Simposio CEA de Robótica, Bioingeniería, Visión por Computador y Automática Marina 2026

Patrullaje y detección autónoma de caídas mediante el robot asistencial ADAM

Lishan, L., Proctor, F., Mora, A., Mendez, A., Garrido, S., Barber, R.

RoboticsLab, Departamento de Ingeniería de Sistemas y Automática, Universidad Carlos III de Madrid, Av. de la Universidad, 30, 28911 Leganés, España.

Resumen

El envejecimiento poblacional global incrementa la demanda de soluciones tecnológicas para la seguridad de los mayores, especialmente ante el riesgo de caídas. Este trabajo presenta un sistema de patrullaje autónomo en simulación mediante el robot asistencial ADAM para la detección de personas en el suelo. La propuesta se centra en una estrategia de navegación que optimiza la selección de *waypoints* diferenciando entre zonas pequeñas, inspeccionadas desde el umbral, y áreas amplias con posicionamiento central. Se integra un sistema de visión multimodal, incorporando un protocolo de verificación ante detecciones inciertas para minimizar falsos positivos. Todo el comportamiento es gestionado por una máquina de estados que coordina el patrullaje, la verificación visual, la aproximación táctica y la respuesta de emergencia. Los resultados validan la eficacia de ADAM como herramienta de apoyo al personal sanitario, mejorando la vigilancia y los tiempos de respuesta en entornos asistenciales.

Palabras clave: Detección de caídas, Patrullaje autónomo, Percepción y detección en robots, Planificación de tareas y movimientos, Navegación autónoma, Robot asistencial, Robot móvil

Autonomous patrolling and fall detection using the ADAM assistive robot

Abstract

Global population aging increases the demand for technological solutions to ensure the safety of the elderly, particularly regarding fall risks. This paper presents a simulated autonomous patrolling system using the ADAM healthcare robot for detecting persons on the ground. The proposal focuses on a navigation strategy that optimizes waypoint selection by differentiating between small areas, inspected from the threshold, and large areas requiring central positioning. A multimodal vision system is integrated, incorporating a verification protocol for uncertain detections to minimize false positives. The entire behavior is managed by a finite state machine that coordinates patrolling, visual verification, tactical approach, and emergency response. Experimental results validate ADAM's effectiveness as a support tool for healthcare personnel, improving surveillance and response times in caregiving environments.

Keywords: Fall detection, Autonomous patrolling, Robot perception and sensing, Task and motion planning, Autonomous navigation, Assistant robot, Mobile robot

1. Introducción

El envejecimiento de la población es un fenómeno creciente a nivel global, lo que plantea importantes desafíos para la salud y la seguridad de las personas mayores. Entre los riesgos más relevantes se encuentran las caídas, que representan una de las principales causas de lesiones graves y hospitalizaciones en adultos mayores. Según la Organización Mundial de la Salud (OMS), aproximadamente un tercio de las personas mayores de 65 años sufre al menos una caída al año. La detección temprana de caídas y la capacidad de respuesta rápida son, por tanto, factores críticos para reducir la gravedad de sus consecuencias. En este contexto, la robótica de servicio surge como una solución prometedora; los robots móviles poseen capacidades que les permiten realizar un patrullaje constante para comprobar el estado de los pacientes, liberando así tiempo del personal de enfermería y garantizando una vigilancia continua que el ojo humano no siempre puede cubrir.

Para abordar este reto, el presente trabajo propone la implementación de un sistema de vigilancia activa mediante el uso

*Autor para correspondencia: lucialis@pa.uc3m.es

del robot móvil manipulador ADAM (Mora et al., 2024). Se introduce una lógica de navegación y un motor de decisiones capaces de adaptarse a distintos escecnarios, mejorando la eficiencia y fiabilidad del patrullaje. En resumen, las principales aportaciones son:

- **Estrategia de navegación diferenciada:** Diseño de un sistema de selección de waypoints basado en el tipo de estancia (zonas pequeñas vs. áreas amplias) para optimizar la eficiencia del patrullaje.
- **Sistema de visión multimodal:** El algoritmo de detección de caídas combina información RGB y nubes de puntos para aumentar la fiabilidad en la identificación.
- **Mecanismo de verificación de incertidumbre:** Implementación de un proceso de aproximación y re-escaneo cuando la detección visual es ambigua, reduciendo las falsas alarmas.
- **Arquitectura de control integrada:** Diseño de una máquina de estados robusta que coordina el patrullaje, la verificación y los protocolos de emergencia en el robot.

La validación de este sistema se ha llevado a cabo en un entorno de simulación avanzado que replica condiciones habitacionales complejas. Los resultados obtenidos muestran que la diferenciación entre el patrullaje desde la puerta y el patrullaje central permite una cobertura exhaustiva. Asimismo, el sistema de visión demuestra ser robusto para distinguir personas caídas, posicionando a ADAM como una plataforma robótica eficaz para la patrulla en entornos asistenciales.

2. Estado del arte

Desarrollar un sistema de patrulla requiere dotar al robot de estrategias de navegación y de algoritmos de detección de caída robustos para cumplir este propósito de forma precisa y segura.

2.1. Sistema de patrulla

En el caso del patrullaje asistencial no solo interesa que el robot vaya de un punto a otro por la ruta más corta, sino que recorra todo el área necesaria para la supervisión. Una técnica eficaz para definir estas rutas de patrullaje es obtener puntos de paso (*waypoints*) por los que el robot debe transitar obligatoriamente para asegurar que no queden zonas sin vigilar. Para calcular la mejor manera de unir estos puntos, se emplean métodos geométricos como la segmentación *watershed* o los diagramas de Voronoi. Estas estrategias dividen el espacio en zonas más simples y trazan caminos seguros que alejan al robot de las paredes, haciendo que la navegación sea mucho más rápida y ordenada (Mora et al., 2022). Por otro lado, sistemas enfocados puramente en la vigilancia introducen algoritmos de patrullaje de revisión rápida (Kachavarapu et al., 2025), que generan y optimizan los *waypoints* de forma matemática para garantizar que el robot tarde el menor tiempo posible en volver a pasar por una zona crítica.

Para resolver los imprevistos que surgen en el trayecto, métodos clásicos como el algoritmo de Dijkstra extendido permiten recalcular la trayectoria y esquivar obstáculos usando mapas que se actualizan al momento con los sensores del robot (Teh et al., 2021). También existen propuestas que usan algoritmos heurísticos para equilibrar el ahorro de batería con una buena cobertura del hospital (Šelek et al., 2023). Más recientemente, se apuesta por el Aprendizaje por Refuerzo Profundo (DRL) y redes neuronales de grafos para ayudar al robot a aprender mejores rutas, evitando repetir los mismos pasillos vacíos y haciendo el patrullaje efectivo. (Tankasala et al., 2024).

2.2. Sistema de percepción de caídas

Para la detección de caídas se utilizan principalmente modelos de aprendizaje profundo que estiman la postura de las personas (Sarobin et al., 2022). Usualmente se basan en MediaPipe, popular por su bajo coste de procesamiento, aunque presenta limitaciones cuando hay varias personas juntas o mobiliario ocluyendo la vista. Por este motivo, en los últimos años se imponen los modelos basados en YOLO, por su capacidad para estimar el esqueleto de varias personas a la vez de forma robusta (Yu et al., 2025). Un ejemplo en robótica es la combinación de redes ConvNeXt y YOLO, que logra una precisión muy alta en entornos domésticos (Sánchez-Girón et al., 2025). También ganan fuerza los modelos *zero-shot*, que no requieren entrenamiento con miles de imágenes específicas de caídas y son útiles para posturas poco comunes (p. ej. personas con andadores o prótesis), reduciendo el margen de error (Zhou et al., 2025).

2.3. Aplicación de detección de caídas para asistencia

Combinando patrullaje y detección de caídas se pueden construir sistemas de vigilancia asistencial. En (Sánchez-Girón et al., 2025) se propone un sistema de patrullaje inteligente con este fin, prestando un servicio ininterrumpido y aliviando la carga de vigilancia del personal clínico (De Miguel et al., 2017). Llevar esto a la práctica en hospitales reales ha demostrado que los robots pueden bloquearse en pasillos o generar falsas alarmas si el sistema no es lo bastante robusto, lo que en ocasiones incrementa la carga del personal (Gebellí and Ros, 2025).

Nuestra propuesta adopta un sistema modular de extremo a extremo que, en lugar de concentrar todas las decisiones en un único modelo, organiza el proceso en módulos independientes orquestados por una máquina de estados. Este diseño favorece la evaluación, el testeo y la detección de errores de cada componente de manera individual, facilitando además el mantenimiento y la extensibilidad del sistema.

3. Robot ADAM simulado

Para las pruebas y la evaluación se utiliza una versión simulada en Gazebo del robot ADAM (*Autonomous Domestic Ambidextrous Manipulator*), un robot destinado a la asistencia física y cuidado de personas mayores en entornos domésticos como hogares y residencias. Este robot se compone de distintos módulos para percibir el entorno, navegar por él y manipular los elementos que se encuentran en él. En este trabajo nos centramos en los dos primeros, de forma que la información percibida por el sensor LiDAR 2D de la base RB1 de Robotnik y la información visual captada por la cámara RGB-D RealSense D435i es procesada para desplazar el robot por el entorno simulado. Los módulos de percepción y visión se integran mediante ROS junto con Gazebo en una arquitectura software que replica la lógica real del robot para maximizar la transferibilidad.

4. Descripción del sistema

El sistema ha sido desarrollado en Python 3 sobre el framework ROS Noetic, siguiendo una arquitectura modular basada en nodos. La implementación se compone de seis nodos agrupados conceptualmente en tres módulos funcionales (Figura 1):

- **Módulo de percepción**: encargado de la detección, análisis de personas e identificación de posibles caídas a partir de la información sensorial.
- **Módulo decisión**: trata de una máquina de estados finitos. Evalúa la información proporcionada por el módulo de percepción y determina la acción a realizar.
- **Módulo de actuación**: incluye el cálculo de *waypoints* sobre el mapa global, la patrulla autónoma del entorno, acercamiento a la persona detectada y la activación de un protocolo de alerta en caso de confirmación de caída.

Además, se han diseñado mensajes ROS específicos:

- **PersonInfo.msg**: contiene un identificador de persona; el estado de percepción que puede ser `OK`, `UNCERTAIN` o `FALL`; el centroide 2D del *bounding box* en imagen y la posición 3D estimada en el mundo.
- **PersonInfoArray.msg**: agrupa un conjunto de mensajes `PersonInfo` en un único mensaje, junto con un `header`.
- **ApproachCommand.msg**: incluye un booleano para activar o desactivar el nodo de acercamiento, el identificador de la persona y el centroide de su *bounding box*.
- **AlertCommand.msg**: contiene un booleano que indica si la alerta está activa, el identificador de la persona, su posición 3D en el mapa y la imagen capturada por el robot en tiempo real.

4.1. Módulo de percepción

En este módulo se encuentra el nodo de detección de caídas, suscrito a los *topics* de la imagen y nube de puntos publicados por la cámara. El sistema considera que una persona ha sufrido una caída cuando presenta una elevada inclinación corporal, aplicando además un filtrado por altura para descartar falsos positivos, como personas tumbadas en cama. Para ello, se emplean modelos de predicción de YOLO11m, elegidos por su equilibrio óptimo entre precisión y velocidad de procesamiento, capacidad para la detección simultánea de múltiples personas y su robustez ante diferencias antropométricas y de posturas.

El algoritmo se estructura en tres fases jerárquicas: detección, estimación de pose y segmentación 3D. De manera que los análisis más costosos se ejecutan solo cuando son necesarios, optimizando eficiencia. Además, se emplea el modo *track* de YOLO para mantener la identidad de cada individuo entre fotogramas y evitar análisis repetitivo.

En la primera fase se lleva a cabo la detección de personas mediante el modelo YOLO Detection, y se aplica un primer filtrado rápido basado en el análisis geométrico del *bounding box* para descartar aquellos casos claramente no relevantes. En concreto, se evalúa la relación de aspecto (ancho/alto) del *bounding box*. Los valores elevados se asocian a una postura vertical, mientras que los valores bajos indican un posible caso de caída y cuando el resultado es ambiguo, la detección se deriva a la segunda fase.

La segunda fase ayuda a excluir casos como personas sentadas o agachadas. Se centra en el estudio de la inclinación del torso de la persona mediante YOLO Pose Estimation, obteniendo los cuatro *keypoints* correspondientes a hombros y caderas. A partir de estos puntos se calcula la pendiente del segmento que une el centro de los hombros con el centro de las caderas. Cuando la pendiente es elevada, la persona es candidata a caída.

En el caso de tener candidatos de caída, se transita a la tercera fase, cuyo objetivo principal es descartar casos en los que no corresponde a una caída, pese a una elevada inclinación corporal, como cuando está recostada sobre una superficie elevada. Para ello, primero se obtiene la máscara 2D de la persona mediante YOLO Segmentation. A continuación, esta máscara se proyecta sobre la nube de puntos, extrayendo los puntos correspondientes a la persona. Luego, se calcula el centroide de dicha nube y se transforma al sistema de referencia del mapa global, obteniendo la posición 3D (x, y, z) de la persona en el entorno. Si el valor de la coordenada z es bajo, quiere decir que la persona está cerca del suelo y, por tanto, trata de una caída.

Con el fin de abordar escenarios multipersona, cada individuo detectado se procesa de forma independiente a lo largo de las distintas fases, guardando la información analizada de cada uno en un mensaje `PersonInfo`. Posteriormente, estos mensajes se agrupan en un `PersonInfoArray`, que se publica como un *topic*. Cabe señalar que una persona se clasifica como `UNCERTAIN` cuando falla la estimación de los cuatro *keypoints* requeridos o el cálculo de la posición 3D de la persona.

Por otra parte, para mantener la coherencia del análisis entre fases y evitar interferencias entre individuos, la imagen de entrada de las dos fases posteriores se construye a partir de la región de interés (ROI) definida por su *bounding box* en la fase de detección. Concretamente, se rellenan los píxeles fuera de la región con valores nulos, limitando la visión al individuo. Se puede ver un ejemplo en la Figura 2.

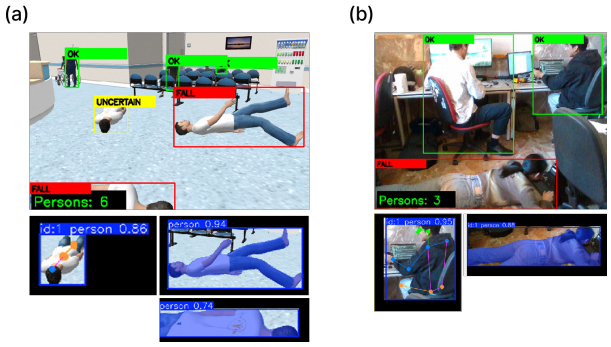

Figura 2: (a) Funcionamiento del nodo de detección de caídas en simulado. (b) Funcionamiento del nodo de deteccion de caídas en real.

4.2. Módulo de decisión

La coordinación completa del sistema se consigue mediante una máquina de estados implementada con SMACH. En total, hay cinco estados y las transiciones entre ellos se pueden visualizar en la Figura 1:

- **Estado INICIO**: fase de espera de 10 segundos para que

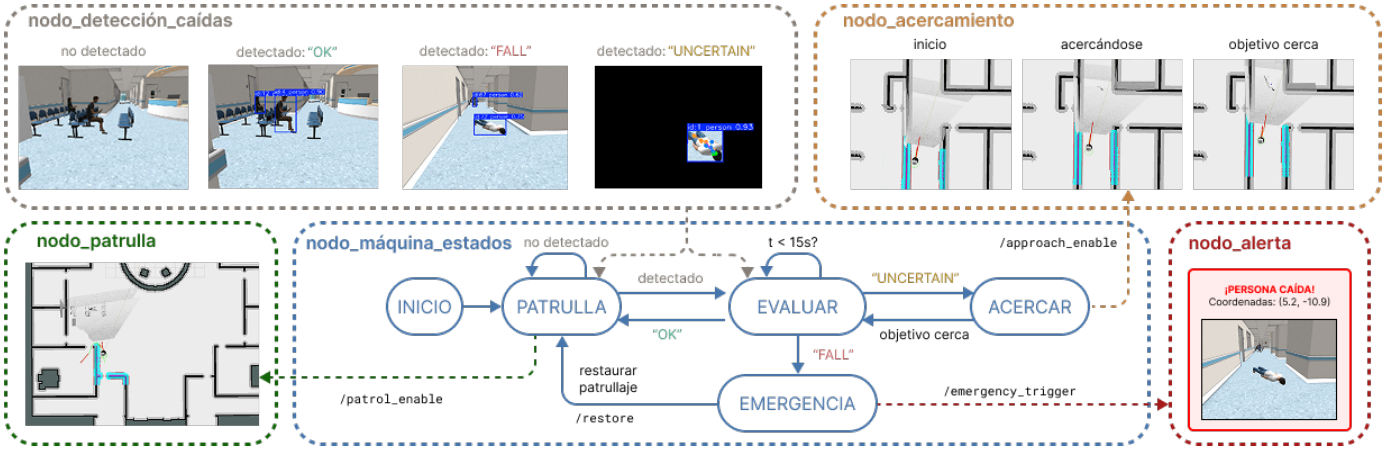

Figura 1: Funcionamiento del sistema de patrulla y detección de caídas. Durante la tarea, se ejecutan cinco nodos. Solo el nodo de detección de caídas permanece siempre activo y publicando información para la máquina de estados; los demás se activan mediante los topics */patrol_enable*, */approach_enable* y */emergency_trigger*, gestionados por la máquina de estados.

el resto de nodos estén operativos. Una vez transcurrido este tiempo, la máquina pasa al estado PATRULLA.

- **Estado PATRULLA**: al entrar en este estado, se activa el nodo de patrulla. En cada ciclo, el estado consulta el *array* de personas publicada por el nodo de detección de caídas. Si aparece alguna persona en estado `FALL` o `UNCERTAIN`, se transita al estado EVALUAR.
- **Estado EVALUAR**: se evaluá el estado de cada persona recibida. Si alguna está bajo el estado `UNCERTAIN`, se transita a ACERCAR para intentar mejorar la observación. Si alguna presenta estado `FALL`, se inicia o se mantiene un temporizador de confirmación (por defecto 15 s). Si el tiempo transcurrido desde la primera detección de caída supera ese umbral, se transita a EMERGENCIA; en caso contrario, se permanece en EVALUAR, evitando así falsas alarmas por detecciones momentáneas. Si todas las personas se encuentran en estado `OK`, se reanuda la misión transitando a PATRULLA.
- **Estado ACERCAR**: se activa el nodo de acercamiento publicando un mensaje con el identificador y el centroide del *bouding box* de las personas en estado `UNCERTAIN`. El nodo de acercamiento utiliza esta información para generar subobjetivos de navegación que acercan el robot a la persona y mejoran el ángulo de visión de la cámara. El estado se ejecuta de forma reiterada comprobando continuamente si sigue habiendo personas `UNCERTAIN`. Si ya no hay ninguna, se desactiva el acercamiento y se transita a EVALUAR para revaluar con la nueva perspectiva.
- **Estado EMERGENCIA**: cuando se entra en este estado, la caída se considera confirmada. Se publica un mensaje con el identificador de la persona, su posición en el mundo y la imagen capturada por el robot en tiempo real. La recuperación de flujo normal queda bajo control del operador, el estado permanece activo hasta que se recibe una señal externa de restauración. En ese momento se para de mandar el mensaje de alerta, se limpia la señal y se transita de nuevo a PATRULLA.

4.3. Módulo de actuación

4.3.1. Nodo Obtención de Waypoints

El algoritmo de generación de *waypoints* obtiene puntos de paso para la patrulla a partir del mapa global de ocupación, con el objetivo de seleccionar un conjunto reducido y representativo para cubrir eficientemente el espacio libre de navegación.

Para este propósito, el sistema implementa distintas estrategias alternativas de segmentación del espacio, entre las que se incluyen *Watershed* y segmentación por habitaciones. Ambas comparten una estructura común en dos etapas: primero, el mapa se divide en regiones navegables y, posteriormente, se selecciona un *waypoint* representativo por cada región. En función del tipo de escenario y de los requisitos de la patrulla, se emplea una u otra estrategia.

Cuando se utiliza *Watershed*, la segmentación se basa en la transformada de distancia del espacio libre, lo que permite identificar regiones bien separadas y homogéneas dentro del entorno navegable. Mientras que en la segmentación por habitaciones, se identifican inicialmente pasos estrechos, típicamente asociados a puertas o conexiones entre espacios, a partir de la distancia a obstáculos. Al eliminar temporalmente estas celdas, el mapa se segmenta en componentes conexas que se interpretan como habitaciones; luego, los pasos estrechos se reasignan a las regiones adyacentes, conservando la coherencia espacial .

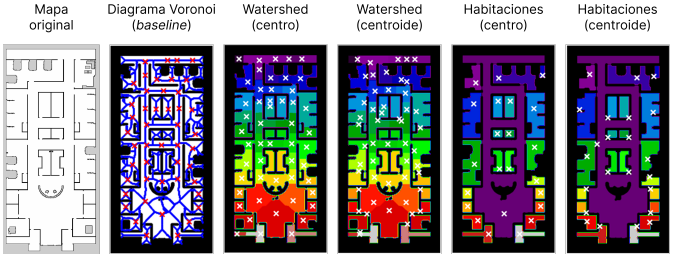

Figura 3: Resultados de las distintas estrategias de obtención de *waypoints* (marcados en cruces). De izquierda a derecha: mapa original; diagrama de Voronoi (*baseline*); segmentación mediante watershed y por habitación, incluyendo la extracción de puntos centrales y centroides de los bordes internos.

Una vez obtenidas las regiones mediante el método seleccionado, se define un único *waypoint* por región siguiendo uno de dos criterios configurables: el punto con máxima distancia a los obstáculos, lo que favorece posiciones centrales y seguras; o un punto obtenido mediante el cálculo del centroide de los bordes internos, los cuales están conectados con otras regiones. Este segundo criterio sitúa los *waypoints* en zonas de tránsito, como puertas o pasillos (ver Figura 3). El proceso de obtención de *waypoints* solo es necesario realizarlo una vez por escenario, y el resultado final es una lista de *waypoints* que se utiliza posteriormente como entrada del nodo de patrulla.

4.3.2. Nodo Patrulla

La activación y desactivación del nodo de patrulla se controla desde la máquina de estados mediante un mensaje booleano. Tras ser activado, el nodo toma la lista de *waypoints* calculada en el nodo obtención de *waypoints* y envía cada punto de manera secuencial como objetivo de navegación. En cada *waypoint* se espera a que la navegación termine correctamente antes de pasar al siguiente; opcionalmente puede ejecutarse un barrido angular al llegar para mejorar la cobertura visual. La lista se recorre iterativamente mientras la patrulla permanezca activa.

La planificación y ejecución de las trayectorias entre *waypoints* se delega al marco de navegación estándar de ROS, `move_base`, el cual combina planificación global y control local teniendo en cuenta la información de los mapas de coste. En concreto, se utiliza el planificador global `navfn/NavfnROS` y el planificador local `TebLocalPlannerROS`.

4.3.3. Nodo Acercamiento

Al igual que el nodo de patrulla, el nodo de acercamiento se activa y desactiva mediante un *topic*, que incluye además el centroide del *bounding box* de la persona detectada. Con esta información, el robot calcula el ángulo de giro necesario para alinearse con la persona y publica un objetivo de navegación a una distancia fija (en este caso 1 m). Este proceso se repite de forma iterativa hasta que la máquina de estados desactiva el acercamiento, lo cual ocurre cuando se ha obtenido una vista suficiente para evaluar la postura de la persona y ya no quedan individuos ambiguos.

4.3.4. Nodo Alerta

El nodo de alerta recibe el mensaje de emergencia, de tipo `AlertCommand`, publicado por la máquina de estados. Cuando el nodo es activado, este muestra una ventana con un aviso de persona caída, las coordenadas y la imagen capturada por el robot en tiempo real. El personal puede analizar la situación a distancia, determinar si se trata de un falso positivo o decidir si es necesario enviar a alguien para atender el incidente. Durante la emergencia, el robot permanece en el lugar donde detectó la caída hasta que el operador indica que la situación está resuelta enviando un mensaje de restauración.

5. Resultados

5.1. Evaluación del sistema de obtención de waypoints

Se compararon las distintas estrategias en cuatro escenarios, agrupados en dos niveles de estructuración (Figura 4), utilizando tres métricas (Tabla 1): número total de *waypoints* generados, porcentaje de cubrimiento del espacio y longitud total de la ruta de patrulla calculada mediante el algoritmo de Dijkstra. Además, se incluyó el método de Voronoi como línea base (ver Figura 3), al tratarse de un enfoque clásico que no realiza una segmentación explícita del entorno.

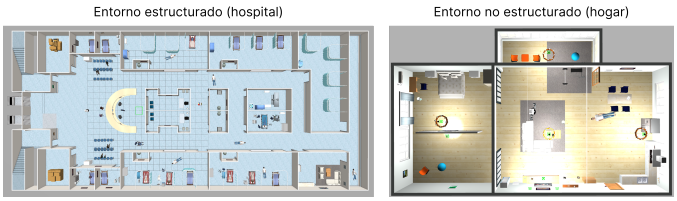

Figura 4: Entornos simulados en Gazebo para experimentos

En entornos estructurados, todos los métodos consiguen obtener un porcentaje de cubrimiento elevado, entorno al 90 %. Entre ellos, el método de Habitaciones destaca por tener aproximadamente la mitad de *waypoints* que el *baseline* y *Watershed*. En particular, la estrategia de centroide logra un recorrido notablemente menor; esto se explica porque, especialmente en habitaciones pequeñas, desde la puerta se puede observar gran parte o incluso todo el interior, evitando desplazamientos innecesarios sin comprometer la cobertura.

Por el contrario, en entornos no estructurados, el método de Habitaciones muestra menos robustez, incluso alcanzando valores de hasta el 53.79 % en el Mundo 2. Esto es debido a que la segmentación de habitaciones depende estrictamente de la presencia de pasos estrechos, los cuales escasean en entornos con divisiones arquitectónicas menos claras. En cambio, el algoritmo *Watershed* resulta ser mucho más robusto, manteniendo un rendimiento consistente en ambas pruebas, especialmente en su variante de centro. Cabe señalar que, aunque este último presenta recorridos más cortos que el *baseline*, la diferencia no es todavía determinante, requeriría realizar pruebas en escenarios de mayor escala para verificar si dicha ventaja se acentúa a medida que aumenta el espacio de navegación.

Tabla 1: Comparación de distintos algoritmos de obtención de *waypoints*

Escenario	Método	Mundo 1			Mundo 2		
		WPs	Cubrimiento	Recorrido	WPs	Cubrimiento	Recorrido
Estructurado	Voronoi	48	96.26 %	337.2 m	90	99.45 %	571.45 m
	Watershed (centro)	43	96.22 %	318.8 m	89	99.46 %	581.80 m
	Watershed (centroide)	43	89.88 %	249.4 m	89	85.71 %	343.90 m
	Habitaciones (centro)	23	95.20 %	289.1 m	48	99.43 %	557.80 m
	Habitaciones (centroide)	23	91.69 %	215.3 m	48	96.22 %	398.40 m
No estructurado	Voronoi	7	97.15 %	36.95 m	9	97.90 %	48.15 m
	Watershed (centro)	7	96.27 %	35.65 m	9	97.92 %	46.10 m
	Watershed (centroide)	7	92.41 %	32.70 m	9	96.32 %	37.35 m
	Habitaciones (centro)	4	87.05 %	28.35 m	2	53.79 %	16.65 m
	Habitaciones (centroide)	4	95.19 %	29.95 m	2	72.89 %	25.85 m

5.2. Evaluación del sistema de detección de caídas

Para evaluar el desempeño del sistema, se escogió tres posturas representativas y escenarios críticos, incluyendo oclusiones por objetos o personas, y condiciones visuales adversas. El robot se mantuvo estático mientras el sujeto adoptaba poses aleatorias dentro de su campo de visión delimitado. Se realizaron 32 iteraciones por cada configuración y los resultados se recogen en la Tabla 2.

Dado que se utilizan modelos preentrenados de YOLO, el rendimiento del sistema está directamente vinculado al de dichos modelos. El sistema presenta un rendimiento significativamente más elevado en la identificación de verdaderos negativos

(no caída) que de verdaderos positivos (caída), con una media de 93.8 % frente a 55.47 %. El análisis de los falsos negativos revela que la mayoría de los errores ocurren bajo ángulos de visión desfavorables, como vistas cenitales. En esta misma línea, posturas más abiertas facilitan la detección, mientras que configuraciones más compacta añade dificultad.

La presencia de oclusiones provoca una disminución de la tasa de éxito de entre un 9.4 % y un 18.7 %, mientras que en condiciones de bajo contraste se observa un descenso cercano al 10 %, lo que sugiere el impacto de condiciones de iluminación desfavorable. Aun así, los resultados siguen evidenciando la viabilidad del módulo de percepción para la detección de caídas, con un rendimiento medio aproximado del 60–65 %.

Cabe señalar que las pruebas en simulado no reproducen toda la complejidad de los escenarios reales. Aunque se realizaron pruebas cualitativas (Figura 2), la validación de su robustez requiere evaluación en entornos reales como trabajo futuro.

Tabla 2: Rendimiento del módulo de percepción en distintos escenarios.

Escenario	Modelo persona	Sin oclusión		Con oclusión	
		No caída	Caída	No caída	Caída
Fondo bajo contraste	Postura de pie	100 %	56.2 %	81.25 %	37.5 %
Fondo alto contraste	Postura de pie	93.8 %	65.6 %	93.75 %	46.9 %
	Postura de marcha	90.6 %	71.9 %	100 %	62.5 %
	Postura sentada	100 %	59.4 %	91.0 %	43.8 %
	Total	96.1 %	63.28 %	91.5 %	47.67 %

5.3. Evaluación del sistema completo

La evaluación del sistema completo se realizó en un entorno hospitalario simulado (Figura 4). A lo largo de la prueba se introdujeron de forma aleatoria personas en distintas posturas simulando situaciones de posible caída mientras el robot recorría los *waypoints*. Se evaluaron un total de 28 eventos (ver Tabla 3). En cada caso, el sistema debía detectar la caída y completar el flujo completo de actuación hasta la generación de la alerta. Tras cada evento, la persona era eliminada del entorno y se enviaba una señal de restauración para continuar la patrulla.

Tabla 3: Evaluación del sistema completa en entorno hospital simulado

Métrica	Total	Casos	Porcentaje (%)	Observaciones
Éxitos totales	24	24/28	85.7	Alerta generada correctamente
Resolución por bounding box	18	18/24	75.0	No requirió análisis de pose
Resolución por estimación de pose	6	6/24	25.0	Casos ambiguos
Fallos totales	4	4/28	14.3	No se generó alerta
Fallo por estimación incorrecta de pose	3	3/4	75.0	Error en inclinación corporal
Fallo por no detección	1	1/4	25.0	Persona no detectada

El sistema logró detectar correctamente la caída en 24 de los 28 casos, alcanzando una tasa de éxito del 85.7 %. En el 75 % de los casos exitosos, la clasificación se resolvió en la primera fase mediante el análisis geométrico del *bounding box*, mientras que el resto requirió la estimación de pose. Los fallos observados se debieron principalmente a errores en la estimación de pose (3 casos) y, en menor medida, a la no detección de la persona (1 caso). En términos de eficiencia, el tiempo medio del robot para completar un ciclo de patrulla fue de 17.5 minutos (velocidad máxima 1 m/s). Por otro lado, el tiempo desde la detección de una persona hasta la identificación de su caída y el cálculo de su posición se sitúa entre 2.09 y 5.88 s (excluyendo tiempos de confirmación), adecuado para un sistema en tiempo real.

Cabe destacar que los resultados obtenidos con el sistema completo superan a los observados en la evaluación aislada del módulo de detección de caídas, lo que pone de manifiesto la efectividad del diseño propuesto. En particular, la capacidad del robot para aproximarse de forma activa a la persona.

6. Conclusiones

En este trabajo se propone un sistema autónomo de vigilancia para la detección de personas caídas en entornos interiores, basado en un diseño modular sobre ROS. El sistema integra percepción visual jerárquica, máquina de estados y navegación autónoma. La evaluación muestra que el enfoque jerárquico de detección permite una identificación eficiente de caídas, resolviendo la mayoría de casos sin análisis costoso, mientras que la generación de *waypoints* adaptativa mantiene un alto cubrimiento con un número reducido de puntos. En simulaciones de entorno hospitalario, el sistema alcanza un 85.7 % de éxito, con fallos principalmente por estimación de pose. Como trabajo futuro, se plantea mejorar la detección de caídas en escenarios de alta complejidad perceptiva, mediante la optimización del algoritmo de aproximación en casos ambiguos. Asimismo, se contempla la evaluación del sistema en entornos reales.

Agradecimientos

Este trabajo ha sido apoyado por el proyecto Advanced Mobile dual-arm Manipulator for Elderly People Attendance (AMME) (PID2022-139227OB-I00), financiado por el Ministerio de Ciencia e Innovación.

Referencias

De Miguel, K., Brunete, A., Hernando, M., Gambao, E., 2017. Home camera-based fall detection system for the elderly. Sensors 17 (12), 2864.

Gebellí, F., Ros, R., 2025. An in-situ participatory approach for assistive robots: methodology and implementation in a healthcare setting. Frontiers in Robotics and AI 12, 1648737.

Kachavarapu, S., Doernbach, T., Gerndt, R., 2025. Fast-revisit coverage path planning for autonomous mobile patrol robots using long-range sensor information. In: 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 7677–7683.

Mora, A., Prados, A., Barber, R., 2022. Segmenting maps by analyzing free and occupied regions with voronoi diagrams. In: ICINCO. pp. 395–402.

Mora, A., Prados, A., Mendez, A., Espinoza, G., Gonzalez, P., Lopez, B., Muñoz, V., Moreno, L., Garrido, S., Barber, R., 2024. Adam: a robotic companion for enhanced quality of life in aging populations. Frontiers in Neurorobotics 18, 1337608.

Sarobin, M. V. R., Anbarasi, L. J., Rukmani, P., Jasmine, S. G., Narendra, M., et al., 2022. Fall detection among elderly person using fallcnn and transfer learning models.

Šelek, A., Seder, M., Petrović, I., 2023. Smooth autonomous patrolling for a differential-drive mobile robot in dynamic environments. Sensors 23 (17), 7421.

Sánchez-Girón, C., Domingo, J. D., García-Bermejo, J. G., Casanova, E. Z., 2025. Navegación proactiva para asistencia tras caídas en un robot social. Simposios del Comité Español de Automática (CEA) 1 (1).

Tankasala, S., Martín-Martín, R., Pryor, M., 2024. Beyond shortsighted navigation: Merging best view trajectory planning with robot navigation. arXiv preprint arXiv:2408.12513.

Teh, C. K., Wong, W. K., Min, T. S., 2021. Extended dijkstra algorithm in path planning for vision based patrol robot. In: 2021 8th International Conference on Computer and Communication Engineering (ICCCE). IEEE, pp. 184–189.

Yu, X., Wang, C., Wu, W., Xiong, S., 2025. A real-time skeleton-based fall detection algorithm based on temporal convolutional networks and transformer encoder. Pervasive and Mobile Computing 107, 102016.

Zhou, T., Iskandar, M. N. S., Chiam, K.-H., 2025. Diffusion models enable zero-shot pose estimation for lower-limb prosthetic users. PLOS Digital Health 4 (3), e0000745.