UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

Tesis

Para obtener el grado de Maestro en Robótica

SEGUIMIENTO DE ADULTOS MAYORES EN MOVIMIENTO UTILIZANDO APRENDIZAJE PROFUNDO Y UN ROBOT MANIPULADOR DE 3 GDL

Presenta:
Ing. Perla Bonifacio Mariano

Director de Tesis:
Dr. Eduardo Sánchez Soto

H. Cd. de Huajuapan de León, Oaxaca, México, 30 de enero de 2026.

A mis hermanas, por ser mi faro de luz.
A ti, Enrique, por permanecer, incluso cuando el camino se volvía incierto.

Agradecimientos

A mi familia y amigos, gracias por su apoyo incondicional, compañía y palabras de aliento a lo largo de este camino. Su confianza y cariño fueron fundamentales para llegar hasta aquí.

A mi director de tesis, el Dr. Eduardo Sánchez Soto, por su invaluable acompañamiento y guía a lo largo de estos años de posgrado. Más allá de su orientación profesional, agradezco sinceramente su calidad humana, la cual fue un pilar fundamental en mi formación académica y personal.

A mis revisores, el Mtro.Moisés Emmanuel Ramírez Guzmán y el Dr. Oscar David Ramírez Cárdenas, por el empeño y la rigurosidad dedicados a la revisión de este proyecto. Sus consejos, orientación y visión fueron determinantes para elevar la calidad de este trabajo; los considero no solo grandes académicos, sino referentes admirables en sus respectivas áreas.

A la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI), por el apoyo económico brindado —becario número 1314421—. Su contribución fue un recurso indispensable para la realización y culminación exitosa de esta investigación.

Finalmente, me agradezco a mí por haber creído en mi potencial incluso en los momentos de mayor incertidumbre. Reconozco mi tenacidad para no desistir, la valentía de incursionar en campos que me eran desconocidos y la disciplina necesaria para cumplir esta meta.

Resumen

Este trabajo presentó el desarrollo y la validación de un sistema de seguimiento robótico para adultos mayores, fundamentado en la integración de un manipulador de 3 GDL con visión estéreo y técnicas de aprendizaje profundo. La arquitectura propuesta se validó en un entorno de simulación ROS-Gazebo, donde se implementaron algoritmos de cinemática directa e inversa para el control del efector final, que es donde se ubicó la cámara. Para la etapa de percepción, se obtuvo un modelo especializado en la detección de personas de la tercera edad mediante la técnica de transferencia de aprendizaje, optimizando una red neuronal convolucional con un conjunto de datos propio.

Asimismo, se desarrolló un algoritmo de estimación de posición que determinó, en tiempo real, la trayectoria del adulto mayor. Este componente de visión por computadora permitió realizar un ajuste dinámico de las articulaciones del robot, estableciendo un control en bucle cerrado que mantuvo el efector final centrado en el objetivo de manera continua. Los resultados experimentales en simulación demostraron la viabilidad técnica del sistema para ejecutar la tarea de seguimiento continuo de personas de la tercera edad en interiores.

Índice

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

La robótica ha evolucionado notablemente, pasando de un enfoque centrado únicamente en robots industriales hacia una nueva etapa caracterizada por sistemas inteligentes capaces de interactuar de forma autónoma con entornos dinámicos y centrados en el ser humano [2]. Esta evolución ha sido impulsada por la integración de disciplinas como el análisis de señales, la teoría de la información y la inteligencia artificial [3], donde la visión por computadora emerge como un proceso fundamental para permitir la interpretación y comprensión de escenas complejas a partir de información visual.

En este contexto surge la robótica de servicio, orientada a asistir, acompañar y facilitar tareas en ámbitos como la salud, la educación y el hogar. Particularmente, el cuidado de adultos mayores constituye un desafío relevante debido al envejecimiento poblacional y a la creciente necesidad de soluciones tecnológicas que complementen la atención humana.

La integración de modelos de aprendizaje profundo en plataformas robóticas como robots manipuladores de múltiples grados de libertad, posibilitan el seguimiento activo de personas en movimiento y una mayor adaptabilidad del sistema ante cambios en el entorno, sentando las bases para soluciones de monitoreo más robustas, autónomas y confiables en escenarios reales.

1.1. Trabajos relacionados

1.1.1. Monitoreo de adultos mayores

El monitoreo de adultos mayores (*elderly monitoring*) engloba un conjunto de tecnologías y sistemas diseñados para apoyar a su supervisión, seguridad y calidad de vida.

En Kim *et al.* [4] realizan una clasificación de estos sistemas en seis funciones principales: (a) actividades diarias, (b) comportamientos anormales, (c) deterioro cognitivo, (d) caídas, (e) localización de personas en interiores y (f) calidad del sueño; para esto identifican 16 tipos de tecnologías de sensores, principalmente los de movimiento y de contacto, así como las cámaras de profundidad. Estos dispositivos se instalan en ubicaciones específicas del hogar (techos, paredes, muebles y electrodomésticos) para recopilar datos sobre las actividades y el comportamiento de los adultos mayores.

A su vez, Nguyen *et al.* [5] destacan la importancia de implementar sistemas no invasivos para el seguimiento de actividades y la localización de las personas mayores, especialmente en hogares inteligentes o residencias asistidas. En este sentido, se han desarrollado soluciones para detectar patrones de actividad, caídas e inactividad prolongada [6–8].

1.1.2. Localización en interiores

Un componente fundamental en los sistemas de monitoreo de adultos mayores es la localización en interiores (*indoor localization*), ya que permite determinar la posición

del individuo dentro de un entorno cerrado, identificar su presencia en zonas potencialmente riesgosas y facilitar su ubicación oportuna en situaciones de emergencia.

En este sentido, numerosos estudios han abordado el problema desde distintos enfoques. Por ejemplo, Karkar *et al.* [9] presentaron un sistema de navegación en interiores (CamNav) basado en visión por computadora que utiliza características de patrones binarios locales multiescalares (MSLBP) para mejorar la precisión en el reconocimiento de lugares. Mientras que Chen [10] utilizó tecnología Beacon para monitorear la seguridad de los adultos mayores en casa, con cámaras que detectaron el uso de electrodomésticos de alto riesgo y alertaron a los cuidadores. Por otro lado, Chen *et al.* [11] propusieron un sistema de primeros auxilios que combinaba la detección de caídas mediante visión y el enrutamiento de rescate a partir de modelos de información de construcción (BIM). En contraste, Lee *et al.* [12] desarrollaron un sistema que empleó etiquetas Bluetooth, sensores de techo y análisis de Big Data para rastrear en tiempo real los movimientos de residentes, personal y visitantes en residencias geriátricas. A diferencia de Mane *et al.* [13], cuyo enfoque incluyó segmentación de imágenes, extracción de características posturales, seguimiento del flujo de movimiento y algoritmos de clasificación para distinguir entre eventos críticos y situaciones no peligrosas para personas de la tercera edad.

1.1.3. Seguimiento de objetos/personas

El seguimiento de objetos (*object tracking*) permite extender la localización en interiores hacia un análisis dinámico del movimiento, al posibilitar el registro de trayectorias y la identificación de patrones de desplazamiento, aspectos clave para la evaluación del comportamiento en entornos cerrados.

Se han propuesto distintos enfoques para afrontar los desafíos en el ámbito de la visión por computadora [14]. Por un lado, Kim y Sim [15] analizaron el seguimiento de objetos específicos en entornos con múltiples objetos en movimiento, utilizando secuencias de video para identificar y rastrear el objetivo en cada cuadro. Este enfoque partió de la generación de imágenes diferenciales mediante técnicas de seguimiento por regiones e implementación del filtro de partículas, logrando un seguimiento preciso, incluso en escenarios complejos donde coexistían múltiples objetivos. En contraste, Kim y Kweon [16] emplearon cámaras en movimiento e integraron técnicas basadas en homografía con el algoritmo de impulso de línea (*online-boosting*) para ajustar el seguimiento a condiciones dinámicas. Otro enfoque fue el presentado por Keivani *et al.* [17], en el que se utilizó un algoritmo K-means modificado para la detección de múltiples objetos en movimiento en tiempo real. Este método operó sin requerir información previa sobre el número de objetos en movimiento, adaptándose a escenarios dinámicos caracterizados por cambios en la iluminación o por trayectorias no lineales. A su vez, el trabajo de Lychkov *et al.* [18] se destacó por las mejoras en la técnica del flujo óptico disperso, lo que permitió un seguimiento más preciso en tiempo real mediante la regeneración automática de puntos característicos perdidos. Esta modificación resultó clave en situaciones en las que los objetos experimentaron oclusiones, rotaciones o cambios de escala, lo que incrementó la versatilidad del sistema.

1.1.4. Robótica basada en visión

En los estudios de Wei y Luo [19], y de Verma *et al.* [20] se empleó un manipulador con cinco grados de libertad (GDL) y una cámara estéreo. No obstante, la modalidad de operación de la cámara presentó variaciones entre ambos trabajos. En Wei y Luo [19]

optaron por la modalidad pasiva y desarrollaron un algoritmo de planificación para el movimiento de la muñeca y la orientación del agarre del robot, aplicando un filtro de Kalman basado en el modelo CAD y la selección dinámica de puntos característicos. Mientras que, en Verma *et al.* [20] utilizaron la modalidad activa y su metodología se basó en características de MSER para la localización de objetos, apoyada en técnicas de agrupamiento basadas en densidad y transformación de homografía.

Estos estudios reflejan un interés creciente en la detección y el seguimiento de objetos, particularmente en el seguimiento humano, lo que evidencia la evolución constante de esta área de investigación y su relevancia en el desarrollo de tecnologías de visión por computadora.

En conclusión, este trabajo se sitúa en la intersección de técnicas de seguimiento de objetos, métodos de localización en interiores y aplicaciones de monitoreo de adultos mayores. Pese a no integrar una localización completa ni analizar detalladamente el movimiento corporal, facilita la adquisición de información relevante sobre la ubicación de una persona mayor dentro de un espacio controlado mediante una solución no intrusiva y fácilmente escalable.

1.2. Planteamiento del problema

En la sociedad contemporánea, el incremento sostenido de la población de adultos mayores plantea desafíos significativos en materia de cuidado, supervisión y atención continua. A medida que la tecnología robótica avanza, los robots de servicio emergen como una solución prometedora para mejorar la calidad de vida de este grupo demográfico. Sin embargo, gran parte de la investigación en este ámbito ha

estado históricamente enfocada en aplicaciones industriales, limitando la exploración de soluciones centradas en escenarios asistenciales.

En el campo de la visión por computadora, si bien existen estudios centrados en el seguimiento de personas, estos se han realizado mayormente en entornos abiertos —como calles o pasillos— y mediante el uso de cámaras estáticas. Esta configuración restringe el campo de visión del sistema, lo que dificulta el seguimiento continuo y preciso de individuos en entornos cerrados complejos.

Por otro lado, los sistemas que integran un robot manipulador con visión por computadora continúan siendo limitados, especialmente en su adopción dentro de entornos domésticos. Lo que se traduce en una falta de desarrollo de soluciones tecnológicas personalizadas para la asistencia y compañía de adultos mayores en sus hogares.

Frente a esta problemática, surge la necesidad de desarrollar un sistema robótico equipado con un sistema de visión por computadora que permita una interacción autónoma y segura con adultos mayores en entornos domésticos, contribuyendo al monitoreo, la asistencia y el apoyo a su vida cotidiana.

Por consiguiente, el objetivo general de esta investigación es desarrollar un algoritmo predictivo de la posición basado en una red neuronal convolucional para identificar, seguir y anticipar el movimiento de adultos mayores en entornos domésticos. Este algoritmo trabajará en conjunto con un robot manipulador de 3 GDL, cuya función principal será evitar los problemas de oclusión comunes en el seguimiento con cámaras estáticas. A través del ajuste dinámico de la posición de la cámara, el robot garantizará un seguimiento continuo del objeto de interés.

1.3. Justificación

En la actualidad, surge una necesidad imperiosa de ofrecer soluciones tecnológicas adaptadas a las demandas cambiantes de la sociedad, con un enfoque particular en el cuidado y el bienestar de las personas mayores. A medida que la población envejece, aumentan las limitaciones físicas y la necesidad de asistencia personalizada, lo que resalta la importancia de la tecnología robótica de servicio. La implementación de un sistema de supervisión con herramientas de inteligencia artificial de última generación, adaptable a un sector vulnerable de la población, se vuelve una herramienta fundamental para garantizar su integridad física.

En este contexto, el presente estudio tiene como objetivo principal el desarrollo de un algoritmo de predicción de posición basado en una red neuronal convolucional integrada en un robot manipulador de 3 GDL. Dicho algoritmo no solo permitirá el seguimiento continuo de adultos mayores para una supervisión constante, sino que también ajustará la orientación del robot de manera dinámica para evitar oclusiones. De esta manera, esta investigación busca ofrecer una solución tecnológica adaptable a las necesidades específicas de los adultos mayores.

1.4. Hipótesis

La combinación de un robot manipulador de 3 GDL y una red neuronal en un entorno de simulación dará como resultado un sistema para el seguimiento de adultos mayores en movimiento. La elección y el entrenamiento de una red neuronal, en conjunto con la integración de la cinemática del manipulador, ofrecerán mejoras en la capacidad

de seguimiento de los adultos mayores.

1.5. Objetivos

1.5.1. General

1. Desarrollar un entorno de simulación en ROS-Gazebo que modele el movimiento de un robot manipulador de 3 GDL . Este entorno será utilizado para el seguimiento de adultos mayores en movimiento mediante una cámara ubicada en el efector final del robot con la ayuda de una red neuronal convolucional (CNN).

1.5.2. Específicos

1. Integrar un conjunto de datos con imágenes de adultos mayores a través de recursos en línea.
2. Seleccionar un algoritmo adecuado para aplicar transferencia de aprendizaje.
3. Entrenar una red neuronal preentrenada para adaptar el modelo a un nuevo conjunto de datos.
4. Desarrollar un robot manipulador de 3 GDL en el entorno de ROS-Gazebo.
5. Diseñar un algoritmo de estimación de la posición a partir de la información proporcionada por el modelo especializado para mover el robot hasta una posición deseada.

6. Integrar el modelo especializado y el algoritmo de estimación de la posición en el entorno de simulación.

7. Realizar un análisis de los resultados obtenidos en la simulación mediante la comparación de la trayectoria real del adulto mayor con la predicha por el algoritmo de estimación de la posición.

1.6. Metodología

Para desarrollar esta investigación, se recurrió a dos metodologías (para sistemas embebidos [21] y en cascada [22]) con el objetivo de obtener una metodología adaptada a las necesidades específicas del proyecto: el entrenamiento de la red neuronal y el desarrollo del robot manipulador; sistemas esenciales para lograr el seguimiento de adultos mayores en movimiento.

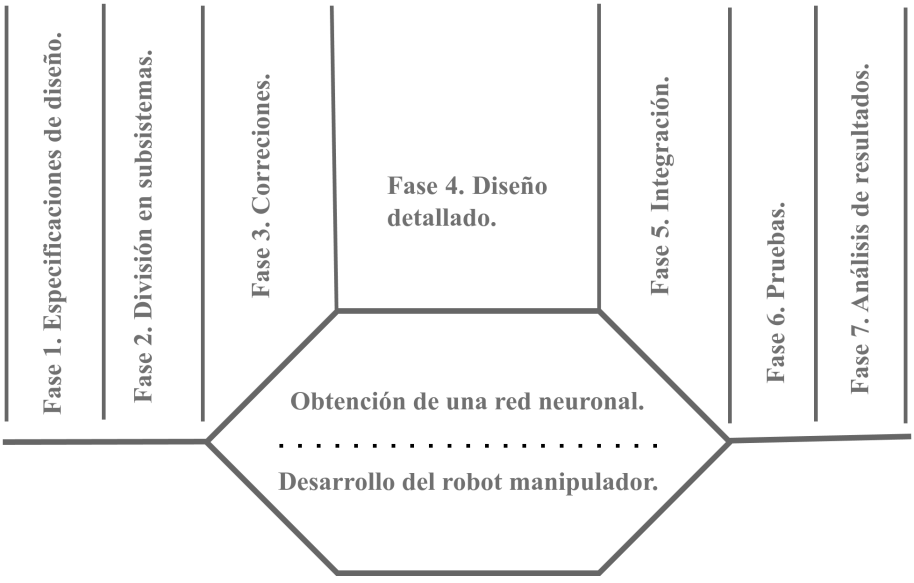


Figura 1.1: Metodología para sistemas embebidos. Adaptado de [21].

En consecuencia, cada fase de la Figura 1.1 implementó de la siguiente manera:

Especificaciones de diseño. Se definieron los requerimientos y herramientas necesarias para diseñar y simular el sistema propuesto.

División en subsistemas. Se dividió el proceso en (a) entrenamiento de la red neuronal e (b) implementación del robot manipulador en el entorno ROS-Gazebo.

Correcciones. Se realizaron modificaciones pertinentes para obtener los elementos necesarios para la integración del sistema completo, dado que cada proceso de la fase anterior fue considerado una caja negra.

Diseño detallado. Se integró la metodología en cascada (ver Figura 1.2) para obtener de una red neuronal enfocada en el reconocimiento de adultos mayores y se desarrolló el robot manipulador.

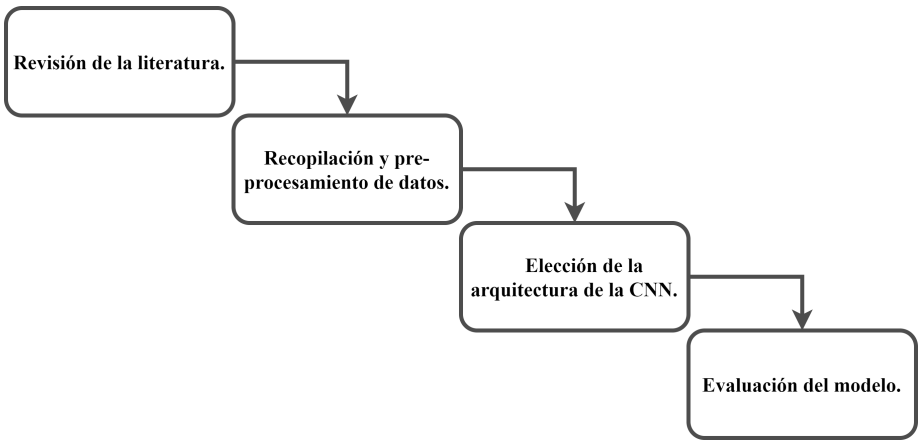


Figura 1.2: Metodología en cascada.

Parte 1: Obtención de la red neuronal

Revisión de la literatura. Se realizó una revisión detallada de la literatura enfocada en CNNs, aprendizaje profundo y aplicaciones específicas en el área de

seguimiento de objetos para identificar las arquitecturas de CNN más relevantes, las técnicas de transferencia de aprendizaje y las herramientas de software más utilizadas.

Recopilación y pre-procesamiento de datos. Para la integración de un conjunto de datos, se hizo uso de imágenes obtenidas de fuentes en línea. Además, se llevó a cabo una etapa de pre-procesamiento de la información en donde se incluyó el aumento de datos y la normalización de los mismos.

Obtención de un modelo especializado. A partir de una red neuronal preentrenada, se obtuvo un modelo especializado en personas de la tercera edad. Se ajustó la arquitectura del modelo de acuerdo con los resultados obtenidos durante las etapas de entrenamiento, validación y prueba.

Evaluación del modelo. Se evaluó el rendimiento del modelo especializado mediante métricas adecuadas para la tarea de detección, como precisión, sensibilidad, especificidad, etc.

Parte 2: Desarrollo del robot

Definición de requisitos. Se especificó el espacio de trabajo y el número de articulaciones necesarias para realizar la tarea objetivo.

Diseño mecánico. Se seleccionó la arquitectura cinemática y se obtuvo el modelo CAD.

Desarrollo del software. Se realizó el análisis cinemático para modelar la relación matemática entre cada una de las articulaciones y el efector final. Adicionalmente, se programó la cinemática inversa para obtener las variables articulares y alcanzar una posición deseada.

Integración en el simulador. A partir del modelo CAD, se generó un formato URDF para validar el modelo cinemático inverso y facilitar la integración del robot en el entorno de simulación.

Integración. Se implementó la red neuronal, el robot y escenarios —modelo 3D de habitaciones al interior de una casa y adultos mayores— en el simulador para poder generar los estímulos necesarios y validar el funcionamiento del algoritmo de estimación de la posición. Este proceso permitió verificar que la interacción y coordinación entre la red neuronal y el robot fueran efectivas, asegurando que el algoritmo de estimación de la posición operará dentro de sus parámetros especificados.

Pruebas. Se realizaron pruebas con múltiples condiciones iniciales para evaluar la robustez del sistema y obtener un panorama completo de su comportamiento.

Análisis de resultados. Se llevó a cabo un análisis detallado de los resultados en simulación. Verificando los tiempos de respuesta, la estabilidad del sistema y la precisión en la salida del algoritmo de estimación de la posición.

1.7. Alcances y delimitaciones

Para el desarrollo del proyecto se tuvieron en consideración los siguientes aspectos.

1.7.1. Alcances

- Desarrollar un algoritmo de estimación de la posición de adultos mayores para permitir que un robot manipulador de 3 GDL realice su seguimiento en un

entorno ROS-Gazebo.

- Este algoritmo permitió que una cámara integrada en el sistema robótico permaneciera siempre centrada en el adulto mayor.

1.7.2. Delimitaciones

- El sistema se acotó a un entorno de simulación.
- El robot manipulador fue de 3 GDL .
- La capacidad de procesamiento del sistema fue una limitación en términos de la velocidad y la detección continua del objetivo.
- Los resultados obtenidos estuvieron específicamente vinculados a la interacción del robot manipulador con los adultos mayores.

Capítulo 2
Marco teórico

2.1. Visión por Computadora

Con la aparición de las computadoras, una de las primeras áreas de investigación fue la Visión por Computadora, que consiste en analizar imágenes mediante computadoras para obtener descripciones de objetos físicos captados por cámaras [23, 24].

En la Figura 2.1 se muestra un ejemplo de la adquisición de una imagen. Este proceso involucra la captura del objeto que posteriormente se analizará y procesará computacionalmente. Esta operación requiere un sistema de digitalización y la intervención de una fuente luminosa para que el sistema reconozca todas las características del objeto observado.

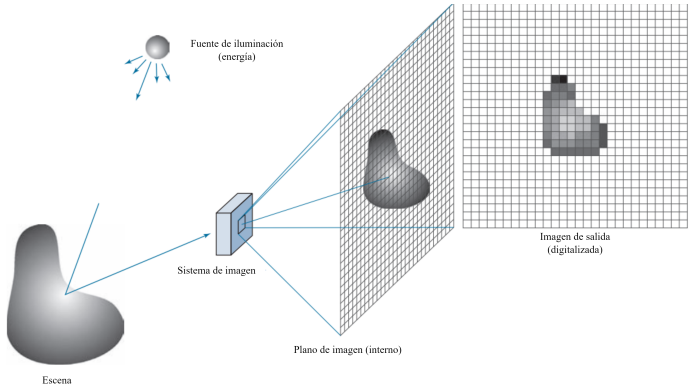


Figura 2.1: Ejemplo de la adquisición de imágenes digitales. Adaptado de [24].

En el esfuerzo por emular el procesamiento visual humano, la Visión por Computado-

ra suele dividirse tradicionalmente en cuatro etapas principales, como se muestra en la Figura 2.2 [25]. Sin embargo, en la actualidad, varios procesos de estos módulos pueden formar un solo subsistema, indicando una mayor integración y complejidad en los sistemas modernos.

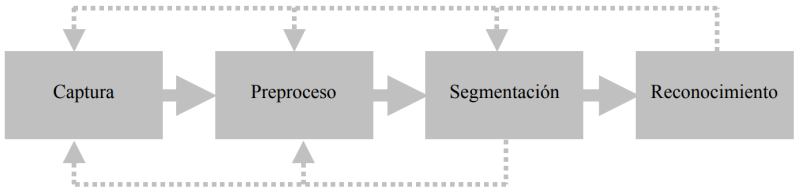


Figura 2.2: Etapas de un sistema de Visión por Computadora. Recuperado de [25].

Esta disciplina, relacionada con otras tecnologías, se destaca principalmente en tres áreas: el procesamiento de imágenes, que busca transformar una imagen en otra; la generación de gráficos por computadora, que convierte la descripción de objetos en imágenes; y el reconocimiento de patrones, que clasifica objetos según sus características entre un conjunto de candidatos [23].

2.2. Procesamiento digital de imágenes

A medida que las computadoras crecieron en potencia y redujeron su costo, hubo una explosión en la gama de aplicaciones para las cuales se contó con la posibilidad práctica, de desempeño y de memoria para el procesamiento de imágenes digitales; se pueden encontrar ejemplos de aplicaciones que van desde la inspección industrial hasta el procesamiento de imágenes médicas [26].

El procesamiento digital de imágenes puede definirse como la aplicación de una serie de operaciones matemáticas a una imagen para obtener un resultado deseado [26].

2.2.1. Operaciones sobre imágenes

Estas operaciones se realizan mediante algoritmos de procesamiento de imágenes, que pueden dividirse en niveles de aplicación y niveles de operación. A medida que la imagen es procesada, se llevan a cabo transformaciones que agrupan píxeles en regiones, extrayendo características significativas que se utilizan para identificar objetos o partes de objetos. Este proceso generalmente culmina en el reconocimiento, donde se deriva una descripción o interpretación de la escena, ver Figura 2.3.

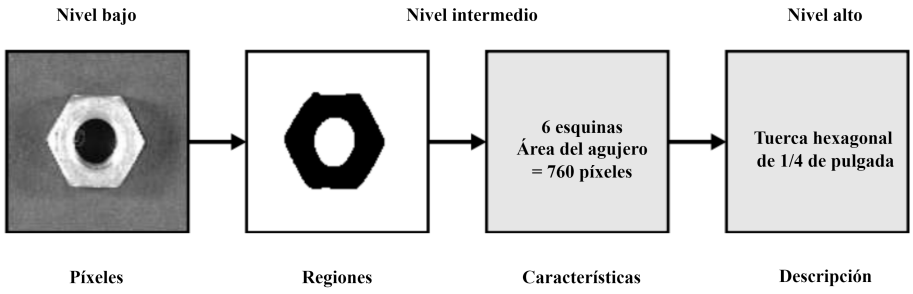


Figura 2.3: Algoritmo de procesamiento de imágenes. Adaptado de [24].

El enfoque en las operaciones de procesamiento de imágenes puede categorizarse en una pirámide, desde operaciones de preprocesamiento en la base hasta operaciones de clasificación y reconocimiento en la cúspide, como se muestra en la Figura 2.4.

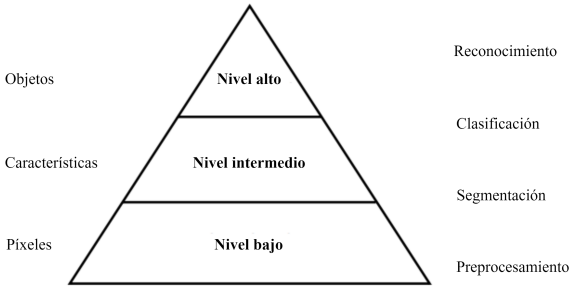


Figura 2.4: Pirámide de procesamiento de imágenes. Adaptado de [26].

En el procesamiento de imágenes, descrito por Vélez *et al.* [25], el proceso comienza con la captura de imágenes mediante sensores, seguida del tratamiento digital en el preprocesamiento para facilitar etapas posteriores, como el filtrado y la mejora de áreas de interés. La segmentación aísla elementos de interés en la escena, mientras que en la cúspide de la pirámide se clasifican y reconocen los objetos segmentados mediante el análisis de características predefinidas. Estas etapas no siempre se siguen de manera secuencial, sino que a veces requieren retroalimentación, como se muestra en la Figura 2.2.

2.3. Calibración de una cámara

El enfoque de la calibración es clave para extraer información tridimensional a partir de imágenes. De acuerdo con Tsai [27], existen dos tipos de datos 3D que pueden inferirse con una cámara calibrada:

- Ubicación de un objeto en el espacio 3D: la posición exacta de un punto en un entorno tridimensional puede determinarse mediante la intersección de dos vistas, lo que permite una representación precisa en el espacio.
- Posición y orientación de una cámara móvil: la calibración de una cámara montada en un robot optimiza su precisión en movimiento, asegurando mediciones correctas y una mejor interpretación del entorno.

Una calibración precisa es esencial en drones, vehículos autónomos y sistemas de visión robótica, ya que garantiza mediciones exactas y una interpretación fiable del entorno [28].

Parámetros

La calibración de una cámara se divide en parámetros intrínsecos y extrínsecos. Los intrínsecos garantizan que los objetos se representen con la escala y la posición correctas en una imagen, ajustando parámetros como la distancia focal, el punto principal y los coeficientes de distorsión para corregir imperfecciones ópticas. Por otro lado, los parámetros extrínsecos determinan la ubicación y orientación de la cámara en el espacio tridimensional, utilizando una matriz de traslación para definir su posición y una matriz de rotación para establecer su inclinación y ángulo, la orientación.

2.4. Aprendizaje automático

Un algoritmo de aprendizaje automático (*machine learning*, ML) extrae patrones o ajusta sus parámetros en función de los datos disponibles en un conjunto de datos, con el objetivo de realizar alguna tarea como predicción o clasificación. Se dice que un programa de computadora aprende de la experiencia E con respecto a alguna clase de tareas y a una medida de rendimiento si su rendimiento en tareas en T, medido por P, mejora con la experiencia E [29].

El ML resulta eficiente en situaciones donde las soluciones existentes requieren un ajuste manual considerable o extensas listas de reglas, como se hacía hace algunos años con los sistemas expertos. A menudo, un algoritmo de ML logra simplificar el código y mejorar su rendimiento. Además, resulta especialmente útil para abordar problemas complejos que carecen de soluciones efectivas mediante enfoques tradicionales. En entornos cambiantes, un sistema de ML puede adaptarse con mayor facilidad a nuevos datos [30].

La tarea principal en el ML consiste en seleccionar un algoritmo de aprendizaje y entrenarlo con ciertos datos. Por lo tanto, los factores determinantes son la elección de un algoritmo adecuado y el uso de los datos adecuados.

La mayoría de los algoritmos de ML requieren una gran cantidad de datos para funcionar correctamente. Pensar que la cantidad es sinónimo de calidad es incorrecto en este contexto específico. La representatividad de los datos con respecto al problema tratado es de suma importancia; en otras palabras, los datos deben ser representativos de los nuevos casos a los que se desea generalizar. El desafío radica en eliminar datos de baja calidad, con errores, valores atípicos, ruido o con información irrelevante [30]. Estos problemas se agravan si se presenta un sobreajuste (*overfitting*), en el que el modelo funciona bien con los datos de entrenamiento, pero no generaliza correctamente, o un subajuste (*underfitting*), que ocurre cuando el modelo no se adapta lo suficiente a los datos de entrenamiento.

2.4.1. Aprendizaje supervisado

En Gerón [30] los sistemas de aprendizaje automático más destacados en la actualidad son aquellos que se entrenan tanto con supervisión humana como sin ella, conocidos como supervisados y no supervisados, respectivamente. Estos sistemas se clasifican según la cantidad y el tipo de supervisión durante su entrenamiento.

En el aprendizaje supervisado, los datos de entrenamiento proporcionados al algoritmo incluyen las soluciones deseadas, llamadas *etiquetas*. Entre sus tareas típicas se encuentran la clasificación o la regresión. Algunos de los algoritmos o modelos más conocidos que utilizan esta categoría de aprendizaje son la regresión lineal, la regresión logística, las máquinas de soporte vectorial y los árboles de decisión. Sin

embargo, entre todos estos métodos, destacan especialmente las redes neuronales, que se analizan con más detalle en la siguiente sección.

2.5. Redes neuronales artificiales

Hoy en día, las redes neuronales artificiales (*Artificial Neural Networks*, ANNs) experimentan un éxito notable, impulsado por las tecnologías y tendencias actuales que proporcionan enormes volúmenes de datos para su entrenamiento. El desempeño de las computadoras actuales también ha contribuido a su posicionamiento, especialmente gracias al desarrollo de potentes tarjetas gráficas (*graphics processing unit*, GPU). Además, se han realizado mejoras significativas en los algoritmos de entrenamiento [31].

En términos generales, las ANNs constituyen el núcleo del aprendizaje profundo (*deep learning*, DL) debido a su versatilidad, potencia y escalabilidad. Esto las convierte en la opción ideal para abordar tareas amplias y complejas de aprendizaje automático, como la clasificación de millones de imágenes, la mejora de los servicios de reconocimiento de voz y la recomendación de vídeos para millones de usuarios diarios [30].

Fundamento biológico

Una neurona es una célula especializada que se encuentra principalmente en la corteza cerebral de los animales. Está compuesta por un cuerpo celular, dendritas, y un axón. Las neuronas reciben señales eléctricas de otras neuronas a través de las sinapsis y forman redes complejas que realizan cálculos sofisticados [29].

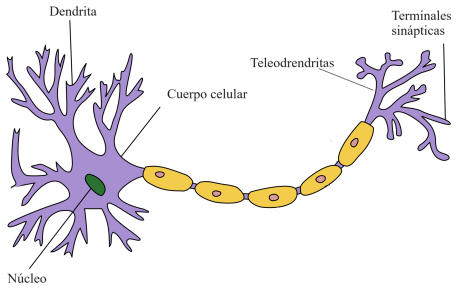
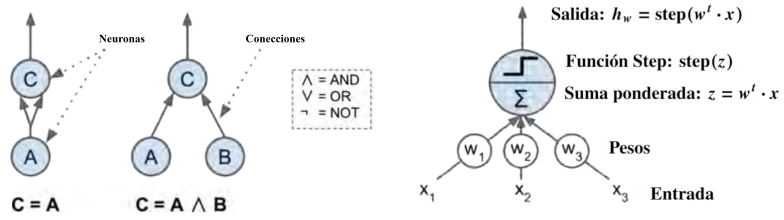


Figura 2.5: Partes principales de una neurona biológica[1].

El diseño de una ANN se basa en el funcionamiento del cerebro humano, en el que las neuronas se conectan entre sí para procesar información de manera similar.

Gerón [30] menciona que una neurona artificial se caracteriza por tener una o más entradas binarias y una salida binaria. Su función principal es activar la salida cuando un número específico de entradas se activa, lo que le permite realizar cálculos lógicos simples .



(a) ANNs realizando cálculos simples.

(b) Arquitectura del LTU.

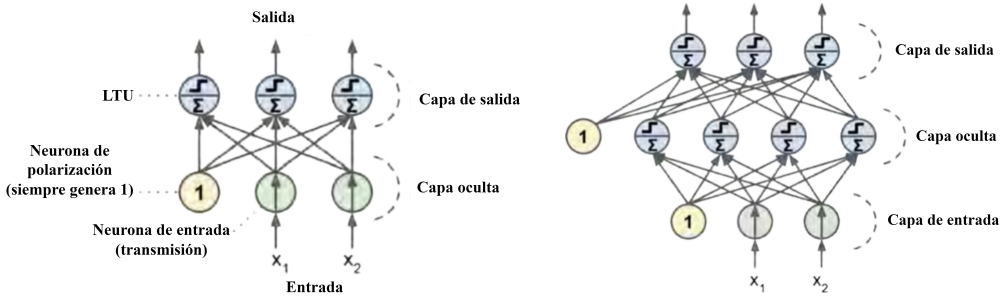
Figura 2.6: Inicios de las ANNs. Adaptado de [30].

El perceptrón, basado en la unidad de umbral lineal (*threshold logic unit*, LTU), es una variante de la ANN. En esta configuración, las entradas y salidas son valores

[1]Recuperado de `https://en.wikipedia.org/wiki/Neuron`.

numéricos, y cada conexión de entrada está ponderada. El perceptrón consta de una única capa de LTU, lo que implica que el límite de decisión de cada neurona de salida es lineal. Esto limita la capacidad de los perceptrones para aprender patrones complejos.

Para abordar esta limitación, se desarrolló el perceptrón multicapa (*multilayer perceptron*, MLP), que consiste en apilar varios perceptrones para formar capas ocultas. Cuando una ANN tiene dos o más capas ocultas, se la conoce como red neuronal profunda (*deep neural network*, DNN).



(a) Perceptrón. (b) Perceptrón multicapa.

Figura 2.7: Perceptrón vs MLP. Adaptado de [30].

Las MLP, también conocidas como redes neuronales de propagación hacia adelante (*feedforward neural network*, FNN), son fundamentales en el aprendizaje profundo. A diferencia de las redes neuronales recurrentes, las FNN no tienen conexiones de retroalimentación.

El proceso de entrenamiento de una red feedforward implica decisiones similares a las de un modelo lineal, como la selección del optimizador, la función de costo y la estructura de las unidades de salida.

Función de costo: es una medida de qué tan bien está funcionando la red neuronal

en comparación con las salidas deseadas durante el entrenamiento. La elección de la función de costo depende del tipo de problema que se aborda.

Optimizador: es el algoritmo que ajusta los pesos de la red neuronal durante el entrenamiento para minimizar la función de costo. Algunos optimizadores comunes incluyen el descenso de gradiente estocástico (*stochastic gradient descent*, SGD), el descenso de gradiente con momento (Momentum), el descenso de gradiente con momento adaptativo (Adam), entre otros. Cada optimizador tiene sus propias ventajas y desventajas en términos de velocidad de convergencia y capacidad para evitar mínimos locales.

Forma de las unidades de salida: se refiere a cómo se estructuran las unidades de salida de la red neuronal. Por ejemplo, en un problema de clasificación binaria, es común tener una única unidad de salida que represente la probabilidad de que la entrada pertenezca a una de las dos clases. En un problema de clasificación multiclase, se pueden utilizar múltiples unidades de salida, una por clase.

Funciones de activación: se utilizan para introducir no linealidad en la red neuronal, lo que les permite modelar relaciones más complejas en los datos. Algunas funciones de activación comunes incluyen la función sigmoide, la función de activación lineal rectificada (ReLU) y la función tangente hiperbólica. Cada función de activación tiene características que se adecúan a distintos problemas y arquitecturas de red.

Al introducir capas ocultas en una FNN, es crucial seleccionar cuidadosamente las funciones de activación. El aprendizaje en estas redes implica calcular gradientes de funciones complejas, un proceso fundamental para ajustar los pesos de la red y

mejorar su desempeño.

Los métodos de aprendizaje supervisado y no supervisado, que son las formas más populares de aprendizaje, se han expresado mediante varias reglas (ver Figura 2.8).

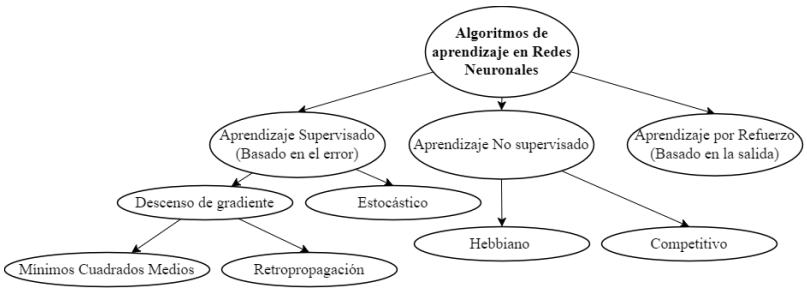


Figura 2.8: Algoritmos de aprendizaje. Adaptado de [32].

2.5.1. Aprendizaje por descenso de gradiente

Rajasekaran y Pai [32] mencionan que este método se basa en minimizar el error E definido en términos de los pesos y de la función de activación de la red. La función de activación debe ser diferenciable, ya que la actualización de los pesos depende del gradiente del error E. Por lo tanto, si ΔW_{ij} es la actualización del peso del enlace que conecta la neurona i-ésima y j-ésima de las capas vecinas, se define como:

$$\Delta W_{ij} = \eta \frac{\partial E}{\partial W_{ij}}$$

Donde η es el parámetro de tasa de aprendizaje y $\frac{\partial E}{\partial W_{ij}}$ es el gradiente de error con respecto al peso W_{ij}.

La regla Delta de Widrow y Hoff y la regla de retropropagación son ejemplos de este tipo de mecanismo de aprendizaje.

2.5.2. Backpropagation

El algoritmo de retropropagación, a menudo simplemente llamado backpropagation (BPN) [29], ha sido fundamental en el resurgimiento del interés por las redes neuronales artificiales. Este método sistemático se utiliza para entrenar redes neuronales artificiales multicapa [32].

En Goodfellow *et al.* [29] mencionan que el proceso de entrenamiento con backpropagation consta de varias fases. En la propagación hacia adelante, la red neuronal feedforward acepta una entrada x y produce una salida $\hat{y}$. La información fluye desde las entradas x hasta las unidades ocultas en cada capa y, finalmente, produce $\hat{y}$. Durante esta fase, se calcula un costo escalar $J(\theta)$.

Luego, en la fase de backpropagation la información del costo se retropropaga a través de la red para calcular el gradiente. Este gradiente se utiliza para ajustar los pesos de la red neuronal, con el objetivo de minimizar $J(\theta)$.

Es importante destacar que el backpropagation no comprende la totalidad del algoritmo de aprendizaje de una red neuronal multicapa, sino que se limita al procedimiento para calcular el gradiente de la función de costo respecto a los parámetros del modelo. Dicho gradiente es posteriormente utilizado por otros algoritmos de optimización, como el descenso de gradiente estocástico, para actualizar los pesos de la red [29].

2.5.3. Aprendizaje profundo

El aprendizaje profundo (*deep learning*, DL) es un subconjunto de la AI en el que sus modelos mejoran su rendimiento a medida que son expuestos a más datos, siempre

que la calidad de los datos y la arquitectura del modelo sean adecuadas [33], ver Tabla 2.1.

Este campo se ha consolidado como una de las principales áreas de investigación, destacando en aplicaciones de visión por computadora [33, 34], al construir sistemas informáticos capaces de resolver con éxito tareas que requieren inteligencia [29]. De ahí que, se empleen modelos no lineales con múltiples capas ocultas, para que el sistema aprenda la compleja relación entre la entrada y la salida [34].

Tabla 2.1: Modelado de DL *vs* ML.

Característica	**Deep Learning**	**Machine Learning**
Dependencia de datos.	Gran cantidad de datos etiquetados.	Reglas específicas creadas por los ingenieros.
Dependencia de hardware.	Requiere hardware especializado (GPU).	Puede funcionar en hardware estándar.
Proceso de ingeniería de características.	Automatización de la extracción de características complejas.	Proceso manual de selección y creación de características.
Tiempo de entrenamiento y ejecución del modelo.	Largo (semanas en grandes conjuntos de datos).	Relativamente corto (segundos a horas).
Pruebas.	Rápidas debido a la generalización de los modelos.	Pueden ser largas para evaluar generalización y precisión.
Percepción e interpretabilidad.	Considerados “cajas negras”.	Más interpretables, basados en reglas lógicas.
Desempeño con el crecimiento de datos exponencial.	Mejora con el crecimiento de datos.	Puede disminuir con el crecimiento de datos, especialmente en modelos basados en reglas.

Información recuperada de [33].

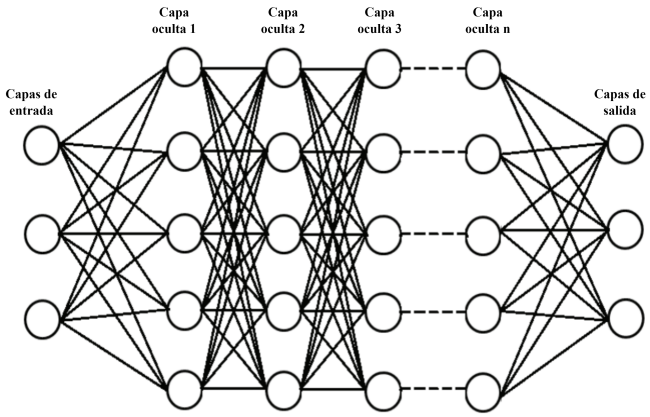


Figura 2.9: Arquitectura de una red profunda. Adaptado de [34].

Entre los tipos de DL más reconocidos se encuentran: el autoencoder (AE), la red de creencias profundas (DBN), la red neuronal recursiva, el aprendizaje profundo de refuerzo directo, la red neuronal recurrente (RNN) y la red neuronal convolucional (*convolutional neural network*, CNN) [31, 34].

Tras revisar el panorama general, el siguiente apartado se centra en las redes neuronales convolucionales, considerado uno de los ejes principales de este trabajo de investigación.

2.5.4. Redes Neuronales Convolucionales

Las redes neuronales convolucionales, propuestas por LeCun en 1989, son un tipo especializado de red neuronal que ha demostrado su éxito en diversas aplicaciones prácticas. Estas redes procesan datos mediante la operación matemática de convolución en al menos una de sus capas [29].

En general, destacan por su eficiencia en la extracción automática de característi-

cas en datos complejos [35], especialmente en el procesamiento de imágenes [36], convirtiéndolas en una herramienta esencial en el campo del aprendizaje profundo.

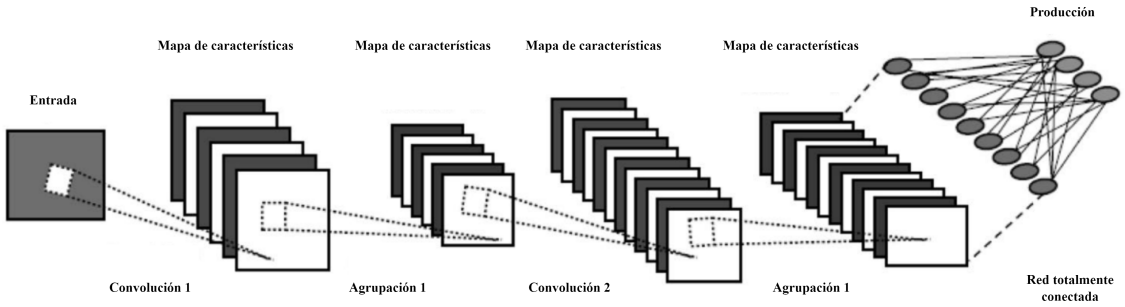


Figura 2.10: Una red neuronal convolucional que incorpora varias capas de convolución y agrupación (pooling). Adaptado de [33].

En Lu *et al.* [36] mencionan que un modelo típico de CNN consta de una capa de entrada, capas ocultas y una capa de salida. La capa oculta, compuesta por varias capas convolucionales (*layers*), capas de agrupación (*pooling*) y capas completamente conectadas (*fully connected layers*, FC) son responsables de la extracción y combinación de características. Los núcleos (*kernels*) convolucionales se utilizan para obtener respuestas de diferentes características y extraer múltiples características locales. Cada capa convolucional desempeña el papel de predicción mediante la propagación hacia adelante (*forward propagation*) y la actualización de parámetros mediante la retropropagación (*backpropagation*). La capa de agrupación puede mejorar la resistencia a la traslación y a la rotación de las imágenes de entrada. Los parámetros del modelo se actualizan mediante el descenso de gradiente. Después del entrenamiento, la red neuronal convolucional puede extraer características jerárquicas, ver Figura 2.10.

Algunas de las arquitecturas CNN comúnmente utilizadas son LeNet, AlexNet, GoogleNet, Network in Network [34].

2.5.5. Arquitectura del modelo YOLOv8

Estos modelos, entrenados con grandes volúmenes de datos, permiten reutilizar representaciones generales y optimizadas para abordar nuevas tareas con mayor eficiencia al reducir el costo computacional y se facilitar la transferencia de conocimiento hacia problemas específicos. Un claro ejemplo de esto es YOLO, una arquitectura de red neuronal que se inspira en el modelo GoogLeNet para la clasificación de imágenes. Al tratarse de una arquitectura de detección en una sola etapa [37], se ha enfocado constantemente en equilibrar la velocidad y la precisión, con el objetivo de ofrecer un rendimiento en tiempo real sin sacrificar la calidad de los resultados de detección [38].

En la Tabla 2.2 se muestra a detalle las especificaciones técnicas de la versión YOLOv8s. Al igual que sus predecesores [39], añade funciones y optimizaciones que lo convierten en una solución eficaz para la detección de objetos en distintas aplicaciones [40].

Tabla 2.2: Especificaciones técnicas de YOLOv8s[2].

Modelo	**Tamaño [px]**	**mAPval**	**Parámetros [M]**	**FLOPs [B]**
YOLOv8s	640	44.9	11.2	28.6

FLOPs (Operaciones de coma flotante por segundo): se refiere al número total de operaciones matemáticas en punto flotante .

La familia YOLOv8 ofrece variantes escalables (n, s, m, l, x) para distintos equilibrios precisión-rendimiento. Se seleccionó YOLOv8s por su óptimo balance: proporciona un mAP50-95 del 44.9 % con solo 11.2M parámetros, permitiendo inferencias en tiempo real (28-40 FPS) utilizando el hardware similar al descrito en el Apéndice A.1,

[2]Recuperado de Especificaciones técnicas de YOLOv8s.

lo que la hace ideal para aplicaciones prácticas que requieren precisión robusta y baja latencia.

El modelo está compuesto por tres capas principales, que conforman su arquitectura base (ver Figura 2.11).

- Red base (*backbone*). Comprende el preprocesamiento inicial con redes como DarkNet y ResNet.
- Cuello (*neck*). Es el puentre entre capas como C2f.
- Predictor (*head*). A través de tres módulos de detección obtiene una predicción.

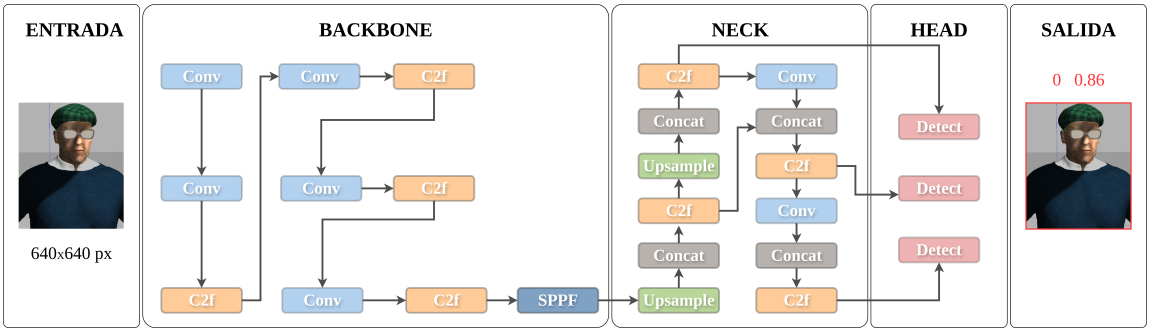


Figura 2.11: Estructura del modelo de detección YOLOv8. Adaptado de [1].

Métricas de evaluación para detección de objetos

Para evaluar el rendimiento de los modelos de detección de objetos, se emplean métricas basadas en las cajas delimitadoras (*bounding boxes*). Estas métricas permiten analizar tanto la exactitud de las predicciones como la capacidad del modelo para localizar correctamente los objetos dentro de una imagen.

Esta combinación de métricas ofrece información detallada sobre la calidad de las detecciones realizadas por el modelo [41].

Box (P, R, mAP50, mAP50-95)

P (Precisión). Indica la exactitud de las detecciones, es decir, cuántas fueron correctas.

R (Recall). Mide la capacidad del modelo para identificar todas las instancias de objetos presentes en las imágenes.

mAP50. Precisión media calculada con un umbral de intersección sobre unión (IoU) de 0.50.

mAP50-95. Promedio de la precisión media obtenido en múltiples umbrales de IoU, desde 0.50 hasta 0.95, lo que brinda una visión más completa del rendimiento del modelo en distintos niveles de dificultad de detección.

2.6. Transferencia de aprendizaje

Los modelos de DL requieren una gran cantidad de datos para su entrenamiento con el fin de aprovechar plenamente su capacidad. Desafortunadamente, la obtención de conjuntos de datos extensos y de alta calidad implica costos elevados y limitaciones prácticas en numerosos dominios de aplicación. Para mitigar los problemas de los conjuntos de datos pequeños, se han propuesto varios métodos como: (a) el aumento de datos (*data augmentation*) que incrementa artificialmente la diversidad del conjunto de entrenamiento y (b) la transferencia de aprendizaje (*transfer learning*, TL) que reutiliza conocimiento previamente adquirido en tareas o dominios relacionados [42].

En la transferencia de aprendizaje se imita la capacidad del sistema visual humano al utilizar conocimientos previos obtenidos en dominios relacionados para abordar

nuevas tareas dentro del mismo dominio. Se puede concebir como un paradigma de aprendizaje especial, en el que los datos de entrenamiento de una distribución distinta se aplican, total o parcialmente, a los datos de prueba [43].

Este método sigue un enfoque común en el que se entrena una red base y luego se transfieren sus primeras n capas a las primeras n capas de una red de destino. Después, las capas restantes de la red de destino se inicializan aleatoriamente y se ajustan para adaptarse a la tarea específica [44], ver Figura 2.12.

Dentro del TL, una de las técnicas más utilizadas es el *fine-tuning*, la cual consiste en inicializar un modelo con pesos preentrenados y ajustar de manera parcial o total sus parámetros utilizando datos del nuevo problema. En este enfoque, las capas iniciales suelen preservar representaciones generales aprendidas previamente, mientras que las capas finales se adaptan progresivamente a la tarea específica, lo que ha demostrado ser un método eficaz y ampliamente adoptado en aplicaciones de visión por computadora y aprendizaje profundo [44].

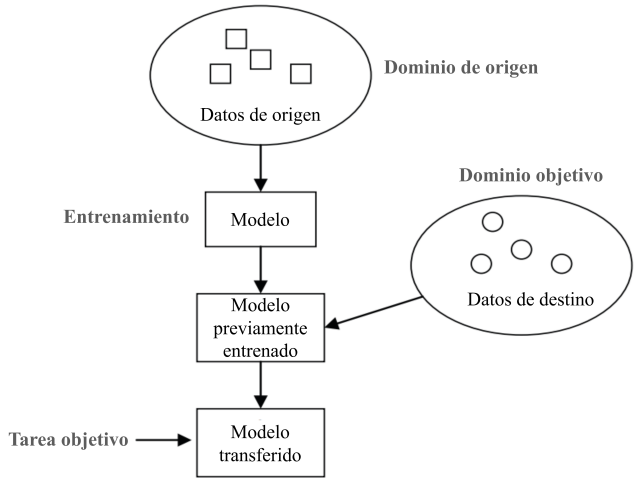


Figura 2.12: Proceso de transferencia de aprendizaje. Adaptado de [44].

2.7. Seguimiento de objetos en movimiento

El seguimiento de objetos en movimiento, una disciplina ampliamente explorada en visión por computadora [45, 46], se revela como un componente crucial para la navegación segura de robots y vehículos móviles en entornos del mundo real [47]. Se identifican cinco métodos principales para abordar este desafío: blobs, contornos, flujo óptico, modelos y redes neuronales [18]. Estas metodologías ofrecen diversas perspectivas y soluciones para abordar la complejidad del seguimiento de objetos en movimiento.

La aplicación de las CNNs en la visión por computadora ha impulsado logros notables en múltiples áreas. Estos incluyen el reconocimiento facial para autenticación y seguridad, la percepción del entorno en vehículos autónomos para una conducción segura, la supervisión automatizada en supermercados para transacciones sin cajeros y el diagnóstico médico avanzado para una atención de salud más precisa y oportuna [35].

2.8. Control visual para seguimiento

Para dotar a un robot de la capacidad de realizar seguimiento visual, es esencial coordinar el sistema visual con el sistema motriz. Esta coordinación, conocida como *visual servoing*, combina visión, cinemática, dinámica, teoría de control y computación, permitiendo que la información visual se incorpore en tiempo real al bucle de control del manipulador [48, 49].

El visual servoing utiliza datos obtenidos de cámaras para guiar el movimiento del ro-

bot hacia una configuración deseada. El control se basa en definir un error visual que se minimiza mediante retroalimentación en tiempo real, lo que convierte la percepción en acción efectiva [50]. Los enfoques más conocidos son Image-Based Visual Servoing (IBVS), donde el error se define en el plano de la imagen, y Position-Based Visual Servoing (PBVS), donde se formula en el espacio cartesiano tridimensional [51].

En PBVS se estima la pose tridimensional del objeto o cámara y se compara con una pose deseada. Este enfoque ofrece interpretaciones físicas claras y trayectorias suaves en 3D, aunque su desempeño depende de la precisión de la calibración de la cámara y de la reconstrucción tridimensional [52, 53].

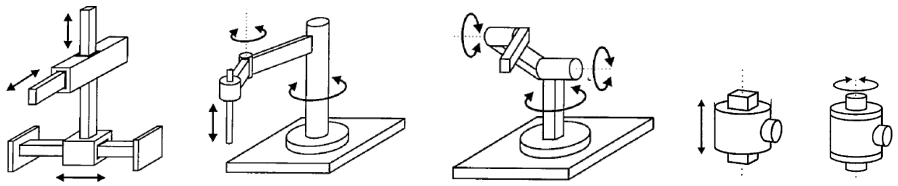
2.9. Análisis cinemático de robots manipuladores

Un robot manipulador se define como un mecanismo diseñado para interactuar con su entorno mediante la manipulación de objetos [54]. Compuesto por una serie de enlaces articulados, formando una cadena cinemática abierta. El movimiento de cada articulación puede ser de desplazamiento, de giro o una combinación de ambas. Sin embargo, en la práctica, los robots sólo emplean la articulación prismática (P) y la rotacional (R) (ver Figuras 2.13(d) y 2.13(e)).

Cada movimiento independiente que una articulación puede realizar respecto de la anterior se conoce como grado de libertad (GDL) [54]. Derivado de esto, la cantidad de grados de libertad de un manipulador se refiere al número de variables de posición independientes que deben ser determinadas para identificar la ubicación de todas las partes del mecanismo [55].

Las combinaciones de articulaciones en un robot dan lugar a diversas configuraciones,

que presentan características que deben considerarse tanto en el diseño, la construcción y la aplicación del robot. Las combinaciones más frecuentes se muestran en la Figura 2.13.



(a) Cartesiano. (b) SCARA. (c) Antropomórfico. (d) P. (e) R.

Figura 2.13: Tipos de configuraciones y articulaciones en robots manipuladores. Recuperado de [54].

Marcos de referencia

Los marcos de referencia permiten describir con precisión la posición y la orientación de cada componente del manipulador. Entre ellos, el marco del efector final (EE) y el punto central de herramienta (TCP) son esenciales para definir la interacción del robot con su entorno. En la Figura 2.14 se presentan cinco marcos estándar, esenciales para la planificación de trayectorias y el control del movimiento. A continuación, se describen cada uno de ellos.

B Corresponde al marco de referencia inicial, es decir, la base del manipulador.

S Se define con respecto al marco base y representa un punto relevante en la tarea.

W Se establece en relación con el marco base y está unido al último eslabón del manipulador, en la zona denominada "muñeca".

T Se define con respecto al marco de la muñeca y se ubica en el extremo de la herramienta que manipula el robot.

G Se establece con respecto al marco de estación y representa la posición final deseada de la herramienta.

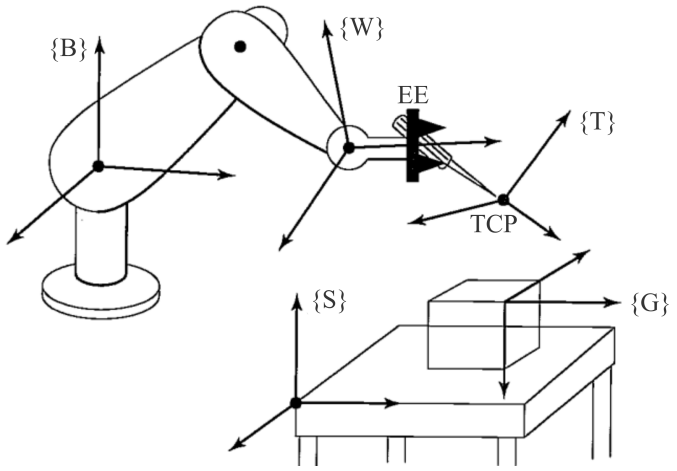


Figura 2.14: Marcos de referencia. Adaptado de [56].

2.9.1. Cinemática directa e inversa

La cinemática analiza el movimiento de los robots manipuladores sin considerar las fuerzas que lo generan. Se divide en dos problemas fundamentales y opuestos entre sí: la cinemática directa y la cinemática inversa, ver Figura 2.15.

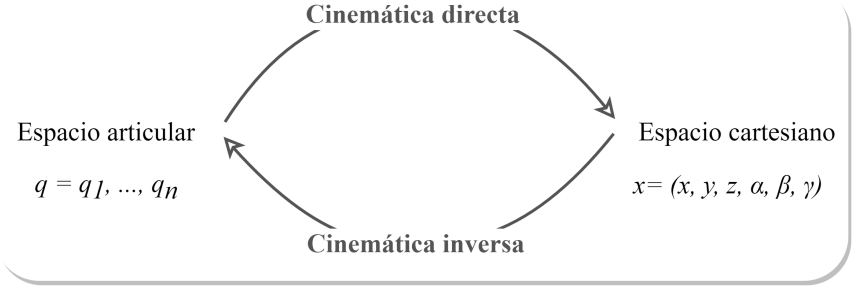


Figura 2.15: Diagrama de relación entre cinemática directa e inversa.

Cinemática directa: determina la pose del efector final en función de las variables de configuración.

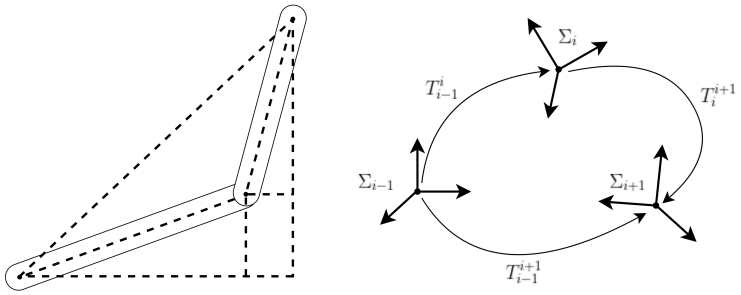
$$x = f(q) \tag{2.1}$$

Cinemática inversa: determina las variables de configuración en función de la pose del efector final.

$$q = g(x) \tag{2.2}$$

2.9.2. Métodos de solución

La solución de la cinemática de un robot, ya sea directa o inversa, se basa en distintos métodos matemáticos. La elección del método depende de la naturaleza del problema y de la complejidad del robot, ver Figura 2.16.



(a) Geométrico. (b) Transformaciones homogéneas.

Figura 2.16: Métodos de solución de la cinemática.

Método geométrico. El método geométrico resuelve la cinemática mediante relaciones geométricas y trigonométricas. Este enfoque modela el robot como una serie de triángulos y figuras interconectadas, aplicando leyes como el teorema

de Pitágoras, la ley de los senos y la ley de los cosenos para calcular directamente los ángulos o las distancias de las articulaciones. Es un método rápido y directo para robots con pocos grados de libertad (2-3), pero inviable para robots más complejos.

Transformaciones homogéneas. Las transformaciones homogéneas resuelven la cinemática de un robot (>3 GDL) de forma sistemática y matricial.

Cinemática directa. Se obtiene al multiplicar secuencialmente las matrices de transformación homogénea de cada articulación (generalmente asignadas mediante el método de Denavit-Hartenberg), lo que da como resultado una única matriz que describe la pose final del robot.

Cinemática inversa. Se resuelve analíticamente al despejar las variables articulares de la ecuación de la cinemática directa. Se iguala la matriz de pose del robot a la pose deseada y se multiplican ambos lados por la inversa de las matrices de articulación para aislar las variables una por una.

En cualquier método de análisis, ya sea mediante transformaciones homogéneas o mediante el método geométrico, es esencial conocer tres elementos fundamentales del robot (ver Figura 2.17): sus parámetros geométricos, su configuración y su pose.

- En la ecuación 2.3 los ángulos de cada articulación son relativos, es decir, medidos con respecto al eslabón anterior.
- En la ecuación 2.4 la posición y orientación del robot se miden de forma absoluta, es decir, con respecto al referencial base (X,Y).

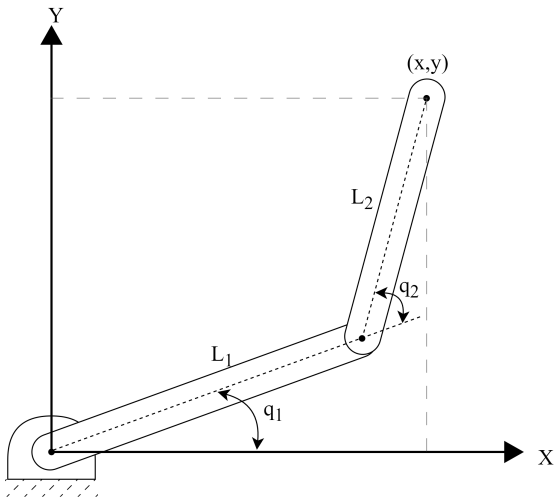


Figura 2.17: Diagrama de un manipulador de 2R.

Párametros geométricos

L_1 = Longitud del eslabón 1.

L_2 = Longitud del eslabón 2.

Configuración

$$q = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} \in \mathbb{R}^2 \tag{2.3}$$

Pose

$$X = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \in \mathbb{R}^3 \tag{2.4}$$

2.10. Entorno de simulación

Los entornos de simulación de robots son plataformas virtuales que permiten el desarrollo, la realización de pruebas y la evaluación de robots en entornos controlados. Los simuladores proporcionan una manera de experimentar con diferentes configuraciones, algoritmos de control y estrategias para establecer la dinámica de las distintas partes que conforman el robot. La ventaja de los simuladores radica en la fabricación de prototipos que se realizará una vez que las simulaciones den los resultados esperados. De esta manera, un simulador permite optimizar el rendimiento de los robots antes de su implementación.

2.10.1. ROS

El sistema operativo de robots (*robot operating system*, ROS) es un kit de desarrollo de software de código abierto para aplicaciones robóticas [57]. Proporciona una estructura flexible que facilita el desarrollo de software para robots y la integración de diferentes componentes de software y hardware.

Conceptos

En la Figura 2.18, se muestra la estructura de comunicación en ROS, destacando la interacción entre nodos, mensajes y tópicos.

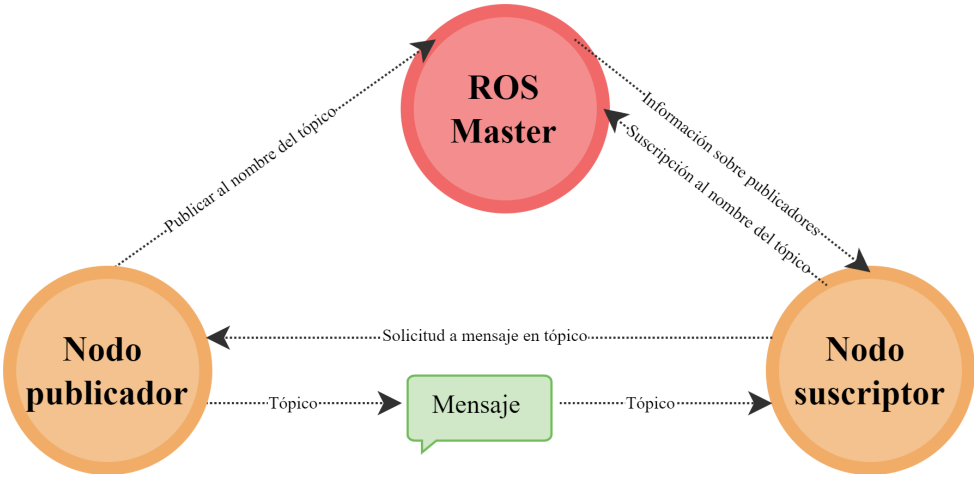


Figura 2.18: Nivel gráfico de cálculo de ROS[3].

Roscore (ROS Master + rosout + un servidor de parámetros). Actúa como el servicio central que coordina la comunicación entre los nodos y es importante para

[3]Recuperado de Introducción a ROS.

iniciar cualquier aplicación en ROS. Lleva un registro de los mensajes y los tipos de mensaje que se realizan cuando ROS es ejecutado.

ROS Master. Es un nodo que proporciona servicios de registro y nombre al resto de los nodos en el sistema. Todos los nodos publicadores y suscriptores se registran en el nodo maestro, que actúa como servidor de parámetros.

Nodo. Es un ejecutable capaz de generar o recibir información en forma de mensajes. Su diseño permite minimizar los fallos, ya que cada nodo opera de manera autónoma. Si uno deja de funcionar, los demás continúan enviando o esperando nuevos mensajes, lo que garantiza la estabilidad y continuidad del sistema.

Tópico. Son los canales de comunicación que utilizan los nodos para intercambiar mensajes. Un mensaje se transmite desde un nodo publicador (*publisher node*) hacia uno o varios nodos suscriptores (*subscriber node*) a una frecuencia determinada, establecida en ambos extremos. Esta frecuencia define el ritmo de envío y recepción de mensajes, asegurando una comunicación eficiente

Mensaje. El mensaje es transmitido por el nodo publicador a través de un tópico, creado dentro de ese mismo nodo. Por su parte, el nodo suscriptor decide a qué nodo o a qué nodos desea suscribirse, lo que permite la recepción de mensajes según su configuración.

Organización del entorno

El desarrollo en ROS se organiza en paquetes, que agrupan nodos, archivos de configuración y librerías necesarias para la ejecución del sistema. Los paquetes se gestionan dentro de un *workspace*, que actúa como el entorno de trabajo donde se gestionan y

construyen los proyectos. Cada nodo dentro de un paquete puede intercambiar información mediante mensajes, lo que facilita la comunicación entre procesos dentro del ecosistema ROS.

Los paquetes permiten organizar el código de manera modular. Dentro de la carpeta src/ de un *workspace*, los paquetes siguen una estructura organizada, como se muestra en la Tabla 2.3.

Tabla 2.3: Estructura de un paquete en ROS.

Elemento	**Descripción**	**Contenido**
package.xml	Configuración del paquete y sus dependencias.	-
CMakeLists.txt	Contiene las reglas de compilación.	-
include/	Almacena archivos de encabezado.	.h
src/	Contiene el código fuente de los nodos.	.cpp, .py
launch/	Archivos de lanzamiento.	.launch
config/	Archivos de configuración del sistema.	.yaml, .xml
scripts/	Scripts y automatizaciones auxiliares.	.py, .sh
worlds/	Contiene archivos de mundos para simulaciones en Gazebo.	.world
models/	Archivos de modelos 3D.	.sdf,.urdf, .dae

2.10.2. Gazebo

Gazebo es una colección de bibliotecas de software de código abierto que se ha estructurado para adaptarse a distintos casos de uso. Cada biblioteca dentro de Gazebo tiene dependencias mínimas, lo que permite su uso en tareas como la codificación de video, la simulación y la gestión de procesos [58].

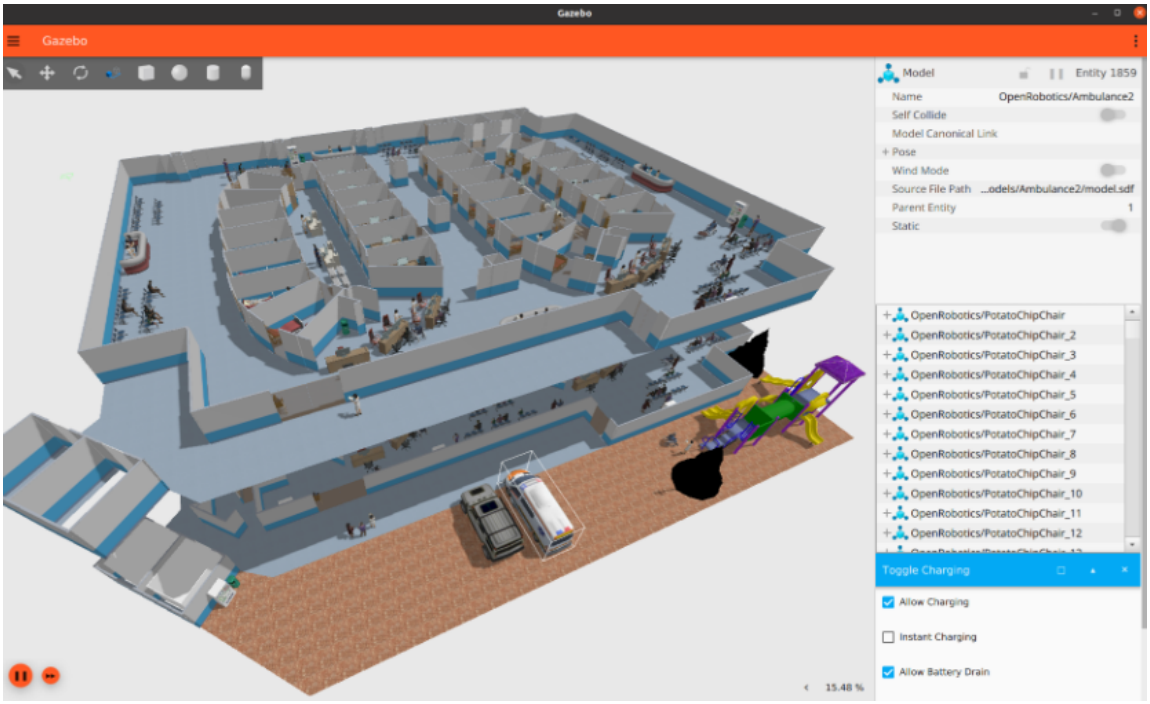


Figura 2.19: Entorno virtual con Gazebo. Recuperado de [58].

Gazebo ofrece una simulación dinámica que modela el comportamiento físico de los robots mediante un motor de simulación. Sus gráficos 3D incorporan iluminación, sombras y sensores que mejoran la visualización de entornos virtuales. Además, permite integrar diversos sensores y utilizar plugins para interactuar con el mundo exterior, ya sea a través de dispositivos o directamente con los robots. Su base de datos de modelos facilita el acceso a robots predefinidos y la importación de modelos propios. También cuenta con comunicación basada en zócalo para una interacción eficiente entre componentes, soporte para simulaciones en la nube y herramientas de línea de comandos que optimizan el desarrollo y la ejecución de simulaciones.

2.10.3. RVIZ

RVIZ (*robot visualization tool*) sirve para visualizar en 3D los datos publicados en ROS, como sensores, modelos de robots, mapas y trayectorias. La Tabla 2.4 presenta los diferentes tipos de displays disponibles en RViz.

Tabla 2.4: Tipos de displays integrados.

Nombre	**Descripción**	**Tipo de mensaje**
Fixed frame	Este marco de referencia fijo que se utiliza para designar el marco del mundo.	
TF	Muestra la jerarquía de transformaciones tf.	
RobotModel	Muestra una representación visual de un robot en la pose correcta (según lo definen las transformaciones TF actuales).	
Image	Crea una nueva ventana de renderizado con una imagen.	sensor_msgs/Image
Pose	Dibuja una pose en forma de flecha.	geometry_msgs/PoseStamped
PointCloud2	Muestra datos de una nube de puntos, con diferentes opciones para modos de renderizado, acumulación, etc.	sensor_msgs/PointCloud2
Marker	Permite mostrar formas primitivas arbitrarias a través de un tópico.	visualization_msgs/Marker
PointStamped	Dibuja un punto como una pequeña esfera.	geometry_msgs/PointStamped

2.10.4. Métricas de evaluación

Para evaluar el desempeño del sistema propuesto, se definieron cuatro métricas que comparan la distancia entre la posición estimada/predicha ($\hat{P}$) y la posición real (P) del modelo de un adulto mayor.

- La raíz del error cuadrático medio (RMSE) mide la magnitud promedio del error, penalizando los errores grandes o atípicos al elevarlos al cuadrado.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(P_i - \hat{P}_i)^2}$$

  Con N como el número de puntos.

- El error absoluto medio (MAE) mide la diferencia absoluta promedio.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|P_i - \hat{P}_i|$$

- El error máximo (MAX_E) mide la peor desviación en la estimación de profundidad/predicción que ha ocurrido en toda la prueba.
- El error final de distancia (FDE) mide la diferencia espacial entre el último punto registrado de la estimación de profundidad/predictor ($\hat{P}_{\text{fin}}$) y el último punto registrado del AM (P_{fin}).

$$\sqrt{\left(x_{P_{fin}} - x_{\hat{P}_{fin}}\right)^2 + \left(y_{P_{fin}} - y_{\hat{P}_{fin}}\right)^2}$$

Donde:

- Posición estimada: hace referencia a la posición del adulto mayor calculada a partir de las mediciones visuales en un instante de tiempo.
- Posición predicha: se refiere a la posición que el sistema calcula que tendrá el adulto mayor en un instante futuro, basado en el historial de su trayectoria.

Capítulo 3
Modelo especializado

En este capítulo se detalla la construcción de un conjunto de datos y el posterior desarrollo de un modelo de red neuronal enfocada en la detección de adultos mayores empleando la técnica de transferencia de aprendizaje.

3.1. Construcción del conjunto de datos

A continuación, se describe el método de recolección de datos y las estrategias de preprocesamiento para garantizar la calidad y representatividad de la información con respecto al problema de detección de adultos mayores.

3.1.1. Búsqueda de imágenes

Se realizó una búsqueda de bancos de imágenes para construir un conjunto de datos adecuado y personalizado. No obstante, se detectaron dos problemas:

- La escasa disponibilidad de bases de datos gratuitas[1],
- Los bancos de imágenes de paga[2] no contaban con una cantidad significativa de imágenes de adultos mayores.

[1]Bancos de imágenes gratuitos: Pexels y Unsplash
[2]Bancos de imágenes de paga como Shutterstock.

A pesar de esta problemática, fue indispensable asegurar una gran cantidad de imágenes consistentes con la calidad requerida. Por ello, se optó por los bancos de imágenes gratuitas.

3.1.2. Características de las imágenes

Las imágenes seleccionadas se caracterizaron por incluir a personas de la tercera edad realizando diversas actividades cotidianas en distintos escenarios.

- Las actividades incluían: estar sentado, caminando, realizando ejercicio, trabajando, corriendo, platicando, escalando, bailando.
- Los escenarios considerados fueron: calles, jardines, hospitales, parques, montañas y viviendas.

En la Figura 3.2 se observan las principales características en el conjunto de datos. Predominaron las personas de tez blanca, al igual que la utilización de los planos medios y americanos en las imágenes (ver Figura 3.1).

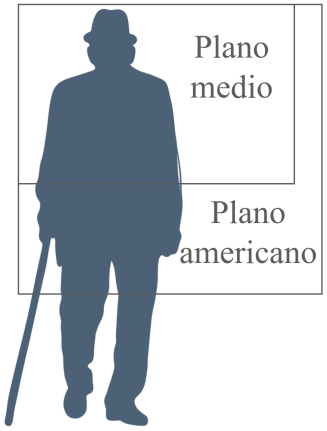


Figura 3.1: Tipos de planos.

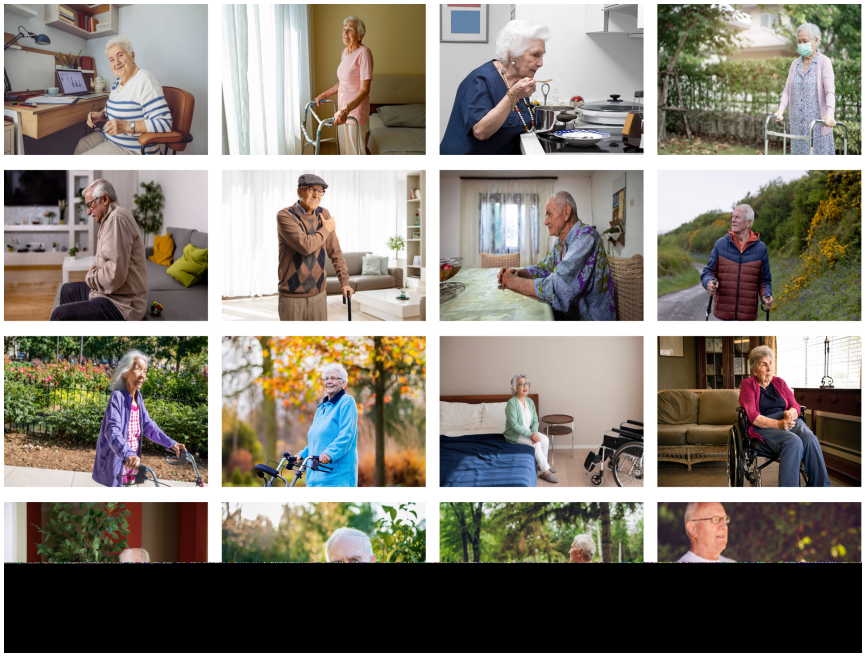


Figura 3.2: Ejemplos de escenas típicas identificadas durante el proceso de recopilación de imágenes de adultos mayores.*

Los datos en bruto fueron almacenados en una carpeta, con las imágenes ordenadas secuencialmente del 1 al número total de imágenes. Durante esta fase, se recolectaron *2315 imágenes*.

3.1.3. Limpieza de datos

En esta etapa, la limpieza de datos se realizó en paralelo a la búsqueda de imágenes. Al buscar las imágenes, se filtraron y descartaron de inmediato aquellas que no correspondían al grupo objetivo, es decir, adultos mayores (ver Figura 3.3).

* Collage hecho a partir de imágenes de iStock.

Figura 3.3: Elección de imágenes.

Este procedimiento garantizó la relevancia y la homogeneidad de la calidad de las imágenes, minimizando la necesidad de procesos de limpieza posteriores.

3.1.4. Gestión de los metadatos

Para cada imagen, se generaron metadatos contextuales (actividad y posición) que se almacenaron en formato JSON. La codificación sigue un sistema alfanumérico estructurado en 4 campos independientes:

```
ImagenN_Actividad+Posición_Género
  [1]       [2]       [3]       [4]
```

Donde:

① N: es un contador que indica el número de imágenes almacenadas.

② Actividad: es un listado de acciones cotidianas.

Caminando [C] Sentado [S] Inclinado [I]

Corriendo [Cr] Recostado [R] Doblado [D]

Parado [P] Hincado [Hi] +...

③ Posición: describe la manera en que los adultos se colocan físicamente frente a la cámara.

[**F**] Frontal [**P**] Perfil [**D**] Diagonal

④ Género: categoría de hombre o mujer.

En la Figura 3.4 se presenta la distribución de cada una de las actividades realizadas por los adultos mayores. Se observa que las categorías PF, PD y SD corresponden a las actividades dominantes, las cuales concentran la mayor cantidad de registros dentro del conjunto analizado.

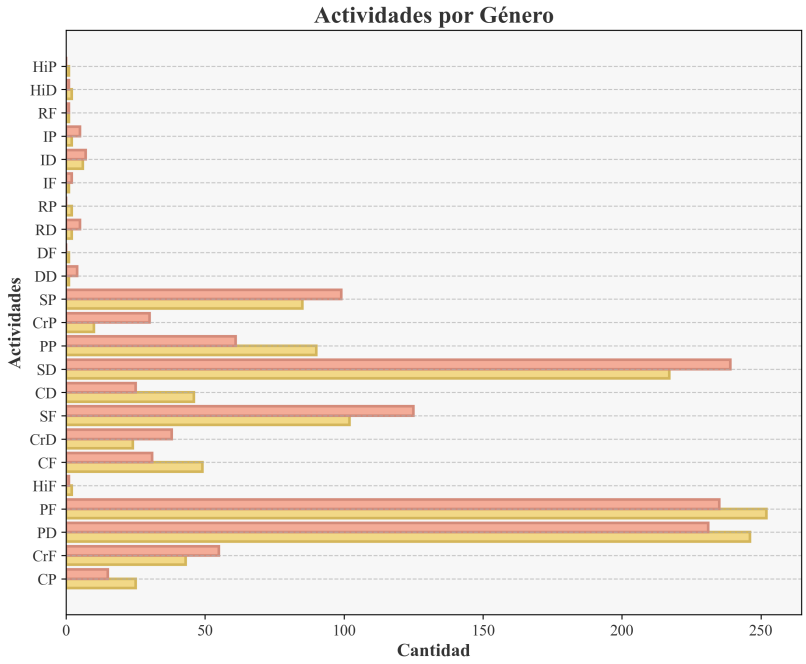


Figura 3.4: Distribución de las actividades.

Etiquetado

Se asignaron etiquetas de clase. El esquema de etiquetado incluyó dos categorías:

[**0**] Hombre [**1**] Mujer

Tras completar esta fase, se evaluó el equilibrio entre las clases. En la Figura 3.5 se muestra la distribución final, obteniendo *1210 etiquetas* por clase.

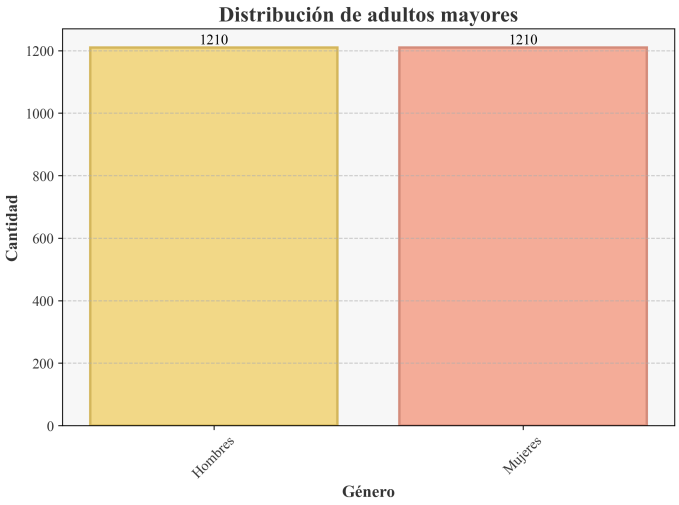


Figura 3.5: Distribución de los adultos mayores.

Anotación

Las anotaciones realizadas en cada imagen se ilustran en la Figura 3.6. Cada instancia fue segmentada manualmente mediante anotaciones poligonales, alcanzando una tasa promedio de aproximadamente 20 imágenes por hora. El tiempo requerido se debió a la ejecución manual del trazado de los polígonos. En consecuencia, las imágenes que presentaban múltiples objetos de interés requirieron una mayor cantidad de tiempo para realizar las segmentaciones.

(a) Imagen original. (b) Imagen segmentada.

Figura 3.6: Anotaciones en el conjunto de datos.

3.1.5. Preprocesamiento

Inicialmente, en lugar de procesar las imágenes completas, se implementó una extracción de regiones de interés (ROI) basada en las coordenadas de las anotaciones del conjunto original (ver Figura 3.7(a)). La salida de este proceso fueron recortes de imágenes focalizados en los adultos mayores con el objetivo de minimizar el ruido que pudiera afectar el rendimiento del modelo, ver Figuras 3.7(b) y 3.7(c).

En los casos en los que una imagen contenía múltiples personas de interés, se ajustaba la cantidad de imágenes generadas. Como resultado, se obtuvieron *2,418 imágenes*.

(a) Imagen original. (b) Recorte 1. (c) Recorte 2.

Figura 3.7: Recorte sobre el área de interés.

Aumento de datos

Las imágenes obtenidas fueron sometidas a operaciones de transformación —geométricas y de color— para aumentar el número de instancias en el conjunto de datos. En consecuencia, este proceso introdujo complejidades técnicas; cada transformación geométrica requirió modificar los valores de las coordenadas de las anotaciones para garantizar su correspondencia espacial. Pese a ello, se generaron *9 imágenes* a partir de una instancia (ver Figura 3.8) aplicando operaciones como rotación, escalamiento y traslación, ver Tabla 3.1.

Tabla 3.1: Operaciones de transformación.

Transformación	Valor
Ruido gaussiano	$\mu = 0$, $\sigma = 25$
Cambio de tono y saturación	Tono = 0, Saturación = 1
Ajuste de brillo	Factor = 1.2
Desenfoque	Kernel= (5, 5)
Rotación	30°
Escalado	0.5
Traslación	50 px [X,Y]
Reflejo	Horizontal y vertical.

Nota: Consultar el Apéndice B.2.

Figura 3.8: Aumento de datos.

Por último, se incorporaron imágenes similares a las presentadas en la Figura 3.9, con el fin de mitigar la inconsistencia detectada en la fase de implementación posterior al primer entrenamiento. Aunque durante el entrenamiento, la validación y las pruebas

se obtuvieron métricas estables y coherentes (ver la Tabla 3.5), el desempeño del modelo en implementación reveló la necesidad de este ajuste.

Figura 3.9: Adición de imágenes al conjunto original.

Redimensionamiento

Después de la etapa de aumento de datos, las imágenes fueron redimensionadas a 640×640 px mediante un proceso de relleno de bordes (ver Figura 3.10), incorporando valores cero para preservar la relación de aspecto original. Este paso fue crítico para garantizar la compatibilidad con el modelo preentrenado, ya que permitió estandarizar los datos de entrada y obtener el conjunto de datos al que se hará referencia a partir de ahora como AM-2025. Dicho nombre proviene de las siglas de Adultos Mayores, mientras que 2025 hace referencia al año de su creación.

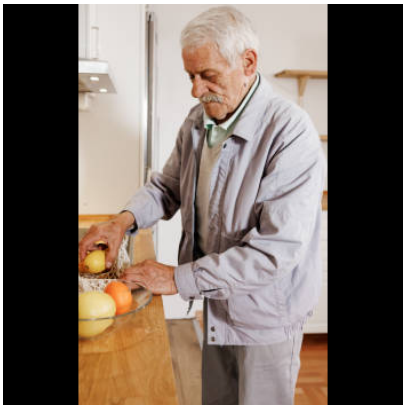


Figura 3.10: Redimensionamiento con relleno de bordes.

División del conjunto de datos

La distribución del conjunto de datos para el entrenamiento del modelo se detalla en la Tabla 3.2 y la proporción de las imágenes en los distintos conjuntos se presenta en la Tabla 3.3.

Tabla 3.2: Distribución del conjunto de datos.

Imágenes			
Aumentadas	**Con contexto**	**Negativas**	**Total**
24180	2209	2418	28807

Con contexto (adultos mayores + entorno), Negativas (fondos + personas <60 años).

Tabla 3.3: División del conjunto de datos.

Entrenamiento	**Validación**	**Prueba**
70 %	15 %	15 %

Para garantizar una mejor división de los datos, se realizaron modificaciones al comando *labelme2yolo* de Labelme (ver Apéndice C.3).

3.2. Transferencia de aprendizaje

La transferencia de aprendizaje es una técnica que permite aprovechar el conocimiento previamente adquirido por un modelo preentrenado para abordar nuevas tareas con menor cantidad de datos y recursos computacionales. A continuación, se describe el enfoque adoptado para utilizar la arquitectura YOLOv8s, detallando los ajustes

realizados para adaptar el modelo a las características específicas del conjunto de datos AM-2025.

3.2.1. Modelo especializado

El proceso para obtener un modelo especializado se desarrolló conforme a las etapas mostradas en la Figura 3.11, las cuales se describen a continuación:

① Elección de un modelo preentrenado (ver sección 2.5.5).
② Generación de un conjunto de datos (ver sección 3.1).
③ Adaptación del modelo utilizando transferencia de aprendizaje.

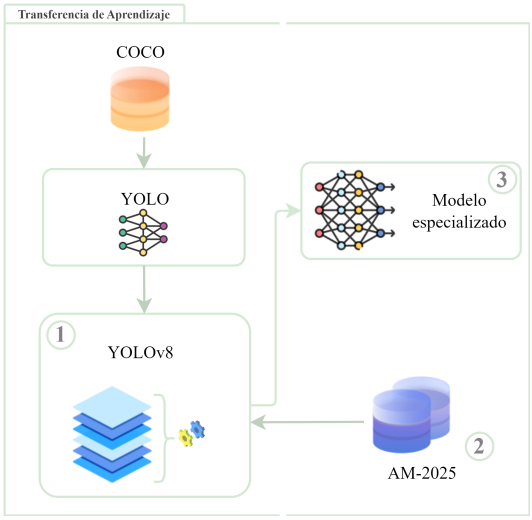


Figura 3.11: Flujo de trabajo.

Para adaptar el modelo YOLOv8s a la tarea de detección, se realizaron las siguientes modificaciones:

- La capa completamente conectada original fue sustituida por una capa de estructura binaria (dos salidas).
- Se reentrenaron todas las capas del modelo (*backbone*, *neck* y *head*) para preservar parcialmente los pesos preentrenados mientras se adaptaban a la nueva

tarea.

Estas adaptaciones permitieron aprovechar el conocimiento adquirido por YOLOv8s en COCO, mientras se especializaba su capacidad de clasificación en el dominio de AM-2025. El proceso se implementó bajo los criterios especificados en las Tablas 3.4 y 3.5.

Tabla 3.4: Hiperparámetros de entrenamiento.

Parámetro	Descripción
Learning rate	Inicial (lr0): 0.01, Final (lrf): 0.1
Batch size	16 muestras por lote
Épocas	50 iteraciones completas
Optimizador	SGD con momentum (0.937)
Función de pérdida	$Loss_{total} = Loss_{cls} + Loss_{box} + Loss_{obj}$

Tabla 3.5: Métricas durante los entrenamientos.

No. Ent.	Épocas	Tiempo total [h]	Precisión	Recall	mAP50	mAP50-95
1	100	9.70	0.9836	0.9855	0.9938	0.9928
2	113	8.02	0.9893	0.9901	0.9938	0.9926
3	50	9.35	0.9761	0.9753	0.9926	0.9895

Derivado del tercer entrenamiento, se presentan los resultados del modelo especializado.

Entrenamiento

De acuerdo con las gráficas presentadas en la Figura 3.12, se observa que tanto la pérdida asociada a las cajas delimitadoras (box_loss) como la pérdida de clasifica-

ción (cls_loss) convergen rápidamente hacia valores cercanos a cero, lo que indica estabilidad durante el proceso de aprendizaje.

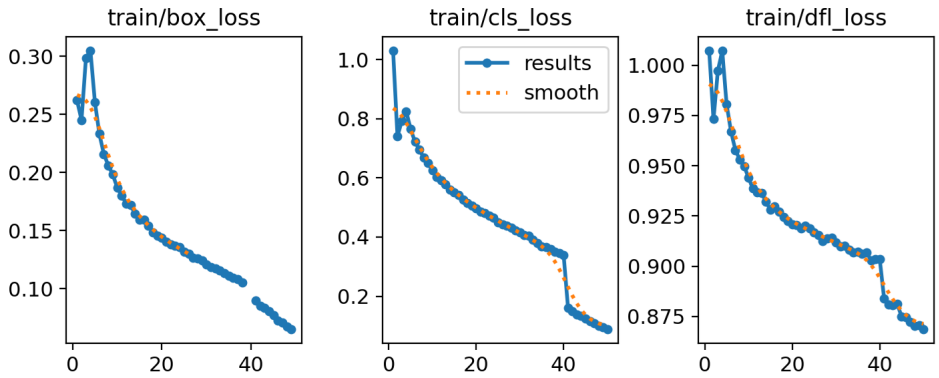


Figura 3.12: Métricas durante el tercer entrenamiento.

Validación

Los resultados en la Figura 3.13 muestran un alto rendimiento del modelo. En la gráfica de box_loss, se observa que el modelo predice con precisión la ubicación y el tamaño de las personas, mostrando valores de pérdida bajos y estables. Paralelamente, la gráfica de cls_loss revela que el modelo presenta una tasa de error baja y una pérdida que converge rápidamente a cero, lo que indica una discriminación efectiva entre clases. Este comportamiento confirma que el modelo ha aprendido tanto a localizar como a identificar correctamente a las personas en el conjunto de validación.

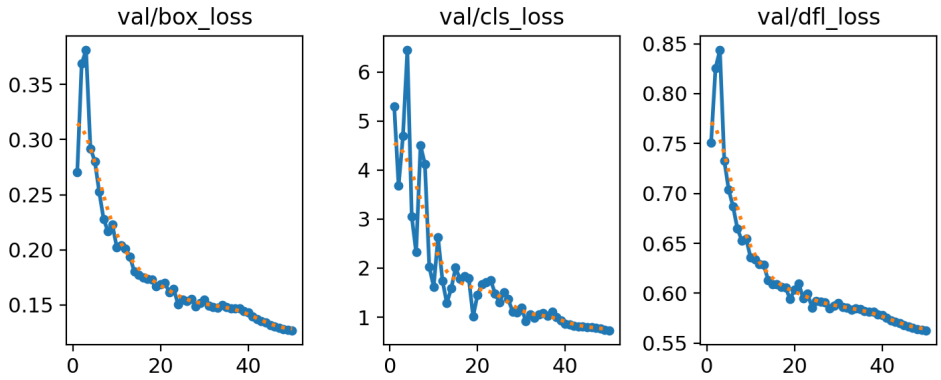


Figura 3.13: Métricas durante la validación.

Durante la etapa de validación, el modelo alcanzó una precisión del 0.976 (metrics/-precision(B)), lo que significa que solo un 0.024 son falsos positivos. Además, muestra un recall del 0.975 (metrics/recall(B)), lo que indica solo un 0.025 de falsos negativos, ver Tabla 3.6.

Tabla 3.6: Métricas durante la validación.

Clase	**Precisión**	**Recall**	**mAP50**	**mAP50-95**
0	0.973	0.980	0.992	0.990
1	0.979	0.971	0.993	0.989
Total	0.976	0.975	0.993	0.989

En cuanto a la exactitud general, en la Figura 3.14 se observa que el modelo logra un mAP50 (metrics/mAP50(B)) del 0.993, lo que demuestra un rendimiento cercano al 1. Su robustez se confirma con un mAP50-95 (metrics/mAP50-95(B)) del 0.989, manteniendo una detección precisa incluso con una superposición del 0.95.

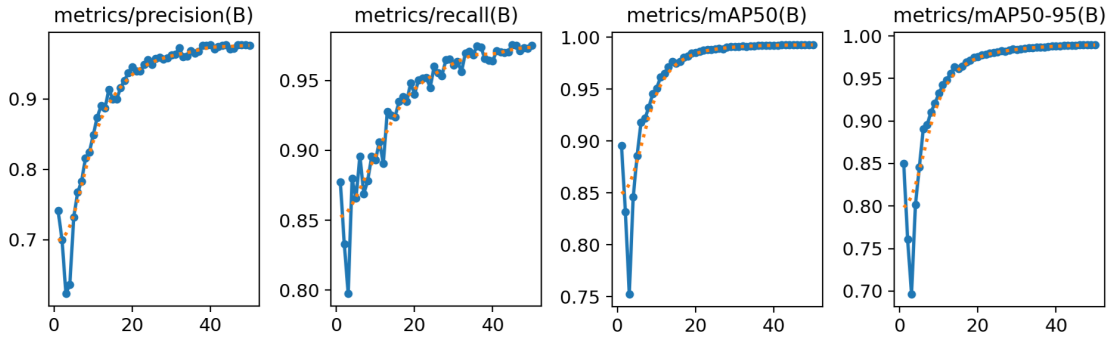


Figura 3.14: Métricas durante la validación.

En el conjunto de validación, algunas imágenes que no contenían objetos etiquetados pertenecientes a las clases 0 o 1 se clasificaron como negativas (*background*). Sin embargo, si el modelo generaba una detección en estas imágenes, dichas predicciones se contabilizan como falsos positivos en la clase correspondiente al background, ver Figura 3.15.

Aunque las imágenes vacías no contribuyeron a las métricas de las clases de interés, los falsos positivos en el background se reflejan en la matriz de confusión como predicciones incorrectas del modelo.

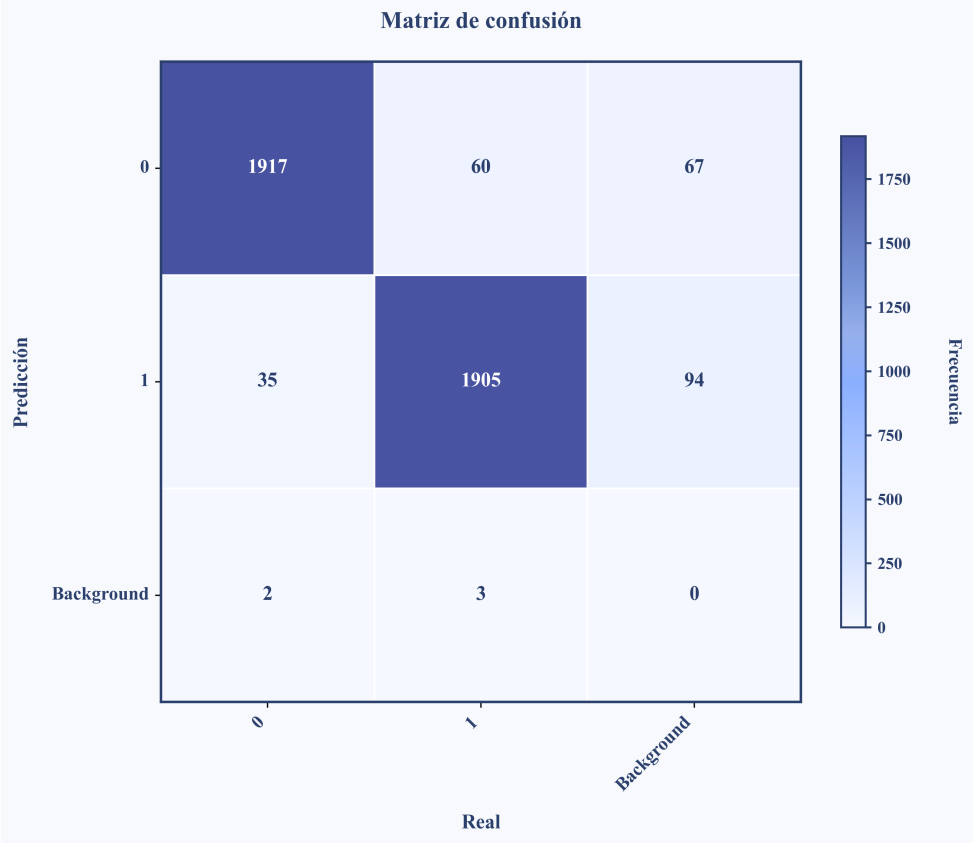


Figura 3.15: Matriz de confusión con el conjunto de validación.

Prueba

Los resultados obtenidos con el conjunto de prueba muestran un alto desempeño en la detección, ver resumen en la Tabla 3.7.

Tabla 3.7: Métricas con el conjunto de prueba.

Clase	Precisión	Recall	mAP50	mAP50-95
0	0.980	0.969	0.983	0.979
1	0.986	0.952	0.974	0.969
Total	0.983	0.960	0.978	0.974

Además, la matriz de confusión (ver Figura 3.16) muestra que el modelo alcanza un rendimiento robusto, con errores mínimos en ambas clases.

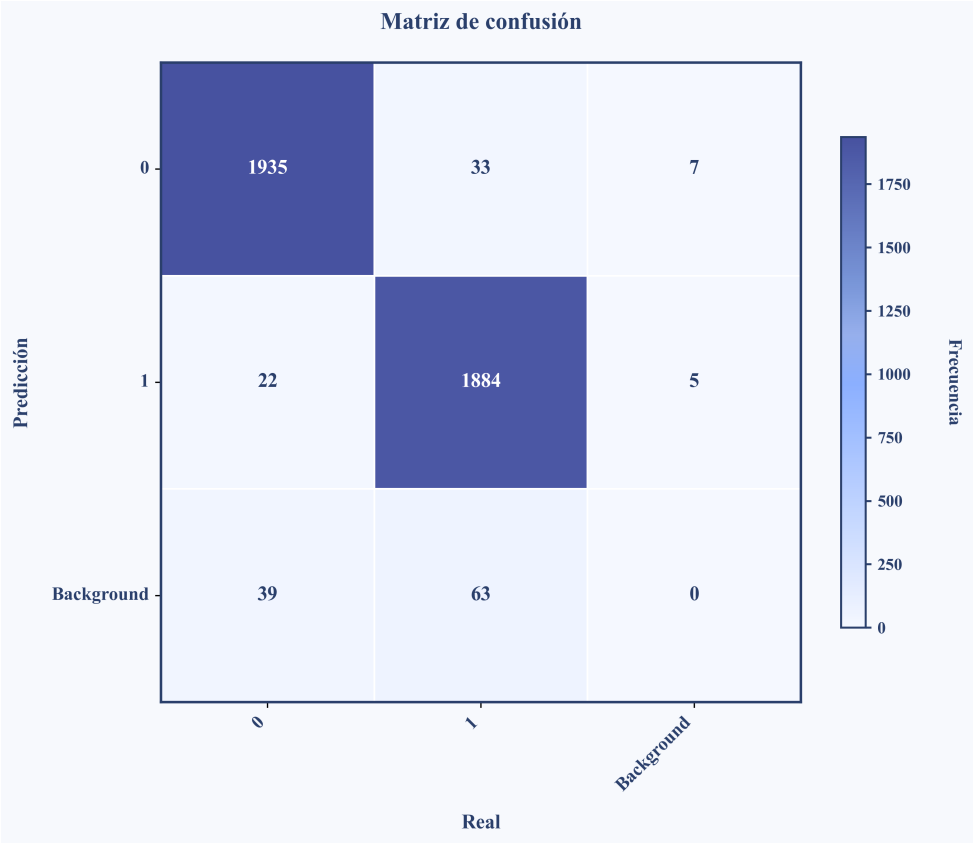


Figura 3.16: Matriz de confusión con el conjunto de prueba.

Prueba en modelos 3D

Al trabajar en un entorno de simulación, fue necesario probar el modelo de detección de adultos mayores con imágenes de modelos 3D. Sin embargo, la cantidad de modelos fue limitada. Por ello, se empleó un conjunto de prueba integrado por 76 imágenes —obtenidas a partir de 36 modelos 3D vistos desde distintas perspectivas— distribuidas según se detalla en la Tabla 3.8.

Tabla 3.8: Distribución de las imágenes de los modelos 3D.

Modelos	**Cantidad**	**Imágenes**
Hombre	14	27
Mujer	22	49

El rendimiento del modelo decreció considerablemente al ser evaluado con imágenes de modelos 3D de adultos mayores, evidenciando un problema de dominio de datos (*domain shift*) (ver Tabla 3.9). Asimismo, se corroboró la importancia de elegir de manera rigurosa el tipo de modelo 3D a implementar en el simulador, dado que, el estilo animación impacta en la detección.

En la Figura 3.17, se muestran ejemplos de las predicciones realizadas sobre imágenes externas al conjunto original, donde se observan cajas delimitadoras para las clases “0” y “1”,y el porcentaje de confianza del modelo asociado a cada predicción.

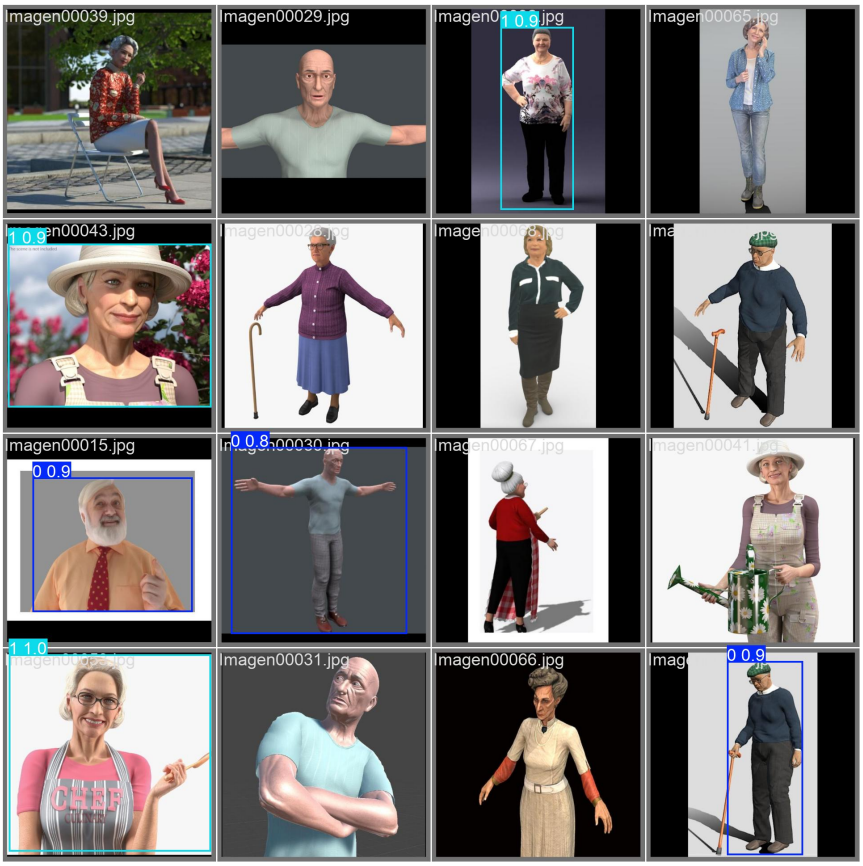


Figura 3.17: Predicciones en un batch de prueba en modelos 3D.

Tabla 3.9: Métricas con el conjunto de prueba (modelos 3D).

Clase	Precisión	Recall	mAP50	mAP50-95
0	1	0.222	0.611	0.591
1	0.75	0.245	0.479	0.385
Total	0.875	0.234	0.545	0.488

Capítulo 4
Robot planar 3R

En este capítulo se presenta el análisis cinemático de un robot planar 3R. El proceso incluyó el diseño CAD del sistema, su conversión al formato URDF y la validación de la cinemática en simulación.

4.1. Análisis cinemático

En esta sección se presenta el análisis cinemático de un robot de 3 GDL en configuración RRR, mejor conocido como manipulador planar 3R. Utilizando el enfoque geométrico (ver Figura 4.1), este análisis contempla dos aspectos: a) la cinemática directa, que permite calcular la posición del efector final a partir de los ángulos articulares conocidos y b) la cinemática inversa, que consiste en determinar los ángulos articulares requeridos para alcanzar una posición objetivo en el espacio de trabajo.

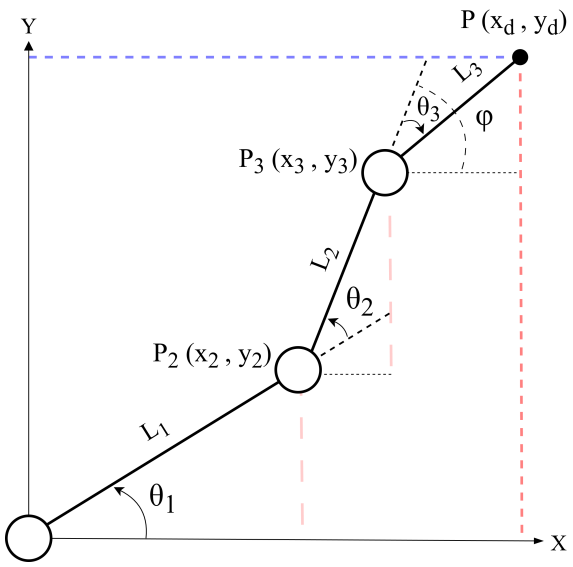


Figura 4.1: Representación funcional del manipulador planar 3R.

4.1.1. Cinemática directa

Para determinar la posición y orientación del robot planar 3R, ver Figura 4.1, se calculan las proyecciones de cada eslabón (L_1, L_2, L_3) sobre los ejes coordenados (X, Y), de modo que la cinemática directa queda definida como:

$$x_d = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2) + L_3 \cos(\theta_1 + \theta_2 + \theta_3) \tag{4.1}$$

$$y_d = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) + L_3 \sin(\theta_1 + \theta_2 + \theta_3) \tag{4.2}$$

$$\phi = \theta_1 + \theta_2 + \theta_3 \tag{4.3}$$

4.1.2. Cinemática inversa

La cinemática inversa consiste en determinar los valores articulares que permiten al efector final alcanzar una posición deseada en el espacio de trabajo. En el caso de un robot 3R, este proceso implica quitar el tercer eslabón (L_3) para simplificar el análisis a un manipulador de 2R que debe alcanzar el punto P_3 (ver Figura 4.2).

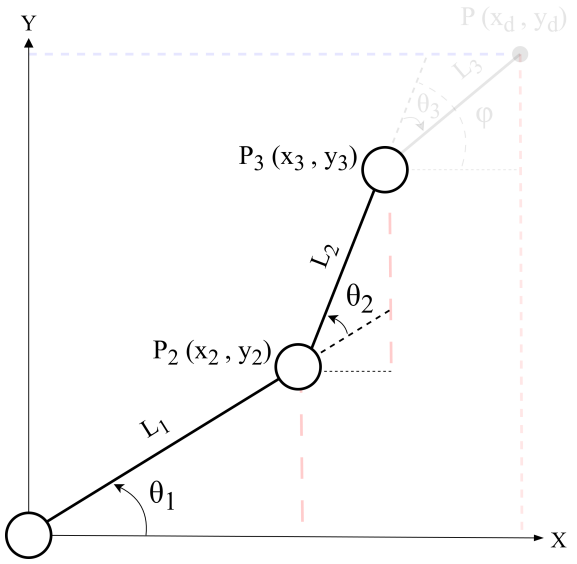
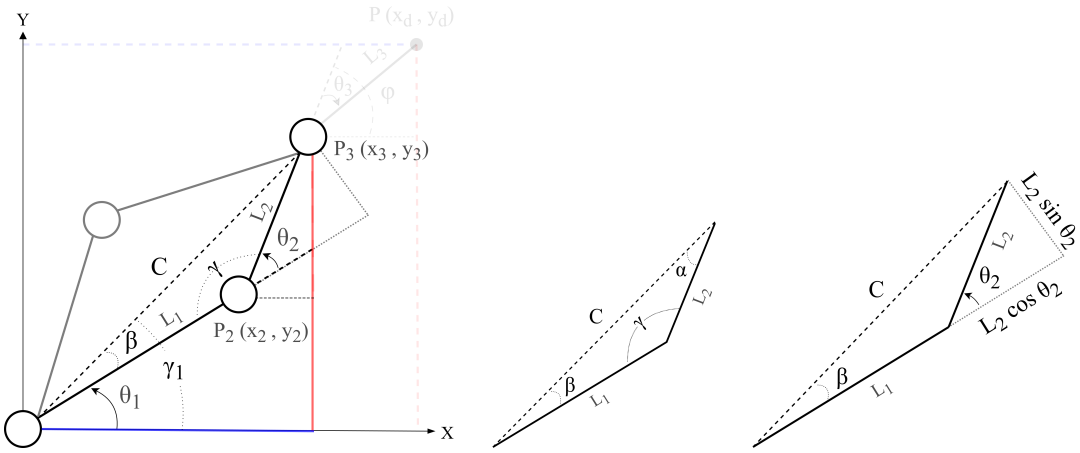


Figura 4.2: Representación funcional del manipulador planar 2R.

Inicialmente, la posición de P_3 queda definida como:

$$x_3 = x_d - L_3 \cos\phi \tag{4.4}$$

$$y_3 = y_d - L_3 \sin\phi \tag{4.5}$$



(a) Manipulador planar 2R. (b) Para obtener θ_2. (c) Para obtener θ_1.

Figura 4.3: Descomposición geométrica.

Después, se realiza una descomposición geométrica para obtener θ_1 y θ_2, ver Figura 4.3.

Para determinar θ_2

En la Figura 4.3(a), se observa que para obtener θ_2, es necesario conocer el ángulo complementario γ y la distancia C.

$$180^\circ = \gamma + \theta_2 \quad \Rightarrow \quad \gamma = 180^\circ - \theta_2 \tag{4.6}$$

$$C^2 = x_3^2 + y_3^2 \quad \Rightarrow \quad C = \sqrt{x_3^2 + y_3^2} \tag{4.7}$$

Aplicando la Ley de Cosenos al triángulo de la Figura 4.3(b), se obtiene $-\cos(\gamma)$ sustituyendo el valor de C en la ecuación (4.8):

$$C^2 = L_1^2 + L_2^2 - 2L_1L_2\cos\gamma \quad \Rightarrow \quad -\cos(\gamma) = \frac{C^2 - L_1^2 - L_2^2}{2L_1L_2} \tag{4.8}$$
$$-\cos(\gamma) = \frac{x_3^2 + y_3^2 - L_1^2 - L_2^2}{2L_1L_2}$$

Sustituyendo (4.6) en (4.8) y aplicando la identidad trigonométrica del coseno $\cos(\pi - \sigma) = -\cos(\sigma)$ se obtiene:

$$-\cos(180° - \theta_2) = \frac{x_3^2 + y_3^2 - L_1^2 - L_2^2}{2L_1L_2} \quad \Rightarrow \quad \cos(\theta_2) = \frac{x_3^2 + y_3^2 - L_1^2 - L_2^2}{2L_1L_2} = D \tag{4.9}$$

En la ecuación (4.10) se define θ_2 como:

$$\tan(\theta_2) = \frac{\sin(\theta_2)}{\cos(\theta_2)} \quad \Rightarrow \quad \theta_2 = \arctan\left(\frac{\sin(\theta_2)}{\cos(\theta_2)}\right) \tag{4.10}$$

Utilizando la identidad trigonométrica $\cos^2(\sigma) + \sin^2(\sigma) = 1$

$$\sin^2(\theta_2) = 1 - \cos^2(\theta_2) \quad \Rightarrow \quad \sin(\theta_2) = \pm\sqrt{1 - \cos^2(\theta_2)} \tag{4.11}$$

Sustituyendo (4.9) y (4.11) en (4.10) se obtiene:

$$\theta_2 = \arctan 2\left(\pm\sqrt{1 - D^2}, D\right) \tag{4.12}$$

Para determinar θ_1

En la Figura 4.3(a), se define a θ_1 como:

$$\theta_1 = \gamma_1 - \beta \tag{4.13}$$

A su vez, γ_1 queda como:

$$\gamma_1 = \arctan\left(\frac{y_3}{x_3}\right) \tag{4.14}$$

Después, en la Figura 4.3(c) se obtiene β.

$$\beta = \arctan 2\left(L_2 \sin(\theta_2), L_1 + L_2 \cos(\theta_2)\right) \tag{4.15}$$

Sustituyendo (4.14) y (4.15) en (4.13).

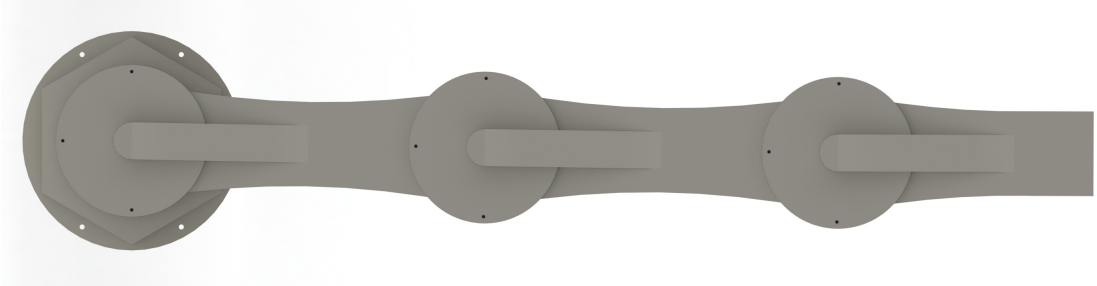
$$\theta_1 = \arctan 2\left(y_3, x_3\right) - \arctan 2\left(L_2 \sin(\theta_2), L_1 + L_2 \cos(\theta_2)\right) \tag{4.16}$$

Regresando al problema de 3R:

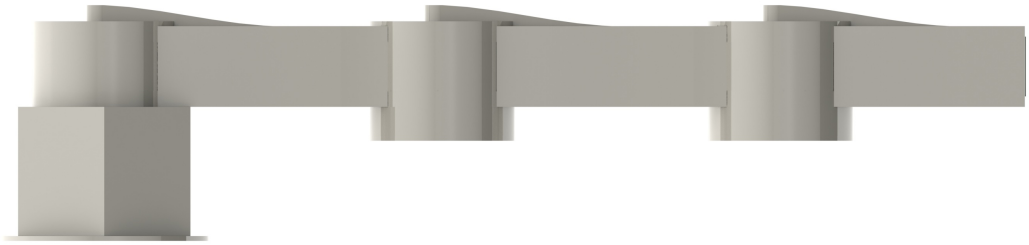
$$\theta_3 = \phi - \theta_1 - \theta_2 \tag{4.17}$$

4.1.3. Diseño CAD

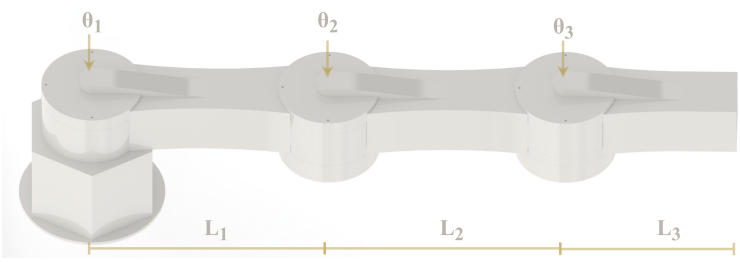
Tomando como punto de partida la representación funcional (ver Figura 4.1), el siguiente paso fue el diseño del robot mediante el uso del software SolidWorks®. El modelo final puede observarse en la Figura 4.4.



(a) Vista superior.



(b) Vista frontal.



(c) Vista isométrica (*home*).

(d) Vista isométrica.

Figura 4.4: Vista del robot.

En el diseño se consideró la posibilidad de construirlo a futuro. Para asegurar la viabilidad del modelo CAD en este aspecto, se seleccionó un servomotor comercial como referencia principal para todas las dimensiones, ver Tabla 4.1 y apéndice D.4.

Tabla 4.1: Dynamixel AX-12A[1].

Parámetro	**Especificaciones**
Dimensiones	$32 \times 50 \times 40\,\text{mm}$
Rango de giro	$0 \sim 300\,^{\circ}$
Par	$1.5\,\text{Nm}$
Velocidad	$59\,\frac{\text{rev}}{\text{min}}$

4.1.4. Descripción del robot en formato URDF

El formato URDF describe la estructura del robot, sus articulaciones, enlaces, sistemas de coordenadas, propiedades físicas básicas (como masa e inercia) y la jerarquía del modelo, lo cual es necesario para que Gazebo interprete adecuadamente el modelo.

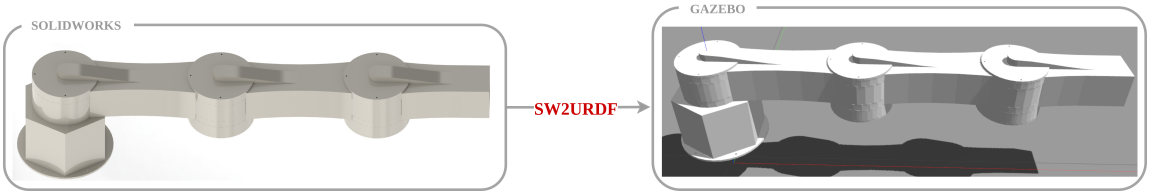


Figura 4.5: Obtención del formato URDF.

A partir del diseño CAD (sección 4.1.3), se obtuvo un archivo .sldasm que fue convertido a .urdf mediante la herramienta *sw2urdf* del software SolidWorks®(Apéndice

[1]Recuperado de https://emanual.robotis.com.

E.5), ver Figura 4.5.

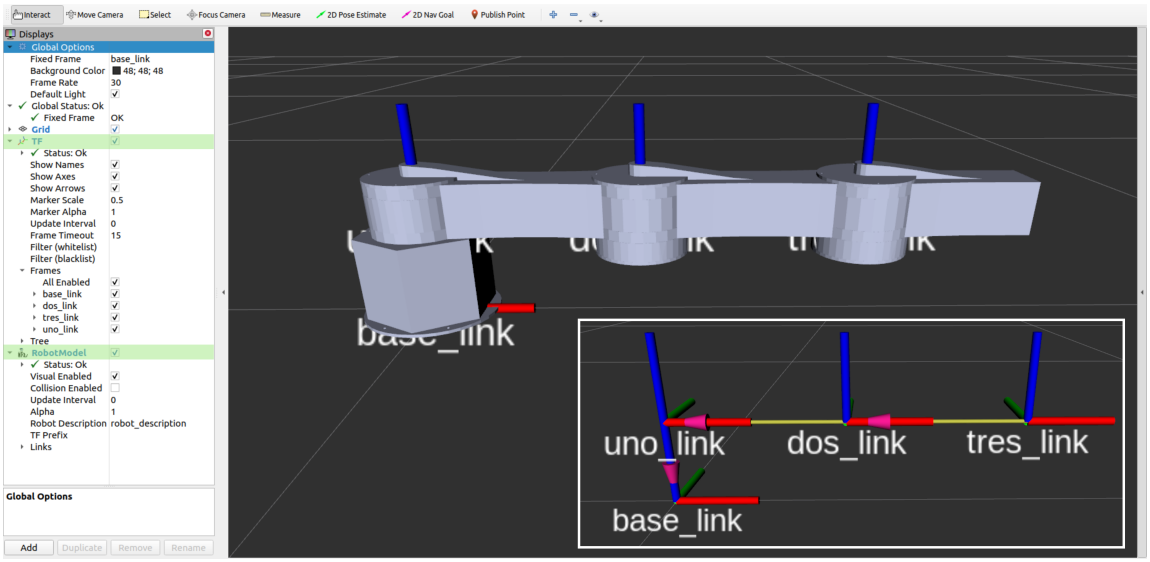


Figura 4.6: Visualización del robot en RVIZ.

Para verificar que el .urdf describe de forma correcta al robot se hace uso de la herramienta RVIZ. En la Figura 4.6 aparecen dos componentes principales: 1) el *TF* para visualizar la posición y la orientación de todos los marcos de referencia de las articulaciones, y 2) *RobotModel*, que muestra el modelo CAD del robot por medio de archivos .stl.

Integración de la cámara RGB-D

Para el desarrollo de una aplicación robótica basada en visión, se incorporó una cámara RGB-D al archivo .urdf del robot, lo que permitió la adquisición simultánea de datos de color y profundidad.

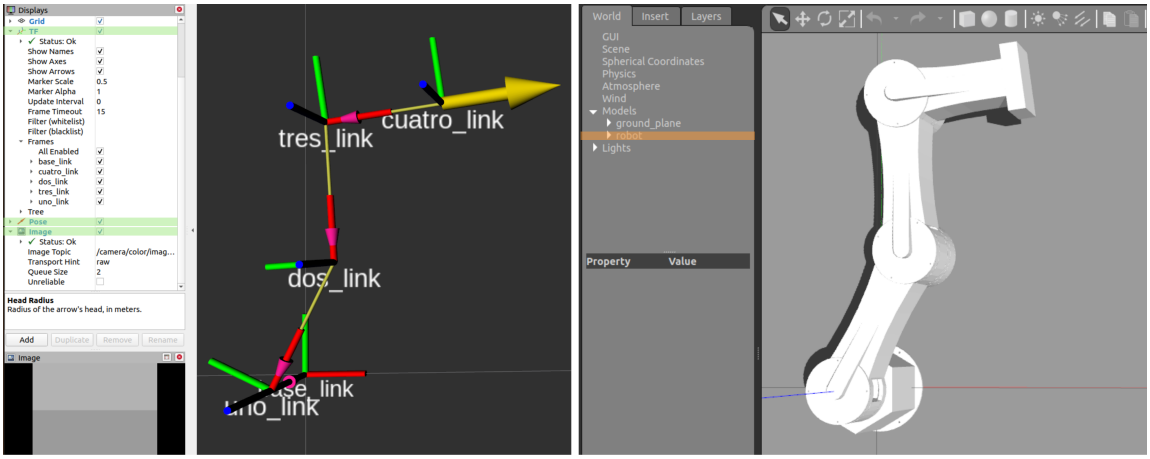


Figura 4.7: Visualización del robot con cámara RGB-D en RViz y en Gazebo.

En la parte izquierda de la Figura 4.7 se presenta la visualización en RViz, donde se observan tres elementos principales: *TF*, *Pose* e *Image*. Estos corresponden, respectivamente, a los sistemas de referencia del robot y de la cámara, una referencia visual del eje Z de la cámara y la visualización de la imagen captada por el sensor. En contraste, en la parte derecha se muestra la simulación en Gazebo, donde únicamente se aprecia el modelo del robot y la cámara integrada.

4.1.5. Espacio de trabajo

En la Figura 4.8 se muestra el espacio de trabajo del robot. El área amarilla representa los límites físicos de cada articulación (Tabla 4.1), definidos en el rango de $0 - 5.23599\,\text{rad}$. La región azul muestra los límites operativos de $0 - 3.14159\,\text{rad}$ establecidos mediante software, que delimitan el área de trabajo real en la simulación.

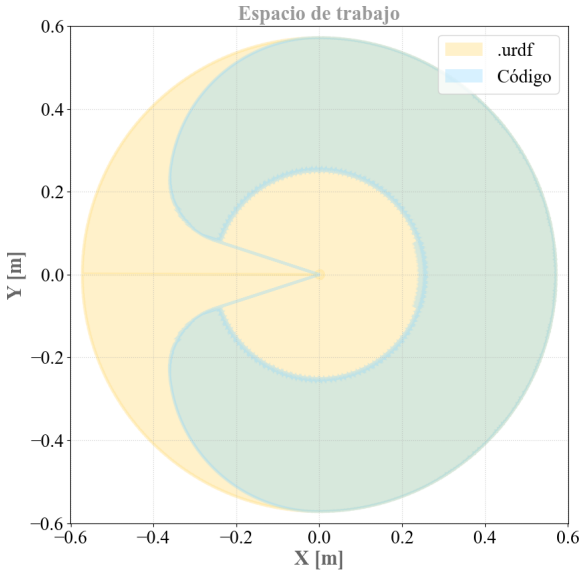


Figura 4.8: Espacio de trabajo teórico y operativo del robot.

4.1.6. Evaluación de la cinemática del robot

La verificación y validación de la cinemática del robot se realizó en el ecosistema ROS, empleando Gazebo como simulador físico y RViz como entorno de visualización. En Gazebo se evaluó el comportamiento dinámico del modelo URDF al ejecutar trayectorias y alcanzar las posiciones deseadas, mientras que en RViz se verificaron las transformaciones de referencia (TF) y las poses publicadas por ROS.

Cinemática directa

Con el objetivo de comprobar que las ecuaciones implementadas en el modelo URDF del robot coinciden con la descripción matemática del mismo (ver sección 4.1), se realizó la verificación de la cinemática directa e inversa.

La Figura 4.9 muestra el diagrama de flujo del algoritmo de la cinemática. Este incluye el cálculo de las variables articulares, la saturación de sus valores para garantizar que se mantengan dentro de los límites operativos y una etapa de verificación que utiliza la cinemática directa para comparar la posición deseada con la posición planeada.

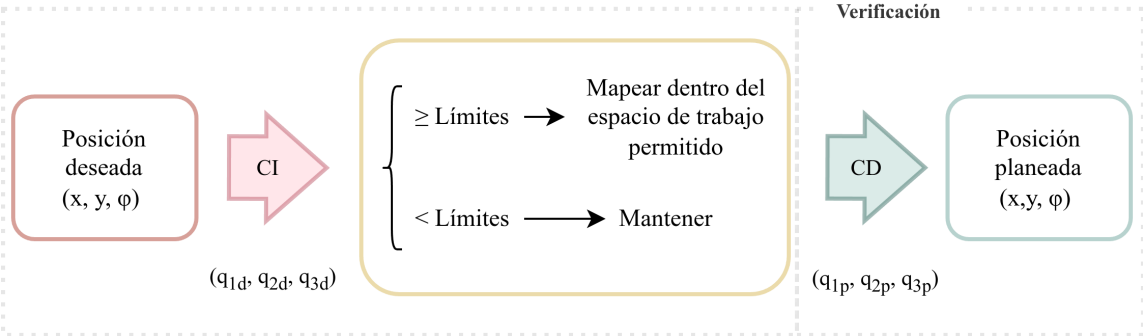


Figura 4.9: Obtención de las variables articulares.

En la Figura 4.10 se comparan las posiciones deseadas $P(x_d, y_d)$ con las posiciones reales alcanzadas por el robot al recibir los valores q_{1d}, q_{2d} y q_{3d}. Los resultados confirman que los cálculos de la cinemática directa permiten al efector final (final de L_3) alcanzar las posiciones objetivo de manera efectiva.

Por otro lado, la Figura 4.11 muestra la evolución temporal de las variables articulares para alcanzar la secuencia de posiciones deseadas.

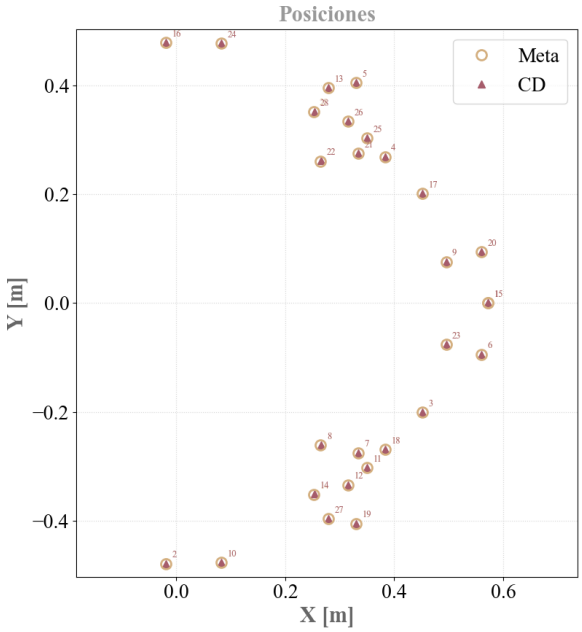


Figura 4.10: Validación de la posición: deseada *vs* real.

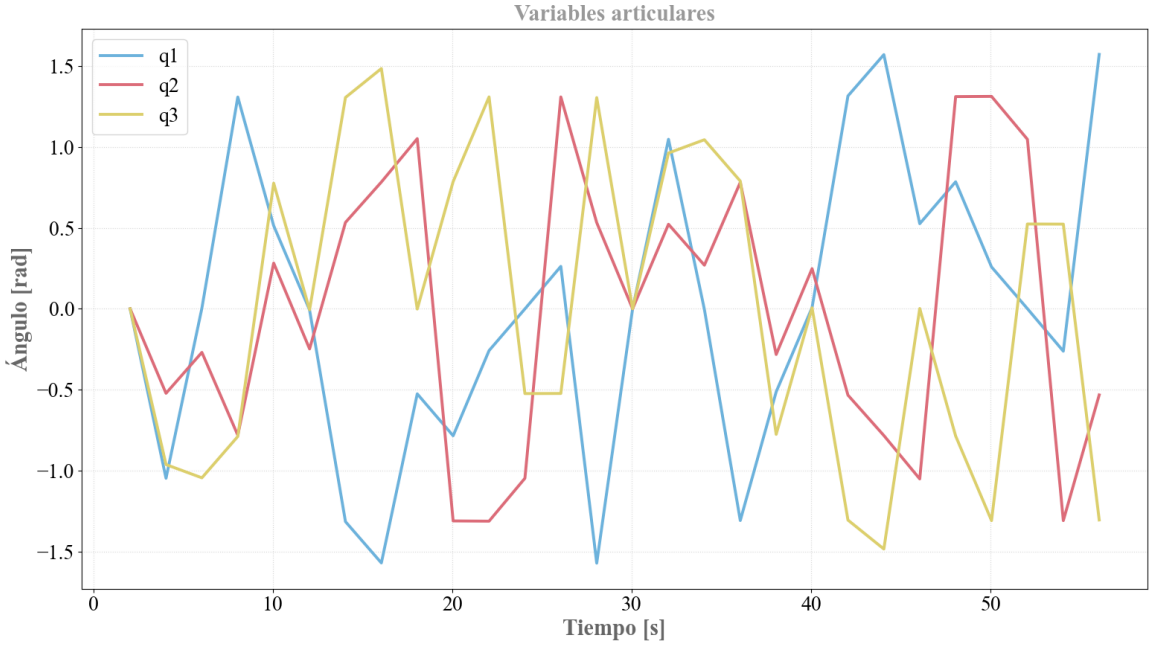


Figura 4.11: Evolución temporal de las variables articulares.

Cinemática inversa

En seguida, se presenta la validación de la cinemática inversa del robot, definiendo el sistema de coordenadas de la cámara como el TCP. La evaluación se centró en el desempeño del robot para alcanzar las posiciones objetivo $P(x_d, y_d, \phi_d)$.

La Figura 4.12 muestra la trayectoria continua del TCP en comparación con el conjunto de posiciones meta establecidas. Se observa que la trayectoria resultante circunda ciertas metas sin alcanzarlas directamente, lo que indica la existencia de puntos inalcanzables dentro del espacio de trabajo.

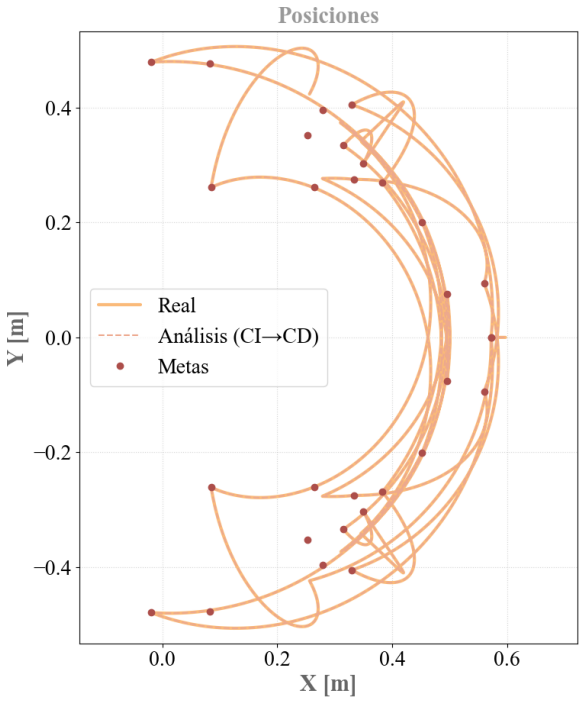


Figura 4.12: Trayectoria continua y puntos meta inalcanzables.

El análisis de las variables articulares (ver Figura 4.12) muestra la evolución temporal de las señales, observándose una superposición prácticamente completa entre las

variables articulares reales q_r y las deseadas q_d, lo que indica un análisis cinemático consistente.

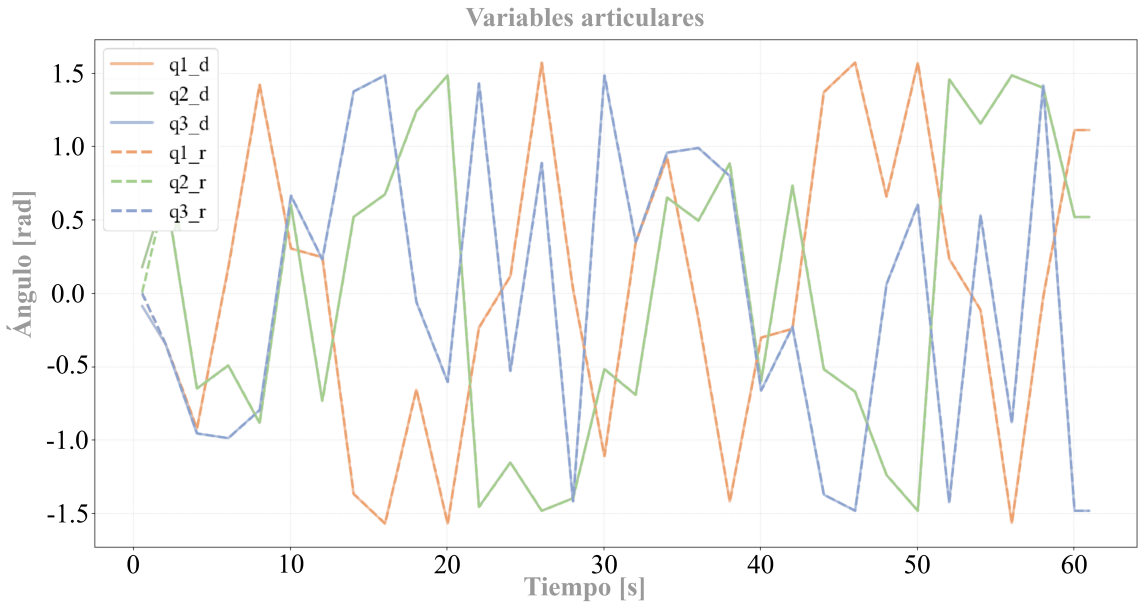


Figura 4.13: Variables articulares: deseadas y reales.

La Figura 4.14 presenta el comportamiento de la orientación del TCP, donde se confirma la superposición entre las señales reales (ϕ_r) y planeadas (ϕ_p), junto con la desviación respecto de la orientación deseada (ϕ_d).

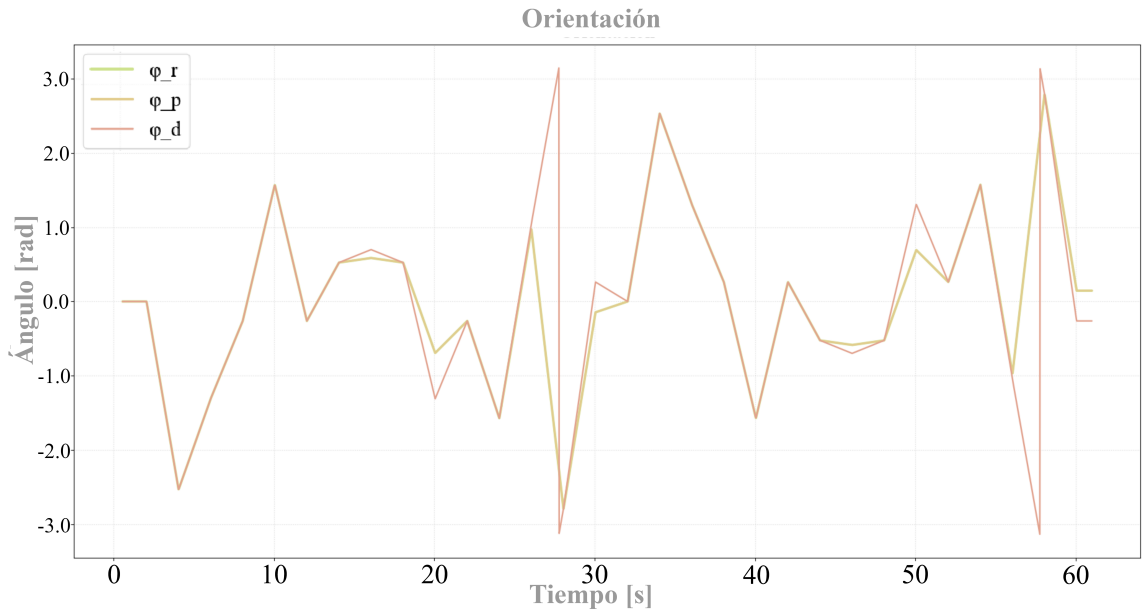


Figura 4.14: Orientación del efector final.

En la Figura 4.15 se muestran picos de error en posición y orientación que se correlacionan temporalmente con las metas identificadas como inalcanzables. Los errores alcanzan valores de 0.3 m en la posición y de 3 rad en la orientación.

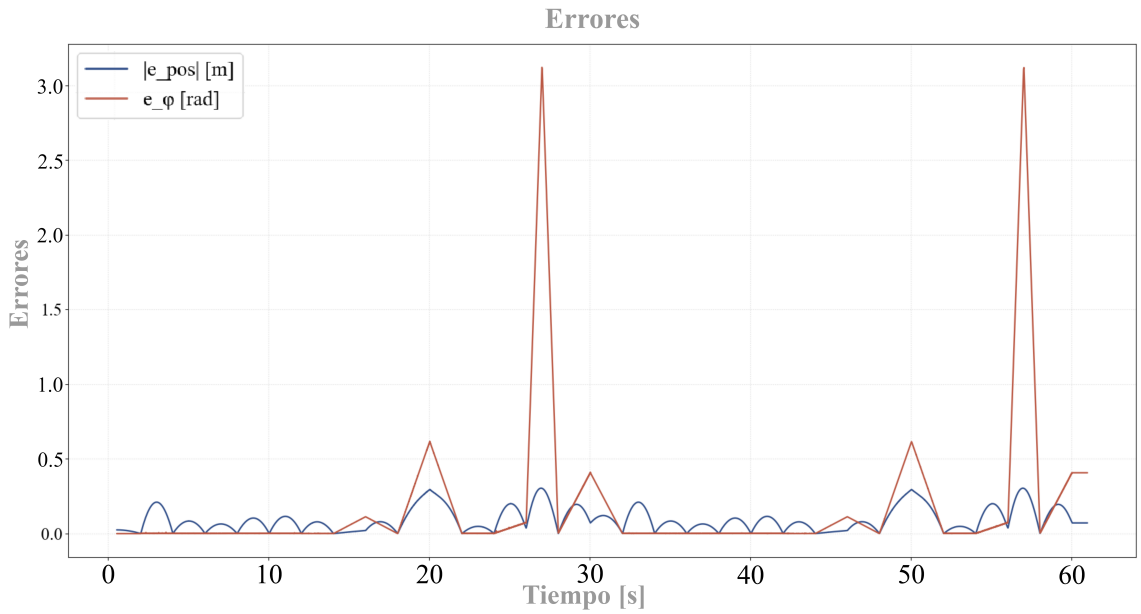


Figura 4.15: Errores asociados a orientaciones inviables.

Los resultados demuestran que la orientación deseada impone limitaciones significativas en la capacidad del robot para alcanzar las posiciones objetivo. La presencia de puntos inalcanzables (Figura 4.12) se atribuye directamente a conflictos entre la orientación deseada y los límites articulares del robot. Específicamente, se identificaron dos casos en los que las coordenadas cartesianas (x,y) eran accesibles, pero la orientación (ϕ) resultaba inviable.

La superposición observada entre las señales q_r y q_p (Figura 4.13) confirma que el sistema de control ejecuta fielmente las trayectorias planeadas. Las discrepancias entre q_p, q_r y q_d validan la efectividad del recorte articular, que previene configuraciones mecánicamente inviables a expensas de la precisión en el seguimiento de trayectorias.

La relación entre los picos de error (Figura 4.15) y las metas inalcanzables refuerza la hipótesis de que las limitaciones articulares constituyen el factor principal de

la degradación del desempeño. Este comportamiento es particularmente evidente en orientación (Figura 4.14), donde ϕ_p y ϕ_r se desvían sistemáticamente de ϕ_d en regiones donde la configuración requerida excede los límites físicos del robot.

Estos resultados demuestran la importancia de considerar las restricciones de orientación durante la planificación de trayectorias.

Capítulo 5
Integración ROS-Gazebo

5.1. Entorno de simulación

En esta sección se describe el entorno de simulación implementado en Gazebo, detallando la configuración del mundo virtual y la integración con ROS para la ejecución de pruebas. La simulación se desarrolló con ROS Noetic y Gazebo Classic (versión 11.15.1).

La integración ROS-Gazebo se presenta en la Figura 5.1 y en el apéndice I.9 se muestra la arquitectura de ROS.

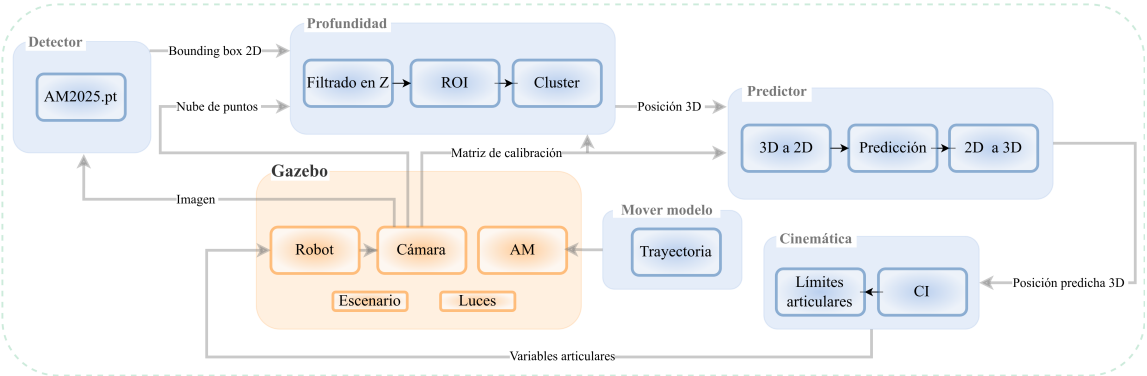


Figura 5.1: Diagrama general de la integración ROS-Gazebo.

El sistema de seguimiento opera mediante la ejecución simultánea de cinco nodos especializados, cada uno cumpliendo una función específica:

1. Nodo *Mover modelo*. La tarea de seguimiento comienza cuando el modelo del adulto mayor (AM) entra por primera vez en el campo de visión de la cámara del robot.

2. Nodo *Detector*. Es el encargado de la detección continua del adulto mayor. Este nodo consume los datos de la cámara y los procesa para identificar la presencia del modelo AM y determinar sus parámetros de imagen, como las coordenadas de su bounding box (ver capitulo 3).

3. Nodo *Profundidad*. Se encarga de calcular la posición tridimensional real del modelo AM en el marco de coordenadas globales de Gazebo. Este nodo suscribe la información de detección y la combina con los datos de profundidad de la cámara (nube de puntos) para transformar las coordenadas del plano de la imagen a la representación cartesiana del mundo (x, y, z) (ver sección 5.1.3).

4. Nodo *Predictor*. Recibe la secuencia de posiciones del modelo AM en el mundo y se encarga de estimar las posiciones a partir de la nube de puntos del sensor de profundidad y los bounding boxes del nodo detector (ver sección 5.1.4).

5. Nodo *Cinemática*. Recibe secuencias de posiciones del nodo predictor para mantener la cámara centrada en la persona de interés. Este nodo resuelve la cinemática inversa para obtener las variables articulares, verifica los límites articulares y genera el movimiento continuo del robot a lo largo de la simulación (ver capitulo 4).

5.1.1. Modelos personalizados

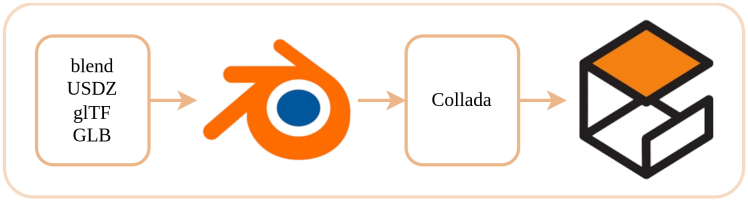
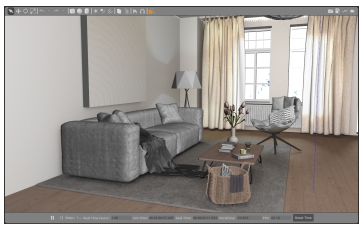
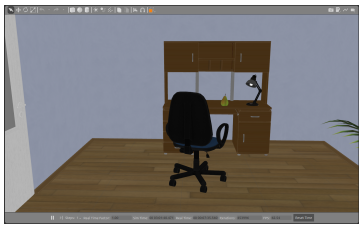


Figura 5.2: Integración de modelos a Gazebo.

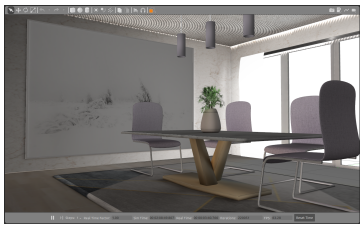
Para generar un entorno de simulación, fue necesario incorporar modelos 3D personalizados —adultos mayores y escenarios— en Gazebo. En Figura 5.2 se muestra el esquema general del proceso de conversión e integración de modelos en la biblioteca del simulador; para más detalles, ver el Apéndice J.10. Como resultado de este proceso, en la Figura 5.3 se observan distintos escenarios.



(a) Sala.



(b) Estudio.



(c) Comedor.

Figura 5.3: Escenarios.

Mientras que, en la Figura 5.4 se muestran tres modelos de adultos mayores. Las diferencias más notables son el género y el nivel de detalle de cada uno.

(a) Modelo 1. (b) Modelo 2. (c) Modelo 3.

Figura 5.4: Modelos 3D de adultos mayores.

Por otro lado, en la Figura 5.5 se presentan los tres modelos en el entorno de simulación Gazebo, ubicados en el escenario de la Figura 5.3(b), con la cámara en distintas posiciones.

(a) Gazebo. (b) Frame 1. (c) Frame 22.

Figura 5.5: Visualización de la detección con RQT.

En la Figura 5.5 se observa que uno de los modelos 3D no fue identificado como adulto mayor, lo que evidenció una limitación del modelo de detección ante geometrías con un bajo nivel de detalle (ver Figura 5.4(c)). En consecuencia, se seleccionaron los

modelos de mayor detalle visual para las pruebas posteriores (ver Figuras 5.4(a) y 5.4(b)).

5.1.2. Integración del sensor

Para la integración del sensor se incorporó una cámara RGBD basada en el modelo Intel RealSense D435. La configuración estéreo (ver Apéndice H.8) se definió con los parámetros de la Tabla 5.1.

Tabla 5.1: Cámara Intel D435[1].

Parámetro	**Especificaciones**
Dimensiones	$90 \times 25 \times 25\,\text{mm}$
FOV RGB	$1.20428\,\text{rad}$
FOV D	$1.5184\,\text{rad}$
Resolución del marco RGB	$1920 \times 1080\,\text{px}$
Resolución del marco D	$1280 \times 720\,\text{px}$
Plano de recorte cercano	$0.3\,\text{m}$
Plano de recorte lejano	$3\,\text{m}$

Nota: Campo de visión (FOV).

Esta configuración optimiza el balance entre el rendimiento de la simulación y la calidad de la nube de puntos. La Figura 5.6 muestra la cámara y la imagen capturada en el entorno de Gazebo. Se observa cómo varían la percepción de profundidad y la resolución en función de la distancia, según los parámetros de recorte definidos para el sensor.

[1]Recuperado de https://realsenseai.com.

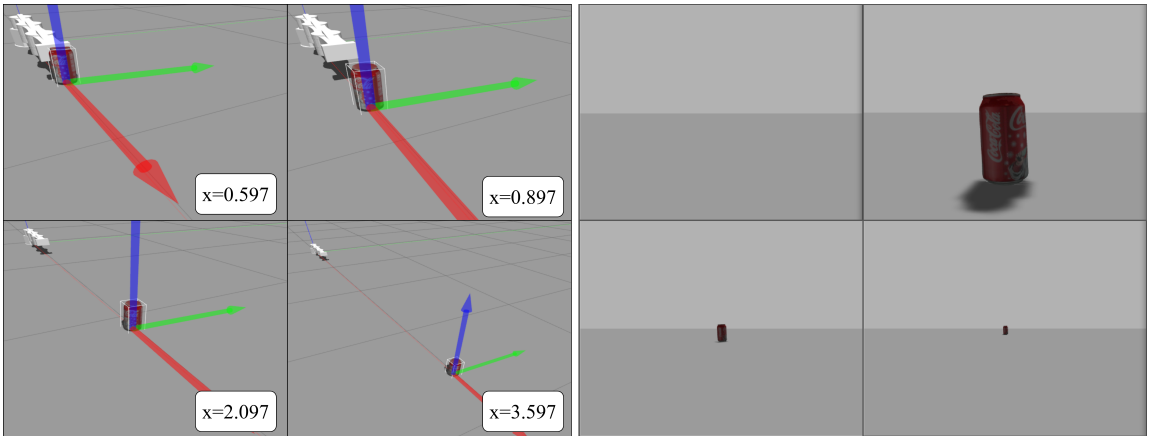


Figura 5.6: Visualización de un objeto capturado por la cámara RGBD a diferentes distancias.

5.1.3. Cálculo de la profundidad

Tras analizar en la sección 5.1.2 la visualización del sensor RGB, en esta sección se presenta la visualización del sensor de profundidad (sensor D), con el fin de complementar el análisis previo.

El comportamiento de la cámara puede observarse en la Figura 5.7, que representa el escenario mostrado en la Figura 5.3(c).

En la interfaz de RVIZ se distinguen tres elementos principales que permiten la interpretación de los datos capturados:

1. Fixed Frame: define el marco de referencia estático sobre el cual se visualizan todos los datos. En este caso, se encuentra configurado respecto del marco óptico de la cámara.

2. PointCloud2: visualiza la nube de puntos tridimensional generada por el sensor de profundidad, lo que permite representar la geometría del entorno.

3. Image: muestra la imagen del entorno simulado en Gazebo, capturada por el sensor RGB.

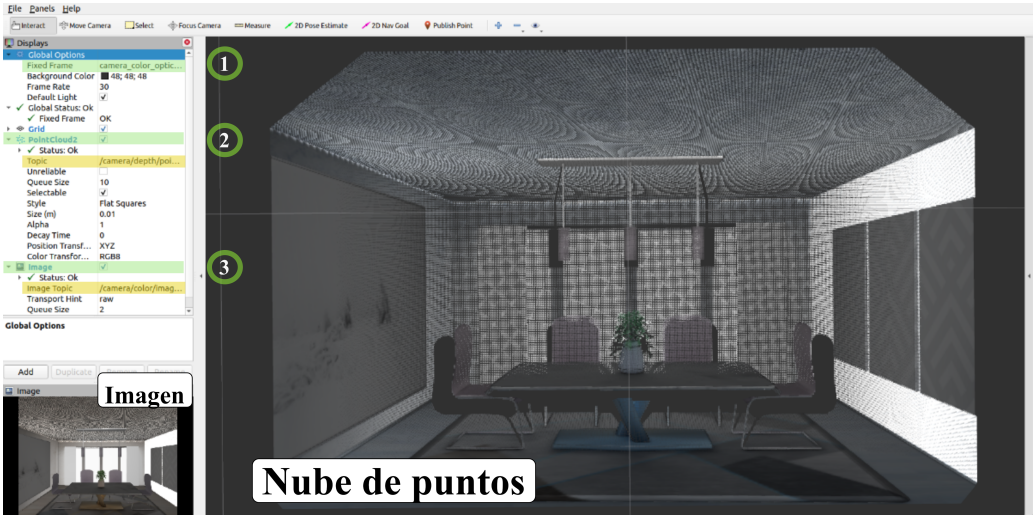


Figura 5.7: Visualización 3D de la cámara en RVIZ.

El cálculo de la profundidad se realiza mediante la integración de datos de detección 2D (nodo detector) y de nubes de puntos 3D (cámara) mediante un proceso de fusión. Primero, se sincronizan temporalmente los mensajes de bounding box y la nube de puntos, aplicando un filtro inicial de profundidad (0-3 m) (Filtro Z) para descartar el ruido. Luego, se proyectan los puntos 3D restantes a coordenadas 2D y se filtran nuevamente usando una región de interés (ROI) reducida basada en la bounding box. Posteriormente, se utiliza el algoritmo DBSCAN para agrupar los puntos 3D resultantes en clusters espaciales y se validan estos clusters seleccionando el que mejor se alinea con el centro 2D del bounding box mediante un cálculo del error de proyección utilizando la inversa de la matriz de calibración $(K^{-1})^2$. Finalmente, se calculan el centroide 3D (posición X, Y, Z) y las dimensiones del cluster validado,

y se publican estos datos como un mensaje para el nodo predictor.

5.1.4. Estimación de la posición

Es el componente central de análisis temporal, su función es anticipar la trayectoria del modelo AM para la planificación del sistema robótico. El proceso inicia al consumir la posición real 3D (x, y, z) publicada por el nodo de profundidad y, con el propósito de mitigar la inestabilidad en la medición de profundidad (z), proyecta dicha posición a las coordenadas estables de la imagen 2D (u, v) utilizando la matriz de calibración K, creando un historial dinámico de píxeles. Este historial se emplea para calcular la predicción del movimiento aplicando una extrapolación lineal a los dos puntos 2D más recientes. Con la predicción 2D y con la profundidad z real del instante actual, se genera y publica la posición predicha 3D para el nodo de la cinemática, mientras que internamente se evalúa y publica el error en píxeles, comparando la precisión de la estimación previa con la posición real observada.

[2]Más información en Multiple View Geometry in Computer Vision.

Capítulo 6
Resultados

En esta sección se presentan los resultados obtenidos tras la integración de los diferentes subsistemas que conforman el sistema de detección de adultos mayores en el simulador.

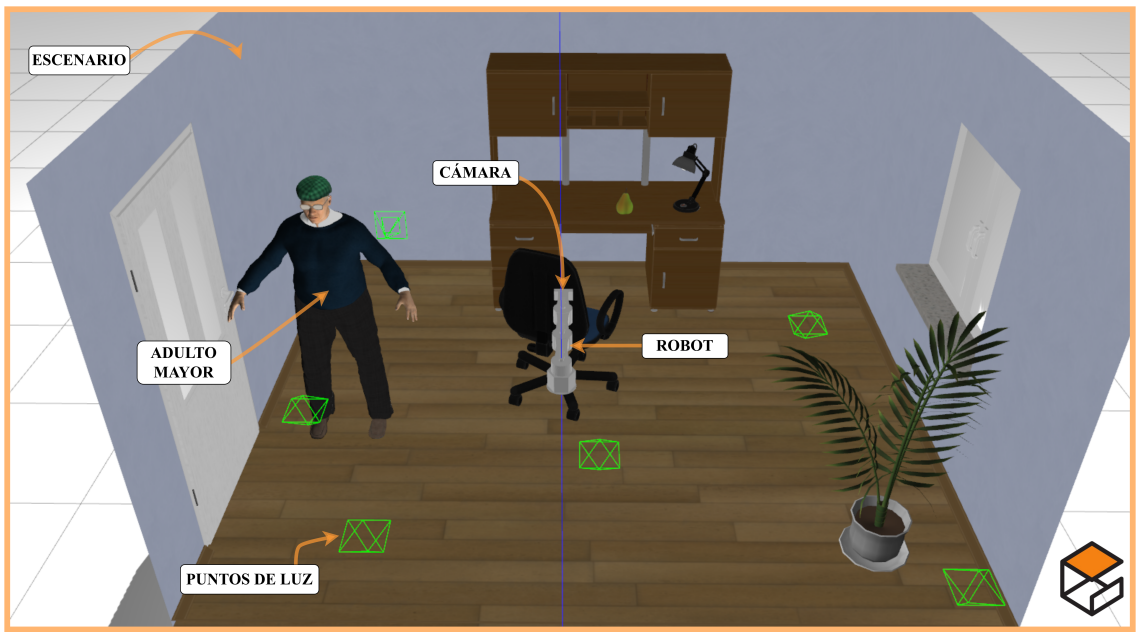


Figura 6.1: Diagrama general.

En la Figura 6.1 se observa que el mundo de Gazebo se compone de cinco elementos principales.

- Modelos 3D de personas de la tercera edad.
- Modelos 3D de habitaciones al interior de una casa.

- Un robot planar 3R, para mover la cámara a la posición deseada.
- Un sensor RGBD, encargado de capturar las imágenes que procesa el modelo de detección.
- Puntos de luz[1], indispensables para acentuar las características de las habitaciones y de los adultos mayores.

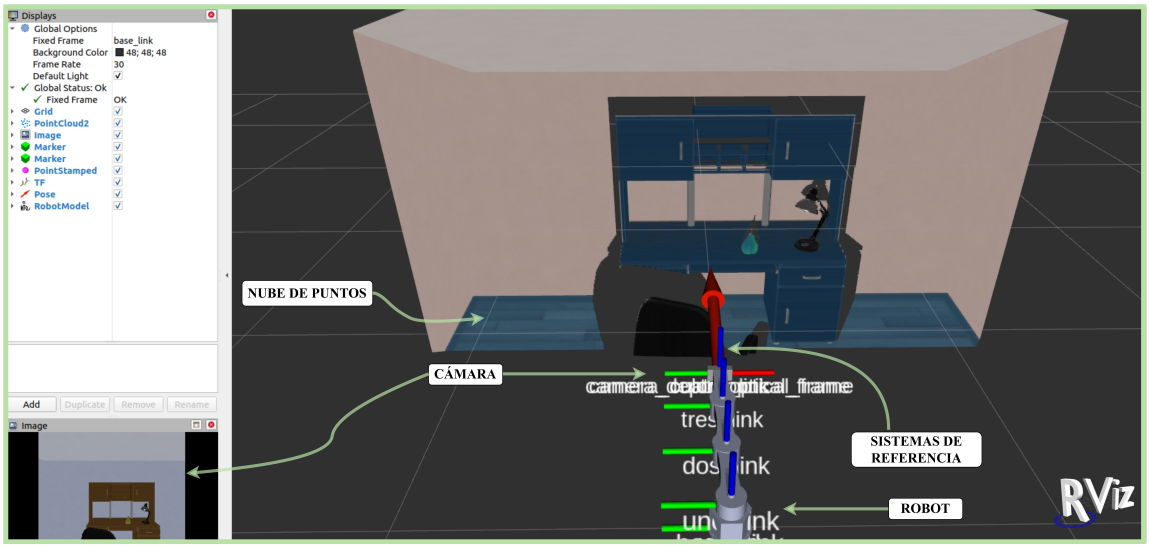


Figura 6.2: Diagrama general.

En la Figura 6.3 se presenta la vista superior de cada escenario, ya que esta permite una visualización más clara de las trayectorias de los adultos mayores (AM). Asimismo, se observa la posición de los puntos de luz en cada escenario.

Por otro lado, en la Tabla 6.1 se muestran las posiciones iniciales de los modelos AM para cada trayectoria y escenario para agregar variabilidad a las pruebas y apreciar distintos comportamientos del sistema ante condiciones iniciales diversas.

[1]Consultar Tipos de componentes de luz.

(a) Sala. (b) Estudio. (c) Comedor.

Figura 6.3: Vista superior de los escenarios bajo diferentes condiciones de iluminación.

Tabla 6.1: Posición de los modelos AM en cada escenario.

		Posición$_{AM1}$ **[m]**		**Posición**$_{AM2}$ **[m]**	
Trayectoria	**Escenario**	**x**	**y**	**x**	**y**
Linea recta	Sala	2.70	2.82	2.70	2.54
	Estudio	2.70	2.74	2.70	2.50
	Comedor	2.00	3.00	2.00	3.00
Diagonal	Sala	4.29	1.70	1.35	-1.28
	Estudio	4.30	1.40	4.50	1.50
	Comedor	0.67	-1.00	3.50	2.00
Zeta	Sala	1.20	-1.36	1.57	-2.34
	Estudio	3.73	1.85	2.80	1.50
	Comedor	4.22	1.64	4.50	2.00
Circulo	Sala	2.70	0.00	1.45	0.00
	Estudio	1.57	0.00	0.82	0.00
	Comedor	2.11	0.00	2.00	0.00

Tras la integración y verificación del sistema, se ejecutaron pruebas en tres escenarios (sala, estudio y comedor). En cada escenario se implementaron:

- Dos perfiles de velocidad para compensar la diferencia visual entre simulación y realidad.
    - V1 ($0.3\,\frac{\mathrm{m}}{\mathrm{s}}$) permitiendo un análisis visual detallado del movimiento del adulto mayor y el comportamiento cinemático del robot.
    - V2 ($1\,\frac{\mathrm{m}}{\mathrm{s}}$) como la velocidad real de un adulto mayor que en el entorno simulado presenta movimientos prácticamente instantáneos. Este umbral de velocidad fue seleccionado con base en lo reportado por Cerda [59], quien señala que velocidades por debajo de este límite representan un posible indicador de problemas en adultos mayores aparentemente sanos.
- Cuatro trayectorias: en línea recta (R), en diagonal (D), en zeta (Z) y en círculo (O), con el propósito de abarcar toda el área en cada escenario.

Con base en las 48 pruebas efectuadas y la información registrada en las tablas del Apéndice L.12, se obtuvieron los promedios correspondientes de las métricas RMSE, MAE, MAX_E y FDE para cada modelo AM.

En primer lugar, las Figuras 6.4 y 6.5 muestran el promedio del RMSE. El error en la posición estimada para AM1 osciló entre 0.21 m y 0.22 m, mientras que, AM2 se encontraba entre 0.19 m y 0.35 m. En tanto, la posición predicha registró un pico de 0.71 m para AM1 y AM2 a $1.0\,\frac{\mathrm{m}}{\mathrm{s}}$.

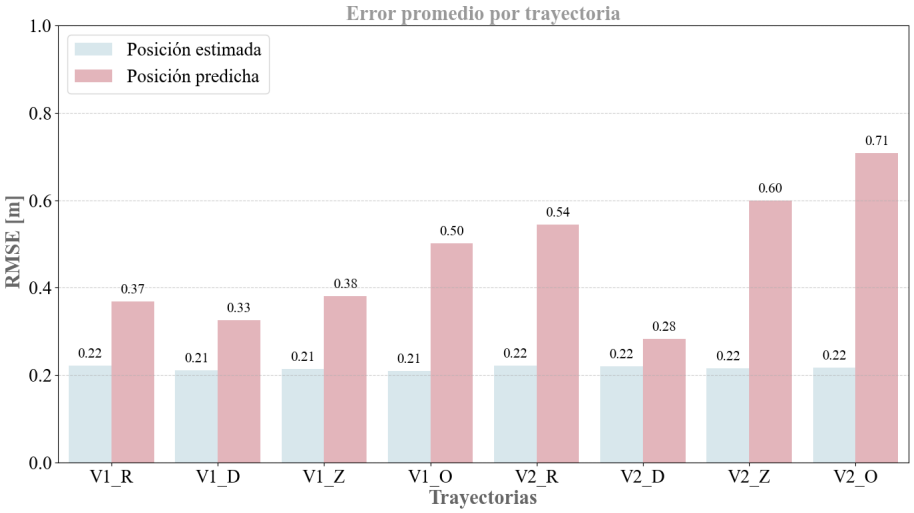


Figura 6.4: Gráfica de RMSE para el modelo AM1.

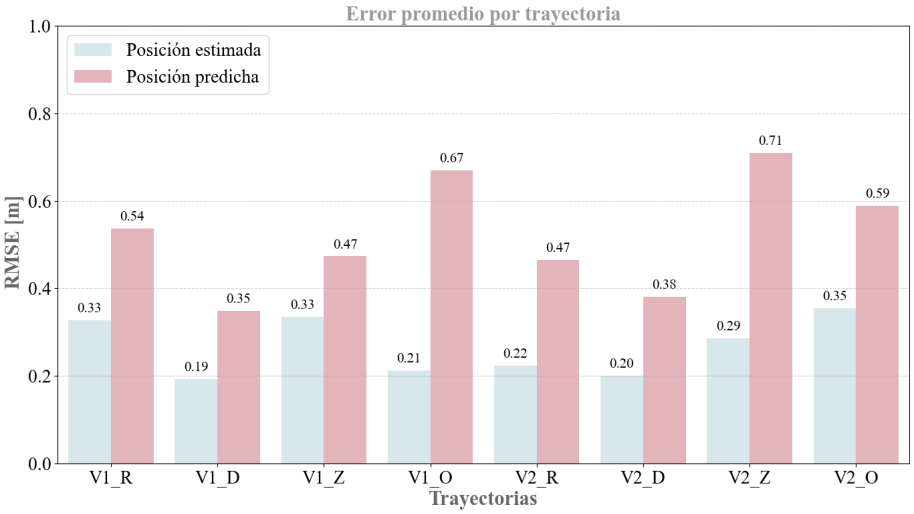


Figura 6.5: Gráfica de RMSE para el modelo AM2.

Posteriormente, las Figuras 6.6 y 6.7 muestran la métrica MAE. El error en la posición estimada de AM1 se situó entre 0.21 m y 0.24 m en contraste con AM2 cuyo rango fue de 0.18 m a 0.28 m. Adicionalmente, el error de predicción más alto fue de 0.56 m para AM1 y de 0.53 m para AM2 con una velocidad de 1.0 $\frac{\text{m}}{\text{s}}$.

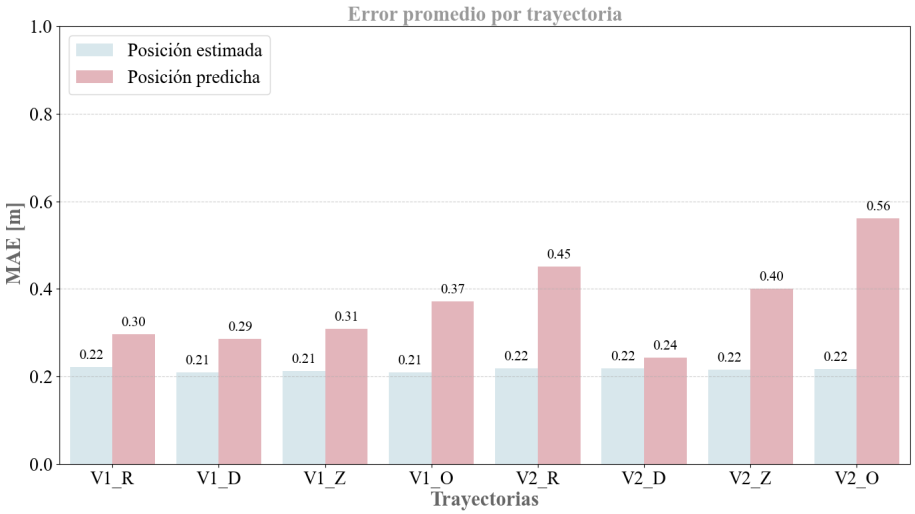


Figura 6.6: Gráfica de MAE para el modelo AM1.

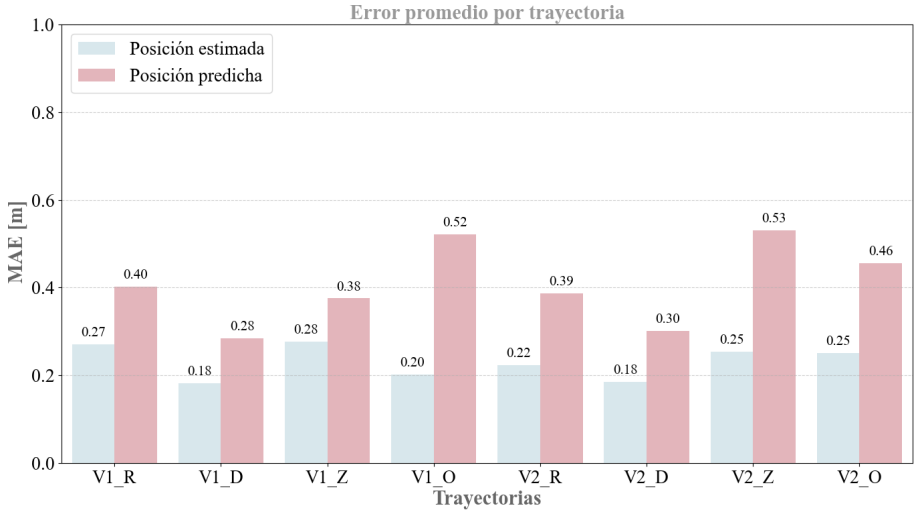


Figura 6.7: Gráfica de MAE para el modelo AM2.

De manera adicional, las Figuras 6.8 y 6.9 presentan el promedio de la métrica MAX_E. El error en la posición estimada para AM1 se mantuvo en un rango de 0.22 m a 0.28 m y AM2 en 0.26 m a 0.73 m. Mientras que, el error de predicción fue de 1.71 m para el AM1 y de 1.67 m para AM2.

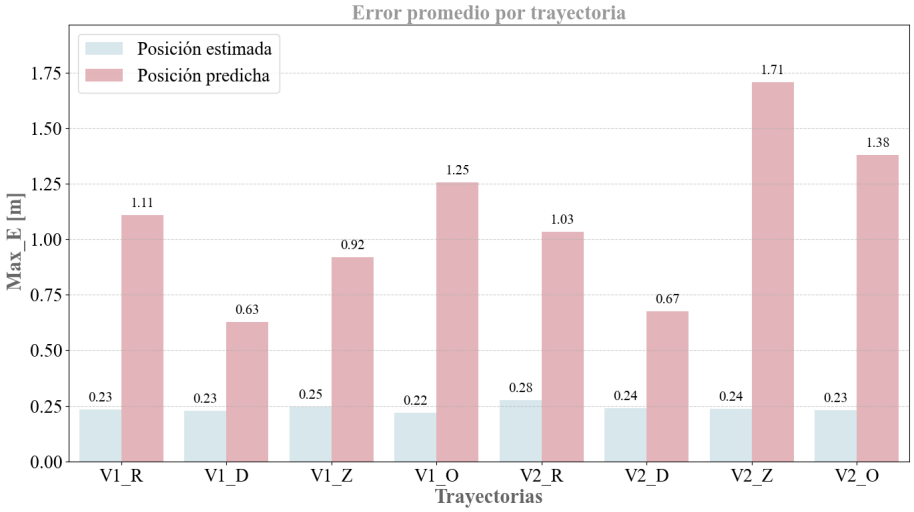


Figura 6.8: Gráfica de MAX_E para el modelo AM1.

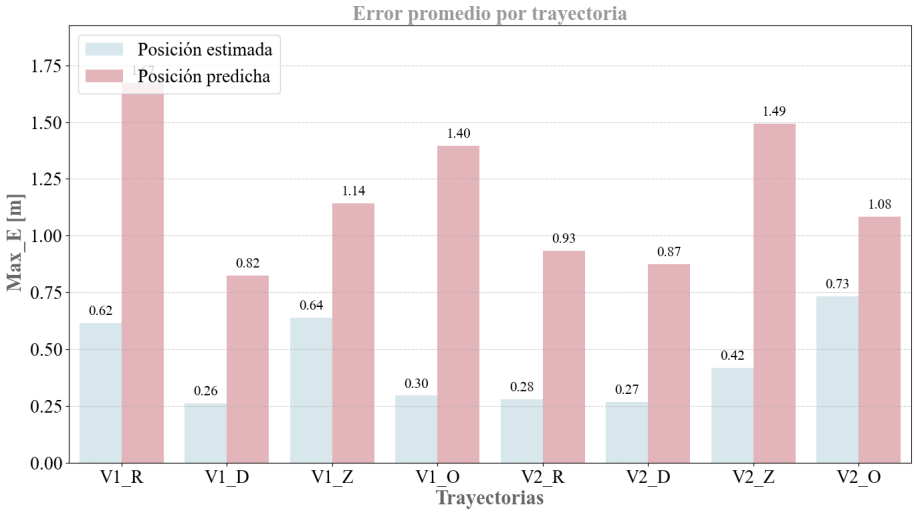


Figura 6.9: Gráfica de MAX_E para el modelo AM2.

Finalmente, las Figuras 6.10 y 6.11 detallan el FDE. El error de la posición estimada para AM1 se mantuvo entre 0.48 m a 1.37 m y en AM2 de 0.44 m a 1.49 m. En contraste, el error promedio más alto en la predicción fue de 1.37 m para AM1 y de 1.51 m para AM2.

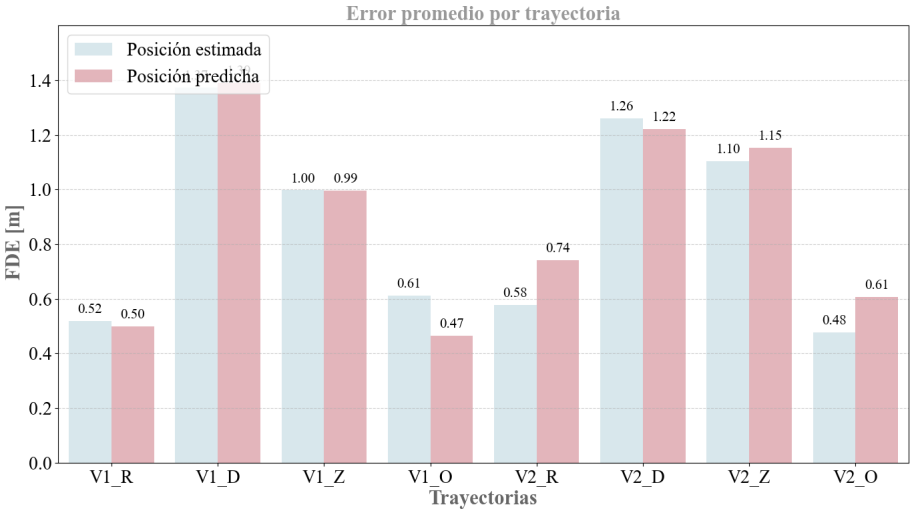


Figura 6.10: Gráfica de FDE para el modelo AM1.

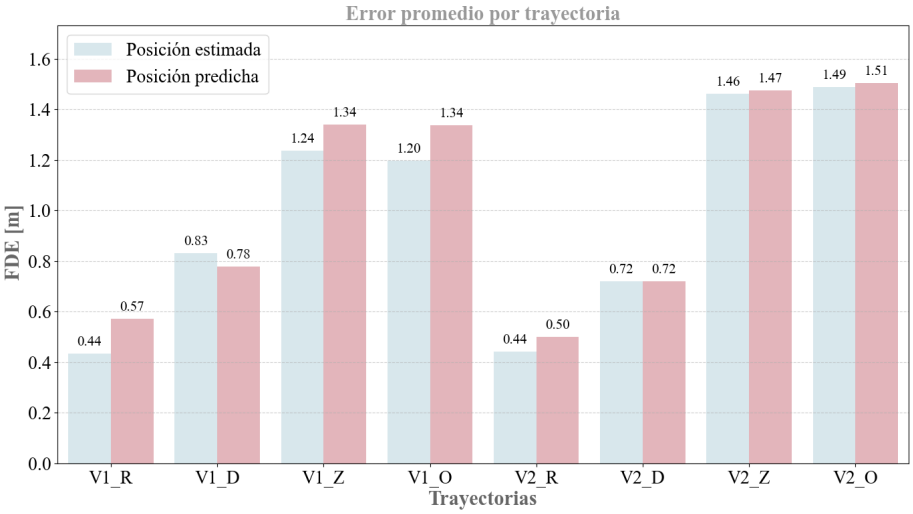


Figura 6.11: Gráfica de FDE para el modelo AM2.

Derivado de lo anterior, se presentan las trayectorias reales, estimadas y predichas para los modelos AM1 y AM2, bajo diferentes escenarios, velocidades e iluminación.

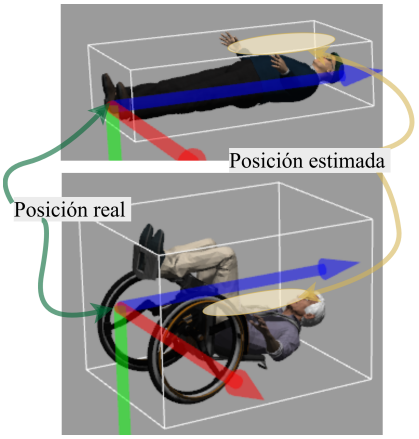
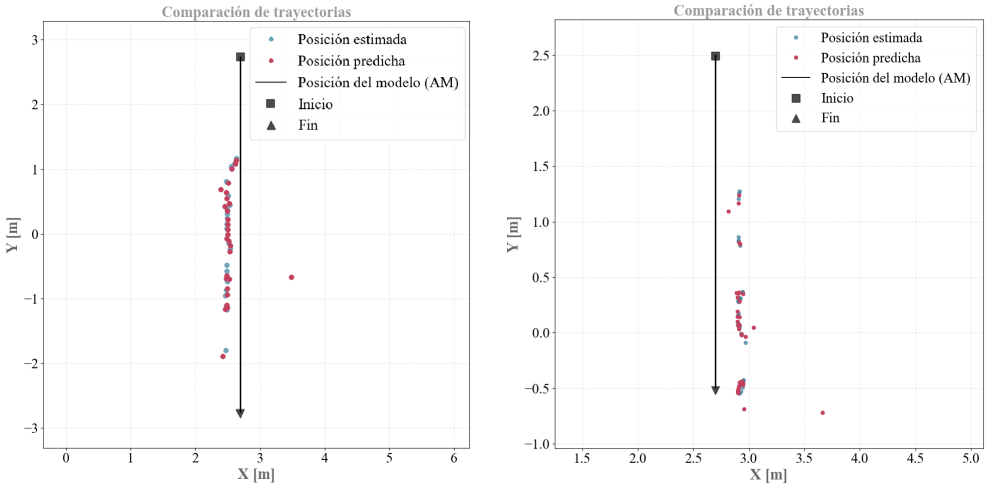


Figura 6.12: Puntos de referencia.

Cabe destacar que, siempre se presentará un offset dada la naturaleza del sensor de profundidad y el punto de referencia que define la posición real de los modelos AM.

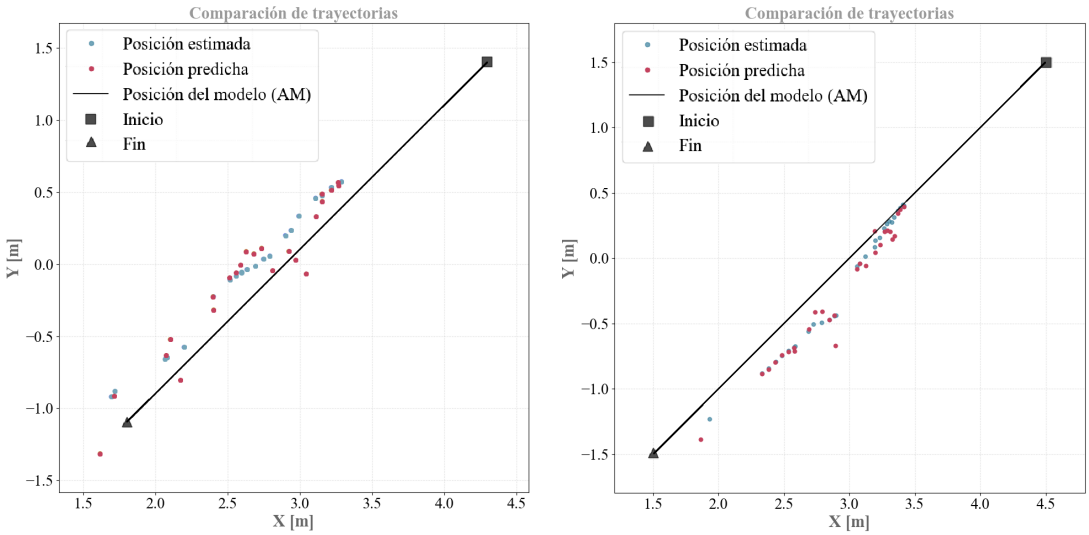
La Figura 6.13 muestra la trayectoria en línea recta para ambos modelos. Se observó una alta coincidencia entre las trayectorias estimada y predicha, con una desviación mínima y continuidad en la secuencia de posiciones.



(a) AM1. (b) AM2.

Figura 6.13: Comparación de la estimación de la posición. Escenario: Estudio. Velocidad: 0.3 $\frac{\text{m}}{\text{s}}$.

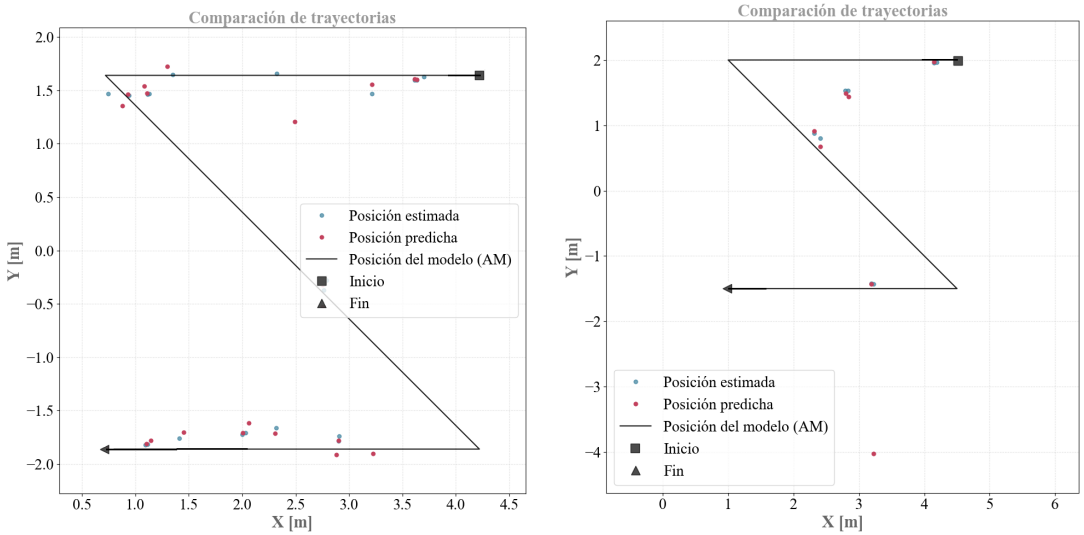
Para la trayectoria diagonal, mostrada en la Figura 6.14, se registró una desviación moderada de las posiciones estimadas y predichas respecto de la posición del modelo. Aunque se mantuvo la continuidad general de la trayectoria, la dispersión de los puntos fue notablemente mayor que en la trayectoria rectilínea.



(a) AM1. (b) AM2.

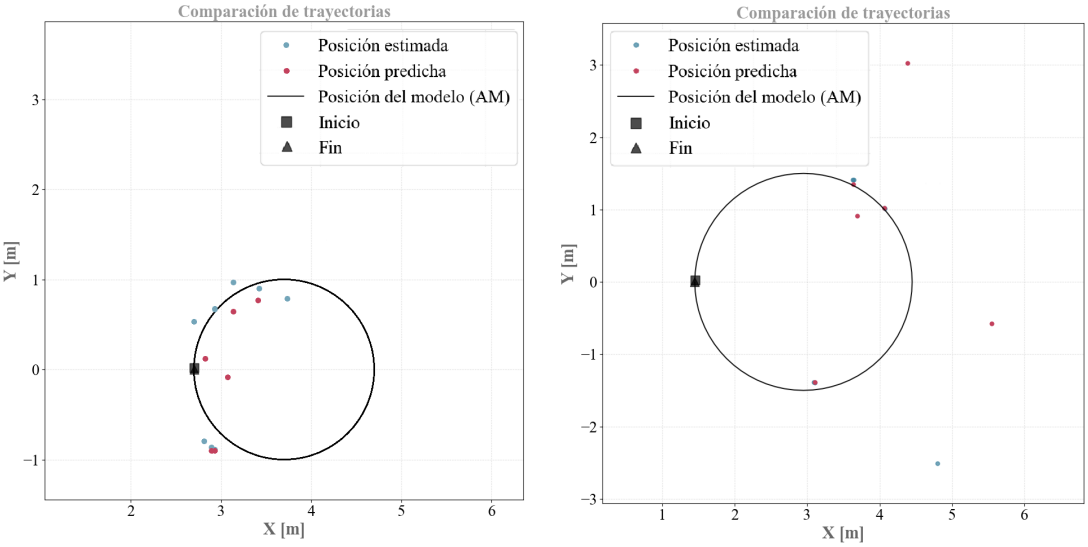
Figura 6.14: Comparación de la estimación de la posición. Escenario: Estudio. Velocidad: 0.3 $\frac{\text{m}}{\text{s}}$.

Por otro lado, los resultados de la trayectoria zeta a una velocidad de 1.0 $\frac{\text{m}}{\text{s}}$ se presentan en la Figura 6.15. Se observó que en los segmentos rectos, el cálculo de la posición fue consistente mientras que en los segmentos diagonales, se registró una mayor desviación en la posición estimada y predicha con respecto a la trayectoria real.



(a) AM1. (b) AM2.

Figura 6.15: Comparación de la estimación de la posición. Escenario: Comedor. Velocidad: 1.0 $\frac{\text{m}}{\text{s}}$.



(a) AM1. (b) AM2.

Figura 6.16: Comparación de la estimación de la posición. Escenario: Sala. Velocidad: 1.0 $\frac{\text{m}}{\text{s}}$.

Finalmente, la Figura 6.16 corresponde a la trayectoria circular a $1.0\,\frac{\mathrm{m}}{\mathrm{s}}$. En este escenario (sala) se detectaron intervalos significativos sin datos de estimación y predicción, lo que resultó en una trayectoria fragmentada para ambos modelos.

La comparación de las trayectorias mostró coherencia con los datos reportados en las Figuras 6.4 a la 6.11. Las trayectorias en línea recta y diagonal (Figuras 6.13 y 6.14), que registraron errores mínimos de RMSE y MAE, demostraron la mayor proximidad visual con la trayectoria real. Por el contrario, la trayectoria circular (Figura 6.16), presentó valores más elevados en métricas como RMSE y FDE, exhibió una mayor desviación en la trayectoria reconstruida. Finalmente, la trayectoria zeta (Figura 6.15) mostró un desempeño intermedio.

Después de analizar el comportamiento del sistema durante el cálculo de la posición estimada y predicha, se muestra el desempeño del robot en las pruebas experimentales.

En la Figura 6.17 es posible observar:

(a) La vista superior del escenario (Estudio) y la posición [0,0] del robot.

(b) Las metas aproximadas (+) limitadas por el espacio de trabajo del robot (semicírculo azul).

(c) Las metas reales (×) que son las posiciones predichas para los modelos AM1 y AM2.

(d) El rango de visión de la cámara que define el comportamiento del robot. Cuando los modelos AM se encuentren cerca (semicírculo amarillo) el robot se retraerá, mientras que, si están lejos (semicírculo rosa) se extenderá.

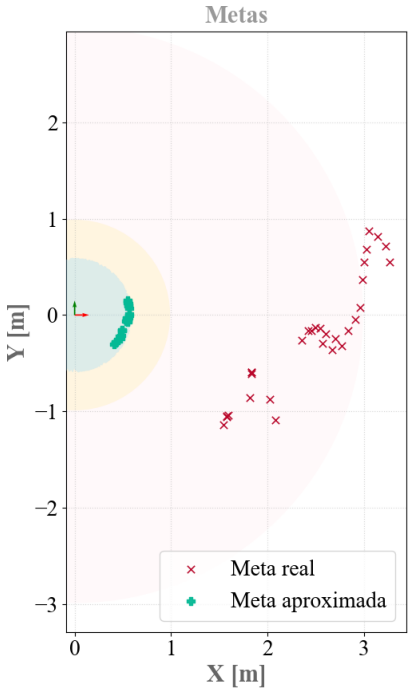


Figura 6.17: Vista superior de la trayectoria diagonal del modelo AM1.

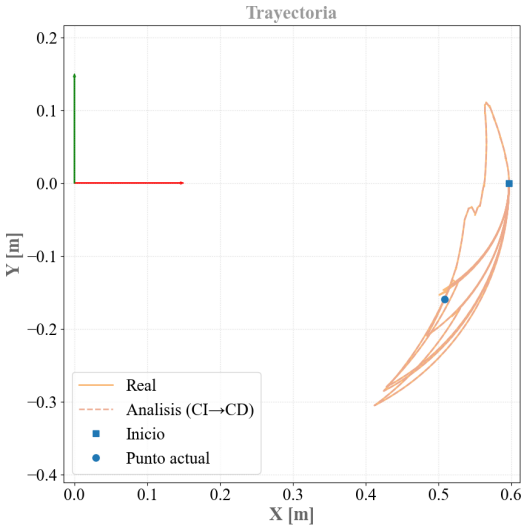


Figura 6.18: Vista a detalle del desplazamiento del robot durante el seguimiento de AM1.

En la Figura 6.18 se muestra el desplazamiento del robot durante la tarea de seguimiento correspondiente al modelo AM1 con una trayectoria diagonal.

Se observó que el análisis cinemático fue consistente y que el movimiento se ejecutó de manera continua y controlada a lo largo de todo el recorrido del adulto mayor en el escenario.

Por otro lado, en la Figura 6.19 se presentan las posiciones de las variables articulares. Los valores deseados (línea continua) y los reales (línea punteada), indicaron que el

sistema de control reprodujo adecuadamente las referencias impuestas.

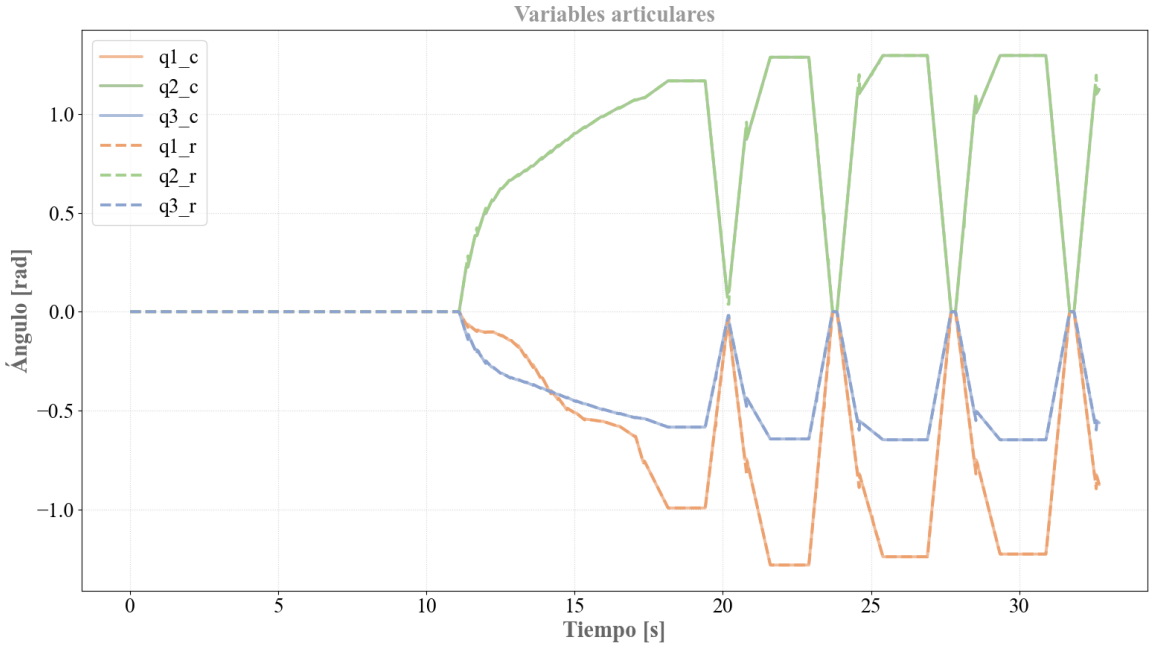


Figura 6.19: Posición angular registrada de las variables articulares.

Finalmente, en la Figura 6.20 se analizaron los errores de posición y orientación. El error de orientación presentó incrementos puntuales porque la orientación deseada se mantuvo fija en home. Cuando el robot rota para ejecutar el movimiento, la orientación real se aleja temporalmente de esa referencia, generando picos en el error hasta que el robot vuelve a alinearse. Mientras que, el error de posición se mantuvo acotado a un rango menor a 0.020 m, lo que constituye un resultado favorable y evidencia una alta capacidad de seguimiento de la trayectoria por parte del robot.

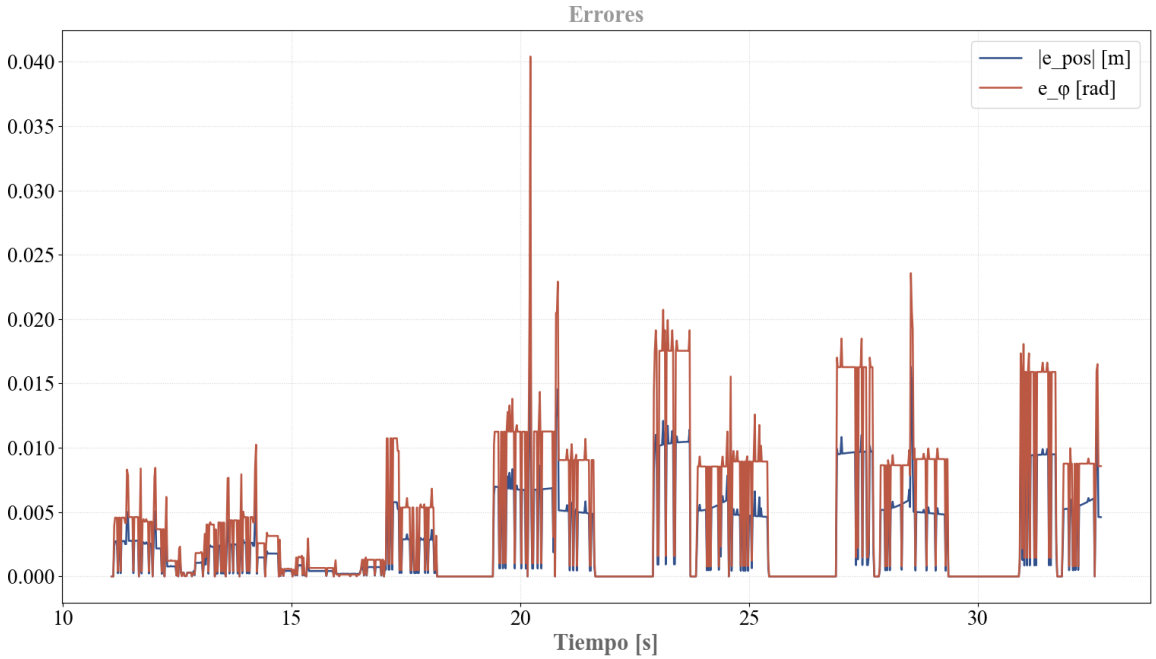


Figura 6.20: Errores de posición y de orientación del robot.

Capítulo 7
Discusión y conclusiones

7.1. Discusión

Tras analizar los datos obtenidos en cada una de las pruebas, se observó que, independientemente de la trayectoria y del escenario, el desempeño en la tarea de seguimiento depende de manera crítica de la detección correcta y continua del modelo especializado. Al tratarse de una cadena de procesamiento, cualquier pérdida en la detección se traduce en un incremento acumulado de los errores de posición en profundidad y de predicción. Este comportamiento evidencia que el modelo especializado constituye el elemento central para el éxito del seguimiento.

El análisis de los resultados permite concluir que múltiples factores inciden de manera significativa en la calidad del seguimiento. En particular, los errores aumentan cuando la trayectoria del adulto mayor inicia deliberadamente fuera del rango de visión de la cámara o cuando el modelo abandona el escenario mientras continúa su movimiento, lo que agrava el error de detección y, por consiguiente, el error de seguimiento. De forma consistente con lo reportado en la sección 5.1.1, las características geométricas y visuales del modelo influyen directamente en su detección, mientras que la iluminación resulta determinante para resaltar cada rasgo. Las pruebas se diseñaron para abarcar variaciones que pueden presentarse en un entorno real: dimensiones distintas de las habitaciones, presencia de elementos que generan oclusiones parciales o totales del adulto mayor y variaciones en la altura del robot en

cada escenario.

Además, la posición del adulto mayor a lo largo de su trayectoria incide directamente en el cálculo de la profundidad. En particular, se encontraron tramos extensos sin detecciones, probablemente asociados a las condiciones específicas de cada escenario y a la rapidez con la que se desplaza. En conjunto, estos resultados evidencian la sensibilidad del modelo especializado a las condiciones del entorno y a la configuración de movimiento del adulto mayor.

7.2. Conclusiones

El presente estudio se centró en el desarrollo y la validación de un entorno de simulación en ROS-Gazebo para modelar el movimiento de un robot planar 3R. El propósito principal fue integrar en este sistema robótico capacidades de visión por computadora para realizar el seguimiento continuo de adultos mayores en movimiento, utilizando una red neuronal. Se concluye que la colaboración entre el robot planar 3R y una red neuronal especializada mediante la técnica de transferencia de aprendizaje resultó sumamente efectiva para la tarea de seguimiento. La clave para la especialización adecuada de la CNN radicó en la selección de un modelo preentrenado y en la integración de un conjunto de datos propio con imágenes relevantes de adultos mayores. La principal contribución de este trabajo radica en la demostración de la capacidad del sistema propuesto para abordar la creciente necesidad de vigilancia y asistencia para este grupo demográfico, que a menudo pasa largos periodos de tiempo en soledad y aislamiento dentro de sus hogares. El valor de esta tesis se extiende a la integración completa del sistema de visión por computadora, desde el diseño y el análisis cinemático del robot manipulador hasta la integración de un con-

junto de datos propio y la especialización de la red neuronal. Este enfoque aplicado propone una alternativa tecnológica que puede mejorar significativamente la calidad de vida y la seguridad de las personas mayores, manteniendo una vigilancia constante y discreta. Las implicaciones prácticas de estos resultados son prometedoras, pues los hallazgos obtenidos en el simulador sientan las bases para la implementación real del sistema. La eficacia demostrada en la tarea de seguimiento de adultos mayores sugiere que el sistema es viable para su despliegue en entornos domésticos reales, lo que permite abrir el camino para el desarrollo de funcionalidades avanzadas.

A pesar de la robustez del sistema, es importante reconocer que existen ciertas limitaciones inherentes al entorno de simulación y al hardware virtual. Específicamente, se identificó que el tamaño del conjunto de datos recopilado para el entrenamiento resultó apenas suficiente; un conjunto de datos más amplio y con una mayor variedad de posturas y contextos habría optimizado la especialización de la red. Adicionalmente, se enfrentaron desafíos como la baja disponibilidad de modelos 3D detallados de adultos mayores, así como el manejo de las condiciones de simulación, como la limitación intencional del rango de visión de la cámara a 3 metros para reflejar condiciones domésticas promedio. Como resultado de esta investigación, se recomiendan futuras líneas de estudio enfocadas en la implementación física y real de este sistema. También se sugiere la incorporación de funcionalidades complementarias, tales como la detección de posiciones de riesgo (caídas), la adición de sensores complementarios a la detección visual y, potencialmente, la especialización del modelo para detectar a otros grupos vulnerables.

Referencias bibliográficas

[1] H. Herfandi, O. S. Sitanggang, M. R. A. Nasution, H. Nguyen, and Y. M. Jang, "Real-time patient indoor health monitoring and location tracking with optical camera communications on the internet of medical things," *Applied Sciences*, vol. 14, no. 3, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/3/1153

[2] A. Stanescu, A. Nita, M. Moisescu, and I. Sacala, "From industrial robotics towards intelligent robotic systems," in *2008 4th International IEEE Conference Intelligent Systems*, vol. 1, 2008, pp. 6–73–6–77.

[3] V. Kumar, Q. Wang, W. Minghua, S. Rizwan, S. M. Shaikh, and X. Liu, "Computer vision based object grasping 6DoF robotic arm using picamera," in *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, 2018, pp. 111–115.

[4] D. Kim, H. Bian, C. Chang, L. Dong, and J. Margrett, "In-home monitoring technology for aging in place: Scoping review," *Interactive Journal of Medical Research*, vol. 11, no. 2, p. e39005, 2022.

[5] L. N. Nguyen, P. Susarla, A. Mukherjee, M. L. Cañellas, C. Álvarez Casado, X. Wu, O. Silvén, D. B. Jayagopi, and M. B. López, "Non-contact multimodal indoor human monitoring systems: A survey," *Information Fusion*, vol. 110, p. 102457, oct 2024.

[6] D. Fuentes, L. Correia, N. Costa, A. Reis, J. Ribeiro, C. Rabadão, J. Barroso, and A. Pereira, "Indoorcare: Low-cost elderly activity monitoring system through image processing," *Sensors*, vol. 21, no. 18, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/18/6051

[7] H. M. Ahmed and B. Abdulrazak, "Monitoring indoor activity of daily living using thermal imaging: A case study," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 9, 2021. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2021.0120902

[8] T. T. Zin, Y. Htet, Y. Akagi, H. Tamura, K. Kondo, S. Araki, and E. Chosa, "Real-time action recognition system for elderly people using

stereo depth camera," *Sensors*, vol. 21, no. 17, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/17/5895

[9] A. G. Karkar, S. Al-Maadeed, J. Kunhoth, and A. Bouridane, "Camnav: a computer-vision indoor navigation system," *The Journal of Supercomputing*, vol. 77, no. 7, pp. 7737–7756, 2021. [Online]. Available: https://doi.org/10.1007/s11227-020-03568-5

[10] M.-Y. Chen, "Establishing a cybersecurity home monitoring system for the elderly," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 7, pp. 4838–4845, 2022.

[11] Y. Chen, Y. Zhang, B. Xiao, and H. Li, "A framework for the elderly first aid system by integrating vision-based fall detection and bim-based indoor rescue routing," *Advanced Engineering Informatics*, vol. 54, p. 101766, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034622002245

[12] C. Lee, E. T.-H. Chu, M. Sie, L. Lin, M. Hong, and C. Huang, "Application of indoor positioning systems in nursing homes: Enhancing resident safety and staff efficiency," *Sensors*, vol. 24, no. 18, p. 6099, 2024.

[13] V. Mane, R. Mahajan, H. A. Durge, P. Ghosh, K. Kadam, I. Mahajan, and P. Muneshwar, "Alert system for non-responsive state of an elderly person," *Journal of Computational Systems and Applications*, 2025.

[14] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: a survey," *ACM Computing Surveys (CSUR)*, vol. 38, no. 4, pp. 1–45, Dec 2006, source: DBLP. [Online]. Available: https://doi.org/10.1145/1177352.1177355

[15] H.-B. Kim and K.-B. Sim, "A particular object tracking in an environment of multiple moving objects," in , 11 2010, pp. 1053–1056.

[16] W. J. Kim and I.-S. Kweon, "Moving object detection and tracking from moving camera," in *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2011, pp. 758–759.

[17] A. Keivani, J.-R. Tapamo, and F. Ghayoor, "Motion-based moving object detection and tracking using automatic k-means," in *2017 IEEE AFRICON*, 2017, pp. 32–37.

[18] I. I. Lychkov, A. N. Alfimtsev, and S. A. Sakulin, "Tracking of moving objects with regeneration of object feature points," in *2018 Global Smart Industry Conference (GloSIC)*, 2018, pp. 1–6.

[19] W. Bing and L. Xiang, "A simulation research on 3D visual servoing robot tracking and grasping a moving object," in *15th International Conference on Mechatronics and Machine Vision in Practice*, 2008, pp. 362–367.

[20] N. K. Verma, A. Mustafa, and A. Salour, "Stereo-vision based object grasping using robotic manipulator," in *2016 11th International Conference on Industrial and Information Systems (ICIIS)*, 2016, pp. 95–100.

[21] A. S. Berger, *Embedded Systems Design: An Introduction to Processes, Tools, and Techniques*. CMP Books, 2002.

[22] I. Sommerville, *Ingeniería de Software*, 9th ed. Atlacomulco 500-5o. piso Col. Industrial Atoto 53519, Naucalpan de Juárez, Estado de México: Pearson Educación, 2011.

[23] A. d. l. Escalera, *Visión por Computador. Fundamentos y Métodos*. Pearson Educacion, 2001.

[24] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. PEARSON; 4th edition, 2018.

[25] J. F. V. Serrano, A. B. M. Díaz, Ángel Sánchez Calle, and J. L. E. Sánchez-Marín, *Visión por Computador*. Madrid: DYKINSON, S.L., 2003.

[26] D. G. Bailey, *Design for Embedded Image Processing on FPGAs*, 1st ed. John Wiley & Sons (Asia) Pte Ltd, 2011.

[27] R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[28] Ultralytics. (2025) Guía de calibración de cámaras para visión por ordenador en 2025. [Online]. Available: https://www.ultralytics.com/es/blog/a-guide-to-camera-calibration-for-computer-vision-in-2025

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[30] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*, 1st ed. 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA: O'Reilly Media, Inc., March 2017.

[31] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 2018, pp. 1–6.

[32] S. Rajasekaran and G. Pai, *Neural Networks, Fuzzy Logic and Genetic Algorithm: synthesis and applications (with CD)*. PHI Learning, 2003.

[33] I. Sarker, "Deep learning: una descripción general completa de técnicas, taxonomía, aplicaciones y direcciones de investigación," *Computación SN. Lic.*, vol. 2, p. 420, 2021. [Online]. Available: https://doi.org/10.1007/s42979-021-00815-1

[34] N. K. Chauhan and K. Singh, "A review on conventional machine learning vs deep learning," in *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, 2018, pp. 347–352.

[35] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, Dec. 2022. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2021.3084827

[36] G. Lu, Q. Hao, K. Kong, J. Yan, H. Li, and X. Li, "Deep convolutional neural networks with transfer learning for neonatal pain expression recognition," in *2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2018, pp. 251–256.

[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[38] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, 2023. [Online]. Available: https://www.mdpi.com/2504-4990/5/4/83

[39] G. Boesch. (2024, oct) Detección de objetos: la guía definitiva para 2025. VISO.AI. [Online]. Available: https://viso.ai/deep-learning/object-detection/

[40] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[41] Ultralytics, "Performance metrics deep dive: How validation metrics from YOLO26 help improve model performance," https://docs.ultralytics.com/es/guides/yolo-performance-metrics/#how-can-validation-metrics-from-yolo26-help-improve-model-performance, 2024.

[42] A. Ramdan, A. Heryana, A. Arisal, R. B. S. Kusumo, and H. F. Pardede, "Transfer learning and fine-tuning for deep learning-based tea diseases detection on small datasets," in *2020 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*, 2020, pp. 206–211.

[43] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1019–1034, 2015.

[44] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *Advances in Neural Information Processing Systems (NIPS)*, vol. 27, 11 2014.

[45] W. Sun, D. Li, W. Jia, P. Li, C. Zhao, and X. Chen, "Small moving object tracking in dynamic video," in *2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, 2015, pp. 239–242.

[46] L. Hao and Z. Shu-kui, "Moving object tracking algorithm based on improved gaussian mixture model," in *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, 2019, pp. 271–275.

[47] S. Kanaki, K. Fujishita, M. Hashimoto, R. Murabayashi, K. Inui, and K. Takahashi, "Cooperative moving-object tracking with multiple mobile sensor nodes — size and posture estimation of moving objects using in-vehicle multilayer laser scanner," in *2016 IEEE International Conference on Industrial Technology (ICIT)*, 2016, pp. 59–64.

[48] J. J. Cabrera García, "Seguimiento visual en mecanismos robóticos con q-learning," Master's thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey, 12 2006. [Online]. Available: http://hdl.handle.net/11285/567650

[49] J. Pomares, R. Gutiérrez, J. M. López, F. Torres, and P. Gil, "Seguimiento de trayectorias 3d mediante control visual basado en imagen," *Grupo de Automática, Robótica y Visión Artificial*, 2004.

[50] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, Oct. 1996.

[51] F. Chaumette and S. Hutchinson, "Visual servo control. part I: Basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, dec 2006.

[52] ——, "Visual servo control. part II: Advanced approaches," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 109–118, mar 2007.

[53] P. I. Corke, *Robotics, Vision and Control*, 2nd ed. Springer, 2017.

[54] A. Barrientos, L. F. Peñín, C. Balaguer, and R. Aracil, *Fundamentos de robótica*, 2nd ed. Edificio Valrealty, 1.a planta Basauri, 17 28023 Aravaca (Madrid): McGraw-Hill/Interamericana de de España, S.A.U., 2007, ISBN: 978-84-481-5636-7.

[55] J. J. Craig, *Robótica, 3era edición.* Pearson Educación, 2006.

[56] ——, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2005.

[57] ROS. (2021) ¿Por qué ROS? [Online]. Available: https://www.ros.org/blog/why-ros/

[58] O. Robotics, "Acerca de gazebo." [Online]. Available: https://gazebosim.org/about

[59] A. Lorena Cerda, "Manejo del trastorno de marcha del adulto mayor," *Revista Médica Clínica Las Condes*, vol. 25, no. 2, pp. 265–275, 2014.

Apéndices

A.1. Software y hardware

En la Tabla 1 se muestran las especificaciones técnicas del sistema operativo y en la Tabla 2 las correspondientes al hardware utilizado durante el desarrollo del proyecto, con el propósito de garantizar un entorno de ejecución estable y eficiente.

Tabla 1: Especificaciones técnicas del sistema operativo.

Parámetro	**Valor**
Sistema Operativo	Ubuntu 20.04 LTS
Arquitectura	x86_64
Procesador	12th Gen Intel(R) Core(TM) i7-12650H
Núcleos	16
RAM instalada	15 GiB
Kernel	5.15.0-105-generic
Shell	/bin/bash
Entorno gráfico	GNOME
Espacio en disco	806 GiB libres de 938 GiB

Tabla 2: Especificaciones técnicas del hardware.

Parámetro	**Valor**
Procesador	12th Gen Intel(R) Core(TM) i7-12650H 2.30 GHz
Memoria RAM	16.0 GB
Tarjeta Gráfica	NVIDIA GeForce GTX 4070
Almacenamiento	1 TB SSD

B.2. Aumento de datos

La Tabla 3 resume las principales bibliotecas y módulos utilizados durante el proceso de aumento de datos. Posteriormente, se presenta el fragmento de código donse se implementan dichas herramientas.

Tabla 3: Bibliotecas para el aumento de datos.

Biblioteca	**Modulo**
Python 3.12.7	json collections → defaultdict base64
Cv2 4.1.0.84	-
Numpy 2.0.2	-
Tensorflow 2.18.0	ImageDataGenerator
Matplotlib 3.9.2	-

```
# Ruido gaussiano
def add_gaussian_noise(image, mean=0, sigma=25):
    gauss = np.random.normal(mean, sigma, image.shape).astype('uint8')
    noisy_image = cv2.add(image, gauss)
    return noisy_image

# Cambio de tono y saturación
def change_hue_saturation(image, hue_shift=0, saturation_scale=1):
    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv_image)
    h = (h.astype(int) + hue_shift) % 180
    h = np.clip(h, 0, 179).astype(np.uint8)
```

```
    s = np.clip(s * saturation_scale, 0, 255).astype(np.uint8)
    modified_hsv = cv2.merge((h, s, v))
    modified_image = cv2.cvtColor(modified_hsv, cv2.COLOR_HSV2BGR)
    return modified_image

# Ajustar el brillo
def adjust_brightness(image, factor=1.2):
    hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    h, s, v = cv2.split(hsv_image)
    v = np.clip(v * factor, 0, 255).astype(np.uint8)
    modified_hsv = cv2.merge((h, s, v))
    modified_image = cv2.cvtColor(modified_hsv, cv2.COLOR_HSV2BGR)
    return modified_image

# Desenfocar la imagen
def blur_image(image, kernel_size=(5, 5)):
    blurred_image = cv2.GaussianBlur(image, kernel_size, 0)
    return blurred_image
```

Código 1: TransformacionesColores.ipynb

```
    transformations = [
        ("rotate", 30),  # Rotación de 30 grados
        ("scale", 0.5, 0.5),  # Escalado a la mitad
        ("translate", 50, 50),  # Traslación de 50 px [X,Y]
        ("flip_horizontal",),  # Reflejo horizontal
        ("flip_vertical",),  # Reflejo vertical
    ]
```

Código 2: TransformacionesGeometricas.ipynb

C.3. División del conjunto de datos

La herramienta empleada para la generación de los metadatos fue LabelMe, la cual dispone del comando *labelme2yolo* que permite convertir los archivos .json a .yaml requerido por YOLO.

Formato de los archivos .json

<class_id><x_center><y_center><width><height>

Formato del archivo .yaml

dataset/
- *images/*
  - *train/*
  - *val/*
- *labels/*
  - *train/*
  - *val/*

El YAML actúa como un archivo de configuración del conjunto de datos, permitiendo que YOLO reconozca dónde están los datos y cómo están organizadas las clases.

Sin embargo, al intentar ejecutar este comando sobre el conjunto de datos, se presentaron diversos problemas, los cuales se detallan a continuación.

Verificación de campos

El problema surgió porque LabelMe admite números enteros en el campo *group_id*, mientras que el comando *labelme2yolo* requiere una cadena de texto.

Normalización y ajuste de coordenadas

Además, en el formato YOLO, las coordenadas de las anotaciones deben estar estrictamente en el rango [0, 1]. Cualquier valor superior a 1 se consideraba corrupto.

Para corregir de forma automática los polígonos que excedían los límites de las imágenes tras el aumento de datos se realizó lo siguiente:

- Detección de las coordenadas de los polígonos que se encontraban fuera de los límites de la imagen.
- Recorte de los polígonos para limitarlos a las dimensiones de las imágenes.

Este procedimiento se centró exclusivamente en modificar y corregir las coordenadas de las anotaciones en los archivos .json.

Mapeo de los ID

Por otro lado, *labelme2yolo* no convertía textos complejos a números, lo que generó identificadores incorrectos. Por ello, se renombraron los campos de las etiquetas para definir el número 0 para hombres y 1 para mujeres.

Del formato JSON a YOLO

Finalmente, se aplicó *labelme2yolo* para obtener el archivo de configuración YAML. Sin embargo, el comando original carecía de la funcionalidad para dividir el conjunto datos en subconjuntos de entrenamiento, validación y prueba. Este aspecto se ajustó y

mejorado durante el segundo entrenamiento, lo que permitió una separación adecuada de los datos y facilitó el proceso de evaluación del rendimiento del modelo.

D.4. Modelo CAD

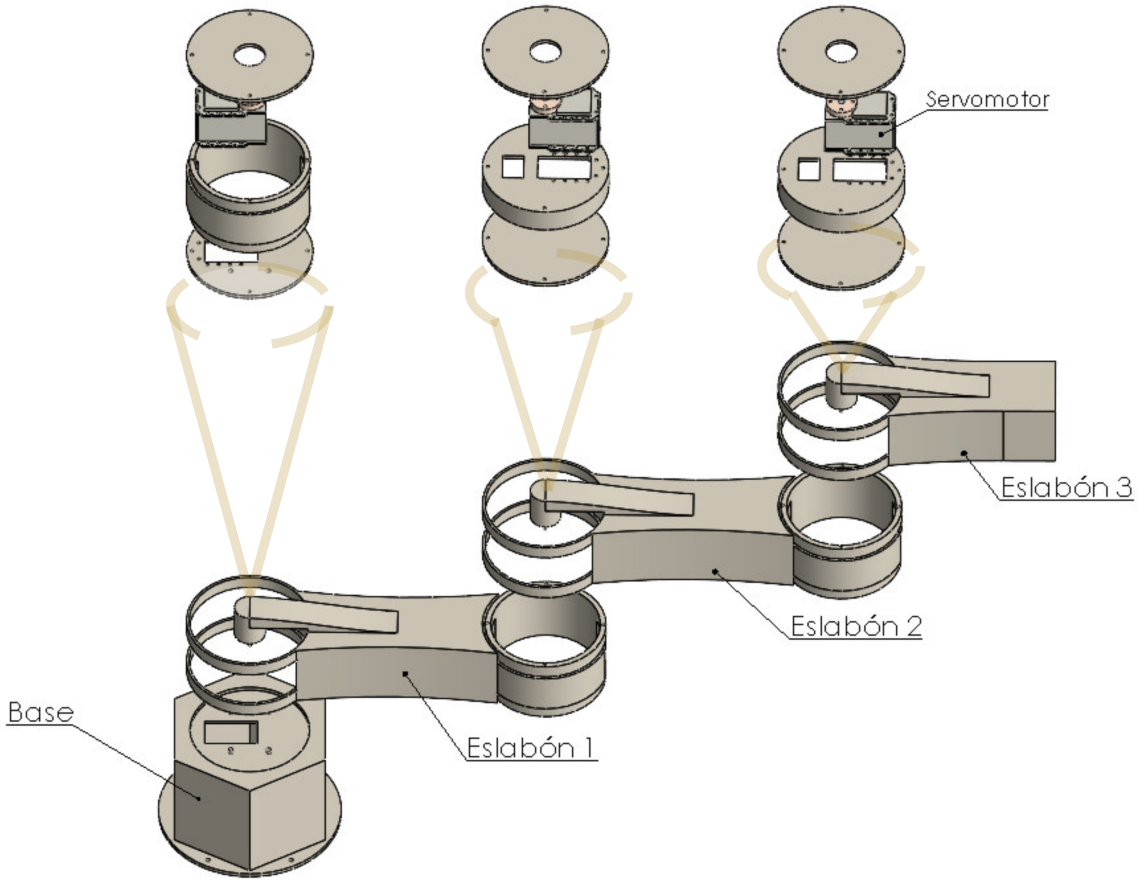


Figura 1: Modelo explosionado.

E.5. Generación del archivo URDF

E.5.1. Verificación del ensamble

1. Corrección de piezas para evitar que dos cuerpos sólidos choquen físicamente.

2. Detección de inferencias (ver la Figura 2).

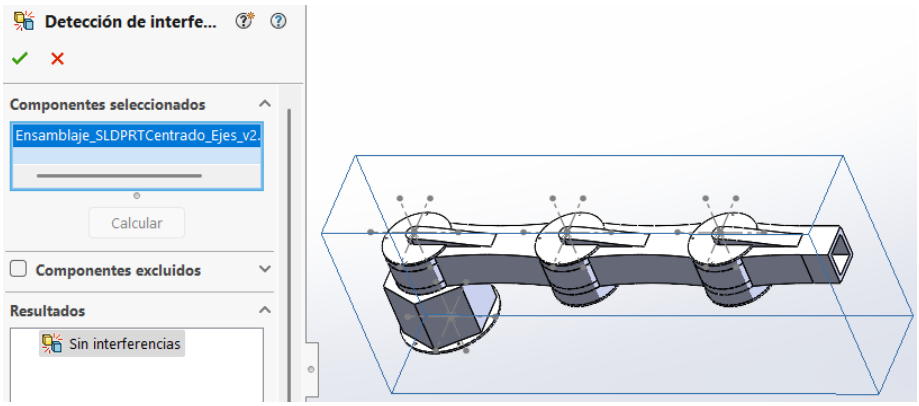


Figura 2: Detección de inferencias.

E.5.2. Paso 1

Dado que el ensamble general se generó a partir de múltiples subensambles, era necesario que:

- Los subensambles se convierten en piezas únicas (.SLDASM ->.SLDPRT).
- Cada .SLDPRT fue colocado en el origen del documento y orientado de acuerdo con la interpretación de SolidWorks®.

- A partir de archivos .SLDPRT generar el ensamble general.

E.5.3. Paso 2

Debido a que los sistemas de coordenadas entre SolidWorks® y CoppeliaSim® no coinciden (ver Figura 3), fue necesario generar un sistema de coordenadas en cada articulación del robot para definir los marcos de referencia de los ejes de rotación, de modo que CoppeliaSim® los interprete correctamente.

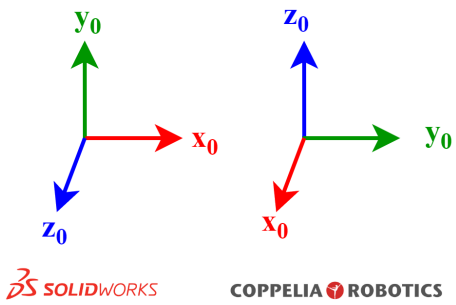


Figura 3: Sistema de coordenas entre softwares.

Paso 2.1

Con la herramienta de SW2URDF se requiere que en cada articulación se defina un eje para indicar su dirección de rotación o traslación.

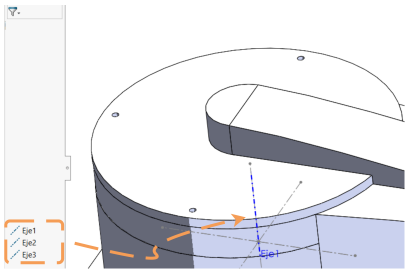


Figura 4: Colocación de ejes en el origen de cada articulación.

E.5.4. Paso 3

Previo al inicio de la exportación del ensamble fue necesario:

1. Activar el complemento SW2URDF (Opciones ->Complementos ->SW2URDF) o descargarlo desde la página: `github.com/ros/solidworks_urdf_exporter/releases`.

   a) Ejecutar el archivo .exe y guardarlo en el mismo directorio de SolidWorks®.

   b) Verificar que el complemento SW2URDF se encuentre activo.

2. Consultar la información en: `wiki.ros.org/urdf/XML/joint`.

3. Ir a Herramientas ->Tools ->Export as URDF

En la Figura 5 se muestra cómo se definen los links (piezas físicas) y los joints (movimiento entre links).

① Se define la base del robot.

② Se define el primer eslabón.

③ Se define el segundo eslabón.

④ Se define el tercer eslabón.

⑤ *Preview and Export* permite corregir la configuración de los joints y los links, ver Figura 6.

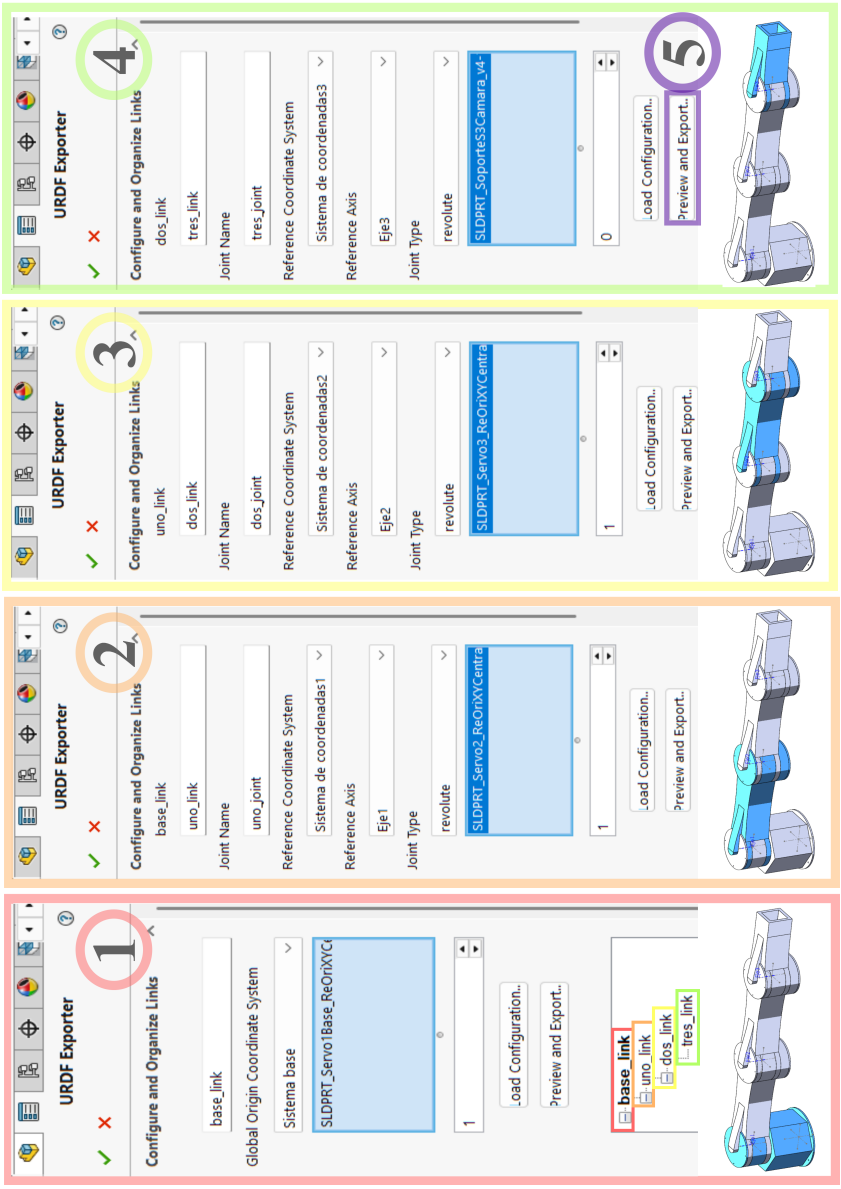
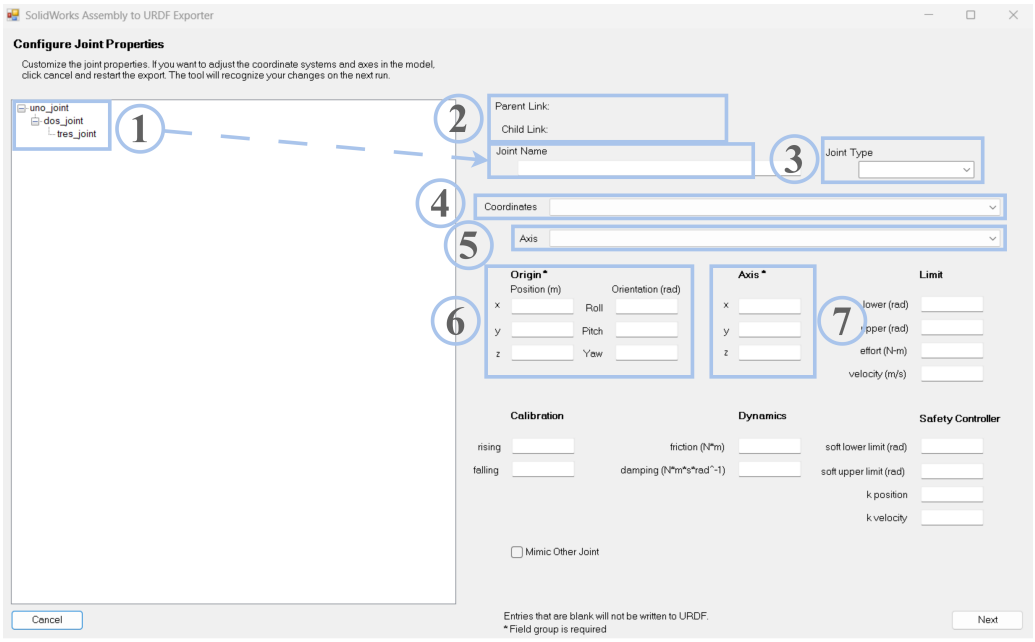
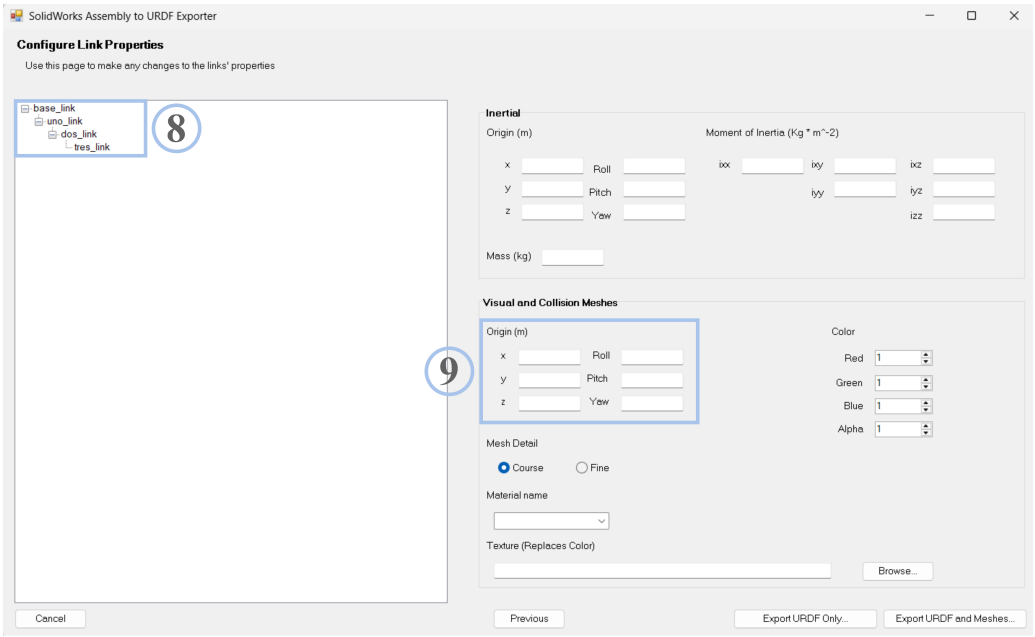


Figura 5: Definición de jerarquías.



(a) Joints



(b) Links

Figura 6: Ventana de configuración.

① Lista de joints.

② Relación entre links y joints (ver la sección Paso 3).

③ Tipo de joints.

④ Sistema de coordenadas para cada joint (ver Paso 2).

⑤ Elección del eje de acción de cada joint.

⑥ Posición de cada joint.

⑦ Orientación del eje de acción.

⑧ Lista de links.

⑨ Posición de cada link.

En la Figura 7 se muestra el resultado correspondiente a los pasos 1, 2 y 3.

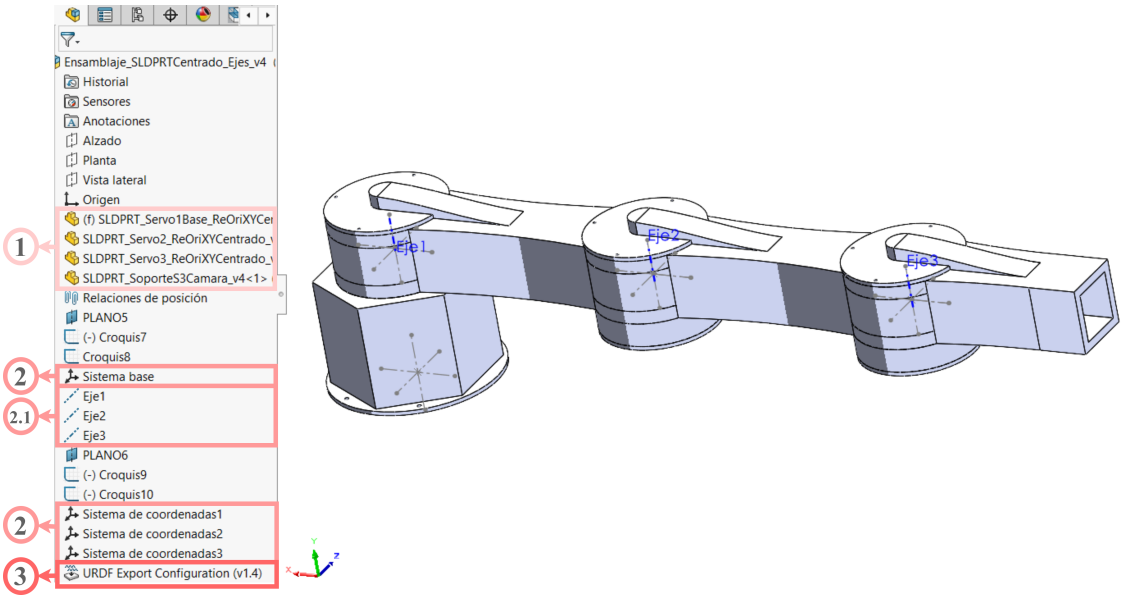


Figura 7: Configuración del modelo CAD previo a la obtención del archivo .urdf.

El complemento SW2URDF genera un conjunto de archivos y carpetas necesarios, entre los que se incluye un archivo URDF que describe la estructura del robot, sus articulaciones, enlaces, sistemas de coordenadas, propiedades físicas básicas (como masa e inercia) y la jerarquía del modelo. También se exportan los archivos de geometría, normalmente en formato STL, que contienen la representación tridimensional de cada componente para su visualización y simulación.

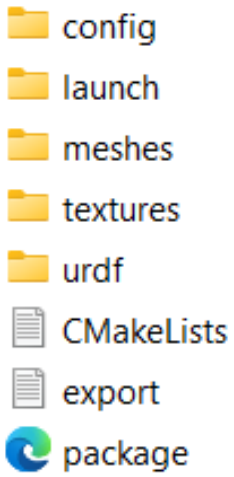


Figura 8: Archivos generados con el complemento SW2URDF.

F.6. Verificación del URDF

Para comprobar que el archivo URDF se generó correctamente, se utilizó CoppeliaSim® (Modules ->Importers ->URDF importer) para visualizar el ensamblaje del robot.

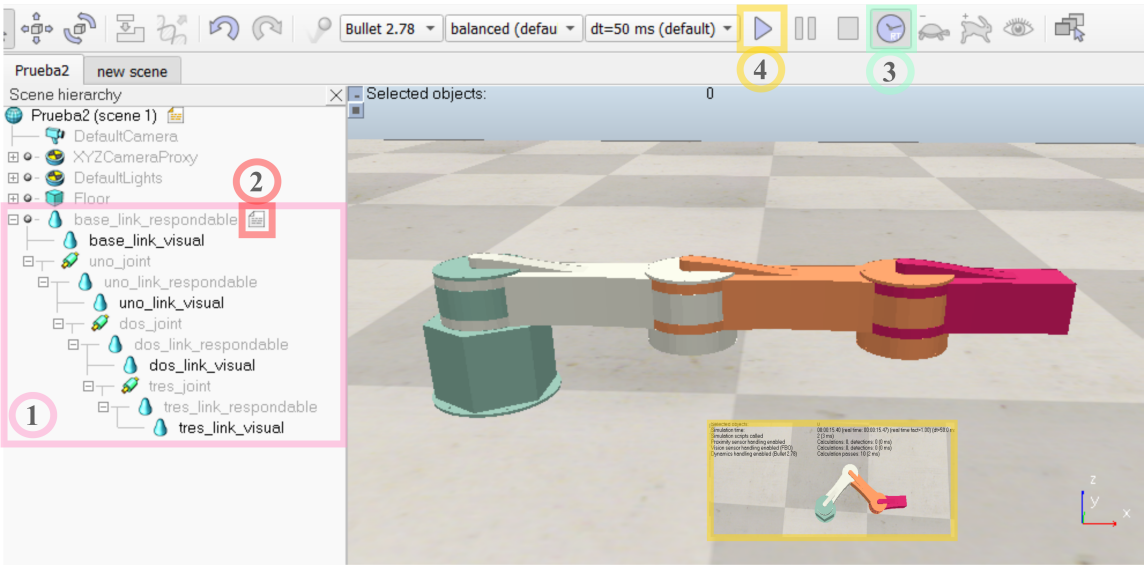


Figura 9: Vista de jerarquía de escenas y la vista 3D correspondiente.

① La jerarquía se crea conectando links mediante joints, colocando dentro de cada link un visual para la apariencia y un respondable para la física. El árbol de objetos refleja esta estructura, en la que el movimiento del link padre afecta al hijo.

② El programa que controla el comportamiento del robot dentro de la simulación (Add ->Associated child script ->Non threaded ->Lua).

```
function sysCall_init()
j1=sim.getObject('./uno_joint')
j2=sim.getObject('./dos_joint')
j3=sim.getObject('./tres_joint')
end
function sysCall_actuation()
sim.setJointTargetPosition(j1, math.rad(50))
```

```
8  sim.setJointTargetPosition(j2, -math.rad(90))
9  sim.setJointTargetPosition(j3, math.rad(45))
10 end
```

Código 3: Prueba1

③ Cuando está activado, la simulación se ejecuta al ritmo del tiempo real, sincronizando la física y los movimientos del robot.

④ Inicia o reanuda la simulación en el escenario.

G.7. Correción del URDF

En la Figura 8 se observa que el complemento SW2URDF generó una carpeta llamada *urdf*, la cual contenía dos archivos: .urdf y .csv. El archivo .csv incluía toda la configuración del modelo CAD definida en la sección E.5.4.

Sin embargo, en el caso de las inercias, estas presentaban valores desproporcionados, por lo que fue necesario modificar manualmente el archivo .urdf. Para corregir la inestabilidad del robot en Gazebo, consecuencia de la exportación directa del modelo CAD, ver Figura 10.

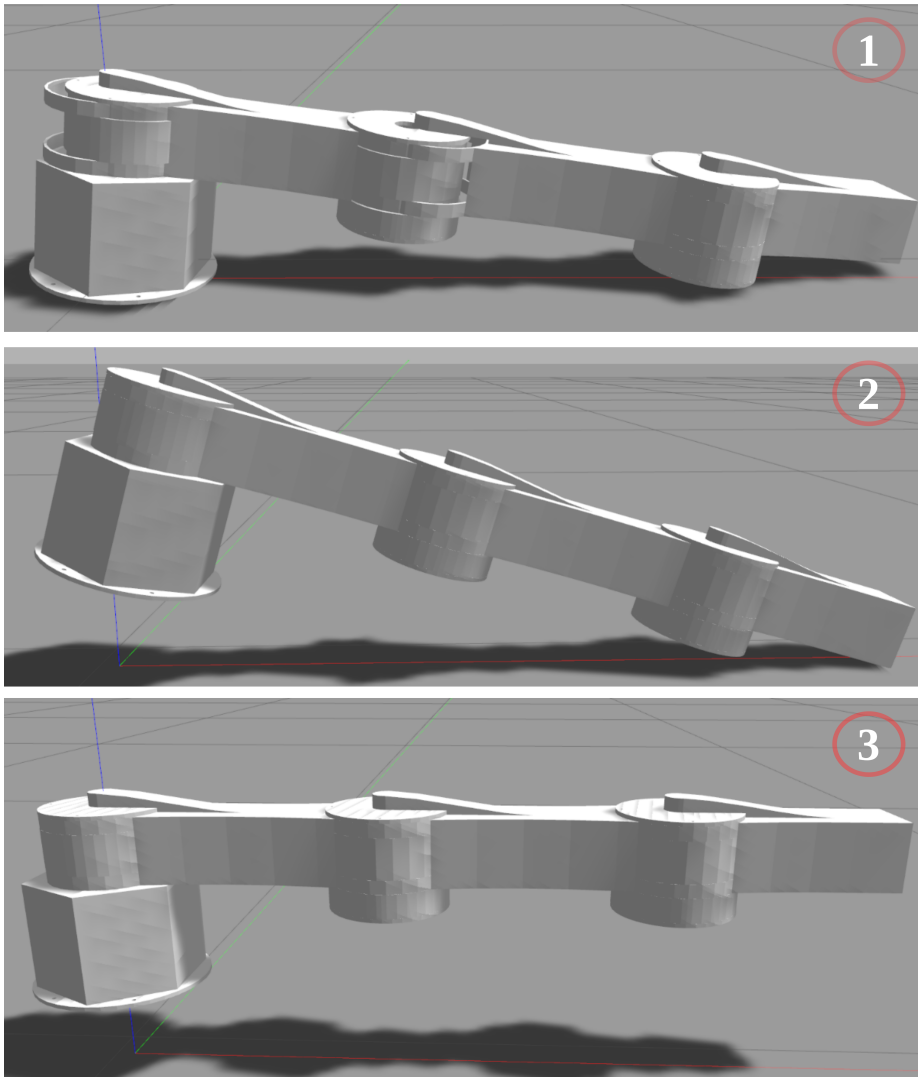


Figura 10: Integración del robot en el simulador Gazebo.

Además, dentro de los problemas detectados se incluyeron: geometrías de <collision>excesivamente complejas (STL), tensores de inercia mal condicionados (con valores cercanos a cero) y joints carentes de límites y amortiguación (<limit>, <dynamics damping>).

Configuración del Link

```
<link
name="base_link">
<inertial>
<origin
xyz="-0.0019449 -2.6858E-16
    0.068385"
rpy="0 0 0" />
<mass
value="0.33901" />
<inertia
ixx="0.00051816"
ixy="-3.3974E-12"
ixz="-3.8478E-07"
iyy="0.0005122"
iyz="1.8168E-17"
izz="0.00071528" />
</inertial>
<visual>
<origin
xyz="0 0 0"
rpy="0 0 0" />
<geometry>
<mesh
filename="...base_link.STL" />
</geometry>
<material
name="">
<color
rgba="0.79216 0.81961 0.93333 1" />
</material>
</visual>
<collision>
<origin
xyz="0 0 0"
rpy="0 0 0" />
<geometry>
<mesh
filename="...base_link.STL" />
</geometry>
</collision>
</link>
```

Código 4: Robot.urdf

Configuración del Joint

```
  <joint
  name="uno_joint"
  type="revolute">
  <origin
  xyz=" 0 0 0.099"
  rpy="0 0 0" />
  <parent
  link="base_link" />
  <child
  link="uno_link" />
  <axis
  xyz="0 0 1" />
  <limit
  lower="0"
  upper="0"
  effort="0"
  velocity="0" />
  </joint>
```

Código 5: Robot.urdf

La solución consistió en redefinir manualmente la física del robot en el URDF. Utilizando en el campo <collision>geometrías simples (<box>) que aproximarán la forma del eslabón, asignando masas realistas e inercias calculadas de forma manual para dichas primitivas y añadiendo dinámicas al joint para garantizar la estabilidad numérica del simulador mediante el uso de los valores presentados en la Tabla 4.1.

Configuración del Link

```
<link name="base_link">
<inertial>
<origin xyz="-0.0019449 0 0.068385" rpy="0 0 0"/>
<mass value="0.33901"/>
<inertia
ixx="0.0009999664967" ixy="0" ixz="0"
iyy="0.0009999664967" iyz="0"
izz="0.0009548781667"/>
</inertial>
<visual>
<origin xyz="0 0 0.068" rpy="0 0 0"/>
<geometry>
<mesh filename="package://robot/meshes/base_link.STL"/>
</geometry>
<material name="">
<color rgba="0.79216 0.81961 0.93333 1"/>
</material>
</visual>
<collision>
<origin xyz="0 0 0.068" rpy="0 0 0"/>
<geometry>
<box size="0.13 0.13 0.136"/>
</geometry>
</collision>
</link>

```

Código 6: Robot.urdf

Configuración del Joint

```
<joint name="uno_joint" type="revolute">
<origin xyz="0 0 0.167" rpy="0 0 0"/>
<parent link="base_link"/>
<child  link="uno_link"/>
<axis xyz="0 0 1"/>
<limit lower="-1.5708" upper="1.5708" effort="1.5" velocity="6.1784655521"/>
<dynamics damping="0.6" friction="0.0"/>
</joint>

```

Código 7: Robot.urdf

H.8. Sensor

En el archivo .urdf del robot, se añadieron los parámetros ópticos y de profundidad del sensor RGB-D, así como los plugins necesarios para su correcta integración y comunicación con ROS.

```
<gazebo reference="cuatro_link">
<sensor name="color_cam" type="camera">
<update_rate>30</update_rate>
<camera>
<horizontal_fov>1.20428</horizontal_fov>
<image>
<width>640</width><height>480</height><format>R8G8B8</format>
<noise><type>gaussian</type><mean>0.0</mean><stddev>0.01</stddev></
   noise>
</image>
<clip><near>0.3</near><far>3.0</far></clip>
</camera>
<plugin name="gazebo_ros_camera_color" filename="libgazebo_ros_camera.
   so">


<sensor name="depth_cam" type="depth">
<update_rate>30</update_rate>
<camera>
<horizontal_fov>1.5184</horizontal_fov>
<image>
```

```
<width>640</width><height>480</height><format>R8G8B8</format>
<noise><type>gaussian</type><mean>0.0</mean><stddev>0.02</stddev></
   noise>
</image>
<clip><near>0.3</near><far>3.0</far></clip>
</camera>
<plugin name="gazebo_ros_depth_camera_depth" filename="
   libgazebo_ros_depth_camera.so">
</plugin>
</sensor>
</gazebo>
```

Código 8: Fragmento de Robot.urdf.

I.9. Arquitectura del sistema

El diagrama de comunicación de la Figura 11 ofrece una representación estructurada de los nodos involucrados, así como de los canales de intercambio de información que posibilitan la coordinación del sistema propuesto en la Figura 5.1.

J.10. Integración de modelos personalizados

Para la incorporación de modelos personalizados en Gazebo (ver Figura 12), se exploraron los repositorios de modelos 3D[1] que ofrecen estos recursos en diversos for-

[1]Repositorio de modelos 3D: Sketchfab, Free3D.

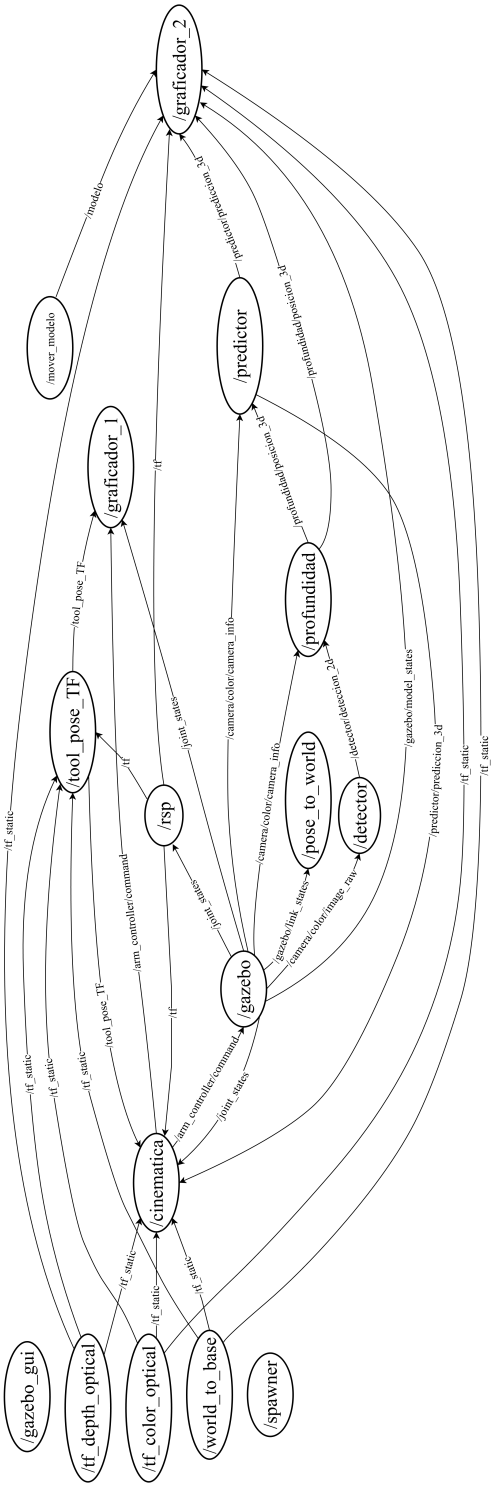


Figura 11: Diagrama de comunicación ROS.

matos. Posteriormente, se utilizó el software Blender (versión 2.82) para convertir dichos modelos al formato .dae (Collada).

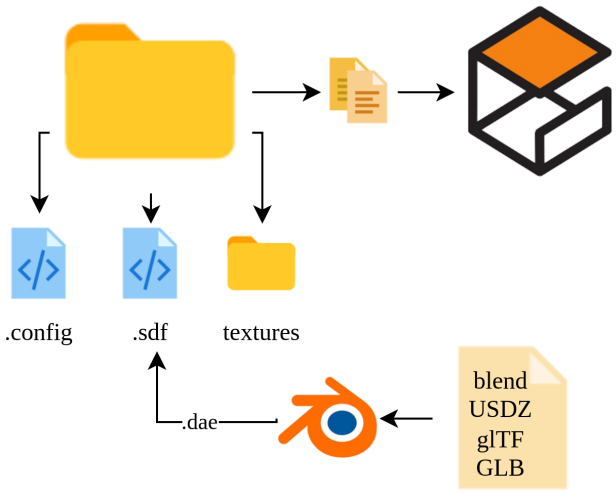


Figura 12: Flujo de trabajo en Gazebo.

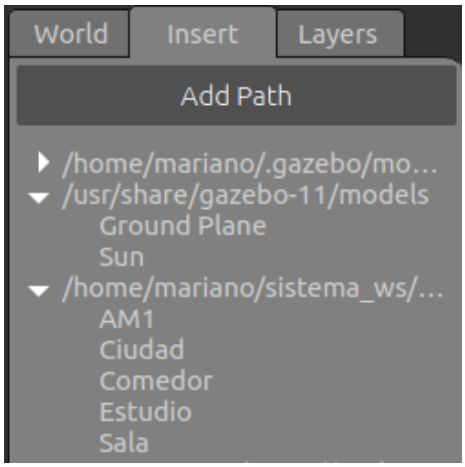


Figura 13: Resultado de la integración de modelos 3D en Gazebo.

Una vez que se obtuvo este archivo en el formato adecuado, se creó una carpeta estructurada con la información requerida para integrar cada uno de los modelos 3D al repositorio de modelos de Gazebo, a través del siguiente comando:

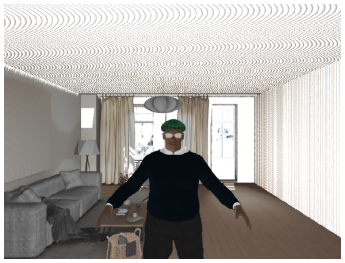
```
gedit ~/.bashrc
export GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:~/sistema_ws/
```

K.11. Escenarios

K.11.1. Configuración general

Tabla 4: Configuración general de los escenarios y modelos.

	Posición [m]			Orientación [rad]			Dimensión [m]			Luces
Escenario/Modelo	**x**	**y**	**z**	**r**	**p**	**y**	**Largo**	**Ancho**	**Alto**	**No.**
Sala	4.50	0.00	0.00	0.00	0.00	-1.57	5.67	9.50	3.23	4.00
Estudio	1.97	0.00	0.02	0.00	0.00	-1.57	4.99	3.92	2.60	3.00
Comedor	2.73	0.00	1.32	0.00	0.00	-1.57	5.70	6.67	2.94	5.00

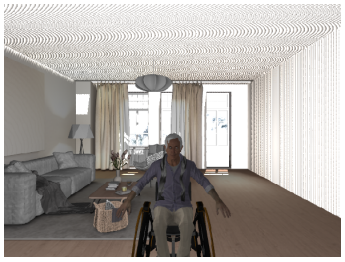


(a) Sala.

(b) Estudio.

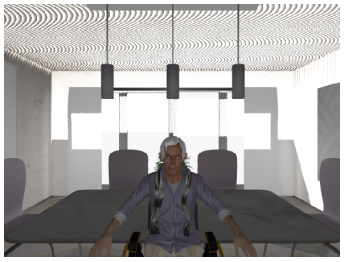
(c) Comedor.

Figura 14: Modelo AM1 con la configuración general de los escenarios.



(a) Sala.

(b) Estudio.



(c) Comedor.

Figura 15: Modelo AM2 con la configuración general de los escenarios.

K.11.2. Condiciones iniciales

Tabla 5: Configuración de los escenarios para el modelo AM1.

		Posición [m]			**Orientación [rad]**		
Escenario	**Modelo**	**x**	**y**	**z**	**r**	**p**	**y**
Sala	AM1	2.70	0.00	0.06	0.00	0.00	-1.57
	Robot	0.00	0.00	1.70	0.00	0.00	0.00
Estudio	AM1	2.70	0.00	0.02	0.00	0.00	-1.57
	Robot	0.00	0.00	1.70	0.00	0.00	0.00
Comedor	AM1	2.00	0.00	0.05	0.00	0.00	-1.57
	Robot	0.00	0.00	1.70	0.00	0.00	0.00

Tabla 6: Configuración de los escenarios para el modelo AM2.

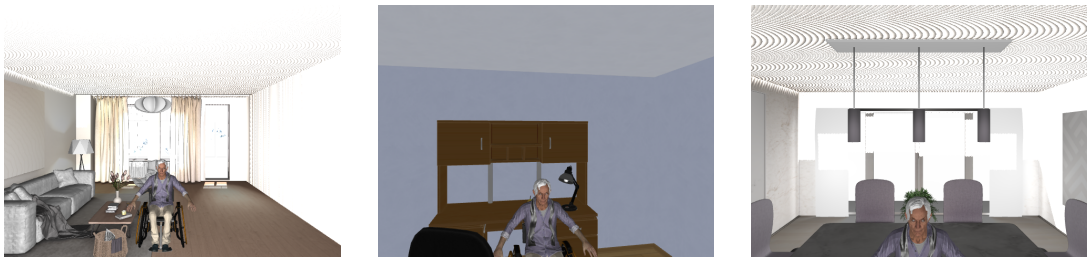
		Posición [m]			**Orientación [rad]**		
Escenario	**Modelo**	**x**	**y**	**z**	**r**	**p**	**y**
Sala	AM2	2.70	0.00	0.06	0.00	0.00	-1.57
	Robot	0.00	0.00	1.30	0.00	0.00	0.00
Estudio	AM2	2.70	0.00	0.02	0.00	0.00	-1.57
	Robot	0.00	0.00	1.30	0.00	0.00	0.00
Comedor	AM2	2.00	0.00	0.05	0.00	0.00	-1.57
	Robot	0.00	0.00	1.30	0.00	0.00	0.00

Tabla 7: Posición de las luces auxiliares.

Posición [m]		Luz_1		Luz_2		Luz_3	
Modelo	**Escenario**	**x**	**y**	**x**	**y**	**x**	**y**
AM1	Sala	-0.30	2.30	0.00	0.00	-0.60	-1.60
	Estudio	0.00	-1.88	0.00	1.70	–	–
	Comedor	0.34	-1.24	0.00	0.85	–	–
AM2	Sala	-0.30	2.30	0.00	0.00	-0.60	-1.60
	Estudio	0.60	1.47	0.75	-1.80	–	–
	Comedor	0.00	-1.50	0.00	1.80	0.15	0.00

(a) Sala. (b) Estudio. (c) Comedor.

Figura 16: Modelo AM1 con la configuración de la Tabla 7.



(a) Sala. (b) Estudio. (c) Comedor.

Figura 17: Modelo AM2 con la configuración de la Tabla 7.

L.12. Desempeño por escenarios

En cada escenario (sala,estudio y comedor) se llevaron a cabo 16 pruebas. Con cada modelo 3D de un adulto mayor (AM1 y AM2) se realizaron 4 trayectorias (lineal, diagonal, zeta y circular) a dos velocidades ($0.3\,\frac{\mathrm{m}}{\mathrm{s}}$ y $1\,\frac{\mathrm{m}}{\mathrm{s}}$). Por lo que, las métricas RMSE, MAE, MAX_E y FDE describen el error al obtener la posición estimada (A) y predicha (B).

L.12.1. Sala

Tabla 8: Error de posición de AM1 a $0.3\,\frac{\mathrm{m}}{\mathrm{s}}$.

Trayectoria	**Lineal**		**Diagonal**		**Zeta**		**Circular**	
Métricas [m]	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
RMSE	0.220	0.317	0.216	0.388	0.218	0.600	0.221	0.779
MAE	0.220	0.285	0.215	0.333	0.218	0.432	0.221	0.526
MAX_E	0.226	0.694	0.228	0.764	0.225	1.432	0.229	2.292
FDE	0.221	0.216	2.319	2.408	1.111	1.111	0.460	0.028

Tabla 9: Error de posición de AM1 a $1\,\frac{\mathrm{m}}{\mathrm{s}}$.

Trayectoria	**Lineal**		**Diagonal**		**Zeta**		**Circular**	
Métricas [m]	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
RMSE	0.220	0.604	0.215	0.216	0.220	0.765	0.225	1.041
MAE	0.220	0.387	0.215	0.216	0.220	0.477	0.225	0.720
MAX_E	0.224	1.610	0.224	0.222	0.229	2.146	0.248	2.488
FDE	0.224	0.245	3.182	3.172	1.130	1.221	0.508	0.162

Tabla 10: Error de posición de AM2 a 0.3 $\frac{m}{s}$.

Trayectoria	**Lineal**		**Diagonal**		**Zeta**		**Circular**	
Métricas [m]	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
RMSE	0.209	0.588	0.171	0.341	0.197	0.396	0.190	1.438
MAE	0.208	0.437	0.163	0.236	0.185	0.302	0.168	1.029
MAX_E	0.244	1.711	0.229	1.159	0.359	0.989	0.280	3.133
FDE	0.827	1.259	0.920	0.888	0.448	0.449	2.308	2.738

Tabla 11: Error de posición de AM2 a 1 $\frac{m}{s}$.

Trayectoria	**Lineal**		**Diagonal**		**Zeta**		**Circular**	
Métricas [m]	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
RMSE	0.210	0.549	0.196	0.354	0.195	0.700	0.644	1.076
MAE	0.209	0.416	0.189	0.304	0.194	0.559	0.344	0.747
MAX_E	0.253	1.149	0.235	0.645	0.219	1.250	1.661	2.041
FDE	0.876	1.060	1.010	1.075	0.919	0.953	2.583	2.558

L.12.2. Estudio

Tabla 12: Error de posición de AM1 a 0.3 $\frac{m}{s}$.

Trayectoria	**Lineal**		**Diagonal**		**Zeta**		**Circular**	
Métricas [m]	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
RMSE	0.219	0.377	0.219	0.251	0.216	0.271	0.202	0.226
MAE	0.219	0.282	0.219	0.245	0.216	0.258	0.202	0.222
MAX_E	0.236	1.688	0.233	0.423	0.234	0.507	0.207	0.346
FDE	0.980	0.910	0.233	0.203	0.289	0.277	0.684	0.668

Tabla 13: Error de posición de AM1 a 1 $\frac{\text{m}}{\text{s}}$.

Trayectoria	**Lineal**		**Diagonal**		**Zeta**		**Circular**	
Métricas [m]	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
RMSE	0.238	0.386	0.226	0.230	0.211	0.435	0.201	0.285
MAE	0.234	0.339	0.226	0.219	0.211	0.324	0.201	0.267
MAX_E	0.366	0.723	0.243	0.360	0.242	1.270	0.209	0.462
FDE	1.228	1.316	0.243	0.123	1.075	1.084	0.703	0.987

Tabla 14: Error de posición de AM2 a 0.3 $\frac{\text{m}}{\text{s}}$.

Trayectoria	**Lineal**		**Diagonal**		**Zeta**		**Circular**	
Métricas [m]	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
RMSE	0.224	0.357	0.214	0.225	0.227	0.287	0.253	0.329
MAE	0.223	0.270	0.198	0.208	0.226	0.266	0.248	0.302
MAX_E	0.304	1.798	0.327	0.409	0.241	0.701	0.377	0.660
FDE	0.205	0.205	0.502	0.374	1.158	1.158	1.085	1.078

Tabla 15: Error de posición de AM2 a 1 $\frac{\text{m}}{\text{s}}$.

Trayectoria	**Lineal**		**Diagonal**		**Zeta**		**Circular**	
Métricas [m]	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
RMSE	0.237	0.323	0.204	0.306	0.233	0.365	0.216	0.219
MAE	0.237	0.288	0.170	0.240	0.233	0.322	0.215	0.219
MAX_E	0.294	0.808	0.313	0.655	0.251	0.701	0.233	0.229
FDE	0.211	0.211	0.559	0.495	1.300	1.301	1.696	1.692

L.12.3. Comedor

Tabla 16: Error de posición de AM1 a 0.3 $\frac{\text{m}}{\text{s}}$.

Trayectoria	**Lineal**		**Diagonal**		**Zeta**		**Circular**	
Métricas [m]	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
RMSE	0.227	0.410	0.198	0.338	0.206	0.270	0.207	0.499
MAE	0.227	0.321	0.196	0.280	0.203	0.236	0.207	0.365
MAX_E	0.241	0.945	0.225	0.699	0.277	0.821	0.217	1.126
FDE	0.357	0.369	1.564	1.562	1.591	1.597	0.687	0.700

Tabla 17: Error de posición de AM1 a 1 $\frac{\text{m}}{\text{s}}$.

Trayectoria	**Lineal**		**Diagonal**		**Zeta**		**Circular**	
Métricas [m]	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
RMSE	0.205	0.642	0.218	0.403	-	-	0.224	0.798
MAE	0.203	0.626	0.216	0.292	-	-	0.224	0.697
MAX_E	0.236	0.768	0.250	1.441	-	-	0.233	1.186
FDE	0.281	0.660	0.359	0.366	-	-	0.217	0.668

Tabla 18: Error de posición de AM2 a 0.3 $\frac{\text{m}}{\text{s}}$.

Trayectoria	**Lineal**		**Diagonal**		**Zeta**		**Circular**	
Métricas [m]	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
RMSE	0.547	0.667	0.192	0.480	0.579	0.737	0.192	0.242
MAE	0.382	0.500	0.183	0.408	0.421	0.558	0.188	0.235
MAX_E	1.301	1.513	0.230	0.906	1.320	1.735	0.229	0.397
FDE	0.273	0.248	1.070	1.070	2.103	2.411	0.196	0.196

Tabla 19: Error de posición de AM2 a 1 $\frac{\text{m}}{\text{s}}$.

Trayectoria	**Lineal**		**Diagonal**		**Zeta**		**Circular**	
Métricas [m]	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
RMSE	0.224	0.524	0.199	0.484	0.428	1.065	0.203	0.472
MAE	0.222	0.457	0.195	0.360	0.335	0.712	0.193	0.403
MAX_E	0.289	0.841	0.250	1.319	0.784	2.529	0.302	0.979
FDE	0.236	0.233	0.593	0.593	2.171	2.171	0.189	0.267