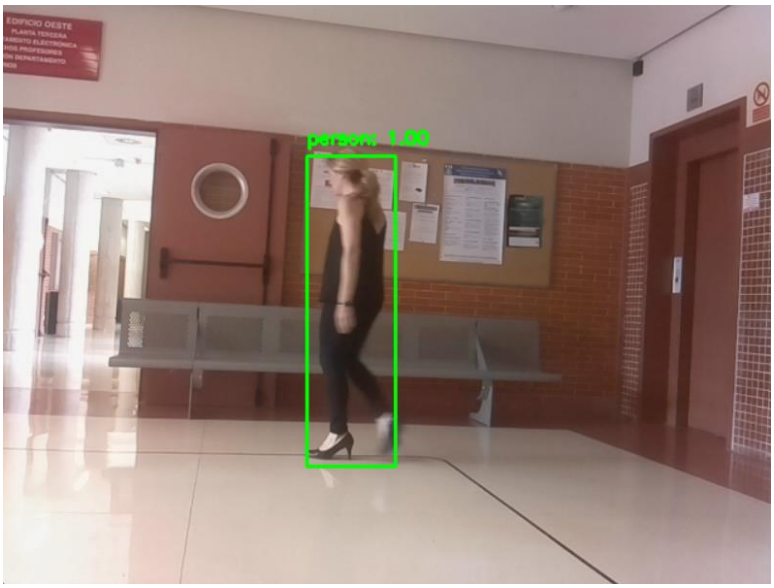


Figura 33: Resultado aplicar NMS a YOLO. (Fuente: Elaboración propia)

Los resultados de YOLO se muestran dibujando un bounding box alrededor de la detección. De este tipo de detección se puede sacar potencial, pues en función de las características del bounding box se puede hacer una distinción inicial entre las poses principales.

El modelo genera una serie de bounding boxes, cada uno con las siguientes características: Coordenadas (x, y) del vértice superior izquierdo del bounding box. Ancho (w) y alto (h) del bounding box. Confianza asociada a la detección (probabilidad de que el objeto sea una persona). Es importante filtrar las detecciones con baja probabilidad, en este caso 50 %.

A partir de los valores proporcionados por YOLO, se calculan las características necesarias para ayudar a clasificar las posturas:

1. **Medidas del bounding box**:
    - Altura y anchura, representan la extensión del bounding box y dan una idea de su tamaño.
2. **Relaciones espaciales:**
    - La relación altura/anchura ayudará a distinguir entre distintas posturas.
3. **Áreas**:
    - Áreas tanto de ocupación como de intersección entre bounding box ayudarán a diferenciar clases como “tumbado”.

Las características extraídas permiten analizar la forma y orientación de la persona en la imagen:

De pie y sentado: En esta postura, la altura del bounding box será significativamente mayor que su ancho, resultando evidente la distinción con el resto de las poses. El refinamiento y la distinción entre estas dos posturas se hará con el análisis de MediaPipe, pues, las características de YOLO no definen con robustez ninguna de las dos.

Caído: En esta postura, la anchura será mayor a la altura, incluso en casos complicados en los que la persona no esté orientada lateralmente a la cámara. Por ello, es un método robusto para diferenciar estas poses del resto.

Tumbado: Se intentará distinguir y filtrar de las poses clasificadas como “caída” aquellas en las que la persona esté “tumbada”. Para ello, mediante YOLO se detectará la clase sofá y la clase cama. Mediante la detección del bounding box de ambas clases y la posición del bounding box respecto al de la clase persona, se podrá realizar una primera distinción en casos en los que la persona se

encuentre tumbada y no caída. Con estas características se podrá definir qué personas se encuentran tumbadas cuando se detecte la clase sofá/cama en la imagen. Aquellas personas que se encuentren predominantemente en la parte inferior del sofá/cama se clasificarán como caídas, (correspondientes, p. ej. a caídas justo a los pies del sofá/cama). También se rechazan detecciones de la clase “tumbado” en aquellas ocasiones en las que la persona se encuentre predominantemente fuera del área de detección del sofá/cama.

Un ejemplo gráfico de qué detecciones se clasificarán como “tumbado” y cuáles no, sería el siguiente (figura 34):

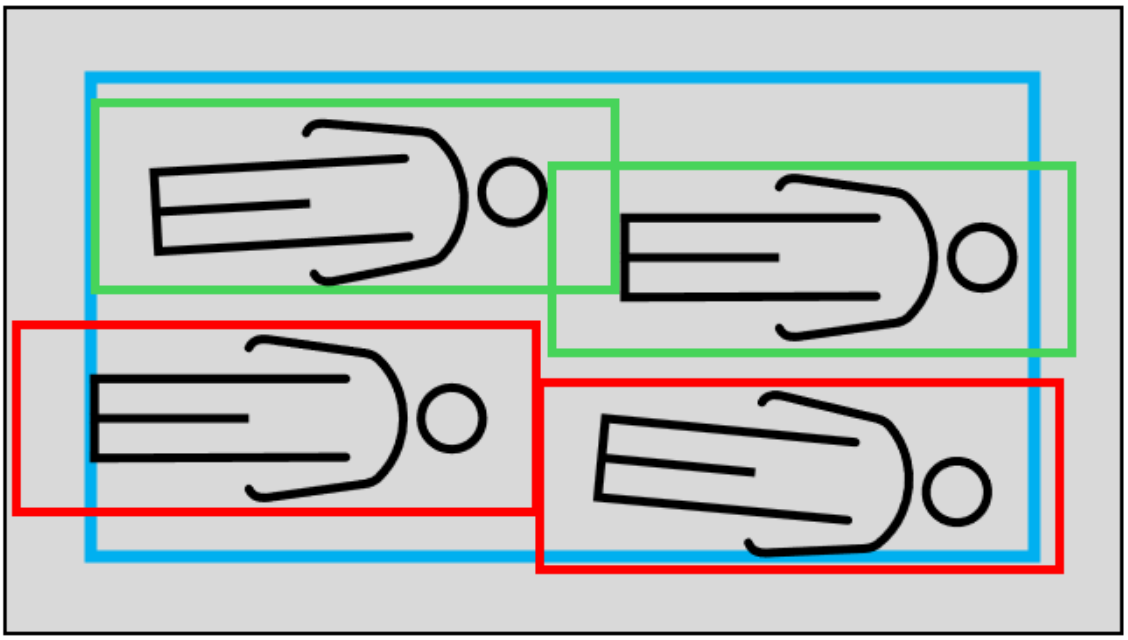


Figura 34: Distinción clase "tumbada" de clase "caída". (Fuente: Elaboración propia)

- En azul: Clase sofá o cama.
- En verde: Clase persona “tumbada”.
- En rojo: Clase persona “caída”.

A continuación, se muestra un listado completo sobre las características concretas extraídas de cada imagen (tabla 8).

Altura del bounding box de la clase persona
Anchura del bounding box de la clase persona
Altura del bounding box de la clase sofá/cama
Anchura del bounding box de la clase sofá/cama
Relación altura/anchura del bounding box de la persona
Área de la persona
Intersección persona con sofá/cama
Área de la mitad superior del sofá/cama
Área de la mitad inferior del sofá/cama
Intersección persona con mitad superior sofá/cama
Intersección persona con mitad inferior sofá/cama
Ocupación de la persona en el sofá/cama
Ocupación de la persona en la mitad superior sofá/cama
Ocupación de la persona en la mitad inferior sofá/cama

Tabla 8: Características extraídas por YOLO.

Usar estas características proporciona varias ventajas:

- Simples y robustas: El ancho y alto del bounding box son datos directos y confiables proporcionados por YOLO. Son fáciles de calcular y tienen un impacto directo en la clasificación de posturas.
- Interpretabilidad: La relación altura/anchura es intuitiva y permite tomar decisiones lógicas para clasificar las posturas.
- Eficiencia computacional: Los cálculos son rápidos, lo que es ideal para aplicaciones en tiempo real o con bajos recursos.

YOLO será útil también para detectar la presencia o no de persona en la imagen, pues si no hay evidencias de bounding box de la clase persona, esta no estará presente en la imagen o, si lo está, no es de una forma evidente.

4.3.2.4. Almacenamiento de características en fichero de texto.

El almacenamiento de las características extraídas es un paso importante para garantizar que puedan ser utilizadas posteriormente en los procesos de entrenamiento del modelo. Para este propósito, se desarrolla un programa en Python que automatiza el proceso de extracción y registro de las características. (“características YOLO a txt.py y características MediaPipe a txt.py”).

El siguiente diagrama muestra resumida la estructura básica del programa de almacenamiento de características (figura 35):

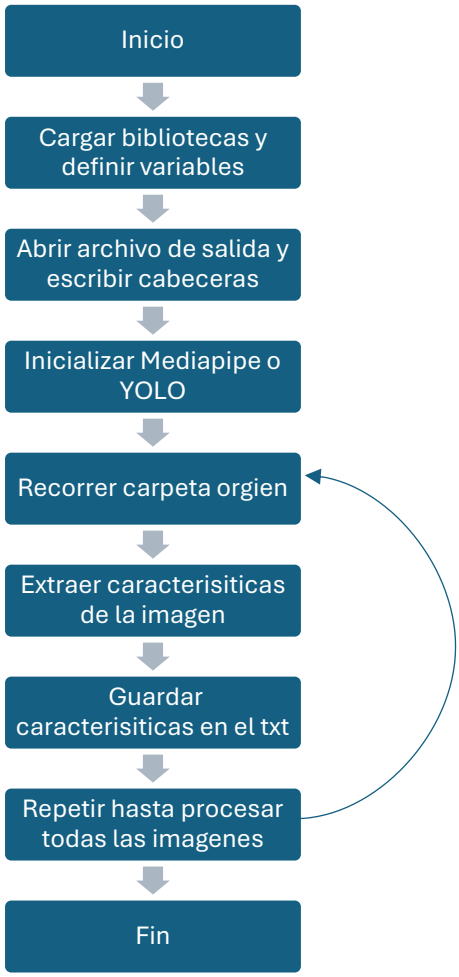


Figura 35: Diagrama de flujo del programa "características YOLO a txt.py y características MediaPipe a txt.py". (Fuente: Elaboración propia)

Se recorrerán las distintas carpetas seleccionadas, según se quiera analizar y extraer características de las imágenes, en el caso “tumbado”, “caída”, “sentado” o “de pie”.

- Para cada imagen, se aplica el algoritmo de MediaPipe o YOLO para detectar las posiciones de los puntos clave y de los bounding boxes.
- Se realizan los cálculos necesarios.
- Se extraen las características relevantes (distancias, ángulos, relaciones espaciales, alto, ancho, etc.).
- Los valores calculados se escriben en un fichero de texto.
- Cada línea del fichero corresponde a una imagen y cada columna a una característica extraída.

Estos pasos se realizan para cada carpeta de imágenes, consiguiendo así tener las características relevantes de cada imagen para las distintas poses. Será muy útil para poder realizar una distinción robusta entre posturas, analizando los resultados obtenidos.

El fichero txt obtenido (figura 36) para cada una de las carpetas y para cada uno de los métodos, contiene el nombre de las características extraídas (ángulos, distancias, relaciones…) y sus valores. Estos valores se recogen con todos los decimales disponibles para asegurar que se capturan todos los detalles posibles para el análisis y no se pierde información valiosa, especialmente cuando se trata de diferencias sutiles entre posiciones o movimientos.

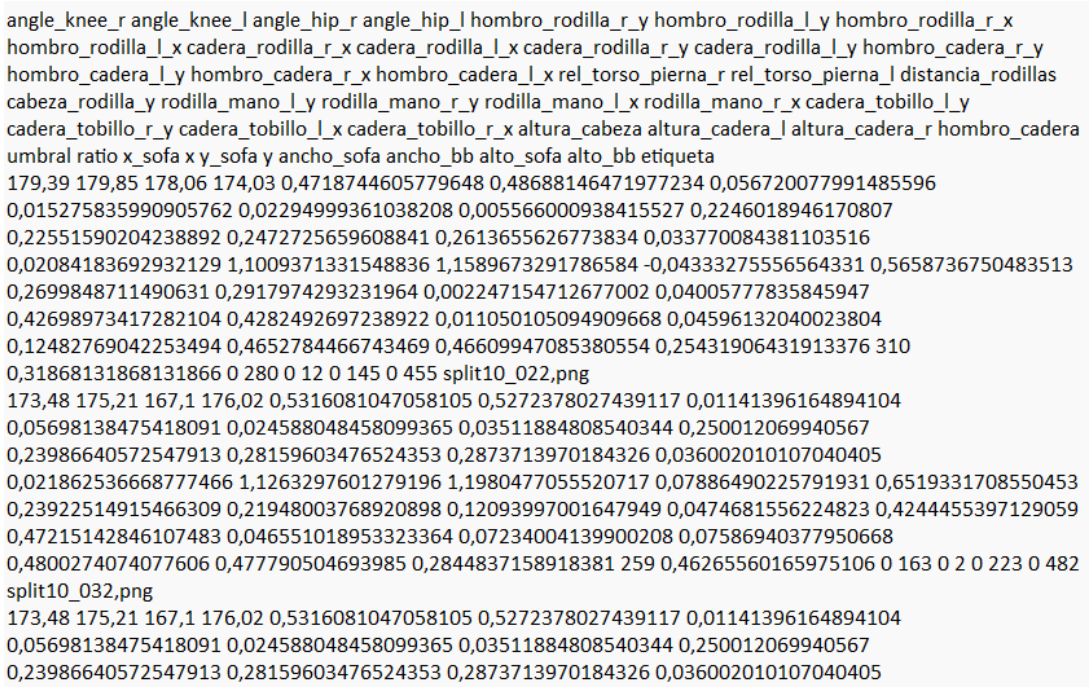

angle_knee_r angle_knee_l angle_hip_r angle_hip_l hombro_rodilla_r_y hombro_rodilla_l_y hombro_rodilla_r_x
hombro_rodilla_l_x cadera_rodilla_r_x cadera_rodilla_l_x cadera_rodilla_r_y cadera_rodilla_l_y hombro_cadera_r_y
hombro_cadera_l_y hombro_cadera_r_x hombro_cadera_l_x rel_torso_pierna_r rel_torso_pierna_l distancia_rodillas
cabeza_rodilla_y rodilla_mano_l_y rodilla_mano_r_y rodilla_mano_l_x rodilla_mano_r_x cadera_tobillo_l_y
cadera_tobillo_r_y cadera_tobillo_l_x cadera_tobillo_r_x altura_cabeza altura_cadera_l altura_cadera_r hombro_cadera
umbral ratio x_sofa x y_sofa y ancho_sofa ancho_bb alto_sofa alto_bb etiqueta
179,39 179,85 178,06 174,03 0,4718744605779648 0,48688146471977234 0,056720077991485596
0,015275835990905762 0,02294999361038208 0,005566000938415527 0,2246018946170807
0,22551590204238892 0,2472725659608841 0,2613655626773834 0,033770084381103516
0,02084183692932129 1,1009371331548836 1,1589673291786584 -0,04333275556564331 0,5658736750483513
0,2699848711490631 0,2917974293231964 0,002247154712677002 0,04005777835845947
0,42698973417282104 0,4282492697238922 0,011050105094909668 0,04596132040023804
0,12482769042253494 0,4652784466743469 0,46609947085380554 0,25431906431913376 310
0,31868131868131866 0 280 0 12 0 145 0 455 split10_022,png
173,48 175,21 167,1 176,02 0,5316081047058105 0,5272378027439117 0,01141396164894104
0,05698138475418091 0,024588048458099365 0,03511884808540344 0,250012069940567
0,23986640572547913 0,28159603476524353 0,2873713970184326 0,036002010107040405
0,021862536668777466 1,1263297601279196 1,1980477055520717 0,07886490225791931 0,6519331708550453
0,23922514915466309 0,21948003768920898 0,12093997001647949 0,04746815562248 23 0,4244455397129059
0,47215142846107483 0,046551018953323364 0,07234004139900208 0,07586940377950668
0,4800274074077606 0,477790504693985 0,2844837158918381 259 0,46265560165975106 0 163 0 2 0 223 0 482
split10_032,png
173,48 175,21 167,1 176,02 0,5316081047058105 0,5272378027439117 0,01141396164894104
0,05698138475418091 0,024588048458099365 0,03511884808540344 0,250012069940567
0,23986640572547913 0,28159603476524353 0,2873713970184326 0,036002010107040405

Figura 36: Fichero de texto con características extraídas de la detección de MediaPipe y YOLO. (Fuente: Elaboración propia)

Estos ficheros de texto muestran los datos extraídos, pero para poder realizar un buen análisis con los datos es necesario organizarlos en forma de tabla. En este caso se ha utilizado Excel al ser una herramienta conocida y porque facilita la visualización, análisis y manipulación de la información de manera clara y ordenada. También es importante la compatibilidad que ofrece Excel con el análisis estadístico, pues será determinante para el desarrollo de umbrales y selección de características.

4.3.2.5. Análisis de datos.

Primero se introducen los datos en una tabla Excel desde los ficheros de texto que contienen las características extraídas. Se organizan en filas para cada imagen y en columnas para cada característica, añadiendo una columna más en la que se etiquetarán los datos de la siguiente manera, según provengan de la carpeta de pie (0), sentado (1), caída (2), desconocido (3) y tumbado (4).

Ordenar los datos en la tabla Excel es realmente útil, pues, se puede acceder al dato o al conjunto de datos deseados de manera rápida y eficaz, incluso si es necesario reemplazar algún conjunto de datos por alguna modificación en el método de detección se puede hacer sin demasiado problema gracias a este orden.

Una pequeña representación de cómo quedaría organizada la tabla Excel con los datos de entrenamiento sería la siguiente (tabla 9):

Imágenes	**Característica 1**	**…**	**Característica N**	**Etiqueta**
Imagen_1_tumbado	…(valores)	…	…(valores)	4
Imagen_n_tumbado	…(valores)	…	…(valores)	4
Imagen_1_desconocido	…(valores)	…	…(valores)	3
Imagen_n_desconocido	…(valores)	…	…(valores)	3
Imagen_1_caída	…(valores)	…	…(valores)	2
Imagen_n_caída	…(valores)	…	…(valores)	2
Imagen_1_sentado	…(valores)	…	…(valores)	1
Imagen_n_sentado	…(valores)	…	…(valores)	1
Imagen_1_depie	…(valores)	…	…(valores)	0
Imagen_n_depie	…(valores)	…	…(valores)	0

Tabla 9: Representación de la tabla de características extraídas.

Con los datos ya organizados se realiza cálculos para obtener información estadística que facilite la comprensión del comportamiento de las características para cada clase. En este caso, se ha optado por extraer para cada conjunto de imágenes separado por clases, calcular los siguientes valores:

- **Promedio:** pues, ayudará a identificar el comportamiento “típico” de la característica para cada clase y reforzar la diferenciación entre estas.
- **Valor máximo:** identifica los casos extremos o límites superiores de cada característica para cada clase.
- **Valor mínimo:** identifica los límites inferiores de cada característica para cada clase.
- **Desviación:** da una idea sobre lo dispersos que son los datos en una clase, identificando si son muy variados o no.
- **Media + desviación:** identifica el límite superior típico de los datos y puede ayudar a establecer un umbral máximo de referencia para cada característica.
- **Media – desviación**: identifica el límite inferior típico de los datos y puede ayudar a establecer un umbral mínimo de referencia para cada característica.

Con esta extracción de métricas para cada característica se puede llevar a cabo una diferenciación entre clases.

Se comparan los promedios, máximos y mínimos para ver qué características son significativamente diferentes entre clases y se usan los valores de media ± desviación para identificar rangos típicos de cada clase y establecer unos umbrales iniciales.

Si una clase tiene una desviación estándar alta, los datos son más variados, lo que podría indicar que esa clase es más difícil de definir con ciertos umbrales. Las clases con desviaciones estándar bajas son más consistentes, lo que puede hacerlas más fáciles de identificar. Por otro lado, los valores que superan la media + desviación o caen por debajo de la media - desviación pueden considerarse fuera de rango, datos extraños u outliners por lo que es necesario analizarlos para ver si tienen sentido, son casos extremos o son errores en las detecciones.

Analizando estas métricas y fijándose bien en si una característica tiene un rango de valores no superpuesto entre clases (por ejemplo, alturas máximas en "de pie" son mayores que en "tumbado"), se puede usar esta métrica para diseñar unas reglas de clasificación inicial. Esto junto a identificar los patrones en los datos para cada clase incluso visualizando mediante gráficos para destacar las características más representativas de cada clase, ayudará a conseguir una clasificación inicial mediante ciertos umbrales obtenidos del análisis.

Un ejemplo de cómo se trabaja analizando las métricas obtenidas serían los siguientes casos:

1. Media de la distancia desde el hombro a la rodilla detectada por MediaPipe medida en el eje vertical para cada clase (tabla 10 y figura 37).

Clase	Media distancia hombro – rodilla en y
Tumbado	0,03781
Caída	0,00652
Sentado	0,18219
De pie	0,34284

Tabla 10: Media de la distancia hombro - rodilla en el eje y por clases.

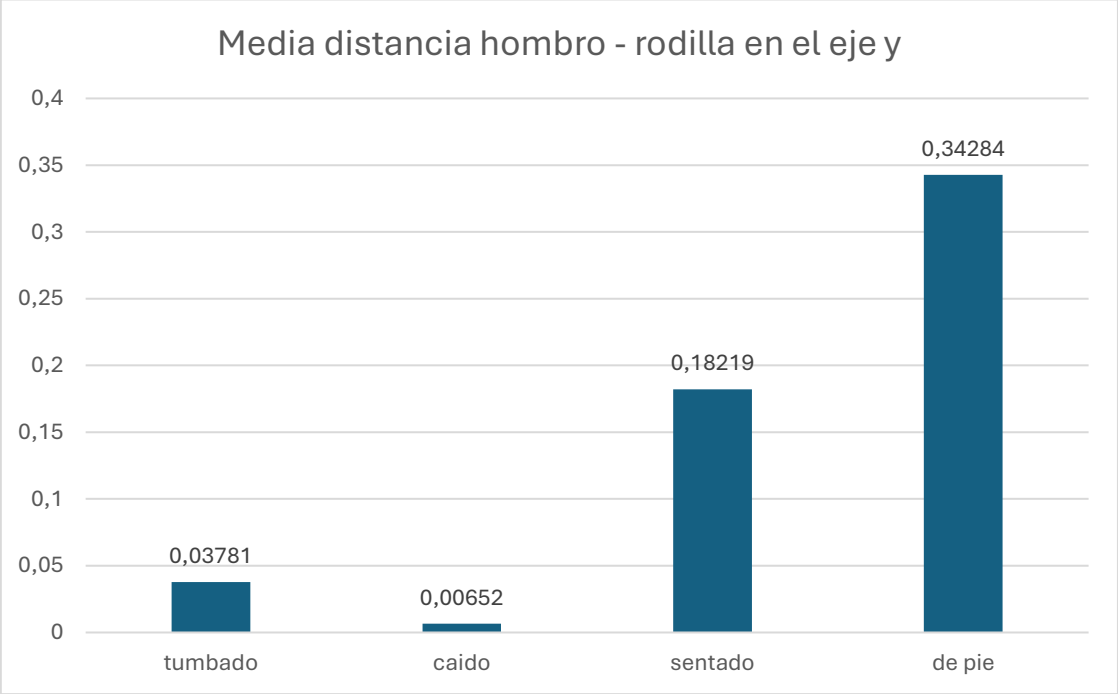


Figura 37: Gráfico de la media de la distancia hombro - rodilla en el eje y por clases. (Fuente: Elaboración propia)

Analizando un poco esta métrica, se ve que valores altos por encima de 0,05 son muy representativos de las clases “sentado” y “de pie”, y valores inferiores se quedan más reservados para las poses tumbadas y caídas. Al igual, se aprecia una diferencia entre las poses “sentado” y “de pie”, donde la media de la clase “sentado” (0,18) es inferior a la de la clase “de pie” (0,34), aproximadamente la mitad.

2. Valores máximos y mínimos de la resta del alto menos el ancho del bounding box de la persona detectada por YOLO para cada clase (tabla 11 y figura 38).

Clase	Máximo alto_bb - ancho_bb	Mínimo alto_bb - ancho_bb
Tumbado	11	-296
Caída	141	-493
Sentado	227	19
De pie	334	40

Tabla 11: Máximos y mínimos del alto menos ancho del bounding box de la persona detectada por clase.

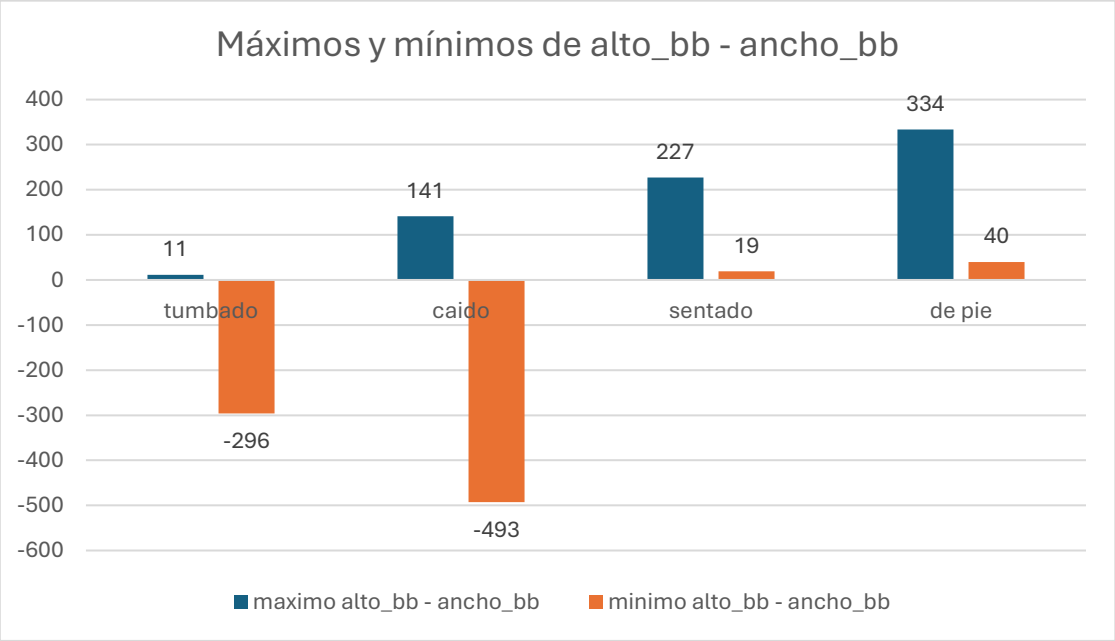


Figura 38: Gráfico de máximos y mínimos del alto menos ancho del bounding box de la persona por clase. (Fuente: Elaboración propia)

En este caso se aprecia una notable diferencia entre las clases, pues son muy representativos los valores negativos para las poses “tumbado” y “caída”, dejando reservados casi exclusivamente los valores positivos para las clases “sentado” y “de pie”.

Este paso se realiza para varias métricas de características de cada pose, pues, resulta interesante conocer desde que valores pueden o no partir los umbrales de distinción para la clasificación final.

4.3.2.6. Entrenamiento del SVM.

Con las características extraídas de todas las imágenes de entrenamiento y ordenadas en una tabla Excel se plantea la opción de entrar un algoritmo SVM que sea capaz de buscar el hiperplano que separa los puntos de datos de las diferentes clases de manera que el margen entre las clases sea el menor posible. Esta clasificación mediante el SVM ayudará a elegir mejores umbrales de decisión para la clasificación "manual" y será otro método de análisis que devolverá un resultado a mayores. Así se podrá contar con dos resultados, el derivado de la clasificación puramente empírica y el del método automático del modelo entrenado.

Lo primero es organizar el archivo Excel de manera que pueda ser convertido en un archivo CSV (command separated values) el cual pueda leer el SVM. El CSV es un formato de archivos utilizado para almacenar datos en forma de texto plano, donde cada valor está separado por una coma.

Cada fila dentro del archivo representa una entrada de datos nueva y cada columna separada por comas u otros delimitadores como punto y coma, representan una característica o un valor nuevo. Generalmente, la primera fila contiene los nombres de las columnas (características) y la última de las columnas contiene la etiqueta del tipo de dato, en este caso la etiqueta 4,3,2,1,0 correspondiente a las poses "tumbado", "desconocido", "caída", "sentado" y "de pie", respectivamente.

Una vez se tiene el archivo ordenado de manera que cada columna sea una característica, cada fila una nueva imagen y la última columna contenga la etiqueta de la clase correspondiente, se puede pasar a entrenar el modelo SVM con los datos extraídos por MediaPipe y YOLO.

Para entrenar y crear el SVM es necesario realizar un código ("SVM con datos.py") en el que se pueda pasar el archivo con los datos de entreno y configurar la división entre conjuntos de entrenamiento y prueba.

Para cada SVM creado será necesario importar las librerías necesarias, leer el archivo CSV y cargarlo en un "*dataframe*". Este *dataframe* es una tabla de datos bidimensional en la que cada fila representa, en este caso, una imagen y cada columna representa una característica. Se realiza un preprocesamiento del archivo de datos y se separan las características y etiquetas, pues el SVM necesita diferenciar entre los datos de entrada y la respuesta esperada.

```
X = df.iloc[:, :-1].values  # Todas las columnas excepto la última
y = df.iloc[:, -1].values   # Solo la última columna (etiquetas)
class_names=df.iloc[:,-1].unique()
```

- X: Se obtienen todas las columnas del *dataframe,* excepto la última, que se usan como características (entrada para el modelo).

- y: Se obtiene solo la última columna del *dataframe*, que se utiliza como etiquetas (salida para el modelo).
- class_names: Se almacenan los nombres únicos de las clases de la última columna para usarlos más adelante en la matriz de confusión.

Se dividen los datos en conjuntos de entrenamiento y prueba, pues el objetivo es poder evaluar de manera objetiva el rendimiento del modelo para no entrenar con todos los datos y luego evaluar con los mismos, pues se estaría cometiendo un error de sobreajuste. Así, el conjunto de entrenamiento se utiliza para ajustar los parámetros del modelo y el conjunto de prueba se utiliza para evaluar el modelo una vez que ha sido entrenado. Con esta separación se puede medir la capacidad del modelo para generalizar, pues un buen rendimiento del modelo sobre el conjunto de prueba refleja que ha aprendido de manera efectiva las relaciones subyacentes en los datos y posee una buena capacidad de generalización.

La división de los datos se consigue de la siguiente forma:

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

En este caso, el tamaño seleccionado para el conjunto de test es del 30% de los datos totales. El uso del parámetro *random_state* nos ayuda a replicar siempre el mismo ensayo controlando la aleatoriedad.

Se crea un SVM con un tipo de *kernel* en este caso “lineal”. El *kernel*, es la función matemática que transforma los datos para que el SVM pueda clasificarlos y el tipo “lineal” pues los datos son linealmente separables.

Una vez obtenido, el modelo se evalúa y se guarda, obteniendo las predicciones, midiendo la precisión del modelo y mostrando un reporte detallado con las siguientes métricas:

- Precisión: muestra qué porcentaje de las predicciones de una clase son correctas.
- *Recall*: muestra qué porcentaje de los ejemplos reales de una clase son correctamente clasificados.
- *F1-score*: muestra el promedio armónico de la precisión y el *recall*, es útil, pues, los datos no están balanceados.
- *Support:* muestra el número de ejemplos reales de cada clase.

```
y_pred = svm.predict(X_test)
print(f"Precisión del modelo: {accuracy_score(y_test, y_pred):.4f}")
print("Reporte de clasificación:")
print(classification_report(y_test, y_pred))
```

Como en el modelo final se pueden dar casos en los que MediaPipe detecte *landmarks* y YOLO no encuentre la clase persona en la imagen o viceversa, se entrenan tres SVM, cada uno con un conjunto de datos que contiene distintas características:

- SVM global: contiene todas las características, las extraídas por MediaPipe y por YOLO, utilizado para cuando en la imagen a analizar ambos modelos sean capaces de detectar.

- SVM MediaPipe: contiene únicamente las características extraídas de los *landmarks* de MediaPipe, pues será útil para el caso en el que en el modelo final YOLO no sea capaz de detectar en la imagen la clase persona y MediaPipe si sea capaz de detectar los puntos clave.

- SVM YOLO: caso contrario al anterior, contiene únicamente las características extraídas del bounding box de YOLO para cuando este sea capaz de detectar persona y MediaPipe no.

Con estos tres SVM se eliminan falsas predicciones en el caso de que alguno de los métodos no aporte los datos correspondientes para la clasificación.

Para cada SVM se obtiene un gráfico denominado matriz de confusión. Es una herramienta fundamental en el análisis de desempeño de los modelos de clasificación, pues proporciona una visión detallada de las predicciones realizadas por el modelo en comparación con las clases verdaderas o reales. Esta matriz es particularmente útil cuando el modelo tiene múltiples clases como este, ya que muestra no solo cuántas predicciones fueron correctas, sino también en qué clases se cometieron errores.

La matriz de confusión es una tabla que compara las predicciones del modelo contra las etiquetas reales (verdaderas) de los datos. En una matriz de confusión, cada fila representa las etiquetas reales y cada columna representa las etiquetas predichas por el modelo.

A continuación, en la tabla 12, se muestra un ejemplo de matriz de confusión básica:

	Predicción positiva	**Predicción negativa**
Real positiva	TP	FN
Real negativa	FP	TN

Tabla 12: Estructura básica de una matriz de confusión.

- (TP): Casos en los que el modelo predijo correctamente la clase positiva (realmente positivo y predicho positivo).

- (TN): Casos en los que el modelo predijo correctamente la clase negativa (realmente negativo y predicho negativo).
- (FP): Casos en los que el modelo predijo incorrectamente la clase positiva (realmente negativo pero predicho positivo).
- (FN): Casos en los que el modelo predijo incorrectamente la clase negativa (realmente positivo pero predicho negativo).

La matriz de confusión obtenida para cada SVM nos da una idea sobre lo bien que generaliza y lo bien que realiza la clasificación autónoma.

Del entrenamiento y creación del SVM se ha diseñado un procedimiento que facilita mucho elegir qué características son relevantes para la clasificación "manual" y qué umbrales se deberían elegir para cada una de estas características en función de la clase. Para ello, se han creado una serie de gráficos que se obtienen graficando para cada característica (columna de X) un histograma que muestra cómo se distribuyen los valores de esa característica para cada clase (y). Esto, como se ha dicho anteriormente, ayuda a entender las características de los datos y cómo se distribuyen entre las diferentes clases de manera visual y clara.

La figura 39 muestra un ejemplo del histograma diseñado a partir de una característica para todas las clases:

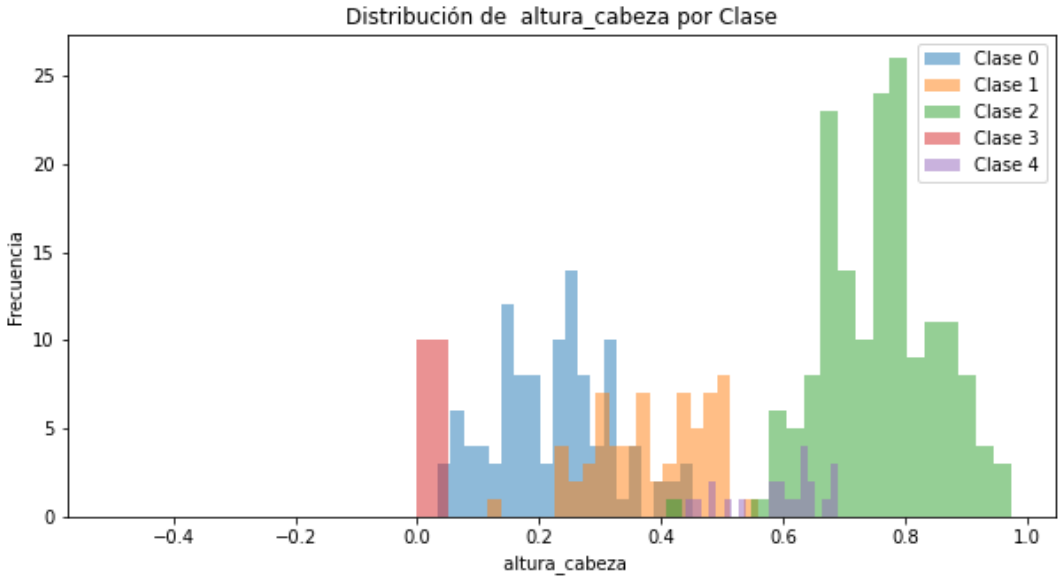


Figura 39: Distribución de altura de la cabeza por clase extraída del SVM. (Fuente: Elaboración propia)

El eje x muestra los valores que alcanza la variable y en el eje y la frecuencia con la que se alcanza cada valor.

4.3.2.7. Selección de características relevantes.

Para la clasificación que se quiere realizar, es necesario filtrar qué características de todas las disponibles por MediaPipe y YOLO son útiles y cuáles no.

Para ello se trabaja juntamente, por un lado, con las métricas extraídas de la tabla Excel, las cuales son más útiles para fijar un valor más concreto de los umbrales de clasificación y, por otro lado, con las gráficas extraídas del entreno del SVM que resultan útiles para de manera visual y rápida decidir qué características son relevantes para cada clase y como dependiendo de la clase la característica toma unos valores u otros.

Hay tres tipos predominantes de distribuciones en las características representadas en los gráficos:

1. En las que se diferencian claramente dos o más clases directamente.
2. En las que una de las clases se diferencia del resto, pero no de manera directa o clara.
3. En las que no se diferencia de manera representativa la característica para ninguna de las clases.

A continuación, se detalla con unos ejemplos la metodología seguida para todas las características, seleccionando aquellas que resultan interesantes y descartando aquellas que no lo son.

- **Tipo 1: Diferencia evidente y clara entre dos o más clases.**

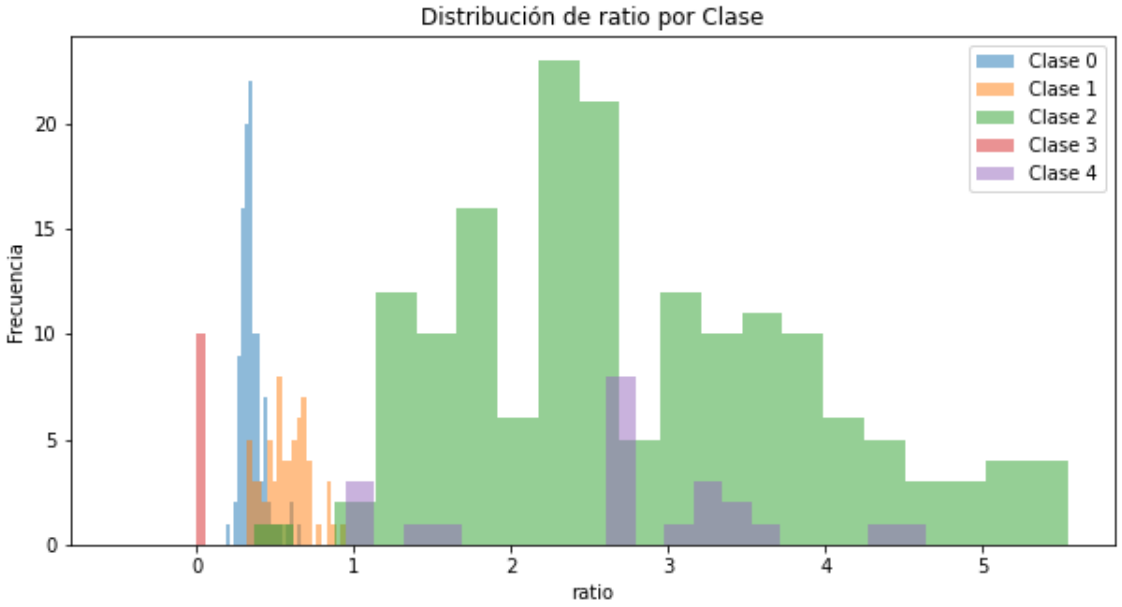


Figura 40: Distribución característica tipo 1. (Fuente: Elaboración propia)

En el gráfico de esta característica (figura 40) se aprecia de manera clara una diferencia evidente entre las clases 0, 1 y 2, las tres poses principales y necesarias a la hora de clasificar, pues es visible que prácticamente donde acaban los valores de una de ellas empieza otra.

Analizando más en profundidad los valores de las métricas (tabla 13) para elegir de manera correcta los umbrales que se seleccionarán en la clasificación final, se aprecia que las diferencias evidentes entre estas quedan reflejadas en el gráfico, separando con cierta precisión las tres clases.

	DE PIE	SENTADO	CAÍDA
Media	0,35	0,57	2,8
Máximo	0,66	0,95	5,54
Mínimo	0,18	0,32	0,36
Desviación	0,07	0,14	1,09

Tabla 13: Métricas de una característica tipo 1.

Se tomarán así pues, valores, por ejemplo superiores a 1 para clasificar la pose caída, valores inferiores a 0,5 para clasificar la pose “de pie”, y valores intermedios para clasificar la pose “sentado”.

- **Tipo 2: Diferencia no evidente de alguna de las clases.**

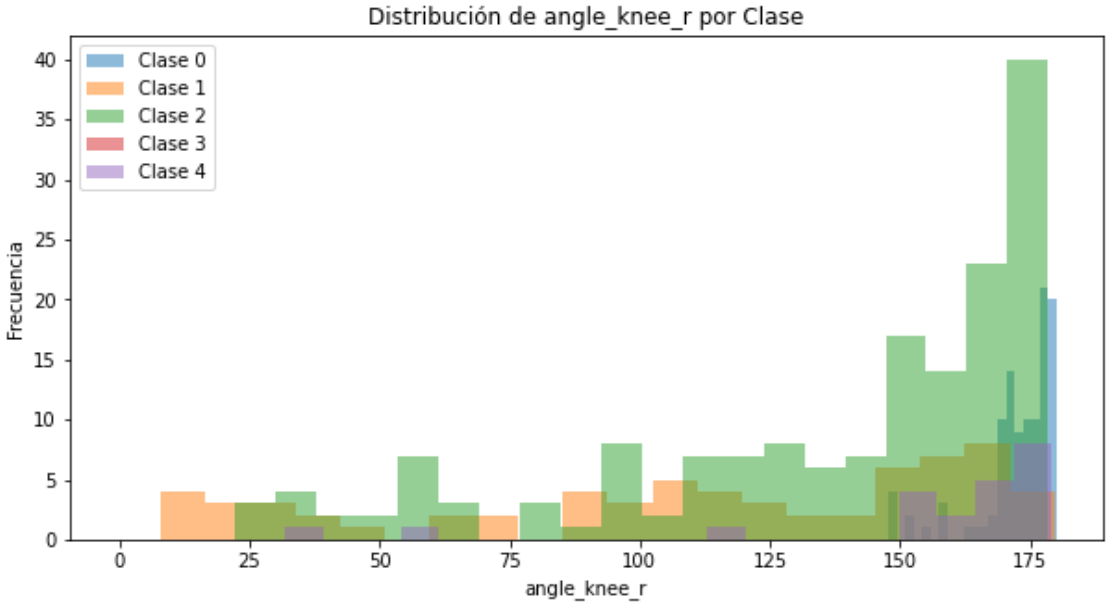


Figura 41: Distribución característica tipo 2. (Fuente: Elaboración propia)

En esta ocasión, en el histograma (figura 41), se aprecia que esta característica toma valores muy variados en todas las clases, pero se ve claramente que la clase 0 (de pie) no toma valores inferiores a 140º-145º en ninguno de los casos

(frecuencia de casos en el eje y), agrupándose los casos más comunes incluso alrededor de valores todavía más altos cercanos a la extensión completa (180º). Esta característica, por lo tanto, se elegiría como representativa de la clase 0 para valores superiores a 140-145.

Para afinar más el valor que posteriormente se utilizará como umbral, hay que fijarse en las métricas extraídas para esta característica (tabla 14), por ejemplo, en los valores mínimos y máximos, en la media y en la dispersión para la clase 0 (de pie) pues es de la que hemos determinado que es representativa.

DE PIE	
Valor máximo	179,97
Valor mínimo	147,86
Media	172,52
Desviación	7,45

Tabla 14: Métricas de una característica tipo 2.

Estas métricas muestran con más precisión que se estaba en lo cierto al determinar un valor de umbral cercano a 140º-145º y que la dispersión de esta característica tiene un valor muy pequeño, por lo que no se va a salir "normalmente" de estos valores típicos.

- **Tipo 3: No hay suficiente evidencia representativa en ninguna de las clases.**

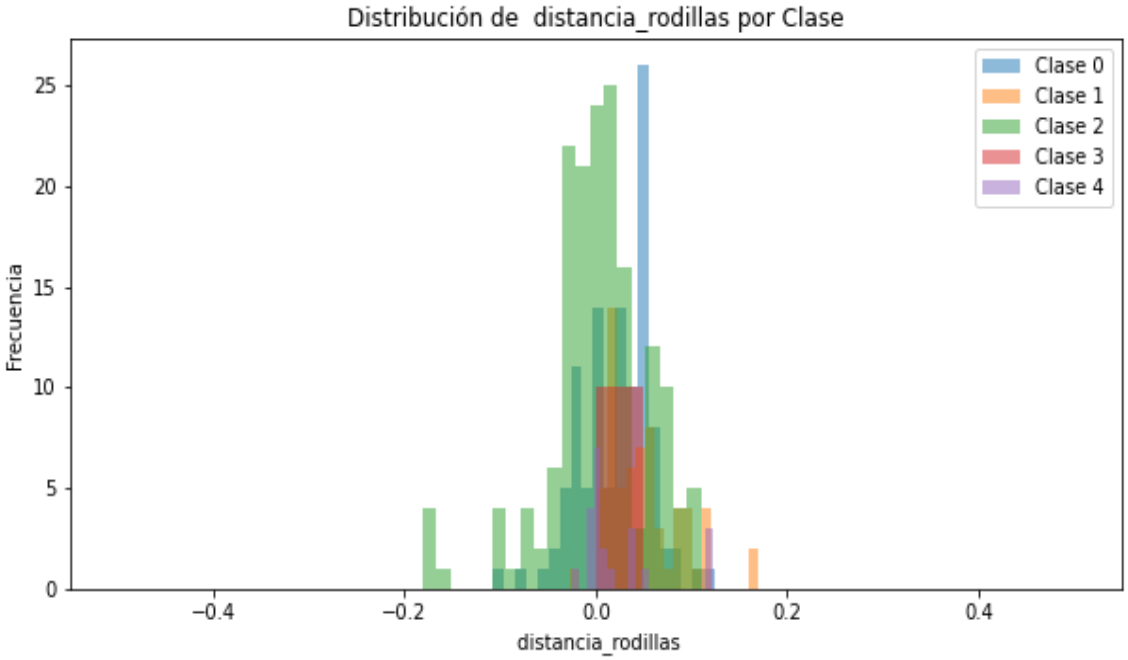


Figura 42: Distribución característica tipo 3. (Fuente: Elaboración propia)

Por último, analizando el gráfico (figura 42), se ve claramente que esta característica no es representativa de ninguna clase, los valores están muy superpuestos y alcanzando valores muy similares para todas ellas.

Este sería un ejemplo de descarte de característica, pues, no es útil para la clasificación final. Aun descartándola de primera vista, se analizan las métricas extraídas para esta característica (tabla 15).

	Media	**Máximo**	**Mínimo**	**Desviación**
DE PIE	0,021	0,124	0,107	0,036
SENTADO	0,051	0,169	-0,028	0,039
CAÍDA	0,003	0,110	-0,181	0,053

Tabla 15: Métricas de una característica tipo 3.

Se aprecia claramente ya con el análisis final de las métricas que esta característica no es suficientemente representativa de ninguna de las clases, pues ninguna de ellas destaca lo suficiente por encima de las demás, como ya se veía en el gráfico.

Esta metodología se sigue para todas las características extraídas, seleccionando aquellas interesantes y representativas y analizando las métricas para fijar unos valores de umbral. La unión de todas las características seleccionadas (tabla 16) de la manera correcta fijará las sentencias de decisión y clasificación de poses en el código final.

Ángulo rodillas	x>140: de pie
Ángulo caderas	x>140: de pie
Distancia vertical hombro-rodillas	x<0,1: caída, x<0: caída
Distancia horizontal hombro-rodillas	x<0,1: de pie, x>0,1: caída
Distancia vertical hombro-caderas	x<0,1: caída
Distancia horizontal hombro-caderas	x>0,05: caída
Distancia vertical cabeza-rodilla	x<0,14
Relación torso-pierna	1<x<2: de pie
Distancia vertical cadera-tobillos	x<0: caída
Altura de la cabeza	x>0,6: caída
Diferencia alto-ancho del bounding box	x<15: caída
Relación ancho/alto del bounding box	x<0,5: de pie
Ocupación persona	x>0,5
Ocupación mitad superior del sofá/cama	x>(ocupación persona/2): tumbado

Tabla 16: Características y umbrales seleccionados.

4.3.3. Arquitectura general del sistema.

Así pues, la estructura final del programa desarrollado se muestra a continuación con el diagrama de bloques de la figura 43:

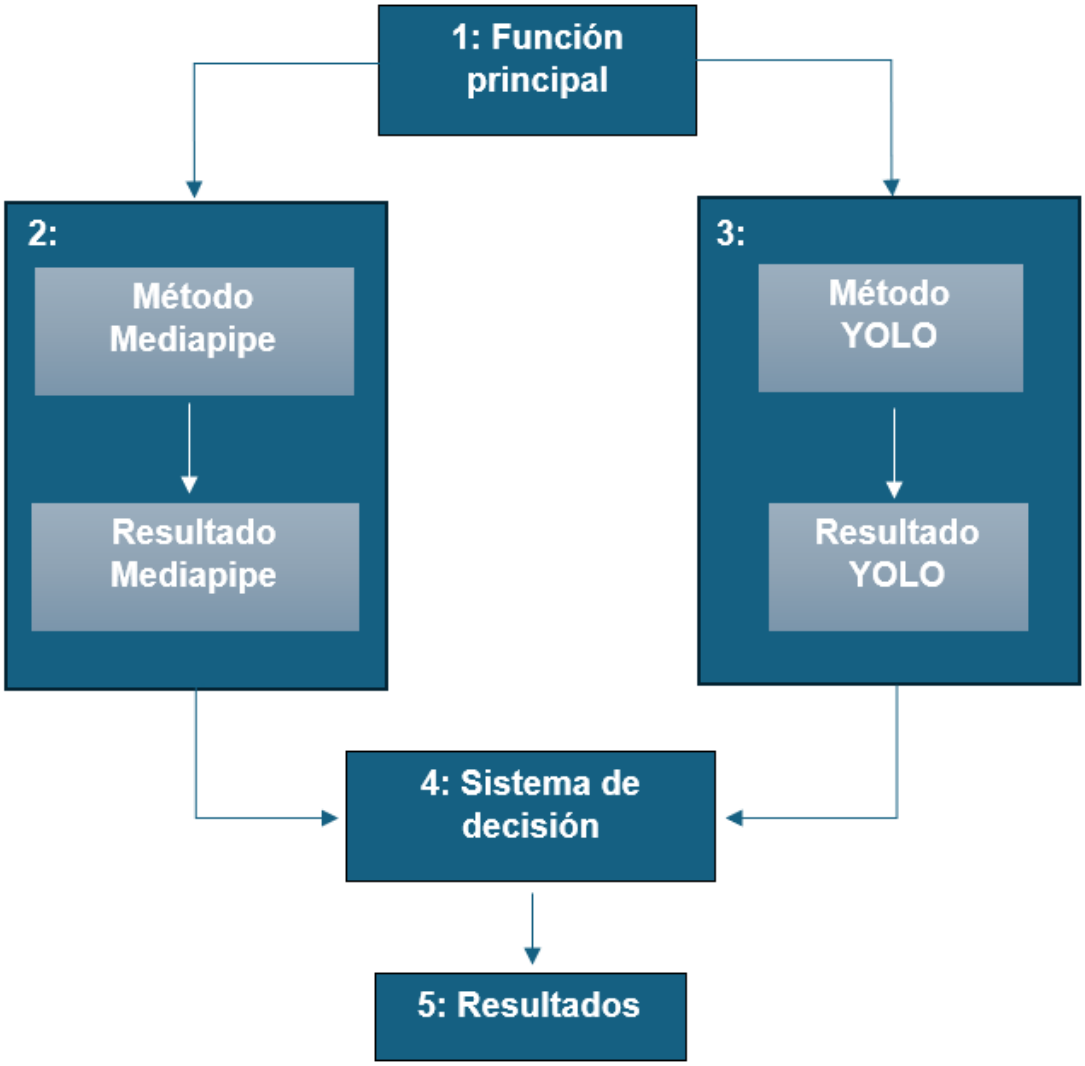


Figura 43: Diagrama de bloques del modelo desarrollado. (Fuente: Elaboración propia)

El código se ha dividido en diferentes partes que se explicarán a continuación con algo más de detalle.

Bloque 1:

- Se importan las librerías necesarias para la correcta ejecución del código.
- Se crea la función principal a la que se le pasa como argumento la imagen a analizar. Esta imagen se lee desde la ubicación del archivo donde se encuentra y se verifica si existe.
- También se cargan los tres modelos de SVM preentrenados.

Bloque 2:

- Se convierte la imagen a RGB, formato necesario para trabajar con ella en MediaPipe.
- Se inicializa MediaPipe Pose y la detección de *landmarks*.
- Si la detección es válida y se encuentran *landmarks* con la suficiente confianza:
    - Se extraen las coordenadas de todos los puntos clave que se consideran necesarios.
    - Se realizan los cálculos como ángulos de articulaciones, relaciones geométricas y distancias entre diferentes puntos del cuerpo.
    - Se pasan las características extraídas al SVM global y al SVM de MediaPipe.
    - Se realiza la predicción del SVM de MediaPipe.
    - Se clasifica la postura y se asigna una etiqueta (“de pie”, “sentado” o “caída”) según las reglas y umbrales de decisión propuestos. Esta clasificación se guarda en la variable “resultado MediaPipe”.
- Si no se detectan *landmarks* se retorna “desconocido”.

Bloque 3:

- Se inicializa el modelo de YOLO para detectar objetos en la imagen.
- Si se ha detectado la clase persona en la imagen con la suficiente confianza:
    - Se extraen los bounding boxes y se calculan las características asociadas a ellos como la altura o la relación alto/ancho…
    - Se analiza la posibilidad de clasificar la pose como “tumbado” mediante la intersección y ocupación con la clase sofá/cama.
    - Se pasan las características extraídas al SVM global y al SVM de YOLO.
    - Se realiza la predicción del SVM de YOLO.
    - Se clasifica la postura y se le asigna una etiqueta (“de pie”, “sentado”, “tumbado” o “caída”) según las reglas y umbrales propuestos. Esta clasificación se guarda en la variable “resultado YOLO”.
- Si no se detecta la clase persona con la suficiente confianza, se retorna “desconocido”.

Bloque 4:

- Según los resultados obtenidos y siguiendo el sistema de decisión diseñado (figura 42) se obtiene un resultado final para la clasificación de la postura presente en la imagen. Este resultado solo podrá ser uno de entre las 5 clases (“de pie”, “sentado”, “tumbado”, “caída” o “desconocido”).

Bloque 5:

- Se muestran los siguientes resultados:
  - El del método empírico resultante del sistema de decisión entre la clasificación realizada en el método basado en las características extraídas de los *landmarks* de MediaPipe y el método basado en las características extraídas de los bounding boxes de YOLO.
  - La predicción del SVM, mostrando si se ha realizado por el SVM global (ambos métodos han detectado o no persona en la imagen), o por el SVM de MediaPipe (YOLO no detecta persona), o por el SVM de YOLO (MediaPipe no detecta persona).
  - También se muestran las imágenes con los *landmarks* y bounding boxes dibujados.

La arquitectura general del sistema desarrollado se resume en el siguiente pseudocódigo:

```
Inicio del programa
Librerías necesarias: OpenCV, NumPy, MediaPipe, Sys, Joblib

Función analizar pose(imagen):
    Cargar imagen
    Cargar modelos SVM y escaladores para:
        - Modelo general
        - Modelo MediaPipe
        - Modelo YOLO
```

```
    Función método MediaPipe(imagen):
        Inicializar MediaPipe Pose
        Detectar poses
        Si se detectan landmarks:
            Verificar visibilidad
            Función calculo ángulos (a, b, c):
                Calcular ángulo entre tres puntos
            Obtener coordenadas de:
```

```
17:                 - Caderas, rodillas, tobillos
18:                 - Hombros, cabeza, muñecas
19:             Calcular características:
20:                 - Ángulos de rodilla y cadera
21:                 - Distancias relativas entre puntos clave
22:                 - Relaciones de torso y piernas

23:             Normalizar características con el escalador
24:             Predecir postura con el SVM MediaPipe

25:             Clasificar postura según umbrales de:
26:                 - Ángulos de cadera y rodilla
27:                 - Proporciones del torso y piernas
28:                 - Altura de la cabeza respecto a las rodillas

29:             Dibujar landmarks sobre la imagen

30:             Retornar:
31:                 - Clasificación MediaPipe
32:                 - Imagen con landmarks
33:                 - Características extraídas
34:                 - Predicción SVM MediaPipe

35:         Si no se detectan landmarks:
36:             Retornar: "desconocido" con valores por defecto
```

```
37:     Función método yolo(imagen):

38:         Cargar modelo YOLO
39:         Obtener etiquetas de clases
40:         Procesar imagen para detectar objetos

41:         Inicializar listas necesarias

42:         Recorrer detecciones:
43:            - Si detección es persona, almacenar sus coordenadas
44:            - Si detección es sofá/cama, almacenar sus coordenadas

45:         Supresión de no máximos (NMS)

46:         Si se detecta una persona:
47:             Calcular características:
48:                 - Altura menos ancho de la caja (umbral)
49:                 - Relación ancho/altura (ratio)
50:                 - Posición relativa respecto a muebles detectados

51:             Si hay un sofá/cama detectado:
52:                 Calcular intersección con la persona
53:                 Determinar si la persona está tumbada

54:             Normalizar características con el escalador
55:             Predecir postura con el SVM YOLO
```

```
56:             Clasificar postura en:                                    3
57:                 - "Tumbado" si la ocupación del sofá/cama es alta
58:                 - "Caída" si el umbral es bajo
59:                 - "Sentado" si la relación ancho/altura es alta

60:             Dibujar boundingboxes sobre la imagen

61:             Retornar:
62:                 - Clasificación YOLO
63:                 - Imagen con detecciones
64:                 - Características extraídas
65:                 - Predicción SVM YOLO
```

```
66:      Sistema de decisión:                                             4
67:         1. Si ambos métodos clasifican igual → Resultado final
68:         2. Si hay conflicto → Priorizar el resultado del método más
                                  fiable
```

```
69:      Resultados finales:                                              5
70:          - Mostrar imagen con detecciones
71:          - Imprimir resultados finales de la clasificación:
72:                - SVM que ha clasificado la imagen y su predicción
73:                - Resultado de la clasificación final y
                     porcentaje de fiabilidad

74: Fin del programa
```

5. Resultados obtenidos.

Como se ha citado con anterioridad, el modelo contará de 2 métodos de clasificación diferentes:

1. Un método utilizando MediaPipe Pose y YOLO y las características calculadas a través de los *landmarks* o puntos clave extraídos del cuerpo de la persona detectada y de las características extraídas de los bounding boxes detectados.

2. Otro método de clasificación automática realizado por el SVM entrenado con todas las características extraídas por MediaPipe y YOLO. Esta clasificación, como ya se ha comentado, contará de 3 SVM, uno entrenado con todas las características extraídas por MediaPipe y YOLO, otro entrenado solo con las características extraídas por MediaPipe y un último entrenado solo con las características extraídas por YOLO. Esto se hace para poder evitar falsos resultados cuando uno de los métodos no sea capaz de detectar persona en el caso en el que sí la haya.

Cada método se ha trabajado por separado hasta conseguir una clasificación robusta para cada uno de ellos y poder implementarlos juntos en el modelo final.

Para todas las pruebas se ha trabajado siempre con el mismo conjunto de imágenes. Este conjunto de prueba se ha construido a partir de las imágenes seleccionadas del *dataset* FPDs y consta de las siguientes imágenes:

DE PIE (0)	**SENTADO (1)**	**CAÍDA (2)**	**DESCONOCIDO (3)**	**TUMBADO (4)**
109	65	164	63	22
Normales (0,1,4)			**No normales (2)**	
196			164	

Tabla 17: Distribución del conjunto de prueba. Dataset FPDs.

No es un conjunto balanceado debido a la selección realizada en el *dataset* y a la prioridad de la clase crítica, en este caso la clase “caída”. Se aprecia un mejor balanceo en la unión de las clases “de pie”, “sentado” y “tumbado”, (“normales”) y la clase crítica “caída”, (“no normal”). Las posturas más comunes en el día a día se identifican como posturas “normales” pues no suponen situaciones de riesgo y la clase “caída” se identifica como “no normal”, pues corresponde a una clase anómala que no tiene que darse.

Luego se añade una cantidad de imágenes en las que no hay personas presentes para ver la eficacia del sistema con respecto a falsos positivos en la detección.

Para el análisis de los resultados se va a seguir una metodología parecida a la que se utiliza en el artículo del cual se ha extraído el *dataset* FPDs.

- Hay persona
  - Se detecta persona (**TP_d**)
    - Se clasifica bien (**TP_c**)
    - Se clasifica mal (**FP_c**)
  - No se detecta persona (**FN_d**)

- No hay persona
  - No se detecta persona (**TN_d**)
  - Se detecta persona (**FP_d**)

- **TP_d**: verdadero positivo en la fase de detección de persona.
- **FN_d:** falso negativo en la fase de detección de persona.
- **TN_d**: verdadero negativo en la fase de detección de persona.
- **FP_d:** falso positivo en la fase de detección de persona.
- **TP_c:** verdadero positivo en la fase de clasificación de pose.
- **FP_c**: falso positivo en la clase de clasificación de pose.

Es interesante separar el análisis de los resultados en la parte de detección y la parte de clasificación, pues que el sistema detecte o no la presencia de persona en la imagen no depende directamente del trabajo realizado en este proyecto si no depende del funcionamiento de los métodos MediaPipe y YOLO preentrenados, en cambio, el acierto o fallo en la clasificación de la pose sí que depende plenamente del funcionamiento del sistema desarrollado en este trabajo.

5.1. Clasificación con MediaPipe y YOLO.

Se trabaja primero con los métodos de clasificación por separado. Cada método se evaluará con el conjunto de pruebas seleccionado, obteniendo unos resultados que serán analizados y comparados para elegir el mejor sistema de decisión para la clasificación final y unión de ambos métodos.

5.1.1. Resultados de la clasificación usando MediaPipe.

El conjunto de prueba se evalúa con la clasificación realiza en MediaPipe, acorde a las características y umbrales seleccionados para este método de la siguiente manera:

```
1:    if      140<(angle_knee)      and      140<(angle_hip)      and
      1<(rel_torso_pierna)<2 and 0.1>(hombro_rodilla_x):
2:          mensaje="de pie"
3:    elif 0.1>(hombro_rodilla_y) and (hombro_rodilla_x)>0.1 and
      (hombro_cadera_y)<0.1 and (hombro_cadera_x)>0.05 and
      cabeza_rodilla_y<0.14 and altura_cabeza>0.6 :
4:          mensaje="caida"
5:    else:
6:          if (hombro_cadera)<0.1:
7:                mensaje="caida"
8:          else:
9:                if(hombro_rodilla_y<0) or (cadera_tobillo_y<0):
10:                     mensaje="caida"
11:               else:
12:                     mensaje="sentado"
```

Se hace un primer filtrado para saber si la pose cumple con los umbrales seleccionados para la clase “de pie” y “caída”, las más genéricas y fáciles de distinguir. Por último, si no se cumple ninguno de estos requisitos anteriores, se clasifica la pose como “sentado”.

Tras pasar por esta clasificación todas las imágenes del conjunto de prueba y organizarlos en una tabla Excel, se obtienen los siguientes resultados para la clasificación con MediaPipe (figura 44).

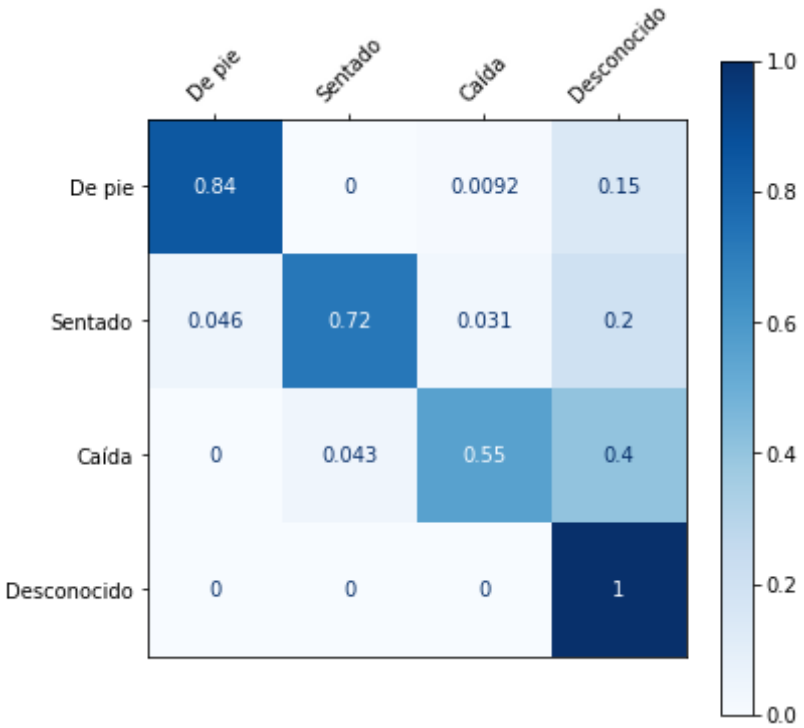


Figura 44: Matriz de confusión normalizada. Método Mediapipe. (Fuente: Elaboración propia)

Estos resultados, como se ha citado anteriormente en la metodología a seguir, se separan entre la parte de detección (tabla 18) del modelo y la parte de clasificación (tabla 19), los cuales desglosados se muestran a continuación:

- Detección:

Detección de MediaPipe	
TP_d	**72%**
FN_d	**28%**
TN_d	**100%**
FP_d	**0%**

Tabla 18: Resultados detección MediaPipe.

- Clasificación:

MEDIAPIPE	**De pie**	**Sentado**	**Caída**
De pie	98,92%	0,00%	1,08%
Sentado	5,77%	90,38%	3,85%
Caída	0,00%	7,14%	92,86%

Tabla 19: Matriz de confusión de clasificación de MediaPipe.

El análisis en profundidad de los resultados obtenidos se muestra en el apartado comparativo entre los resultados de ambos métodos (MediaPipe y YOLO).

En esta prueba no se ha evaluado el subconjunto correspondiente a la clase "tumbado" pues esta clasificación se realizará únicamente a través de los métodos que ofrece YOLO.

5.1.2. Resultados de la clasificación usando YOLO.

El conjunto de prueba se evalúa con la clasificación realizada en MediaPipe, acorde a las características y umbrales seleccionados para este método de la siguiente manera:

```
1:    if sofa_cama:
2:          area_persona = alto_bb*ancho_bb
3:          interseccion = calcular_interseccion([x, y, ancho_bb,
                        alto_bb], [x2, y2, ancho_bb2, alto_bb2])
4:          y_mitad_sofa = int(y2 + alto_bb2 / 2)
5:          mitad_superior_sofa = [x2, y2, ancho_bb2,
                              y_mitad_sofa - y2]
6:          mitad_inferior_sofa = [x2, y_mitad_sofa, x2+ ancho_bb2,
                              y2 + alto_bb2]
7:          interseccion50s = calcular_interseccion([x, y, ancho_bb,
                        alto_bb], mitad_superior_sofa)
8:          interseccion50i = calcular_interseccion([x, y, ancho_bb,
                        alto_bb], mitad_inferior_sofa)
9:          ocupacion_persona = interseccion/area_persona
10:         ocupacion50s=interseccion50s/area_persona
11:         ocupacion50i=interseccion50i/area_persona
12:         if ocupacion_persona > 0.5:
13:               if ocupacion50s > (ocupacion_persona/2):
14:                     tumbado =True
15:               else:
16:                     tumbado=False
```

```
17:   if umbral < 15:
18:         if tumbado:
19:               mensaje ="tumbado"
20:         else:
21:               mensaje="caida"
22:   else:
23:         if ratio > 0.5:
24:               mensaje="sentado"
25:         else:
26:               mensaje="de pie"
```

En esta ocasión se empieza por determinar si existe la clase “sofá” o “cama” en la imagen y a continuación se calculan los parámetros necesarios para posteriormente realizar la clasificación en base a la posibilidad de la existencia de la clase “tumbado”. Si esta es posible que exista, las poses que se detectarían como “caída” se clasificarán como “tumbado”, de lo contrario se distingue únicamente “caída”, “de pie” y “sentado”.

Tras pasar por esta clasificación todas las imágenes del conjunto de prueba y organizarlos en una tabla Excel, se obtienen los siguientes resultados (figura 45) para la clasificación con YOLO.

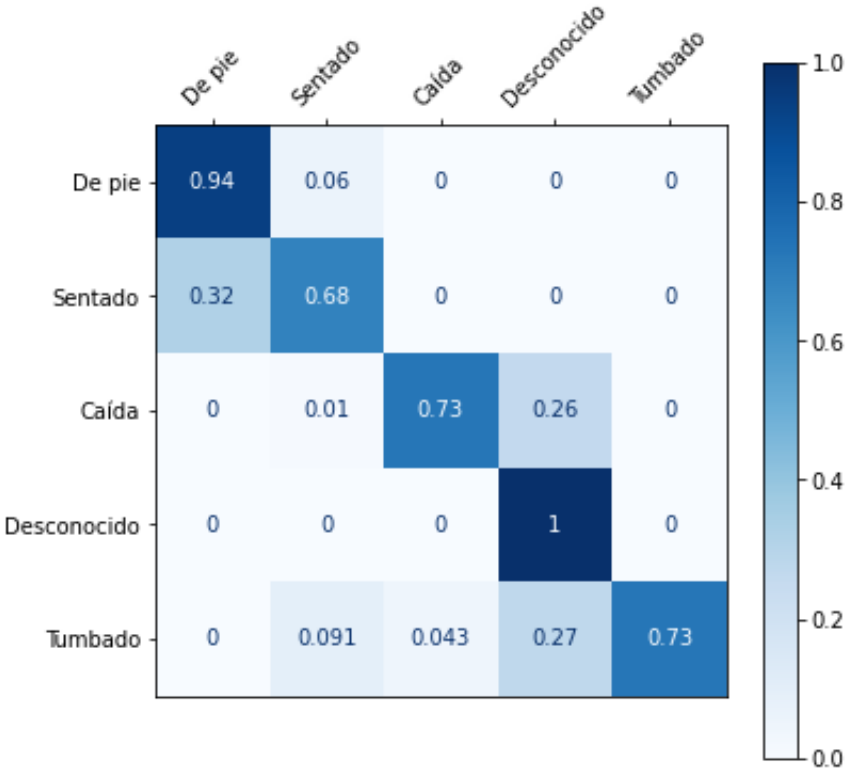


Figura 45: Matriz de confusión normalizada. Método YOLO. (Fuente: Elaboración propia)

Al igual que para MediaPipe, se separan estos resultados entre detección (tabla 20) y clasificación (tabla 21).

- Detección:

Detección de YOLO	
TP_d	87%
FN_d	13%
TN_d	100%
FP_d	0%

Tabla 20: Resultados detección YOLO.

- Clasificación:

YOLO	De pie	Sentado	Caída	Tumbado
De pie	94,50%	5,50%	0,00%	0,00%
Sentado	32,31%	67,69%	0,00%	0,00%
Caída	0,00%	0,83%	99.17%	0,00%
Tumbado	0,00%	0,00%	0,00%	99,9%

Tabla 21: Matriz de confusión de clasificación de YOLO.

5.1.3. Comparativa de los métodos individuales.

En este apartado se compararán los resultados obtenidos para sendos métodos (MediaPipe y YOLO), pues es esencial para conseguir una clasificación robusta y lo más eficaz posible.

- **Comparativa en la detección.**

	MEDIAPIPE	YOLO
TP_d	72%	87%
FN_d	28%	13%
TN_d	100%	100%
FP_d	0%	0%

Tabla 22: Comparativa resultados clasificación de métodos individuales.

Comparando los resultados obtenidos en ambos métodos (tabla 22), se observa que YOLO es mucho más eficaz a la hora de realizar la detección de persona cuando sí que la hay, lo que se ha clasificado como TP_d, más concretamente, una mejora del 15% respecto a la detección de MediaPipe. Con respecto a los falsos positivos en la detección cuando no hay persona, ambos métodos se comportan siempre de manera muy robusta, obteniendo un 100% de acierto.

Para visualizar mejor la diferencia en la detección, cuando sí que hay persona entre los dos métodos, se puede observar el gráfico inferior (figura 46):

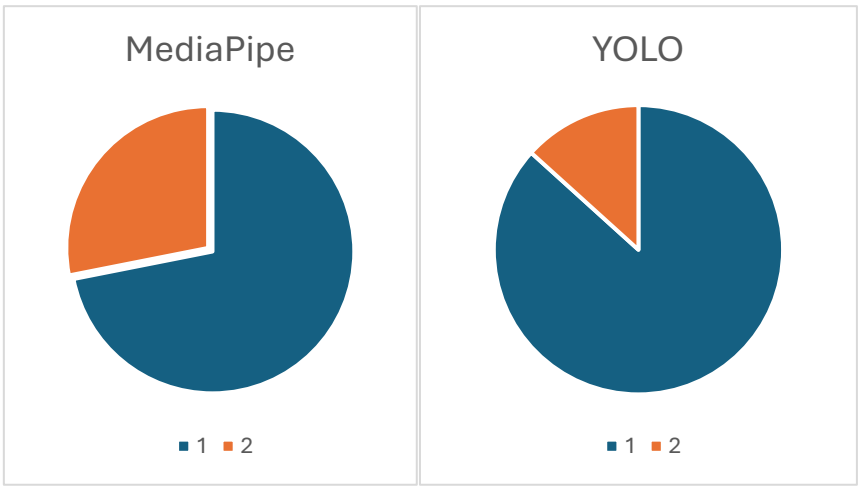


Figura 46: Gráfico comparativo clasificación de métodos individuales. (Fuente: Elaboración propia)

Tras observar los resultados en la detección, se elige como prioritaria la detección mediante YOLO.

- **Comparativa caída.**

Se pasará a realizar una comparativa en la parte de clasificación para ambos métodos, separándolos por poses. Se empieza por la clase crítica, la pose "caída" (tabla 23).

	De pie	Sentado	Caída	Tumbado
MediaPipe	0,00%	7,14%	92,86%	--
YOLO	0,00%	0,83%	99.17%	0,00%

Tabla 23: Comparativa en la clasificación de pose caída de métodos individuales.

Ambos métodos muestran un acierto superior al del 90%, lo que es muy positivo. Aun así, se aprecia cómo YOLO se acerca mucho más al 100% de eficacia. Por lo tanto, se priorizará la clasificación de esta pose mediante este método. Para visualizar mejor la diferencia entre ambos métodos, se observa el gráfico inferior (figura 47).

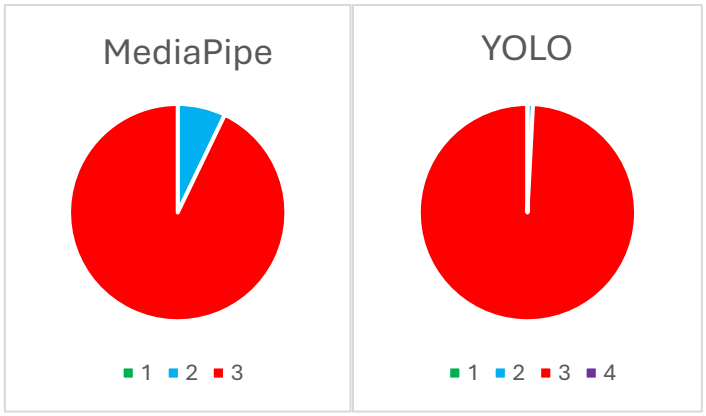


Figura 47: Gráfico comparativo de clasificación de pose caída de métodos individuales. (Fuente: Elaboración propia)

- **Comparativa resto poses.**

Se pasa a comparar los resultados obtenidos para las poses que se podrían clasificar como normales.

	De pie	Sentado	Caída	Tumbado
MediaPipe	98,92%	0,00%	1,08%	--
YOLO	94,50%	5,50%	0,00%	0,00%

Tabla 24: Comparativa clasificación de pie de métodos individuales.

Comenzando por la pose “de pie” (tabla 24), en la que ambos métodos obtienen un acierto superior al 90%, se puede considerar que la clasificación es correcta. Se aprecia una pequeña diferencia a favor del método MediaPipe, obteniendo un 4,42% más de acierto y acercándose a ese 100% de efectividad. Aunque ambos métodos demuestran ser robustos, se prioriza la clasificación de MediaPipe para esta pose. La diferencia se puede apreciar en el gráfico inferior (figura 48).

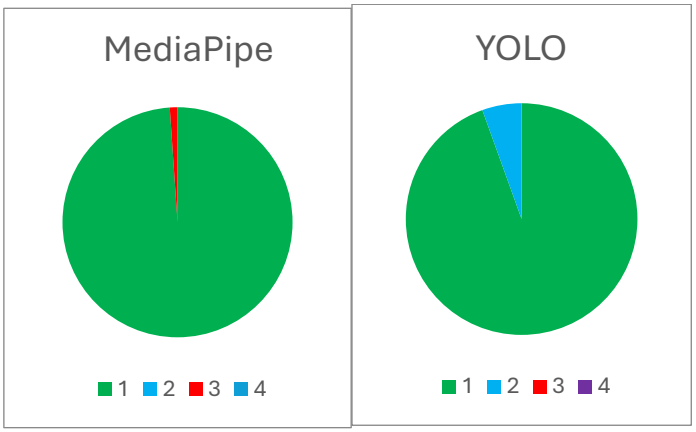


Figura 48: Gráfico comparativo clasificación de pie de métodos individuales. (Fuente: Elaboración propia)

Por último, se analiza la clasificación en la pose “sentado” para ambos métodos (tabla 25):

	De pie	Sentado	Caída	Tumbado
MediaPipe	5,77%	90,38%	3,85%	5,77%
YOLO	32,31%	67,69%	0,00%	0,00%

Tabla 25: Comparativa clasificación pose sentado de métodos individuales.

En esta ocasión sí que se aprecia una notable diferencia en los resultados obtenidos por ambos métodos. MediaPipe ofrece una eficacia del 90%, mientras que YOLO cae hasta el 67%. Esto es debido a la hora de cómo ambos métodos clasifican las poses. MediaPipe es capaz de extraer información de articulaciones, relaciones corporales y demás pudiendo ser mucho más preciso a la hora de diferenciar poses, mientras que YOLO solo es capaz de extraer información de las medidas del boundingbox, por lo que es muy fácil que se confunda con la pose “de pie” como es evidente en los resultados obtenidos.

Por estas razones se priorizará la respuesta de MediaPipe para esta pose. La diferencia entre los resultados de ambos métodos se muestra en el gráfico inferior (figura 49).

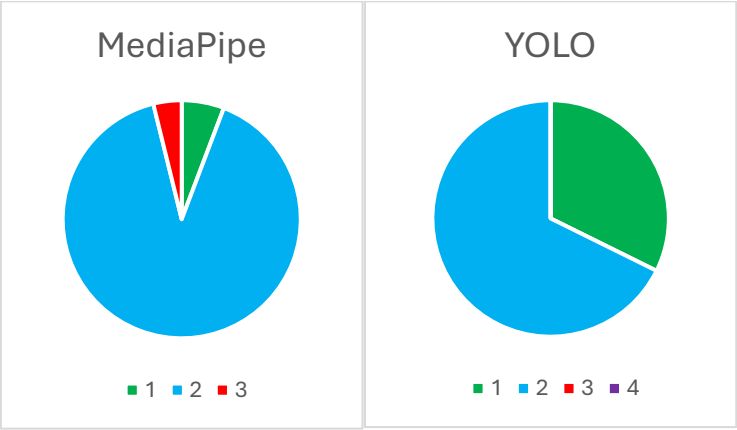


Figura 49: Gráfico comparativo clasificación pose sentado de métodos individuales. (Fuente: Elaboración propia)

Basándose en los resultados obtenidos en la detección y clasificación individual de ambos métodos en las pruebas, se crea un sistema de decisión para el modelo.

5.1.4. Implementación de un sistema de decisión basado en las fortalezas de cada método.

Se desarrolla un sistema de decisión basándose en las fortalezas y resultados de cada método. El siguiente esquema (figura 50) lo muestra con claridad:

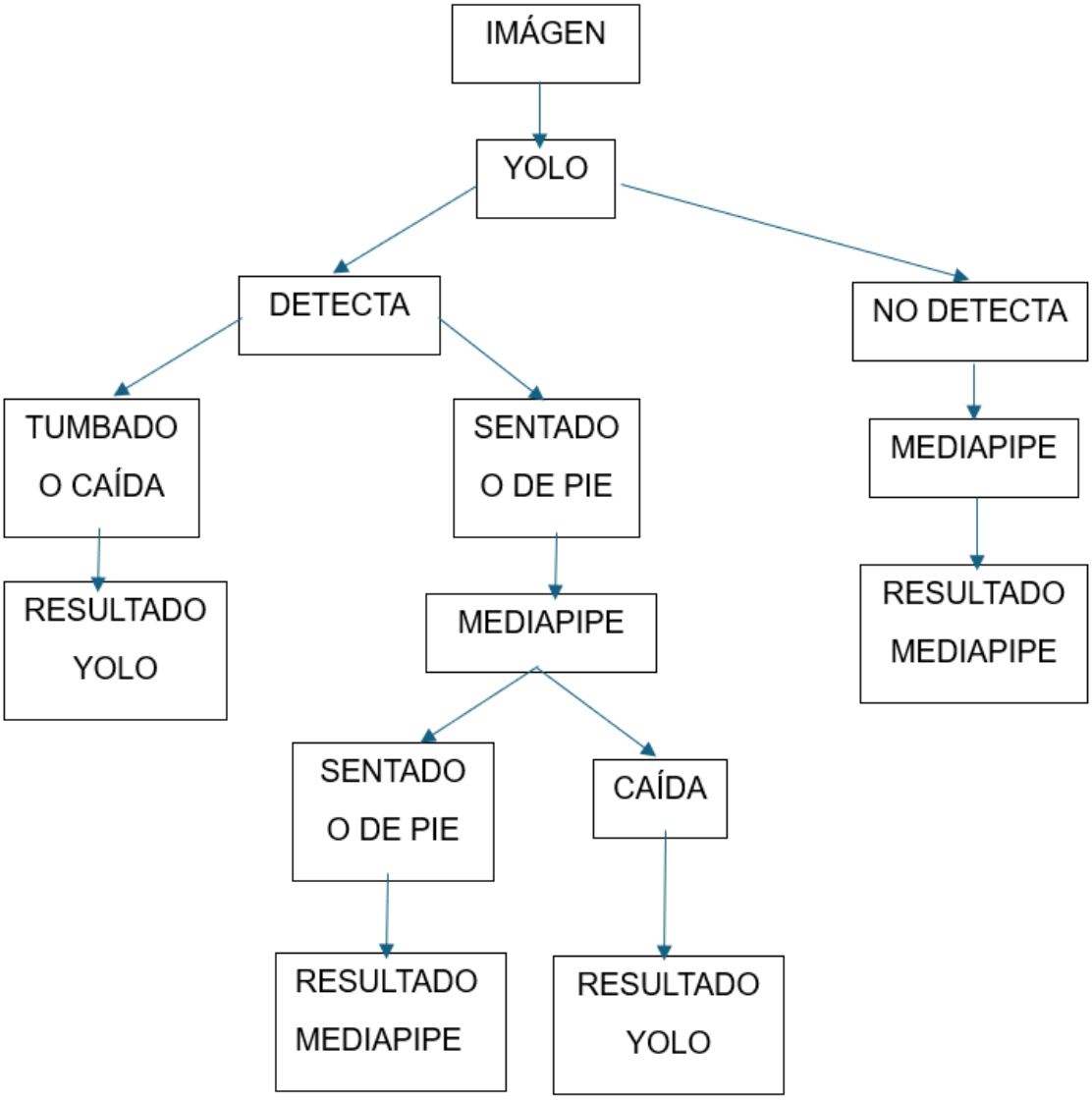


Figura 50: Árbol de decisión para la clasificación final. (Fuente: Elaboración propia)

5.2. Clasificación del SVM.

Para el método de clasificación automático se entrena un algoritmo SVM que, acorde a las características extraídas de cada imagen, clasificará la pose presente.

El primer paso es entrenarla y obtener unas medidas de los resultados que ayudarán a saber cómo generaliza el algoritmo entrenado y si ha separado bien las clases.

5.2.1. Entrenamiento y uso del SVM con datos combinados de MediaPipe y YOLO.

Se empieza por el algoritmo SVM entrenado con las características extraídas tanto de MediaPipe como de YOLO, se denominará SVM_Global, pues se ejecutará cuando ambos métodos sean capaces de extraer información de la imagen.

```
1:    nombres_caracteristicas = [“angle_knee_r”, “angle_knee_l”,
      “angle_hip_r”, “angle_hip_l”, “hombro_rodilla_r_y”,
      “hombro_rodilla_l_y”, “hombro_rodilla_r_x”,
      “hombro_rodilla_l_x”, “cadera_rodilla_r_x”,
      “cadera_rodilla_l_x”, “cadera_rodilla_r_y”,
      “cadera_rodilla_l_y”, “hombro_cadera_r_y”, “hombro_cadera_l_y”,
      “hombro_cadera_r_x”, “hombro_cadera_l_x”, “rel_torso_pierna_r”,
      “rel_torso_pierna_l”, “distancia_rodillas”, “cabeza_rodilla_y”,
      “rodilla_mano_l_y”, “rodilla_mano_r_y”, “rodilla_mano_l_x”,
      “rodilla_mano_r_x”, “cadera_tobillo_l_y”, “cadera_tobillo_r_y”,
      “cadera_tobillo_l_x”, “cadera_tobillo_r_x”, “altura_cabeza”,
      “altura_cadera_l”, “altura_cadera_r”, “hombro_cadera”, “umbral”,
      “ratio”, “x_sofa”, “x”, “y_sofa”, “y”, “ancho_sofa”, “ancho_bb”,
      “alto_sofa”, “alto_bb”]
```

Entrenando y probando el SVM con el conjunto de imágenes de prueba, se obtienen siguientes resultados (tabla 26):

Precisión del modelo: 0,9820

Reporte de clasificación:

	precision	*recall*	*f1-score*	*support*
0	0,97	1,00	0,98	32
1	1,00	0,95	0,97	19
2	0,98	1,00	0,99	48
3	1,00	1,00	1,00	4
4	1,00	0,88	0,93	8
Macro avg	0,99	0,96	0,98	111
Weighted avg	0,98	0,98	0,98	111

Tabla 26: Reporte de clasificación SVM global.

La precisión es la exactitud general del modelo, es una métrica global del modelo, no específica de cada clase y muestra que clasifica correctamente el 98.20% de las muestras totales.

Basándose en el análisis de las métricas, se puede deducir que las clases 2 y 3 son perfectamente clasificadas (*recall* y *f1-score* de 1,00). La clase 4 tiene el peor desempeño, con un *recall* de 0,88 (fallando en reconocer un 12% de las muestras).

También se genera la matriz de confusión, que muestra cómo se distribuyen las predicciones para cada clase de manera gráfica (figura 51):

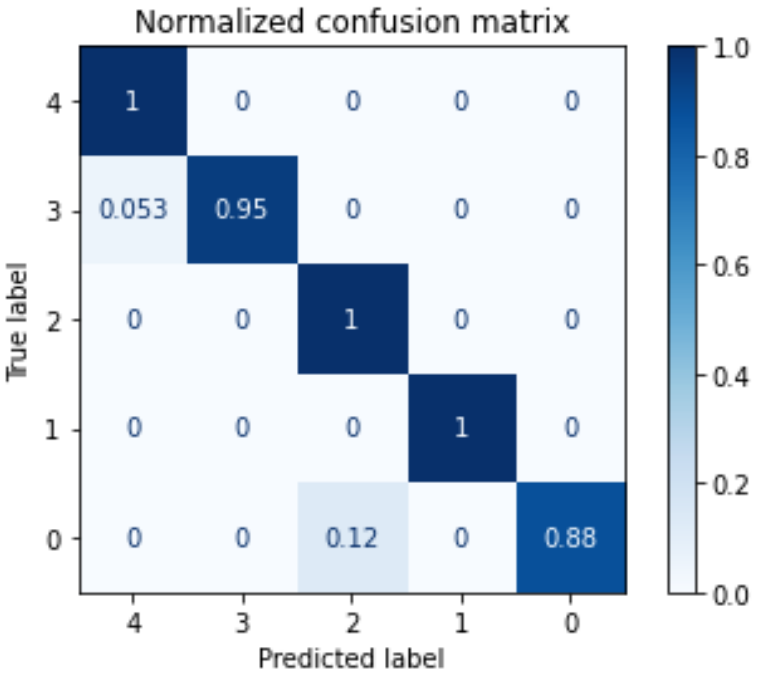


Figura 51: Matriz de confusión normalizada rango [0-1]. SVM global. (Fuente: Elaboración propia)

Como conclusión, el modelo entrenado tiene un desempeño excelente con una *accuracy* general del 98,2%. La clase 4 (con menos soporte) es la más problemática, con un *recall* más bajo y errores pequeños. Se da como bueno el entreno de este SVM y se usará para la clasificación definitiva.

5.2.2. Clasificación en escenarios específicos.

También se entrena un algoritmo SVM únicamente con características de cada uno de los métodos para cuando uno de ellos no sea capaz de extraer información de la imagen. Esto ayudará a evitar errores en la clasificación por parte del SVM global al faltarle características.

5.2.2.1. *SVM con datos únicamente de MediaPipe.*

En este primer caso, se entrena un SVM con las características extraídas a través de MediaPipe para cuando YOLO no sea capaz de detectar la clase persona.

```
1:    nombres_caracteristicas = [“angle_knee_r”, “angle_knee_l”,
      “angle_hip_r”, “angle_hip_l”, “hombro_rodilla_r_y”,
      “hombro_rodilla_l_y”, “hombro_rodilla_r_x”,
      “hombro_rodilla_l_x”, “cadera_rodilla_r_x”,
      “cadera_rodilla_l_x”, “cadera_rodilla_r_y”,
      “cadera_rodilla_l_y”, “hombro_cadera_r_y”, “hombro_cadera_l_y”,
      “hombro_cadera_r_x”, “hombro_cadera_l_x”, “rel_torso_pierna_r”,
      “rel_torso_pierna_l”, “distancia_rodillas”, “cabeza_rodilla_y”,
      “rodilla_mano_l_y”, “rodilla_mano_r_y”, “rodilla_mano_l_x”,
      “rodilla_mano_r_x”, “cadera_tobillo_l_y”, “cadera_tobillo_r_y”,
      “cadera_tobillo_l_x”, “cadera_tobillo_r_x”, “altura_cabeza”,
      “altura_cadera_l”, “altura_cadera_r”, “hombro_cadera”]
```

Entrenando y probando el SVM con el conjunto de imágenes de prueba, se obtienen siguientes resultados (tabla 27 y figura 52):

Precisión del modelo: 0,9725

Reporte de clasificación:

	precision	***recall***	***f1-score***	***support***
0	1,00	0,94	0,97	35
1	0,88	0,94	0,91	16
2	0,98	1,00	0,99	58
Macro avg	0,96	0,96	0,96	109
Weighted avg	0,97	0,97	0,97	109

Tabla 27: Reporte de clasificación SVM MediaPipe.

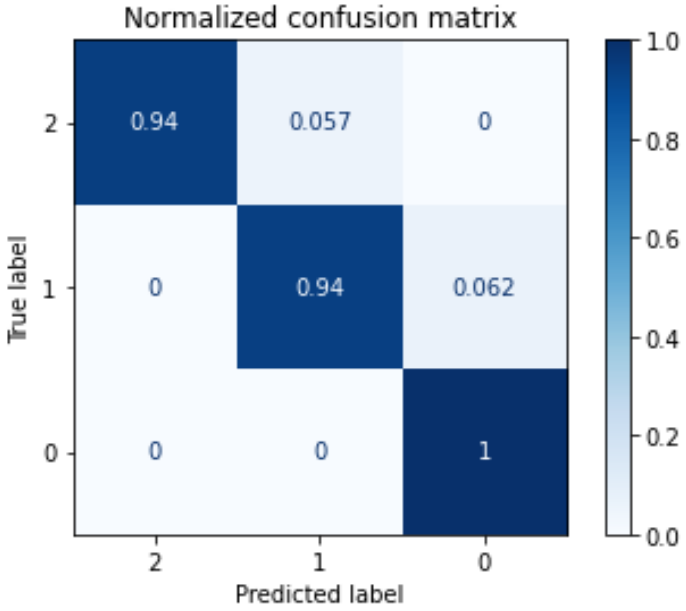


Figura 52: Matriz de confusión normalizada rango [0-1]. SVM MediaPipe. (Fuente: Elaboración propia)

Analizando las métricas, se puede decir que el modelo tiene un desempeño general muy bueno, pues clasifica correctamente el 97,25% de las muestras y posee un balance fuerte entre precisión y *recall*. La clase 0 posee una excelente precisión, aunque pierde algunos casos reales (*recall* 94%), la clase 2 se clasifica casi a la perfección, probablemente por su mayor soporte. La clase 1 es la más problemática con la precisión más baja del 88%, esto puede deberse a que tiene el menor soporte.

5.2.2.2. SVM con datos únicamente de YOLO.

Por último, se entrena un SVM solamente con las características extraídas por YOLO para cuando MediaPipe no detecte *landmarks* en la imagen.

```
1:    nombres_caracteristicas = ['umbral','ratio', 'x_sofa', 'x',
      'y_sofa', 'y', 'ancho_sofa', 'ancho_bb', 'alto_sofa', 'alto_bb']
```

Entrenando y probando el SVM con el conjunto de imágenes de prueba, se obtienen siguientes resultados (tabla 28 y figura 53):

Precisión del modelo: 0,9355

Reporte de clasificación:

	precision	***recall***	***f1-score***	***support***
0	0,97	0,91	0,94	35
1	0,84	0,95	0,89	22
2	1,00	0,97	0,98	32
4	0,75	0,75	0,75	4
Macro avg	0,89	0,90	0,89	93
Weighted avg	0,94	0,94	0,94	93

Tabla 28: Reporte de clasificación SVM YOLO.

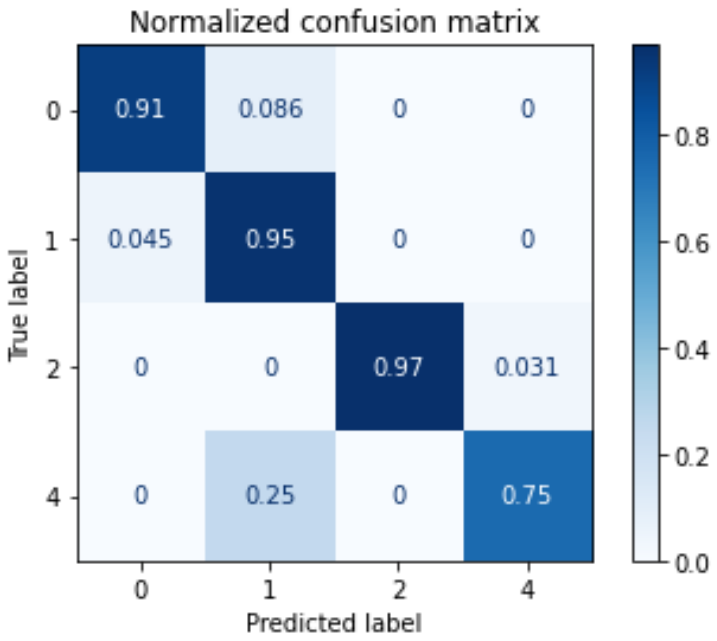


Figura 53: Matriz de confusión normalizada rango [0-1]. SVM YOLO. (Fuente: Elaboración propia)

Aunque el desempeño general del modelo es bueno (93% de precisión), hay un desbalance en el rendimiento entre algunas clases. Las clases 0, 1 y 2 tienen buenos desempeños, pero la clase 4 tiene problemas debido posiblemente al bajo número de muestras.

5.3. Validación.

En este apartado se validarán los resultados del modelo final implementado en su conjunto a través del conjunto de imágenes de prueba para ver su rendimiento en general y si fuera necesario realizar algún cambio. Se lleva a cabo la validación con el conjunto de prueba, pues será indispensable comparar los resultados obtenidos del modelo completo con los obtenidos por los métodos individuales, para verificar que las decisiones tomadas en la lógica del código han sido correctas.

5.3.1. Evaluación del rendimiento del sistema en imágenes de prueba.

Como se ha dicho anteriormente, se extraen los resultados obtenidos por el modelo final sobre el conjunto de prueba.

Se recuerda la estructura de dicho conjunto (tabla 29):

DE PIE	SENTADO	CAÍDA	DESCONOCIDO	TUMBADO
109	65	164	63	22

Tabla 29: Conjunto de prueba.

La matriz de confusión ofrece los resultados globales obtenidos por el modelo (figura 54). Como en ocasiones anteriores, se separarán los resultados de la parte de detección (tabla 30 y figura 55) y de la parte de clasificación (tabla 31 y figura 56).

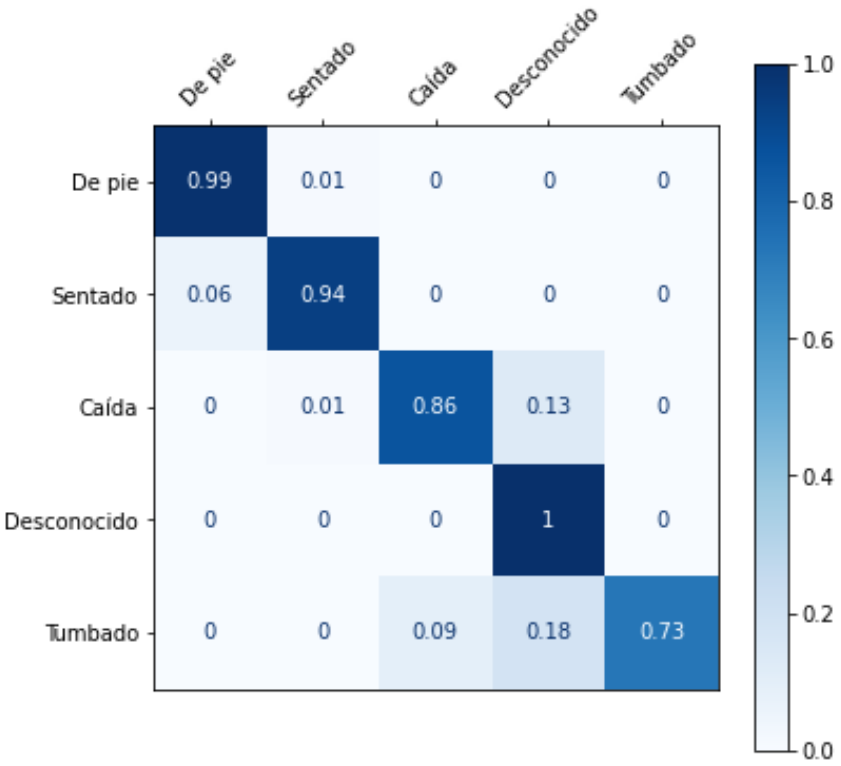


Figura 54: Matriz de confusión conjunto de prueba método manual. (Fuente: Elaboración propia)

- **Detección.**

TP_d	**92,78%**
FN_d	**7,22%**
TN_d	**100,00%**
FP_d	**0,00%**

Tabla 30: Resultados detección conjunto de prueba.

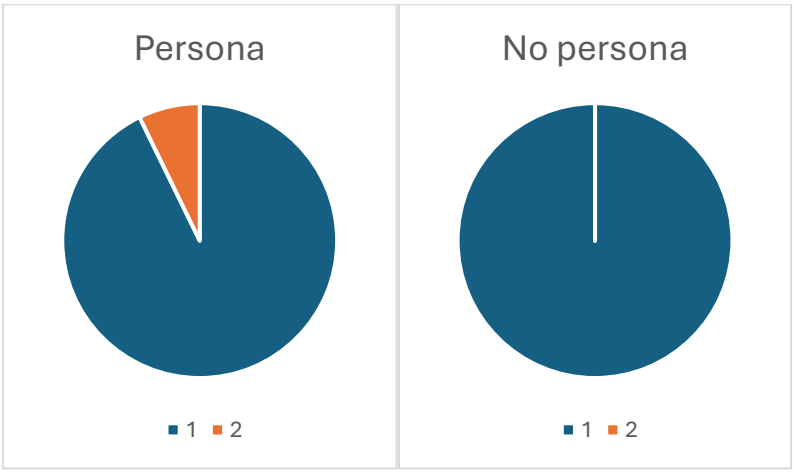


Figura 55: Gráfico resultados detección conjunto de prueba. (Fuente: Elaboración propia)

- **Clasificación.**

	De pie	Sentado	Caída	Tumbado
De pie	99,08%	0,92%	0,00%	0,00%
Sentado	6,15%	93,85%	0,00%	0,00%
Caída	0,00%	0,61%	99,39%	0,00%
Tumbado	0,00%	0.00%	11,11%	88,89%

Tabla 31: Resultados clasificación conjunto de prueba método manual.

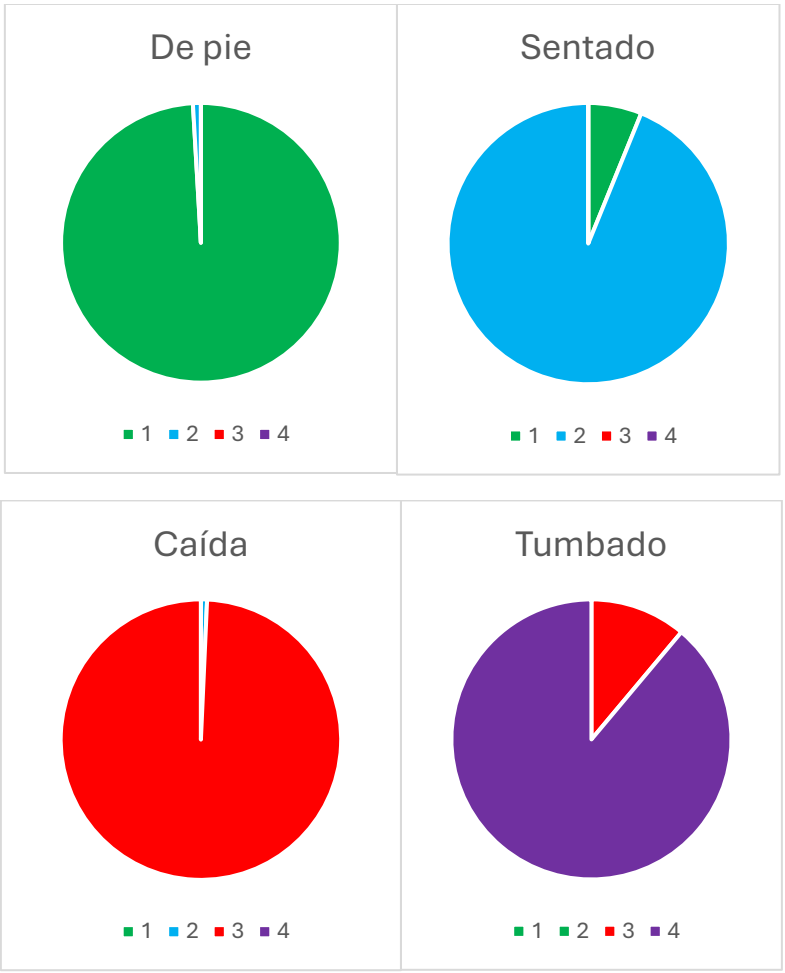


Figura 56: Gráfico resultados clasificación conjunto de prueba método manual. (Fuente: Elaboración propia)

También se muestran los resultados finales de la clasificación automática mediante el algoritmo SVM entrenado (figura 57). Se recuerda que los resultados en la parte de detección, al depender directamente de los métodos MediaPipe y YOLO, no arrojan ninguna diferencia con los obtenidos anteriormente. Los resultados de la parte de clasificación (tabla 32 y figura 58) se muestran a continuación:

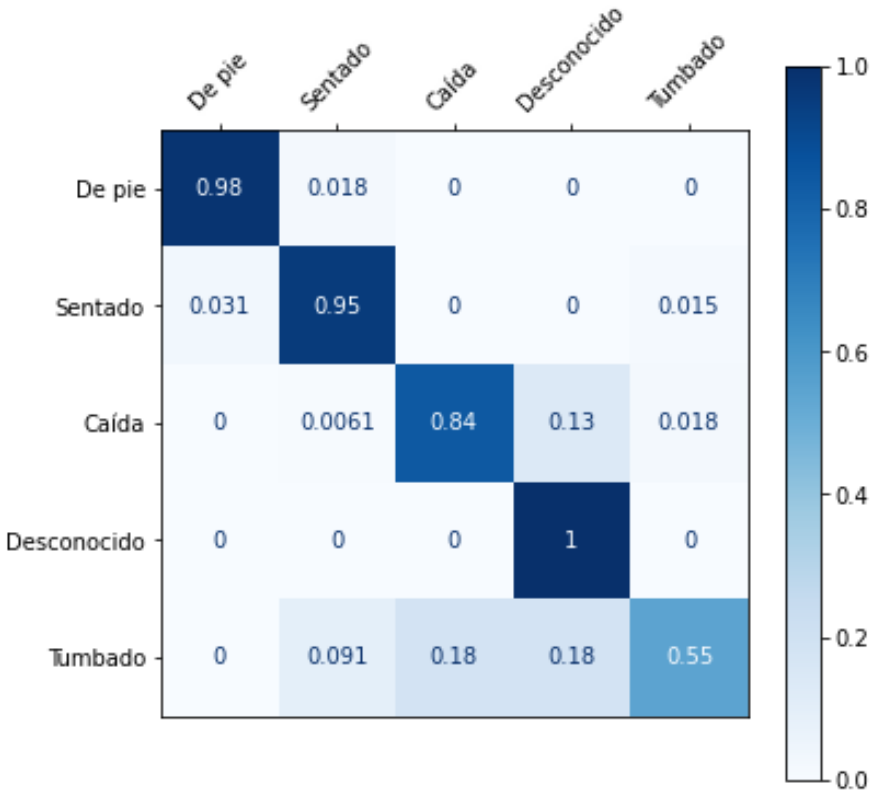
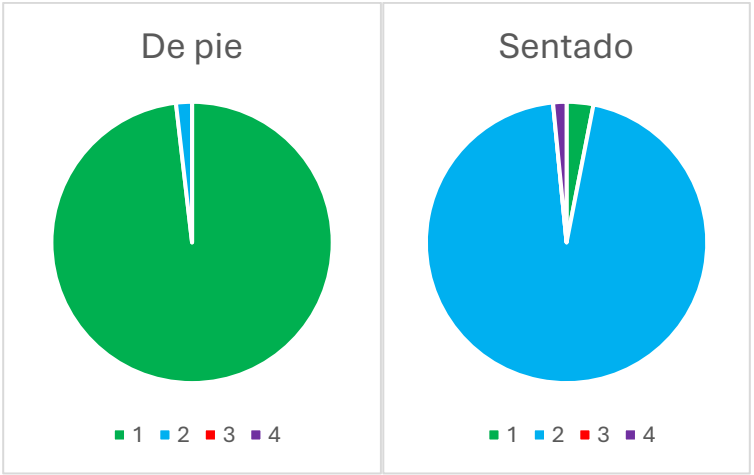


Figura 57: Matriz de confusión conjunto de prueba método SVM. (Fuente: Elaboración propia)

- **Clasificación.**

	De pie	Sentado	Caída	Tumbado
De pie	98,17%	1,83%	0,00%	0,00%
Sentado	3,08%	95,38%	0,00%	1,54%
Caída	0,00%	0,70%	97,18%	2,11%
Tumbado	0,00%	11,11%	22,22%	66,67%

Tabla 32: Resultados clasificación conjunto de prueba método SVM.



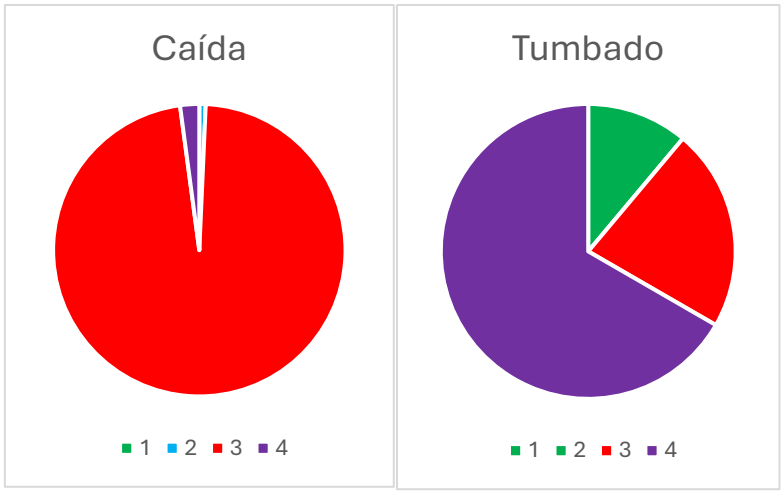


Figura 58: Gráfico resultados clasificación conjunto de prueba método SVM. (Fuente: Elaboración propia)

Para poder analizar con rigor, los resultados obtenidos del modelo final y del SVM para el conjunto de prueba se comparan con los resultados obtenidos por los métodos individuales a continuación.

5.3.2. Comparativa entre los métodos individuales y la combinación final.

Como se ha citado anteriormente, se analizan los resultados obtenidos en el conjunto de prueba (tabla 33):

- **Detección.**

	MEDIAPIPE	YOLO	FINAL
TP_d	72%	87%	92,78%
FN_d	28%	13%	7,22%
TN_d	100%	100%	100%
FP_d	0%	0%	0%

Tabla 33: Comparativa resultados detección. Métodos individuales VS Modelo final.

Se aprecia así una notable mejora al unir ambos métodos según el árbol de decisión. Cuando el método que mejor resultado ha obtenido en la detección, es decir, YOLO, no detecta, se depende de la detección de *landmarks* de MediaPipe, cubriendo más casos y mejorando bastante la detección. Como se puede ver, supera de manera cómoda el 90% de los casos.

Se demuestra, así, la eficacia en este apartado de la unión de ambos métodos.

- **Clasificación.**

Para esta parte se había elegido como prioritario el método con mejores resultados individuales (tabla 34).

Para la clasificación de la pose “caída” se prioriza la decisión de YOLO, pues, sus resultados, como se ha visto anteriormente, son mejores a los de MediaPipe (98.35% de acierto para YOLO y 92.86% de acierto para MediaPipe). Para la clasificación de las poses “de pie” y “sentado” se priorizaba la decisión de MediaPipe, pues, sus resultados han sido mejores en la fase de entrenamiento (98.92% de acierto para MediaPipe y 94.5% de acierto para YOLO en la pose “de pie”. 90.38% de acierto para MediaPipe y 67.69% de acierto para YOLO en la pose “sentado”). Por último, para la clasificación de la pose “tumbado” se depende completamente de YOLO, pues, es el método en el que se ha basado totalmente esta clasificación.

	TP_c	**FP_c**
MEDIAPIPE	94,05%	5,95%
YOLO	90,32%	9,68%
FINAL	95,10%	4,90%
SVM	89,35%	10,65%

Tabla 34: Resultados generales de cada método. Conjunto de prueba.

Los resultados obtenidos por los métodos individuales, eligiendo el que mejor resultado ha obtenido, son buenos, superando siempre el 90%, pero la unión de ambos métodos demuestra superioridad, aunque no de manera muy exagerada, pues los métodos individuales ya tenían una buena precisión. Se mejora así todavía más la clasificación y se hace más robusta.

En lo que respecta al algoritmo SVM, no mejora en exceso la clasificación realizada de forma “empírica” a través de los umbrales de decisión de las características extraídas por los métodos individuales. Por se ofrecen sus resultados como una respuesta más, pues, se considera un método complementario y prácticamente igual de fiable, y resulta interesante comparar la respuesta de un sistema creado “manualmente” a base del análisis riguroso de las características y un algoritmo de decisión autónoma como es el SVM.

La unión de ambos métodos, lo que consigue es aumentar el espectro en la detección y la clasificación, no descartando una imagen ni seleccionando una postura de primeras sin haber comprobado con el resultado obtenido por el otro método.

5.4. Resultados obtenidos en el conjunto de test.

Ver cómo generaliza el modelo desarrollado es esencial para sacar conclusiones firmes sobre su rendimiento, por ello, se prueba mediante imágenes de test, las cuales no han sido nunca utilizadas para el entrenamiento ni decisiones tomadas con el modelo realizado. El conjunto de imágenes de test es el siguiente y sigue una distribución parecida al conjunto de prueba (tabla 35).

DE PIE	SENTADO	CAÍDA	TUMBADO
74	85	140	23
NORMALES			CAÍDA
182			140

Tabla 35: Conjunto de test.

Las matrices de confusión obtenidas para el modelo final (figura 59) y el modelo del SVM (figura 60) son las siguientes:

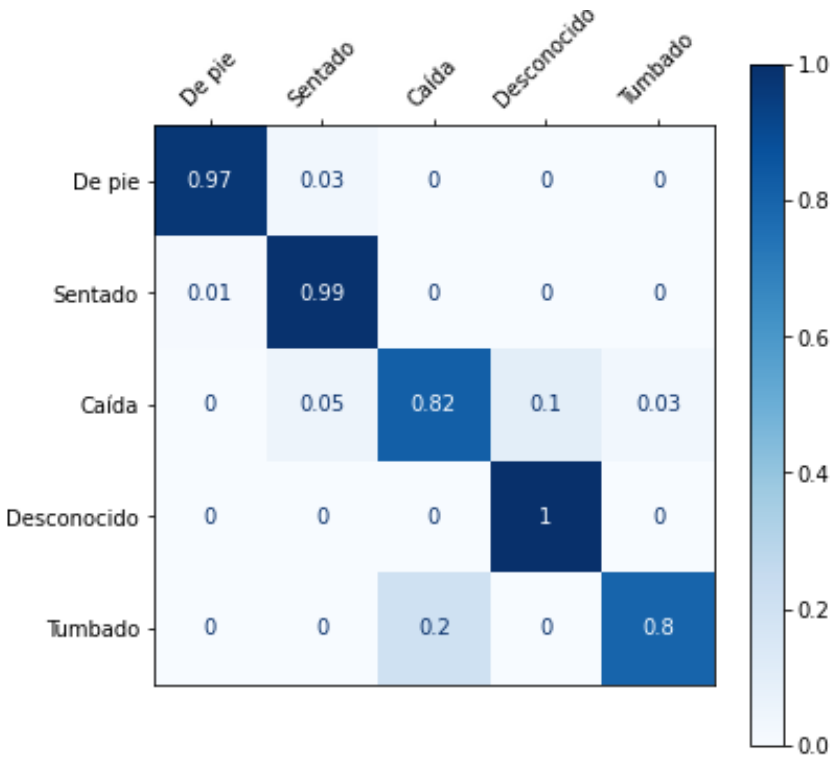


Figura 59: Matriz de confusión de test método manual. (Fuente: Elaboración propia)

Los resultados obtenidos para el modelo final con el conjunto de imágenes de test son muy similares a los obtenidos en las pruebas realizadas. Se puede decir que el modelo generaliza bastante bien. Salvo en las clases más conflictivas por la variabilidad en las posturas (pose “caída”) y la dificultad en los métodos de detección (pose “tumbado”) en los que los resultados disminuyen un poco su eficacia, en el resto de las poses se parecía una muy buena precisión.

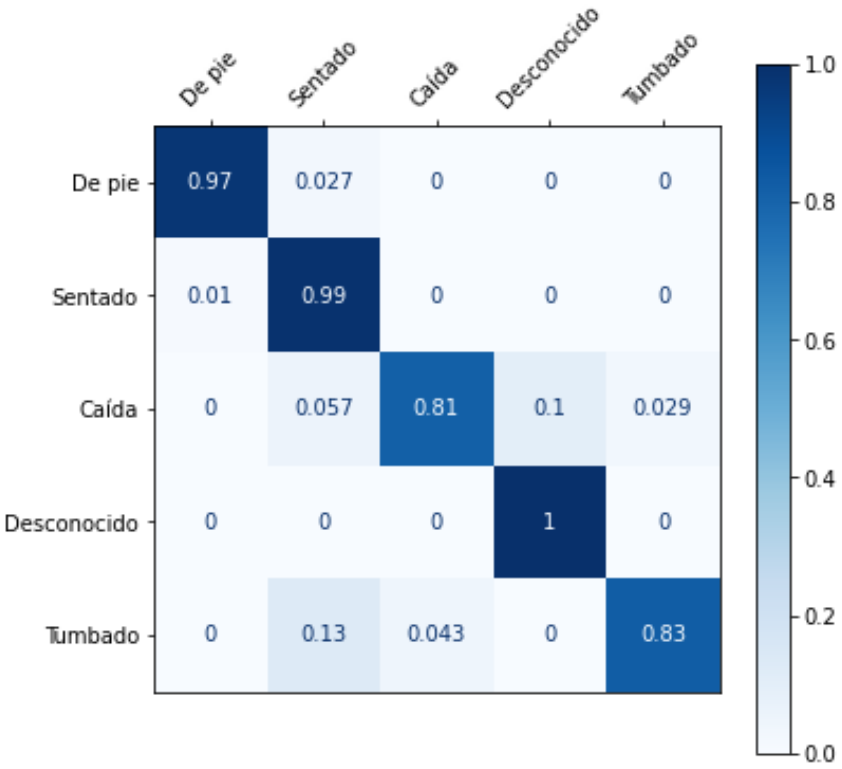


Figura 60: Matriz de confusión de test método SVM. (Fuente: Elaboración propia)

Con el modelo automático del clasificador SVM pasa lo mismo. Se puede ver que los resultados obtenidos son buenos y similares a los obtenidos anteriormente. Como conclusión a este análisis preliminar, el SVM generaliza también con mucha precisión en la clasificación para estas imágenes de test.

- **Detección.**

En este caso (tabla 36) se aprecia que los resultados obtenidos para la detección de personas son muy buenos, superando con creces el 90% de acierto.

TP_d	**95,67%**
FN_d	**4,33%**
TN_d	**100,00%**
FP_d	**0,00%**

Tabla 36: Resultados detección test.

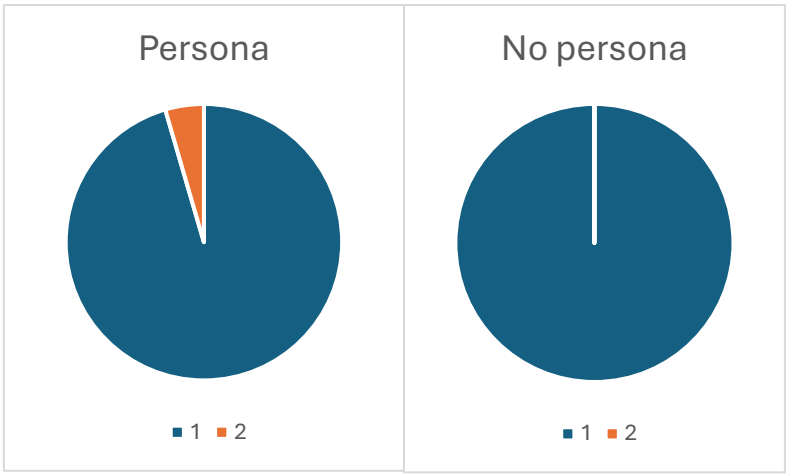


Figura 61: Gráfico resultados detección test. (Fuente: Elaboración propia)

- **Clasificación.**

Para la parte de clasificación se analizarán los resultados obtenidos para el método de clasificación mediante el árbol de decisión y los umbrales calculados y el método automático de la clasificación del SVM.

Empezando por el método “manual”, estos son los resultados que se obtienen (tabla 37):

Manual	De pie	Sentado	Caída	Tumbado
De pie	97,30%	2,70%	0,00%	0,00%
Sentado	0,01%	99,90%	0,00%	0,00%
Caída	0,00%	5.55%	91,27%	3,18%
Tumbado	0,00%	0,00%	20,00%	80,00%

Tabla 37: Resultados clasificación test método manual.

La precisión en la clasificación es muy buena para casi todas las clases, superando el 90% incluso llegando prácticamente al 100% en la clase “sentado”. Para la clase “tumbado” el rendimiento cae hasta el 65%, confundiéndose en la mayoría de los casos con la clase “caída”, error previsible por el método de clasificación de la pose “tumbado”.

Los errores que se cometen en la clasificación en ningún momento son graves, pues las clases confundidas son clases cercanas y semejantes a la pose a clasificar.

Visualmente, se aprecian los resultados obtenidos en el siguiente gráfico (figura 62):

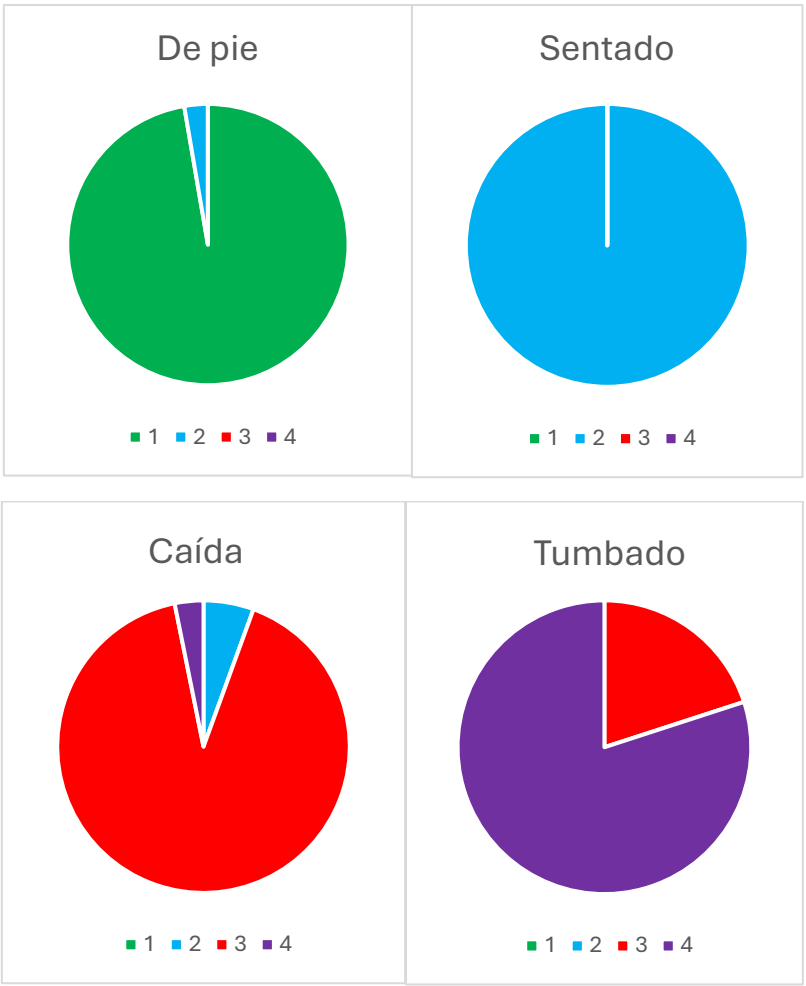


Figura 62: Gráfico resultados clasificación test método manual. (Fuente: Elaboración propia)

En lo que respecta a la clasificación SVM se obtienen los siguientes resultados (tabla 38):

SVM	De pie	Sentado	Caída	Tumbado
De pie	97,30%	2,70%	0,00%	0,00%
Sentado	0,01%	99,9%	0,00%	0,00%
Caída	0,00%	6,35%	90,48%	3,17%
Tumbado	0,00%	13,04%	4,35%	82,61%

Tabla 38: Resultados clasificación test método SVM.

En esta ocasión, con el conjunto de test, se ha comportado mejor este método, pues los resultados obtenidos con respecto al método "manual" así lo demuestran. En esta ocasión la pose más conflictiva "tumbado" se trabaja de manera más efectiva y el SVM es capaz de separar mejor esta clase del resto.

Al igual que en el método anterior, los errores cometidos en la clasificación no son graves, pues cuando se producen son con clases cercanas y muchos casos muy similares en postura.

Los resultados de manera gráfica se visualizan en el siguiente gráfico (figura 63):

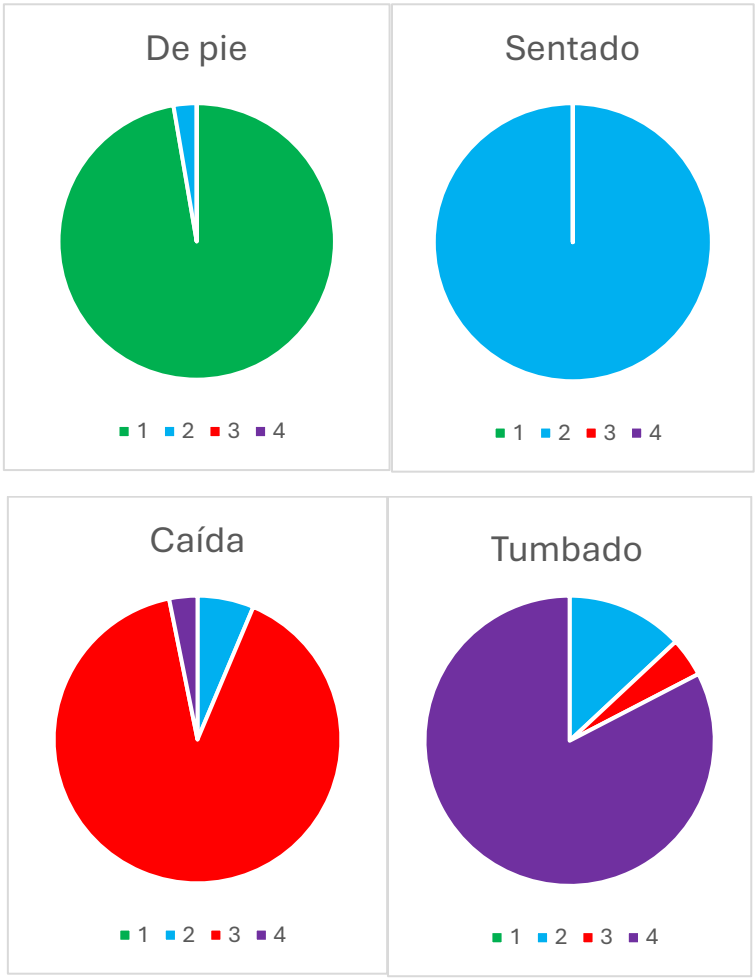


Figura 63: Gráfico resultados clasificación test método SVM. (Fuente: Elaboración propia)

Como resumen final a estos resultados se muestra el acierto global que han tenido ambos métodos (tabla 39). Con esto queda demostrado el buen funcionamiento tanto del método manual, conseguido de manera "empírica" con el análisis riguroso de las características extraídas y con una buena selección de umbrales, como del método de clasificación automático del SVM, en el que se parecía la fortaleza de generalizar muy bien en imágenes completamente nuevas a partir de un entreno previo.

	TP_c	**FP_c**
MANUAL	92,12%	7,86%
SVM	92,57%	7,43%

Tabla 39: Resultados generales de la clasificación. Conjunto de test.

5.5. Resultados en imágenes.

A continuación, se muestran algunos resultados en forma de imágenes de cómo detectan los modelos y su resultado para que se aprecie la variedad de imágenes con las que se ha trabajado en cuanto a poses, escenarios e iluminación.

5.5.1. Detecciones clase “de pie”.

Figura 64: Ejemplo de resultados del modelo en imágenes. Clase “de pie”. (Fuente: Elaboración propia)

5.5.2. Detecciones clase “sentado”.

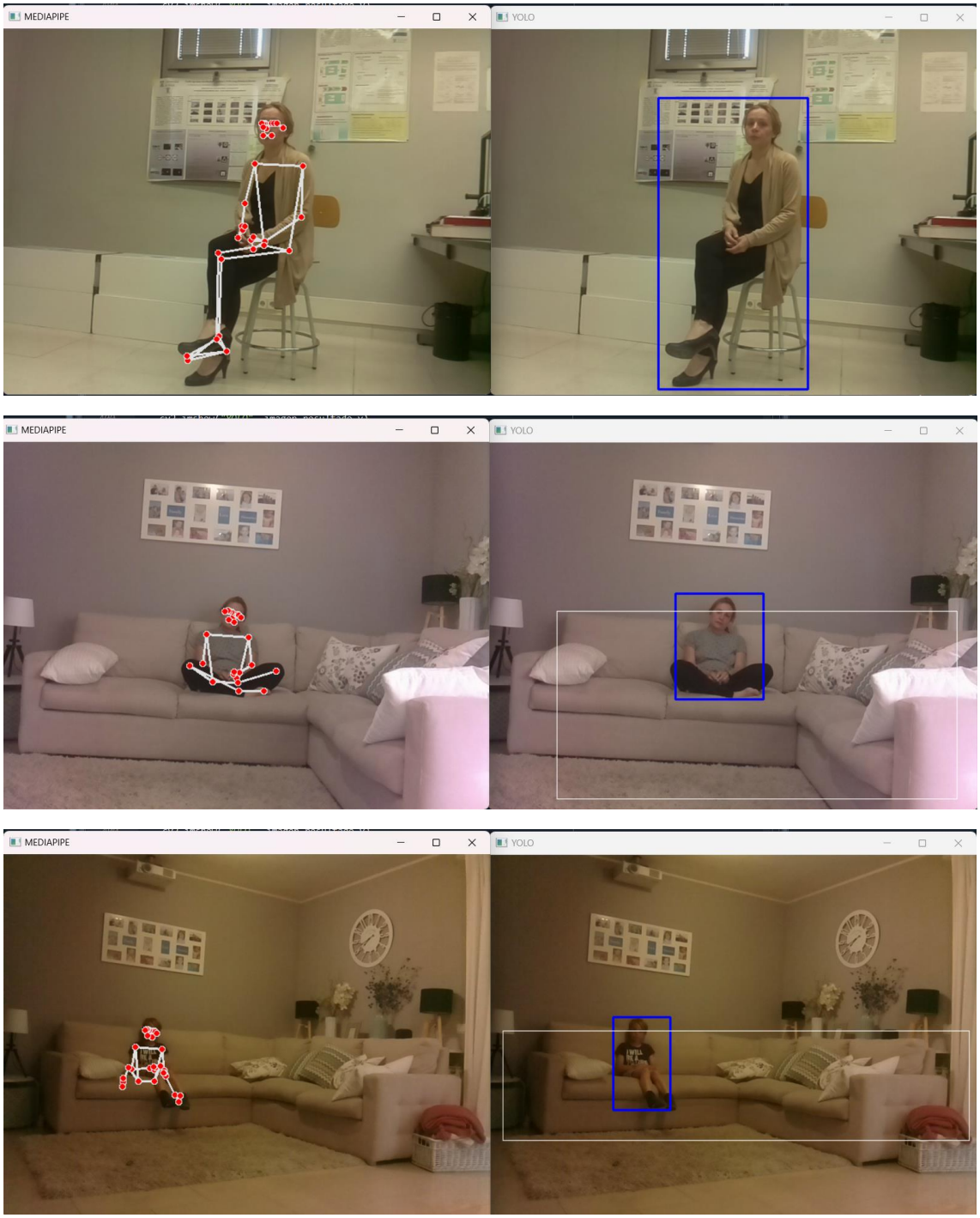


Figura 65: Ejemplo de resultados del modelo en imágenes. Clase “sentado”. (Fuente: Elaboración propia)

5.5.3. Detecciones clase “tumbado”.

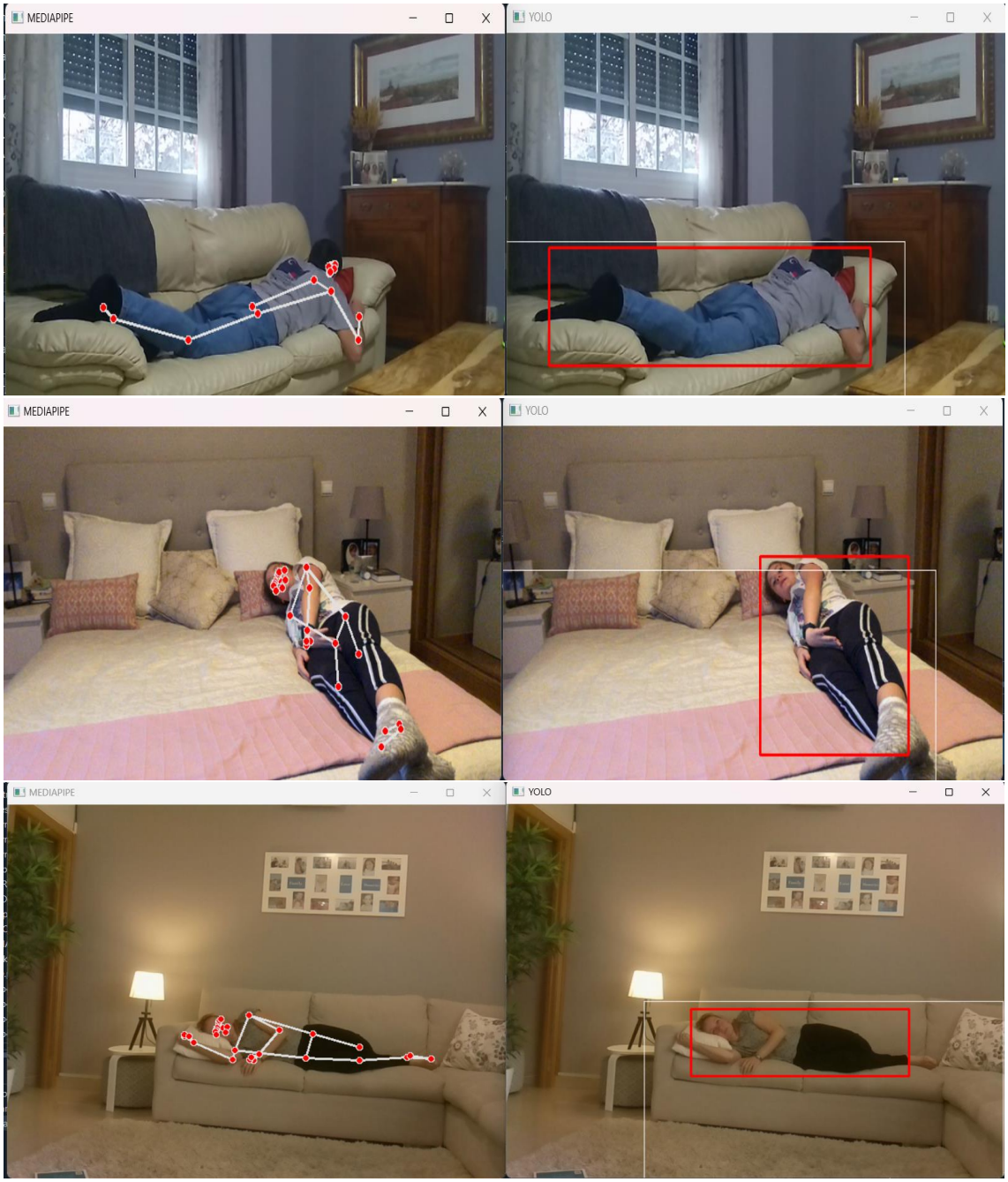


Figura 66: Ejemplo de resultados del modelo en imágenes. Clase “tumbado”. (Fuente: Elaboración propia)

5.5.4. Detecciones clase “caída”.

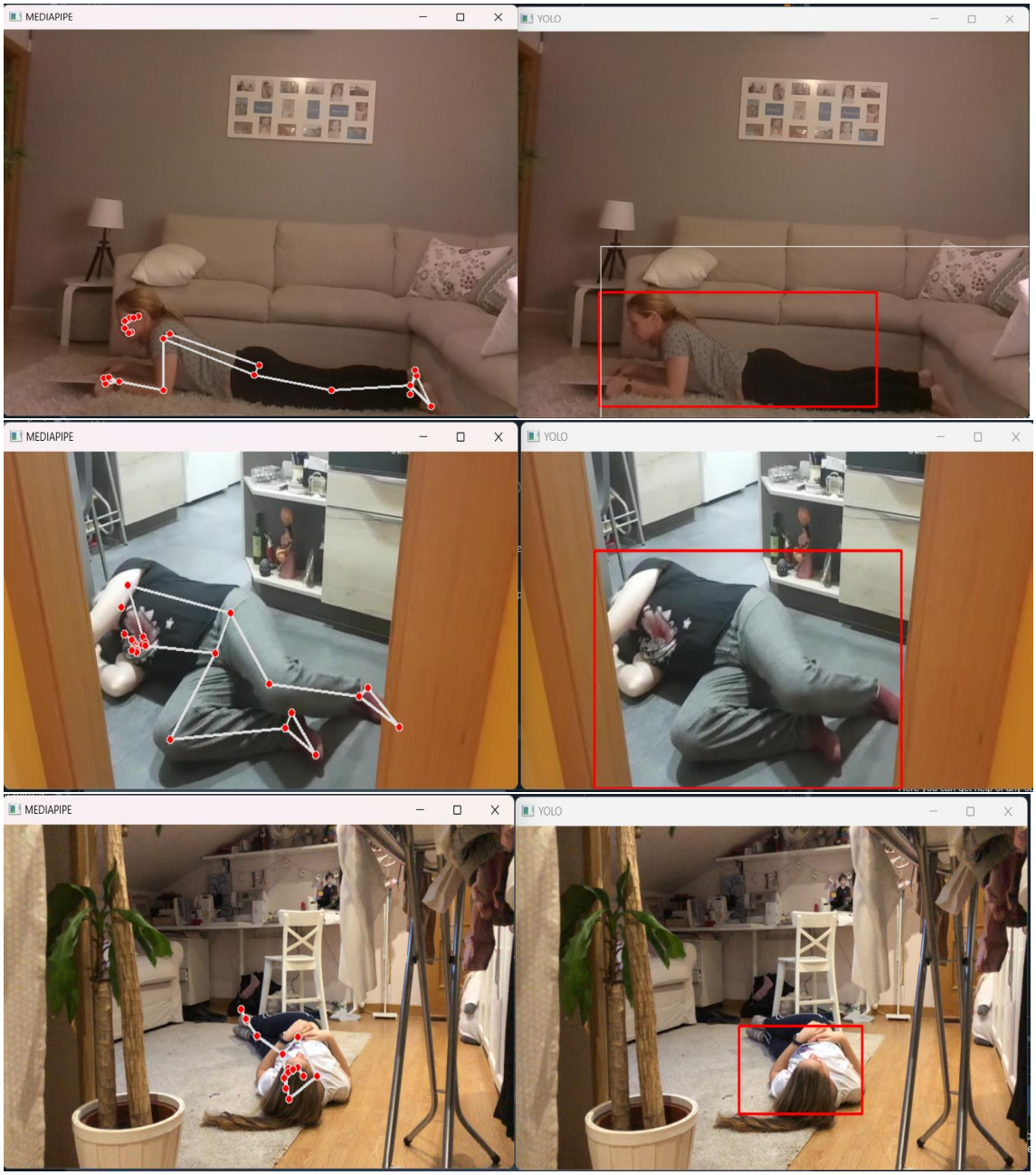


Figura 67: Ejemplo de resultados del modelo en imágenes. Clase “caída”. (Fuente: Elaboración propia)

6. Conclusiones y líneas futuras.

6.1. Conclusiones.

El presente trabajo ha desarrollado un sistema basado en visión artificial para la caracterización de la posición de personas en entornos domésticos, con un enfoque especial en personas que viven solas. Se ha diseñado una solución mediante una combinación de MediaPipe y YOLO para la extracción de características y clasificación basada en el análisis de estas en 4 clases principales: “de pie”, “sentado”, “tumbado” y “caída”.

La estructura básica del proyecto pasa por un análisis de la imagen mediante MediaPipe Pose y YOLO del que se extraen una serie de características relevantes y se seleccionan los umbrales mediante los que se realiza la clasificación final. También se ha experimentado en paralelo con un clasificador SVM al que se le pasan las características extraídas por MediaPipe y YOLO para una clasificación autónoma.

En el proceso se ha pasado por una fase de estudio de efectividad de las dos herramientas que se utilizan (MediaPipe Pose y YOLO) para elegir en qué situaciones es más robusta una que otra y así conseguir un sistema de decisión basado en los puntos fuertes de cada uno.

Se ha realizado un análisis detallado de las características extraídas de las detecciones por ambas herramientas para poder elegir cuáles de ellas eran más relevantes y distintivas de cada clase. Junto con la creación del SVM, a partir de la recolección de estas características se ha podido mejorar su elección y la creación de umbrales de clasificación más precisos.

La caracterización presentó ciertos desafíos a la hora de no inducir solapamientos con las demás clases con la selección de características y los umbrales adecuados. Estas dificultades fueron superadas con dedicación y un esfuerzo constante gracias a un exhaustivo trabajo de análisis y optimización.

Los resultados obtenidos en este proyecto han sido altamente satisfactorios, no solo en las pruebas de entrenamiento, sino también en los test realizados, donde el modelo ha demostrado una precisión destacable en la clasificación de las posturas. Estos avances no solo contribuyen al campo de la visión artificial, sino que tienen un impacto directo en la sociedad, especialmente en la detección temprana de caídas.

Desde una perspectiva práctica, este sistema y su capacidad de detección en entornos domésticos representan un paso importante hacia una mayor seguridad y bienestar para personas, especialmente aquellas de avanzada edad que viven solas o tienen problemas de movilidad.

6.2. Líneas futuras de trabajo y mejoras.

Aunque los resultados obtenidos son satisfactorios, existen varias áreas que pueden ser optimizadas y que podrían abrir nuevas posibilidades de aplicación, pues este trabajo no ha sido desarrollado con la idea de cerrar la línea de trabajo al terminarlo.

Una mejora clave sería explorar arquitecturas más complejas y avanzadas, como redes neuronales entrenadas específicamente para esta labor y conseguir un *dataset* mucho más amplio utilizado para entrenar el modelo, ya que podrían proporcionar una mayor precisión y robustez en la clasificación de posturas complicadas, especialmente en condiciones difíciles o entornos variados, incorporando una mayor diversidad de escenarios y condiciones, ayudando a mejorar la generalización del modelo y permitiendo su implementación en una mayor variedad de situaciones reales. Incluso se podrían considerar enfoques que incluyan técnicas de aprendizaje automático, permitiendo que el modelo se adapte y aprenda de nuevos datos en tiempo real, mejorando su rendimiento a medida que se recopilan más muestras.

Una de las limitaciones encontradas en este proyecto ha sido la posibilidad de trabajar con imágenes en tiempo real, lo que permitiría la monitorización continua del movimiento. Esto va de la mano con explorar arquitecturas más complejas y tener un equipo de trabajo potente para poder procesar toda esta información. Además, se podría explorar la integración de otros sensores o modalidades de datos, como sensores de movimiento o cámaras adicionales, para obtener una visión más completa y precisa del entorno y las personas, aumentando la fiabilidad del sistema.

Una línea futura de trabajo clara sería trabajar en la conectividad con los servicios médicos y familiares para poder mandar mensajes de alerta al detectar caídas. También profundizar en la posibilidad de emitir recomendaciones cuando, por ejemplo, el sistema detecte la postura “sentado” varias veces seguidas, evitando situaciones de sedentarismo.

Estas mejoras y líneas de investigación abren nuevas oportunidades para desarrollar sistemas de visión artificial aún más eficientes y adaptables, con un impacto directo en la mejora de la seguridad y calidad de vida de las personas, especialmente aquellas en situaciones vulnerables.

7. Bibliografía.

[1] Instituto Nacional de Seguridad y Salud en el Trabajo. (2012). Accidente en casa también ocurren.

https://www.insst.es/documents/94886/375493/Tr%C3%ADptico.+Accidente.+en+casa+tambi%C3%A9n+ocurren+-+A%C3%B1o+2012+%28en+cat%C3%A1logo%29.pdf/8c955888-f343-4442-ba00-f13147d3927b?t=1685613056653

Consultado por última vez el 12 de noviembre de 2024

[2] Organización Mundial de la Salud. (2021). Caídas.

https://www.who.int/es/news-room/fact-sheets/detail/falls

Consultado por última vez el 12 de noviembre de 2024

[3] Pontificia Universidad Católica de Chile. (2021, 6 de septiembre). Entre 7 y 11,5 horas es el tiempo que pasamos sentados durante el día en promedio.

https://www.uc.cl/noticias/entre-7-y-11-5-horas-es-el-tiempo-que-pasamos-sentados-durante-el-dia-en-promedio/

Consultado por última vez el 12 de noviembre de 2024

[4] Sanidad, Ministerio de. (2017). *Encuesta Nacional de Salud 2017: Actividad física.*

https://www.sanidad.gob.es/estadEstudios/estadisticas/encuestaNacional/encuestaNac2017/ACTIVIDAD_FISICA.pdf

Consultado por última vez el 12 de noviembre de 2024

[5] INTEC ROBOTS S.L. (s.f.). TEMI - Robot asistente personal - Telepresencia.

https://intecrobots.com/temi-robot-asistente-personal-telepresencia-intec-robots/

Consultado por última vez el 3 de marzo de 2025

[6] *Wikipedia.* (2023). *Visión artificial.*

https://es.wikipedia.org/wiki/Visi%C3%B3n_artificial

Consultado por última vez el 22 de noviembre de 2024

[7] *IBM.* (n.d.). *Machine learning.*

https://www.ibm.com/es-es/topics/*machine-learning*

Consultado por última vez el 22 de noviembre de 2024

[8] *IBM.* (n.d.). *Deep learning.*

https://www.ibm.com/es-es/topics/*deep-learning*

Consultado por última vez el 27 de noviembre de 2025

[9] *MathWorks.* (n.d.). *Convolutional neural network.*

https://es.mathworks.com/discovery/convolutional-neural-network.html

Consultado por última vez el 14 de febrero de 2025

[10] *Wikipedia. (s.f.). Postura.*

https://es.wikipedia.org/wiki/Postura

Consultado por última vez el 19 de diciembre de 2024

[11] *López, J. (2021, 22 de noviembre). Estimación de la pose con deep learning. Campus Big Data*

https://www.campusbigdata.com/blog/estimacion-de-la-pose-con-*deep-learning/*

Consultado por última vez el 14 de febrero de 2025

[12] *Roboflow. (2021, diciembre 21). What is OpenPose?*

https://blog.roboflow.com/what-is-openpose/

Consultado por última vez el 3 de marzo de 2025

[13] Jardin, A. (n.d.). ZED-OpenPose. GitHub.

https://github.com/adujardin/zed-openpose?ref=blog.roboflow.com

Consultado por última vez el 3 de marzo de 2025

[14] Encord. (n.d.). AlphaPose definition.

https://encord.com/glossary/alphapose-definition/

Consultado por última vez el 3 de marzo de 2025

[15] Viso.ai. (n.d.). AlphaPose: Real-time multi-person pose estimation.

https://viso.ai/*deep-learning*/alphapose/

Consultado por última vez el 3 de marzo de 2025

[16] Del Valle Hernández, L. (2020, octubre 13). Sensor Kinect e inteligencia artificial [Audio podcast]. Programar Fácil.

https://programarfacil.com/podcast/86-sensor-kinect-inteligencia-artificial/

Consultado por última vez el 3 de marzo de 2025

[17] Hacedores. (n.d.). Introducción a Kinect.

https://hacedores.com/introduccion-a-kinect/

Consultado por última vez el 3 de marzo de 2025

[18] Elforaici, I., El Abed, A., & Kharroubi, S. (2018). Reconocimiento de posturas utilizando una cámara RGB-D: explorando modelado corporal 3D y enfoques de aprendizaje profundo. arXiv.

https://arxiv.org/abs/1810.00308

Consultado por última vez el 13 de Enero de 2025

[19] Zhu, X., Zhang, Y., Wang, L., & Li, J. (2024). Combined MediaPipe and YOLOv5 range of motion assessment system for spinal diseases and frozen shoulder. Scientific Reports, 14(1), 12345.

https://doi.org/10.1038/s41598-024-66221-8

Consultado por última vez el 13 de Enero de 2025

[20] Wikipedia. (n.d.). Anaconda (distribución de Python).

https://es.wikipedia.org/wiki/Anaconda_(distribuci%C3%B3n_de_Python)

Consultado por última vez el 19 de diciembre de 2024

[21] Crehana. (2020, 22 de septiembre). ¿Qué es OpenCV?

https://www.crehana.com/blog/transformacion-digital/que-es-opencv/

Consultado por última vez el 20 de diciembre de 2024

[22] Innovatiana. (n.d.). MediaPipe.

https://es.innovatiana.com/post/MediaPipe-101

Consultado por última vez el 20 de diciembre de 2024

[23] Google AI. (n.d.). Pose Landmarker. Google AI.

https://ai.google.dev/edge/MediaPipe/solutions/vision/pose_*landmark*er?hl=es-419

Consultado por última vez el 21 de Diciembre de 2024

[24] Wikipedia. (s.f.). Algoritmo You Only Look Once (YOLO). Wikipedia.

https://es.wikipedia.org/wiki/Algoritmo_You_Only_Look_Once_(YOLO)

Consultado por última vez el 21 de diciembre de 2024

[25] Viso AI. (n.d.). YOLOv3 overview: The basics of YOLO and how it works.

https://viso.ai/*deep-learning*/yolov3-overview/

Consultado por última vez el 21 de diciembre de 2024

[26] Elakkiya, R., Kumar, P. M., Krishnan, M., & Balamurugan, B. (2021). Vehicle detection from aerial images using deep learning: A comparative study. ResearchGate.

https://www.researchgate.net/publication/350502286

Consultado por última vez el 5 de marzo de 2025

[27] Reddie, P. J. (s.f.). YOLO: Real-Time Object Detection. Pjreddie.com.

https://pjreddie.com/darknet/yolo/

Consultado por última vez el 21 de diciembre de 2024

[28] GeeksforGeeks. (n.d.). Support Vector Machine algorithm.

https://www.geeksforgeeks.org/*support*-vector-*machine*-algorithm/

Consultado por última vez el 10 de enero de 2025

[29] Fallen People Detection Capabilities Using Assistive Robot. S. Maldonado-Bascón, C. Iglesias-Iglesias, P. Martín-Martín, S. Lafuente-Arroyo. Electronics 2019.

https://gram.web.uah.es/data/datasets/fpds/index.html

Consultado por última vez el 11 de enero de 2025

[30] IASLAB-RGBD (Antonello, M.; Carraro, M.; Pierobon, M.; Menegatti, E. Fast and robust detection of fallen people from a mobile robot. 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)).

https://gram.web.uah.es/data/datasets/fpds/index.html

Consultado por última vez el 11 de enero de 2025

[31] Up-Fall Detection (Kwolek, B.; Kepski, M. Human fall detection on embedded platform using depth maps and wireless accelerometer. Computer methods and programs in biomedicine 2014).

https://gram.web.uah.es/data/datasets/fpds/index.html

Consultado por última vez el 11 de enero de 2025

[32] Elderly Set. S. Maldonado-Bascón, C. Iglesias-Iglesias, P. Martín-Martín, S. Lafuente-Arroyo. Electronics 2019.

https://gram.web.uah.es/data/datasets/fpds/index.html

Consultado por última vez el 11 de enero de 2025

[33] Gaby Sol. (2020). *OmesTutorials2020.* GitHub.

https://github.com/GabySol/OmesTutorials2020

Consultado por última vez el 23 de enero de 2025

[34] GitHub. (2020). *MediaPipe Pose Classification.*

https://github.com/google/MediaPipe/blob/master/docs/solutions/pose_classification.md

Consultado por última vez el 23 de enero de 2025

[35] Scikit-learn. (n.d.). *Confusion matrix.*

https://scikit-learn.org/1.5/auto_examples/model_selection/plot_confusion_matrix.html

Consultado por última vez el 24 de enero de 2025

[36] TensorFlow Blog. (2021). *High fidelity pose tracking with MediaPipe, BlazePose, and TFJS.*

https://blog.tensorflow.org/2021/05/high-fidelity-pose-tracking-with-MediaPipe-blazepose-and-tfjs.html

Consultado por última vez el 24 de enero de 2025

[37] UVA. (2023). *Documento.*

https://apps.stic.uva.es/guias_docentes/uploads/2023/452/42388/1/Documento.pdf

Consultado por última vez el 22 de noviembre de 2024

[38] Yolo11 Fall Detection. (n.d.). *YOLO v3 Object Detection Track.* GitHub.

https://github.com/freedomwebtech/yolo11fall/blob/main/yolo11objectdetection-track.py

Consultado por última vez el 26 de enero de 2025

8. Anexos:

8.1. Anexo 1: Lista de programas desarrollados.

“FUNCIÓN principal TFG Rubén.py”: Función principal del proyecto. Desarrollada para conseguir una clasificación de la postura de la persona presente en una imagen entre 5 clases predominantes, “de pie”, “sentado”, “tumbado”, “caída” y “desconocido”.

“clasificador imágenes en carpetas.py”: Función desarrollada para conseguir clasificar y filtrar las imágenes de los *datasets* utilizados a lo largo del proyecto.

“características MEDIAPIPE a txt.py”: Función desarrollada para pasar todas las características extraídas por MediaPipe a un txt. Se ha utilizado para ordenar estas características por clases y poder trabajar con ellas para la clasificación final.

“características YOLO a txt.py”: Función desarrollada para pasar todas las características extraídas por YOLO a un txt. Se ha utilizado para ordenar estas características por clases y poder trabajar con ellas para la clasificación final.

“SVM con datos.py”: Función desarrollada para crear y entrenar los algoritmos SVM utilizados en el proyecto con los datos extraídos a través de MediaPipe y YOLO. De este programa surgen los archivos con los SVM entrenados y sus escaladores para normalizar las características.

- “SVM_MY.pkl” y “scaler_MY.pkl”
- “SVM_YOLO.pkl” y “scaler_YOLO.pkl”
- “SVM_MEDIAPIPE.pkl” y “scaler_MEDIAPIPE.pkl”

8.2. Anexo 2: Lista de librerías del entorno de Python.

Nombre	Versión
absl-py	2.1.0
alabaster	0.7.12
arrow	1.2.3
astroid	2.14.2
asttokens	2.0.5
atomicwrites	1.4.0
attrs	23.2.0
autopep8	2.0.4
babel	2.11.0
backcall	0.2.0
bcrypt	3.2.0
beautifulsoup4	4.12.2
binaryornot	0.4.4
black	23.11.0
blas	1.0
bleach	4.1.0
bottleneck	1.3.7
brotli-python	1.0.9
ca-certificates	2024.9.24
certifi	2024.8.30
cffi	1.16.0
chardet	4.0.0
charset-normalizer	2.0.4
click	8.1.7
cloudpickle	2.2.1
colorama	0.4.6
comm	0.2.1
contourpy	1.2.0
cookiecutter	2.6.0
cryptography	42.0.5
cycler	0.12.1
debugpy	1.6.7
decorator	5.1.1
defusedxml	0.7.1
diff-match-patch	20200713
dill	0.3.7
docstring-to-markdown	0.11
docutils	0.18.1
exceptiongroup	1.2.0
executing	0.8.3
filelock	3.16.1

flake8	7.0.0
flatbuffers	24.3.25
fonttools	4.50.0
fsspec	2024.10.0
icc_rt	2022.1.0
icu	73.1
idna	3.4
imagesize	1.4.1
importlib-metadata	7.1.0
importlib-resources	6.4.0
importlib_metadata	7.0.1
inflection	0.5.1
intel-openmp	2021.4.0
intervaltree	3.1.0
ipy*kernel*	6.28.0
ipython	8.15.0
isort	5.9.3
jaraco.classes	3.2.1
jax	0.4.25
jedi	0.18.1
jellyfish	1.0.1
jinja2	3.1.3
joblib	1.4.2
jpeg	9e
jsonschema	4.19.2
jsonschema-specifications	2023.7.1
jupyter_client	8.6.0
jupyter_core	5.5.0
jupyterlab_pygments	0.2.2
keyring	24.3.1
kiwisolver	1.4.5
krb5	1.20.1
lazy-object-proxy	1.6.0
libclang	14.0.6
libclang13	14.0.6
libpng	1.6.39
libpq	12.17
libsodium	1.0.18
libspatialindex	1.9.3
lz4-c	1.9.4
markdown-it-py	2.2.0
markupsafe	2.1.3
matplotlib	3.8.3
matplotlib-inline	0.1.6
mccabe	0.7.0
mdurl	0.1.0

MediaPipe	0.10.11
mistune	2.0.4
mkl	2021.4.0
mkl-service	2.4.0
mkl_fft	1.3.1
mkl_random	1.2.2
ml-dtypes	0.3.2
more-itertools	10.1.0
mpmath	1.3.0
mypy_extensions	1.0.0
nbclient	0.8.0
nbconvert	7.10.0
nbformat	5.9.2
nest-asyncio	1.6.0
networkx	3.2.1
numexpr	2.8.4
numpy	1.26.4
numpy-base	1.24.3
numpydoc	1.5.0
opencv-contrib-python	4.9.0.80
opencv-python	4.10.0.84
openssl	3.0.13
opt-einsum	3.3.0
packaging	24.0
pandas	2.0.3
pandocfilters	1.5.0
paramiko	2.8.1
parso	0.8.3
pathspec	0.10.3
pexpect	4.8.0
pickleshare	0.7.5
pillow	10.2.0
pip	23.3.1
platformdirs	3.10.0
pluggy	1.0.0
ply	3.11
pooch	1.8.2
prompt-toolkit	3.0.43
protobuf	3.20.3
psutil	5.9.0
ptyprocess	0.7.0
pure_eval	0.2.2
py-cpuinfo	9.0.0
pycodestyle	2.11.1
pycparser	2.21
pydocstyle	6.3.0

pyflakes	3.2.0
pygments	2.15.1
pylint	2.16.2
pylint-venv	3.0.3
pyls-spyder	0.4.0
pynacl	1.5.0
pyparsing	3.1.2
pyqt	5.15.10
pyqt5-sip	12.13.0
pyqtwebengine	5.15.10
pysocks	1.7.1
python	3.9.19
python-dateutil	2.9.0.post0
python-fastjsonschema	2.16.2
python-lsp-black	2.0.0
python-lsp-jsonrpc	1.1.2
python-lsp-server	1.10.0
python-slugify	5.0.2
python-tzdata	2023.3
pytoolconfig	1.2.6
pytz	2023.3.post1
pywin32	305
pywin32-ctypes	0.2.2
pyyaml	6.0.1
pyzmq	25.1.2
qdarkstyle	3.2.3
qstylizer	0.2.2
qt-main	5.15.2
qt-webengine	5.15.9
qtawesome	1.2.2
qtconsole	5.5.1
qtpy	2.4.1
referencing	0.30.2
requests	2.31.0
rich	13.3.5
rope	1.12.0
rpds-py	0.10.6
rtree	1.0.1
scikit-learn	1.3.0
scipy	1.12.0
seaborn	0.13.2
setuptools	68.2.2
sip	6.7.12
six	1.16.0
snowballstemmer	2.2.0
sortedcontainers	2.4.0

sounddevice	0.4.6
soupsieve	2.5
sphinx	5.0.2
sphinxcontrib-applehelp	1.0.2
sphinxcontrib-devhelp	1.0.2
sphinxcontrib-htmlhelp	2.0.0
sphinxcontrib-jsmath	1.0.1
sphinxcontrib-qthelp	1.0.3
sphinxcontrib-serializinghtml	1.1.5
spyder	5.5.1
spyder-*kernel*s	2.5.0
sqlite	3.41.2
stack_data	0.2.0
sympy	1.13.1
text-unidecode	1.3
textdistance	4.2.1
threadpoolctl	3.5.0
three-merge	0.1.1
tinycss2	1.2.1
tomli	2.0.1
tomlkit	0.11.1
torch	2.5.1
torchvision	0.20.1
tornado	6.3.3
tqdm	4.67.0
traitlets	5.7.1
typing-extensions	4.9.0
typing_extensions	4.9.0
tzdata	2024a
ujson	5.4.0
ultralytics	8.3.36
ultralytics-thop	2.0.12
unidecode	1.2.0
urllib3	2.1.0
vc	14.2
vs2015_runtime	14.27.29016
watchdog	2.1.6
wcwidth	0.2.5
webencodings	0.5.1
whatthepatch	1.0.2
wheel	0.41.2
win_inet_pton	1.1.0
wrapt	1.14.1
xz	5.4.6
yaml	0.2.5
yapf	0.40.2

zeromq	4.3.5
zipp	3.18.1
zlib	1.2.13
zstd	1.5.5