UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

Máster Universitario en Robótica

"Diseño y simulación de un sistema automatizado para la navegación y almacenamiento eficiente de paquetes en centros logísticos utilizando el Robot Colaborativo UR10e en conjunto con un Robot Móvil y la integración de visión artificial y marcadores ArUco mediante la herramienta CoppeliaSim (V-REP)"

Trabajo de Fin de Máster

Curso 2022-2023

Autor: Nathalie Patricia Brito Santos

Tutor: Prof. Dr. Oscar Reinoso García

AGRADECIMIENTOS

Realizar mi Máster en el extranjero ha sido un viaje lleno de desafíos y aprendizajes inolvidables. Desde el inicio, supe que seguir mi sueño no sería fácil, pero nunca imaginé cuánto impactaría en mi vida. Detrás de las fotografías en Instagram, descubrí que la verdadera historia estaba llena de momentos de superación y crecimiento. Llegar a España con mi maleta rebosante de alegría y determinación fue solo el comienzo de un camino que me enfrentaría a cambios de horario, choques culturales y una nueva dinámica educativa. Estos desafíos se volvieron poderosos impulsores de mi desarrollo personal y académico, y hoy, al mirar hacia atrás, puedo decir con gratitud que cada obstáculo superado ha forjado mi fortaleza y me ha llevado a alcanzar metas que nunca creí posibles.

Ha sido un largo camino, ha habido risas, lágrimas, abrazos a distancia y más, por tal razón, es fundamental recordar y agradecer. Primero a Dios por permitirme vivir este proceso y terminar bien en salud física y emocional. Luego a mis padres por ser siempre inspiración de superación personal y profesional. A mi Moi hermosa por ser mi confidente, mi cómplice y mi apoyo, por darme ánimo, fuerzas y acompañarme cada día hasta quedarme dormida; nuestro amor es un regalo que valoro más que cualquier tesoro. A mis amigas que desde la distancia estuvieron ahí con mucho amor para mí. A cada uno de mis compañeros de estudios, aventuras y travesía por darme ánimo, alegría, fuerza y calor familiar en España. A mi prima que vive en Madrid por sus consejos y compañía. A mis compañeros de trabajo por ser los mejores porristas. A mis hermanas y hermano por su apoyo inquebrantable dándome fuerzas en los momentos de debilidad.

A mis abuelas, una en el cielo y otra presente en mi vida, les dedico un agradecimiento especial. A pesar de la distancia física, siento su amor y cuidado en cada paso que doy. Son amor y están presentes en los momentos más importantes de mi vida. Su legado vive en mí y me inspiran a seguir adelante con valentía.

Al Instituto Especializado de Estudios Superiores Loyola (IEESL), mi amada alma mater, quiero expresar mi profundo agradecimiento por todo el apoyo incondicional que he

recibido a lo largo de mi trayectoria. En este lugar tan especial, no solo he crecido académicamente desde muy temprana edad, sino que también he encontrado un hogar donde hoy en día tengo el privilegio de laborar.

Al Ministerio de Educación Superior de mi país, República Dominicana, gracias por crear oportunidades para que los jóvenes de mi país podemos estudiar en el extranjero y ampliar nuestros horizontes. Estoy sumamente agradecida por su inversión en el futuro de la nación y por ser parte esencial para yo poder hacer realidad mis aspiraciones.

A Valgrai, la Valencian Graduate School and Research Network of Artificial Intelligence, mi gratitud es infinita. Gracias por ser una entidad comprometida con el avance de la ciencia y por apoyarme en la realización de mis estudios en robótica. Su apoyo incondicional y sus recursos han sido fundamentales para mi crecimiento académico y profesional. La colaboración con expertos en el campo y el acceso a tecnología de vanguardia me ha permitido explorar nuevas fronteras en mi campo de estudio. Estoy profundamente agradecida por la confianza que han depositado en mí y por ser un pilar en mi camino hacia el éxito.

Por último, pero no menos importante, deseo extender mi agradecimiento a la Universidad Miguel Hernández (UMH) y a todas las personas que forman parte de ella. A mis estimados profesores, su sabiduría y pasión por la enseñanza han dejado una marca imborrable en mi desarrollo académico. A mi tutor de tesis, el Dr. Oscar Reinoso García, gracias por su orientación constante. A Joaquín Pastor, cuyo apoyo e inspiración han sido un faro en mi camino, gracias por creer en mí incluso cuando dudaba de mí misma. Siempre recordaré esas palabras de aliento y motivación que me han impulsado a superar mis propios límites.

¡Gracias a todos!

RESUMEN

En la era de la economía globalizada, las dinámicas de competencia y creciente demanda ponen a prueba a los centros logísticos, que se enfrentan al indispensable reto de optimizar la eficiencia en el almacenamiento de paquetes. Este desafío no solo es crítico para mantener la competitividad, sino también para satisfacer las demandas en constante expansión de la cadena de suministro. Paralelamente, el impulso hacia la implementación de soluciones robóticas en estas operaciones introduce una preocupación adicional: la seguridad en la navegación de los robots. Así, los desafíos de optimizar la eficiencia y asegurar la navegación segura de los robots, se convierten en piezas fundamentales en el rompecabezas de la modernización de los centros logísticos.

Antes de implementar dichos sistemas en un entorno real, es de vital importancia llevar a cabo una fase de simulación. Las simulaciones brindan la posibilidad de realizar pruebas rigurosas, identificar posibles fallos y asegurar la fiabilidad del sistema en un entorno controlado, minimizando así los riesgos y costes asociados con los errores operacionales en la fase de implementación real.

En este contexto, este trabajo de fin de máster busca abordar estos desafíos presentes en la logística. Presenta el diseño y la simulación de un sistema automatizado orientado a mejorar la navegación y el almacenamiento de paquetes en centros logísticos. La solución se basa en la utilización de un Robot Colaborativo UR10e de Universal Robots, en conjunción con un Robot Móvil. Este sistema innovador integra tecnología de visión artificial y marcadores ArUco, herramientas fundamentales para su desarrollo y prueba a través de la plataforma de simulación CoppeliaSim (V-REP).

El sistema automatizado que se presenta en este trabajo de fin de máster para la optimización del almacenamiento implica una coordinación meticulosa entre el Robot Móvil y el Robot Colaborativo UR10e. Ambos están equipados con visión artificial para detectar los marcadores ArUco, que sirven para identificar cada paquete y trazar la ruta que el Robot Móvil debe seguir dentro del almacén. En la práctica, el Robot Móvil

transporta al Robot Colaborativo UR10e al lugar correcto en el almacén, donde el UR10e recoge los paquetes de la cinta transportadora y los almacena de manera eficiente en los estantes una vez que el Robot Móvil le haya transportado al lugar establecido para cada paquete.

Este trabajo de fin de máster implicó un estudio detallado de áreas como los Robots Colaborativos, Robots Móviles (plataforma móvil), Norma ISO para Robots Colaborativos, el uso de la visión artificial en los Robots Colaborativos y móviles, los sistemas de evasión de obstáculos, las Marcas ArUco como referencia visual y los métodos de clasificación de paquetes en centros logísticos. Esta investigación representa un avance significativo en la automatización de los centros logísticos, proporcionando una solución innovadora y prometedora para mejorar la eficiencia en el almacenamiento de paquetes. La metodología empleada y la demostración práctica de cómo la integración estratégica de tecnologías avanzadas puede resolver desafíos logísticos específicos pueden ser útiles para futuras investigaciones y desarrollos en este campo, sirviendo como un punto de referencia en la exploración de soluciones similares.

Palabras clave: Robot Colaborativo, UR10e, Robot Móvil, Visión Artificial, Marcadores ArUco, Python, CoppeliaSim, Clasificación de Paquetes, Centros Logísticos, Simulación, Automatización, Robótica, Eficiencia de Almacenamiento, Navegación de Robots.

ABSTRACT

In the era of globalized economics, the dynamics of competition and growing demand challenge logistics centers, which face the crucial task of optimizing package storage efficiency. This challenge is not only critical to maintaining competitiveness but also to satisfying the ever-expanding demands of the supply chain. In parallel, the drive to implement robotic solutions in these operations introduces an additional concern: ensuring safe robot navigation. Thus, optimizing efficiency and securing robot navigation become fundamental pieces in the puzzle of modernizing logistics centers.

Before implementing such systems in a real environment, it is vitally important to undergo a simulation phase. Simulations provide the opportunity for rigorous testing, identification of potential failures, and ensuring the system's reliability in a controlled environment, thus minimizing the risks and costs associated with operational errors in the actual implementation phase.

In this context, this master's thesis seeks to address these logistical challenges. It presents the design and simulation of an automated system aimed at improving package navigation and storage in logistics centers. The solution is based on the use of a UR10e Collaborative Robot from Universal Robots, in conjunction with a Mobile Robot. This innovative system integrates artificial vision technology and ArUco markers, essential tools for its development and testing through the CoppeliaSim (V-REP) simulation platform.

The automated system presented in this master's thesis for storage optimization involves meticulous coordination between the Mobile Robot and the UR10e Collaborative Robot. Both are equipped with artificial vision to detect ArUco markers, which are used to identify each package and plot the route the Mobile Robot should follow within the warehouse. In practice, the Mobile Robot transports the UR10e Collaborative Robot to the correct location in the warehouse, where the UR10e picks up the packages from the conveyor

belt and efficiently stores them on the shelves once the Mobile Robot has transported it to the established location for each package.

This master's thesis involved a detailed study of areas such as Collaborative Robots, Mobile Robots, ISO Standards for Collaborative Robots, the use of artificial vision in Collaborative and Mobile Robots, obstacle avoidance systems, ArUco Marks as visual references, and package classification methods in logistics centers. This research represents a significant advance in the automation of logistics centers, providing an innovative and promising solution to improve package storage efficiency. The methodology used and the practical demonstration of how the strategic integration of advanced technologies can address specific logistical challenges can be useful for future research and developments in this field, serving as a reference point in the exploration of similar solutions.

Keywords: Collaborative Robot, UR10e, Mobile Robot, Computer Vision, ArUco Markers, Python, CoppeliaSim, Package Classification, Logistics Centers, Simulation, Automation, Robotics, Storage Efficiency, Robot Navigation.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1 Motivación

En la industria logística, la eficiencia y precisión en el almacenamiento de paquetes son aspectos cruciales. Enfrentándose a volúmenes crecientes de paquetes, los centros logísticos requieren soluciones innovadoras que optimicen estos procesos. En este contexto, este TFM plantea el diseño y la simulación de un sistema automatizado para el almacenamiento inteligente de paquetes utilizando robótica avanzada, tecnología de visión artificial y marcadores ArUco.

El objetivo es desarrollar una solución tecnológica que integra robots colaborativos, como el UR10e de Universal Robots, y un Robot Móvil Autónomo (AMR) para mejorar las operaciones de los centros logísticos. La visión por computadora y los marcadores ArUco permiten una identificación precisa de los paquetes y la determinación de su ubicación óptima de almacenamiento.

Además, la simulación en el entorno CoppeliaSim permite la validación de este sistema antes de su implementación real, garantizando su funcionalidad y efectividad. El resultado es una solución que puede incrementar la productividad y mejorar la eficiencia en los centros logísticos.

Este TFM pretende impactar tanto el ámbito académico como el industrial. Desde el punto de vista académico, contribuyendo al conocimiento en el campo de la robótica y la automatización de procesos logísticos, brindando una base sólida para futuras investigaciones y desarrollos en este ámbito. Desde una perspectiva industrial, ofreciendo una solución tecnológica aplicable en centros logísticos reales, que puede generar mejoras significativas en la eficiencia operativa, la precisión en el almacenamiento de paquetes y la optimización de recursos.

1.2 Objetivos

El propósito de este Trabajo de Fin de Máster (TFM) consiste en desarrollar una solución integrada que optimice la eficiencia del almacenamiento y transporte de paquetes en

centros logísticos mediante la implementación del Robot Colaborativo UR10e de Universal Robots, un Robot Móvil Autónomo (AMR) y la visión artificial con marcadores ArUco. En consecuencia, el objetivo principal se orienta hacia el mejoramiento de la eficiencia y precisión en los procesos de manipulación y almacenamiento de paquetes, enfocándose en la optimización del uso del espacio y la minimización de errores.

Para cumplir con este propósito, se necesitará:

- Integrar el Robot Colaborativo UR10e de Universal Robots y una plataforma móvil para la manipulación, almacenamiento y transporte eficiente de paquetes en centros logísticos.
- Incorporar los marcadores ArUco para la identificación y localización precisa de paquetes utilizando visión artificial.
- Utilizar los marcadores ArUco para trazar la ruta de navegación de la plataforma móvil, permitiendo una navegación precisa y segura dentro del entorno de los centros logísticos.
- Simular y validar el sistema completo, incluyendo robots y algoritmos de visión artificial, en el entorno CoppeliaSim.

Al cumplir estos objetivos, se espera contribuir a la investigación en robótica aplicada a la logística, proporcionando soluciones efectivas para el almacenamiento optimizado de paquetes en los centros logísticos. Además, este TFM servirá como una base sólida para el conocimiento y experiencia en el uso de robots colaborativos, robot móvil, visión artificial y marcadores ArUco, fortaleciendo mi desarrollo profesional como docente universitaria y la especialización en el área de la robótica que busco alcanzar con este trabajo final.

1.3 Estructura de la memoria del TFM

La memoria del presente Trabajo Fin de Máster se estructura en ocho capítulos, los cuales abordan de manera detallada la ejecución y desarrollo de cada uno de los pasos

necesarios para lograr la combinación de tecnologías que permitan mejorar significativamente tanto la eficiencia en la clasificación de paquetes como la seguridad en la navegación de robots colaborativos en entornos logísticos. A continuación, se proporcionará una breve descripción de cada capítulo:

Capítulo 1. Introducción: En este capítulo, se presenta la motivación y el contexto del proyecto, enfatizando en la necesidad y la relevancia de optimizar los procesos de almacenamiento y transporte de paquetes en los centros logísticos mediante la implementación de un sistema integrado que utilice el Robot Colaborativo UR10e de Universal Robots, un Robot Móvil y la visión artificial con marcadores ArUco. Asimismo, se definen los objetivos del TFM y se presenta un resumen de la estructura de la memoria, proporcionando una visión general del contenido de cada capítulo.

Capítulo 2. Estado del arte: En este capítulo, se realiza una revisión y análisis del estado actual de las tecnologías y normativas relacionadas con el sistema desarrollado en este trabajo de fin de máster. Se abordan los robots colaborativos, destacando el UR10e de Universal Robots y la norma ISO para este tipo de robots. Se explora el campo de los Robots Móviles, sus aplicaciones en los centros logísticos, y las certificaciones pertinentes. Asimismo, se examina la aplicación de la visión artificial en estos robots y el uso de marcadores ArUco como referencia visual en sistemas de almacenamiento y transporte de paquetes. Este capítulo proporciona un marco teórico para comprender mejor el contexto y las bases de la solución propuesta en este TFM.

Capítulo 3. Metodología (diseño y funcionalidad del sistema): Este capítulo empieza con una introducción al simulador de robótica CoppeliaSim (anteriormente V-REP). Se describe luego el proceso de creación de un entorno de simulación, detallando cada paso para construir una estación que integra el robot colaborativo UR10e, el Robot Móvil, y el sistema de visión artificial basado en marcadores ArUco. Se explican las características, capacidades y funciones del UR10e y de la plataforma móvil, así como su interacción y

coordinación dentro del sistema. También se detalla la funcionalidad de los marcadores ArUco en la identificación y localización de objetos.

Capítulo 4. Implementación y simulación del sistema: Este capítulo se enfoca en la ejecución y simulación del sistema en el entorno de simulación CoppeliaSim. Destaca la configuración detallada del sistema de almacenamiento, la activación y control del brazo robótico UR10e y del Robot Móvil, y la puesta en práctica de la lógica de programación desarrollada para la manipulación y transporte de paquetes. Se explica en profundidad cómo se implementa y utiliza la visión artificial en la simulación para la identificación y localización de paquetes, haciendo uso de los marcadores ArUco. Finalmente, se detallan los aspectos más relevantes de la interfaz de programación utilizada para el control del sistema, con énfasis en los comandos clave y la interacción con CoppeliaSim a través de la API Remota.

Capítulo 5. Resultados: En este capítulo, se presentan los resultados obtenidos de la simulación del sistema. Se evidencia la eficacia del sistema automatizado creado, con imágenes de las simulaciones realizadas, mostrando la correcta manipulación y transporte de paquetes.

Capítulo 6. Conclusiones y líneas futuras de trabajo: En este capítulo, se exponen las conclusiones del proyecto, comparándolas con los objetivos establecidos. También, Se realiza una discusión en términos de eficiencia y precisión del sistema de almacenamiento y transporte de paquetes propuesto, comparándolo con métodos convencionales. Además, se plantean posibles mejoras y líneas de investigación futuras, con énfasis en la escalabilidad y adaptabilidad del sistema a diferentes entornos logísticos y tipos de paquetes.

Capítulo 7. Bibliografía: En este capítulo se citan todas las referencias bibliográficas utilizadas en el trabajo, incluyendo artículos científicos, normativas, libros y otros recursos relevantes. Las referencias se presentan ordenadas y se proporciona el código correspondiente para su cita en la memoria.

Capítulo 8. Anexos: En este último capítulo, se proporciona información adicional y relevante como el glosario de las palabras clave, las hojas de características técnicas de los robots utilizados, detalles de los marcadores ArUco utilizados y los códigos de los programas desarrollados para el robot colaborativo UR10e y el Robot Móvil.

2. ESTADO DEL ARTE

2.1 Robots Colaborativos

Los robots colaborativos, comúnmente conocidos como cobots, son una clase de robots diseñados para interactuar y trabajar en colaboración directa con seres humanos en entornos de trabajo compartidos. A diferencia de los robots industriales tradicionales, los cobots están diseñados para ser seguros y capaces de trabajar codo a codo con los trabajadores sin necesidad de barreras físicas de seguridad, [1].

Una característica fundamental de los cobots es su capacidad de colaboración. Estos robots están diseñados para complementar y asistir a los trabajadores humanos en tareas específicas, ya sea realizando tareas repetitivas y tediosas o brindando soporte en tareas que requieren fuerza o precisión. Los cobots son programables y flexibles, lo que les permite adaptarse a diferentes entornos y tareas con relativa facilidad.

La evolución de los robots colaborativos ha sido notable en los últimos años. A partir de la introducción de estándares de seguridad personal en los dispositivos de asistencia inteligente por parte de la Asociación de Industrias Robóticas (RIA) en 2003, surgió el término "robot colaborativo" que se popularizó rápidamente. En ese mismo año, KUKA se destacó al desarrollar el primer robot colaborativo comercial capaz de realizar tareas automatizadas y simplificadas en entornos industriales repetitivos.

Desde entonces, se han logrado avances significativos en la creación de cobots con características específicas, como la opción de uno o dos brazos y entre seis y siete grados de libertad. Estas mejoras han permitido una mayor flexibilidad y adaptabilidad en la interacción con los seres humanos, impulsando así el campo de los robots colaborativos. Además, la estandarización y regulación de la seguridad en los cobots ha sido un factor clave para su aceptación y aplicación en diversas industrias, [2].

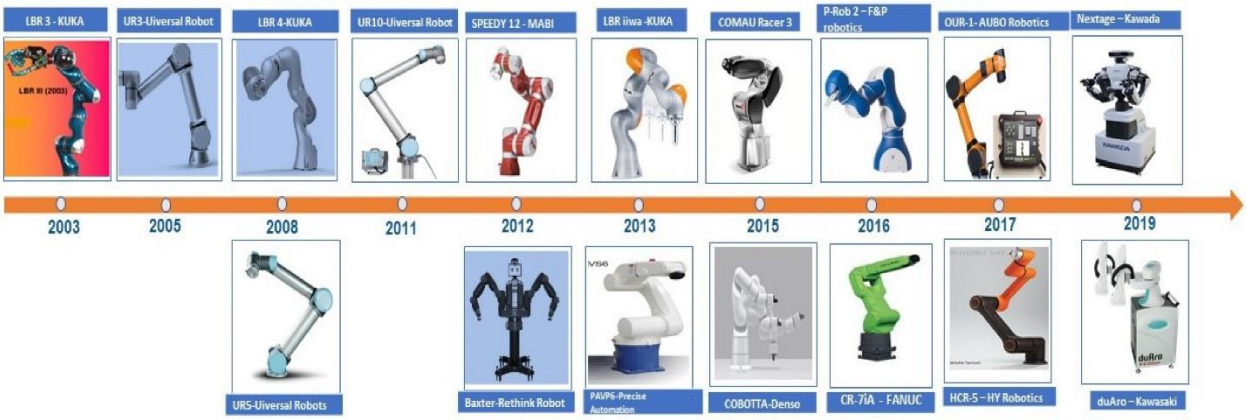


Figura 2.1 Línea de tiempo: Desarrollo de robots colaborativos desde el 2003, [2].

Robots colaborativos o cobots en el mercado actual.

Marca	Nombre de los Robots	Descripción	Precio (Estimado)	Carga Máxima (Estimada)	Alcance del Brazo (Estimado)
Universal Robots	UR3e, UR5e, UR10e, UR16e, UR20.	Robots colaborativos flexibles y fáciles de usar.	$35,000 - $70,000	3 kg - 20 kg	500 mm - 1300 mm
ABB	YuMi, IRB 14000 YuMi	Robots colaborativos con alta precisión y flexibilidad.	$50,000 - $100,000	0.5 kg - 10 kg	559 mm - 1173 mm
KUKA	LBR iiwa	Robot colaborativo con sensores de fuerza y torque integrados.	$60,000 - $100,000	7 kg	820 mm

Yaskawa	HC10, MOTOMAN-PL	Robots colaborativos versátiles y precisos.	$30,000 - $50,000	5 kg - 15 kg	600 mm - 900 mm
FANUC	CR-35iA	Robot colaborativo con gran capacidad de carga.	$80,000 - $120,000	35 kg	1813 mm
Omron	TM Series	Robots colaborativos compactos y fáciles de programar.	$30,000 - $50,000	4 kg - 12 kg	700 mm - 1300 mm
Stäubli	TX2 Series	Robots colaborativos con alta precisión y seguridad.	$60,000 - $100,000	5 kg - 15 kg	720 mm - 1450 mm
Techman Robots	TM5-700, TM12-AG	Robots colaborativos fáciles de usar y programar.	$20,000 - $40,000	4 kg - 12 kg	700 mm - 1300 mm

Tabla 2.1 Robots colaborativos en el mercado actual, elaboración propia.

Las estimaciones de precios, carga máxima y alcance del brazo robot colaborativo puede variar según las configuraciones y accesorios adicionales seleccionados para cada modelo.

Diferencias entre cobots y robots industriales tradicionales.

En el ámbito de la robótica, es importante diferenciar entre los robots industriales y los robots colaborativos, también conocidos como cobots. Estos dos tipos de robots presentan características distintivas que los hacen adecuados para diferentes aplicaciones y entornos de trabajo. Dentro de estas características están las siguientes, [3]:

Tamaño: Los robots industriales se caracterizan por su tamaño considerable, lo cual puede influir en la organización de la cadena de producción en una fábrica. Por otro lado, los cobots son mucho más pequeños, lo que los hace más adaptables y no limita tanto el área de trabajo.

Zona de trabajo: Mientras que los robots industriales están fijos en una zona de trabajo específica y requieren más espacio para realizar sus tareas, los cobots no necesitan tanta amplitud y, además, al ser móviles, pueden ser trasladados de una zona de la fábrica a otra con facilidad.

Nivel de dificultad en la programación: La programación de robots industriales suele ser compleja y requerir conocimientos especializados, lo cual limita el número de personas capaces de manejar estos dispositivos. En contraste, los cobots están diseñados para ser fáciles de usar, con interfaces intuitivas que no requieren formación previa, lo que permite a un mayor número de personas manipularlos y reprogramarlos para diversas tareas.

Proceso de producción: Los robots industriales son ideales para grandes tiradas de producción y soportan el traslado de objetos pesados, siendo muy eficientes en estas condiciones. Por su parte, los cobots son más versátiles y se adaptan mejor a tiradas cortas que priorizan la personalización del producto. Además, son ideales para la colaboración con seres humanos y pueden redistribuirse para diferentes tareas.

Rentabilidad e implantación: Los cobots suelen ser más económicos que los robots industriales, lo que se refleja en un retorno de inversión (ROI) más rápido. Mientras que

los robots industriales también pueden amortizarse a largo plazo, los cobots a veces logran hacerlo en menos de un año. Esto explica por qué cada vez más pymes optan por cobots como una incorporación rentable a sus fábricas.

Nivel de colaboración y seguridad: Los robots industriales no suelen colaborar directamente con los operarios de la fábrica, ya que están diseñados para realizar tareas específicas y "servir" al operario. Por el contrario, los cobots están diseñados para cooperar con los seres humanos y se enfocan en ser dispositivos seguros. No requieren vallados físicos, ya que sus sensores inteligentes actúan como sistemas de seguridad, detectando el movimiento y deteniéndose automáticamente para proteger al operario.

Mayor margen de acción frente a errores: Los fallos en los dispositivos utilizados en las fábricas pueden resultar costosos. Los robots industriales tienden a ser más rígidos en sus tareas y dependen en gran medida de los proveedores de robótica industrial, lo que puede aumentar los costos. Por otro lado, los cobots se caracterizan por su dinamismo y facilidad de reprogramación, lo que los convierte en una herramienta útil para solucionar errores de producción y reduce la dependencia de servicios técnicos externos.

Principales componentes y tecnologías utilizadas en los cobots.

La capacidad de los robots colaborativos para interactuar de manera segura y eficiente con los seres humanos se sustenta en una serie de componentes y tecnologías fundamentales. Estos elementos desempeñan un papel crucial en el diseño y la funcionalidad de los cobots. Los principales componentes y tecnologías que permiten el desarrollo de los robots colaborativos son:

Sensores:

Los sensores desempeñan un papel crucial en la capacidad de un cobot para percibir y comprender su entorno. Estos sensores pueden incluir cámaras, sensores de fuerza, sensores táctiles y sensores de proximidad, entre otros. La información recopilada por

estos sensores permite al cobot detectar la presencia y posición de los seres humanos, así como interactuar de manera segura con ellos, [4].

Actuadores:

Los actuadores son los componentes responsables de generar el movimiento y la acción en un cobot. Los cobots suelen utilizar actuadores eléctricos, como motores y servomotores, que les brindan una alta precisión y control en sus movimientos. Estos actuadores permiten al cobot realizar tareas delicadas y colaborativas con los seres humanos de manera segura, [5].

Controladores y software:

Los cobots están equipados con controladores y software especializados que permiten la programación y control del robot. Estos sistemas de control pueden variar desde interfaces gráficas intuitivas hasta lenguajes de programación específicos. El software de programación permite a los operadores configurar el comportamiento del cobot y adaptarlo a diferentes tareas y escenarios de colaboración, [6].

Sistemas de seguridad:

La seguridad es un aspecto fundamental en los cobots, especialmente al trabajar en estrecha colaboración con los seres humanos. Los cobots están equipados con sistemas de seguridad avanzados, como sensores de colisión, limitadores de fuerza y controladores de velocidad, que les permiten detectar situaciones peligrosas y reaccionar de manera adecuada para evitar lesiones, [7].

Funcionamiento de los Robots Colaborativos.

Vistos los principales componentes y tecnologías utilizadas en los cobots, el siguiente diagrama representa las principales etapas del funcionamiento de un robot colaborativo. Los sensores capturan información del entorno y la transmiten al sistema de procesamiento de datos. El procesamiento de datos se encarga de analizar y procesar la información recibida. A partir de ahí, el control y la planificación toman decisiones sobre

las acciones a realizar por el robot. Finalmente, los actuadores ejecutan las acciones físicas del robot en respuesta a las instrucciones del control y la planificación, [8].

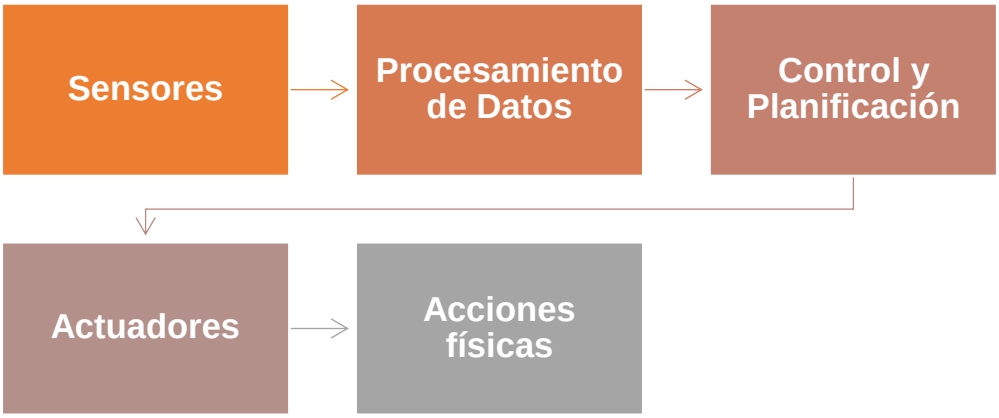


Figura 2.2 Diagrama en bloque del funcionamiento básico de los Cobots, elaboración propia.

Aplicaciones de los Robots Colaborativos.

La aplicación de robots colaborativos se extiende a diversas industrias y sectores, ofreciendo soluciones innovadoras para mejorar la eficiencia, la seguridad y la productividad. Estos robots, diseñados para trabajar en colaboración directa con los seres humanos, han encontrado su lugar en entornos de manufactura, logística, salud, alimentación, automoción, agricultura, construcción, educación y formación, entre otros. A continuación, se presenta una tabla descriptiva con ejemplos concretos de aplicaciones de robots colaborativos en cada industria, destacando la marca, la fecha y el lugar donde se llevaron a cabo, [9].

Sector	Aplicación	Ejemplos reales del uso de Cobots
Manufactura	- Montaje de piezas	- UR5 de Universal Robots utilizado en BMW (2019, Alemania) para el ensamblaje de componentes automotrices.

	- Manipulación y empaquetado de productos	- Yaskawa HC10 utilizado por Flex Ltd. (2020, Japón) para la manipulación y empaquetado de productos electrónicos.
Logística	- Clasificación y paletización de productos	- Fanuc CR-35iA utilizado por DHL (2018, Singapur) para la clasificación y paletización automatizada en centros logísticos.
	- Gestión de inventario	- Omron TM Series utilizado en almacenes (2021, Estados Unidos) para la gestión automatizada de inventario.
Salud	- Asistencia en cirugías colaborativas	- RAS Robotic Ankle Systems utilizado en cirugías de tobillo (2020, Estados Unidos)
	- Transporte de medicamentos y equipos médicos	- ABB IRB 910SC utilizado en hospitales (2019, Suiza) para el transporte de medicamentos y equipos médicos.
Alimentación	- Procesamiento y envasado de alimentos	- Stäubli TX2 utilizado en la industria alimentaria (2020, Francia) para el procesamiento y envasado de alimentos.
	- Manipulación de alimentos delicados y sensibles	- Yaskawa Motoman SDA10D utilizado para la manipulación segura de alimentos (2019, Japón).
Automoción	- Manipulación de piezas en líneas de ensamblaje	- KUKA LBR iiwa utilizado en Volkswagen (2017, Alemania) para la manipulación de piezas en líneas de ensamblaje.
	- Colocación y soldadura de componentes automotrices	- Fanuc CRX-10iA utilizado en la industria automotriz (2020, Japón) para la colocación y soldadura de componentes.
Agricultura	- Siembra y cosecha automatizada	- Agrobot utilizado en la agricultura (2021, España) para la siembra y cosecha automatizada.

	- Clasificación y empaquetado de productos agrícolas	- Ecorobotix utilizado en la clasificación y empaquetado de productos agrícolas (2019, Suiza).
Construcción	- Levantamiento y colocación de materiales	- Built Robotics utilizado en la construcción (2020, Estados Unidos) para el levantamiento y colocación de materiales.
	- Inspección y mantenimiento de estructuras	- Doosan Robotics utilizado en la inspección y mantenimiento de estructuras (2021, Corea del Sur).
Educación y formación	- Asistencia en la enseñanza y demostración de conceptos	- NAO utilizado en instituciones educativas (2018, Francia) para la asistencia en la enseñanza y demostración de conceptos.
	- Programación y prácticas de laboratorio	- ABB YuMi utilizado en programas educativos y prácticas de laboratorio (2020, Suecia).

Tabla 2.2 Aplicación de robots colaborativos en distintos sectores, elaboración propia.

2.2 Norma ISO para Robots Colaborativos

La norma ISO/TS 15066:2016 establece los estándares y directrices para el trabajo seguro con robots colaborativos. Esta norma es de vital importancia en el campo de la robótica colaborativa, ya que proporciona pautas específicas para garantizar la seguridad de los trabajadores que interactúan con los cobots.

La norma ISO/TS 15066:2016 aborda aspectos clave relacionados con la seguridad humana al trabajar en colaboración con los robots. Establece los límites de fuerza y presión que un cobot puede ejercer sobre un ser humano, así como la velocidad y el control de movimiento que deben mantenerse para evitar lesiones. Además, proporciona directrices sobre la implementación de medidas de seguridad adicionales, como

sensores de proximidad y sistemas de parada de emergencia, para garantizar la detección temprana de situaciones potencialmente peligrosas, [10].

La aplicación de la norma ISO/TS 15066:2016 es esencial para asegurar que los cobots sean utilizados de manera segura y efectiva en entornos de trabajo colaborativos. Al seguir estas normas, se reduce significativamente el riesgo de accidentes y lesiones, creando un ambiente laboral más seguro y confiable para los trabajadores.

La norma técnica ISO/TS 15066 establece cuatro modos de servicio colaborativos que pueden ser utilizados individualmente o en combinación, dependiendo de la aplicación y el diseño del sistema robótico, [11]. Estos modos de servicio aseguran la seguridad en la interacción entre humanos y robots en entornos colaborativos:

Parada de seguridad vigilada: En este modo, el robot se detiene cuando interactúa con el operador en el espacio de colaboración. Sin embargo, este estado de detención es vigilado, lo que significa que el accionamiento del robot puede permanecer conectado. Esto permite una interacción más fluida y flexible entre el robot y el operador.

Guiado manual STOP: Este modo garantiza la seguridad en la colaboración entre humanos y robots, ya que el robot es guiado manualmente a una velocidad reducida de manera segura. Esto permite al operador tener un mayor control y precisión al interactuar con el robot, minimizando los riesgos de accidentes o lesiones.

Limitación de fuerza y potencia: Este modo se utiliza cuando se permite el contacto físico intencionado o accidental entre el sistema robótico (incluyendo la pieza de trabajo) y una persona. Para garantizar la seguridad, se limita la fuerza y la potencia del robot a niveles que no representen riesgos de lesiones. Para lograr esto, se requieren robots diseñados específicamente para este modo de servicio, y la norma ISO/TS 15066 establece límites de carga biomecánicos que no deben superarse en caso de colisión del robot con partes del cuerpo.

Monitorización de la velocidad y la distancia: Este modo se considera el futuro de la colaboración hombre-robot. Consiste en monitorear la velocidad y las trayectorias de movimiento del robot, y corregirlas según la velocidad y posición del operador dentro del

espacio de protección. Esta monitorización y corrección en tiempo real garantiza una colaboración segura y evita posibles colisiones o situaciones peligrosas.

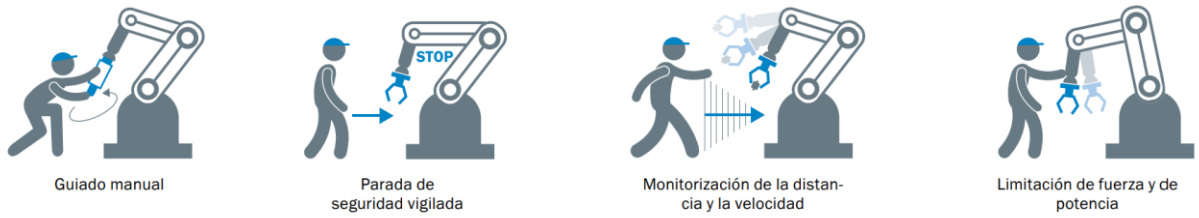


Figura 2.3 Modos de servicio colaborativos según ISO/TS 15066, [12].

2.3 Visión artificial en los Robots Colaborativos

La visión artificial ha desempeñado un papel crucial en la mejora de las capacidades de los robots colaborativos en una variedad de aplicaciones, incluida la industria logística. Esta tecnología permite a los robots percibir y comprender su entorno mediante el procesamiento de imágenes y la extracción de información visual. En el contexto de los robots colaborativos, la visión artificial se utiliza para tareas como la detección y seguimiento de objetos, la navegación autónoma, la clasificación de paquetes y la interacción segura con los humanos, [13].

La detección y seguimiento de objetos son funciones fundamentales en la visión artificial de los robots colaborativos. Los sistemas de visión basados en cámaras permiten a los robots identificar y localizar objetos en su entorno. Esto se logra mediante el análisis de características visuales, como colores, formas y texturas, utilizando algoritmos de procesamiento de imágenes. Algunas técnicas comunes utilizadas para la detección y seguimiento de objetos incluyen el filtrado de color, la segmentación de imágenes y los algoritmos de seguimiento visual basados en características.

La navegación autónoma es otro aspecto clave habilitado por la visión artificial en los robots colaborativos. Los sistemas de visión permiten a los robots construir mapas del entorno y planificar rutas seguras y eficientes para moverse dentro de él. Esto se logra utilizando técnicas de percepción visual, como la detección de obstáculos, la estimación de la profundidad y el reconocimiento de patrones de navegación. Algunos algoritmos ampliamente utilizados en la navegación autónoma de robots colaborativos incluyen el

mapeo simultáneo y la localización (SLAM, por sus siglas en inglés) y la planificación de trayectorias basada en visión, [14].

En la industria logística, la clasificación de paquetes es una tarea crítica que se beneficia enormemente de la visión artificial. Los sistemas de visión permiten a los robots colaborativos identificar y clasificar los paquetes en función de características visuales como formas, tamaños y etiquetas. Esto se logra mediante técnicas de procesamiento de imágenes, como la segmentación semántica, el reconocimiento de objetos y el aprendizaje automático. Algunos enfoques utilizados para la clasificación de paquetes en la industria logística incluyen el uso de redes neuronales convolucionales (CNN) y algoritmos de aprendizaje profundo, [15].

Es importante destacar que la visión artificial en los robots colaborativos requiere una combinación de hardware y software adecuados. Los sistemas de visión suelen incluir cámaras de alta resolución y sensores de profundidad, que capturan la información visual necesaria para las tareas de percepción. Además, se utilizan algoritmos y técnicas de procesamiento de imágenes avanzados, como la visión estéreo y la fusión sensorial, para mejorar la calidad y precisión de la información visual capturada.

2.4 Robots Móviles

Un robot móvil se define comúnmente como una máquina autónoma o semi-autónoma que tiene la capacidad de moverse en un entorno físico. Estos robots están equipados con sistemas de locomoción, que pueden incluir ruedas, orugas, patas o incluso propulsores para robots submarinos o voladores. Los robots móviles utilizan una variedad de sensores y algoritmos de navegación para percibir su entorno y tomar decisiones de movimiento, [16].

Es importante tener en cuenta que, las configuraciones cinemáticas de los robots móviles se refieren a la forma en que los mecanismos de movimiento del robot están dispuestos y cómo estos interactúan con su entorno. La elección de la configuración cinemática para un robot móvil tiene una influencia significativa en sus capacidades y aplicaciones.

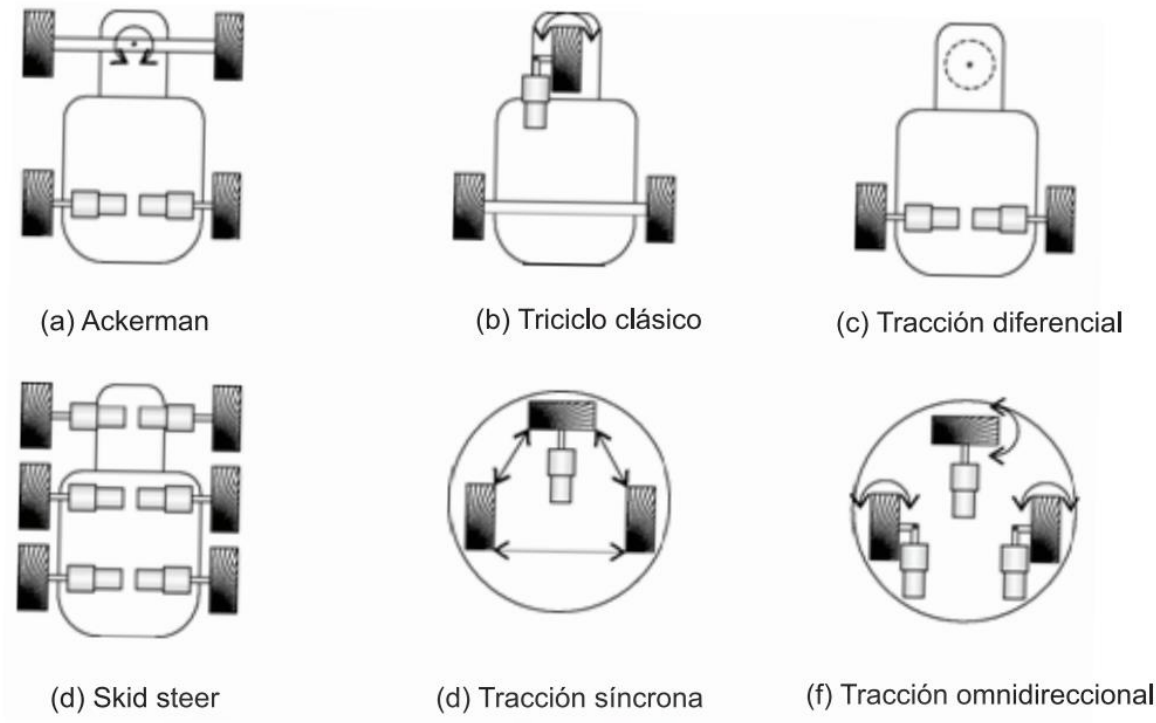


Figura 2.4 configuraciones más comunes de los robots móviles, [17].

En la siguiente tabla se presentan una comparación de diferentes configuraciones cinemáticas en robots móviles:

Nombre	Configuración Cinemática	Ventajas	Aplicaciones	Ejemplo Real
Robot Diferencial	Dos ruedas motrices en el mismo eje que se pueden controlar de	Versátil y capaz de manejar una variedad de terrenos.	Tareas de transporte y manipulación en diversos entornos.	Clearpath Jackal UGV

	manera independiente.			
Robot de Dirección Carro (Ackermann)	Ruedas delanteras pueden girar mientras que las ruedas traseras son fijas.	Permite curvas suaves, diseño comúnmente utilizado en vehículos.	Ideal para tareas de transporte en almacenes grandes con pasillos amplios.	Segway RMP 440 SE
Robot Triciclo	Dos ruedas son fijas y la tercera rueda puede girar para cambiar la dirección.	Simple y estable.	Utilizado en tareas de transporte y monitoreo.	TurtleBot 3 Burger
Robot Omnidireccional	Puede moverse en cualquier dirección sin tener que girar primero.	Alta maniobrabilidad, útil en espacios reducidos.	Ideal en entornos con espacio limitado y muchos obstáculos.	KUKA youBot

Robot Skid Steer	Utilizan un método de control de la dirección en el que las ruedas de un lado del vehículo se mueven a una velocidad diferente que las del otro lado para permitir que el robot gire.	Tienen la capacidad de girar en su propio eje, lo que proporciona una gran maniobrabilidad. Además, su diseño es más simple y compacto que el de otros tipos de robots móviles.	Este tipo de robot es comúnmente usado en tareas de inspección, exploración, y operaciones de búsqueda y rescate donde la habilidad de maniobrar en espacios estrechos es crucial.	El robot móvil Clearpath Jackal.
Robot Tracción Sincrónica	Utilizan una serie de ruedas que se mueven a la misma velocidad y en la misma dirección para proporcionar movimiento.	Proporcionan una conducción suave y estable, que es ideal para aplicaciones donde el robot necesita transportar cargas delicadas o pesadas.	Son comúnmente utilizados en aplicaciones de logística y almacenamiento para el transporte de mercancías.	El robot móvil MiR500 de Mobile Industrial Robots.

Tabla 2.3 comparación de diferentes configuraciones cinemáticas en robots móviles, elaboración propia.

Los robots móviles son una amplia categoría de robots que abarca diferentes tipos basados en su nivel de autonomía, funcionalidad y aplicación, tales como:

Vehículos Guiados Automáticamente (AGV): Los AGV son robots móviles que siguen una ruta predefinida o guiada, a menudo a través de cables o marcas en el suelo o mediante sistemas de navegación láser. Se utilizan para el transporte de materiales en entornos industriales, como almacenes o fábricas. Los AGV tienen una capacidad limitada para adaptarse a los cambios en el entorno; si se encuentra un obstáculo en su camino, el AGV se detendrá hasta que se retire el obstáculo. Algunos AGV más avanzados pueden tener cierta capacidad para navegar alrededor de obstáculos, pero en general, los AGV requieren un entorno de operación bastante estructurado, [18].

Robots Móviles Autónomos (AMR): A diferencia de los AGV, los AMR tienen una mayor capacidad para navegar de forma autónoma en su entorno. Utilizan sensores y algoritmos avanzados para percibir su entorno, planificar rutas y evitar obstáculos. Esto les permite operar en entornos menos estructurados y más dinámicos que los AGV. En lugar de seguir rutas predefinidas, los AMR pueden determinar su propio camino a un destino dado. También pueden adaptarse a los cambios en el entorno, como la reubicación de objetos o la presencia de personas u otros robots. Los AMR se utilizan para una amplia variedad de tareas, incluyendo el transporte de materiales, la inspección y la limpieza, [19].

Otros tipos de robots, basados en su aplicación y funcionalidad, son los descritos en la siguiente tabla:

Tipo de Robot	Funcionalidad y Diferencia a otros	Aplicación	Ejemplo Real
Robots Móviles Inteligentes (AIV)	Estos robots utilizan inteligencia artificial para tomar decisiones autónomas	Se utilizan en una amplia variedad de tareas, desde la	Aethon Tug

	y aprender de sus experiencias. Se diferencian de otros tipos de robots móviles en su capacidad para aprender y adaptarse a su entorno.	entrega de productos hasta la limpieza de entornos.	
Vehículos Guiados por Visión (VGV)	Utilizan cámaras y software de visión para navegar. Se diferencian de otros robots móviles en que utilizan la visión como su principal medio de navegación.	A menudo se utilizan en tareas de inspección y mapeo, así como en aplicaciones de almacén y logística.	Seegrid VGVs
Vehículo Autónomo (SDV)	Este tipo de vehículo puede navegar sin la ayuda de un conductor humano. Se diferencian de otros robots móviles en que están diseñados específicamente para el transporte de pasajeros o bienes en carreteras públicas.	Principalmente se utilizan en el transporte de personas y mercancías.	Tesla Autopilot
Robot de Entrega Autónomo (ADV)	Este robot puede llevar a cabo la entrega de productos de forma autónoma. Se diferencian de otros robots móviles en que su principal tarea es la entrega de productos.	Se utilizan para la entrega de alimentos, medicamentos y otros productos.	Starship Technologies

Vehículos Terrestres No Tripulados (UGV)	Estos vehículos pueden operar en la superficie terrestre sin un operador humano a bordo. Se diferencian de otros robots móviles en que suelen estar diseñados para operar en condiciones difíciles o peligrosas.	A menudo se utilizan en aplicaciones militares, de seguridad y de búsqueda y rescate.	QinetiQ MAARS
Robot Móvil Autónomo Inteligente (SAMR)	Estos robots utilizan la inteligencia artificial para adaptarse a su entorno y realizar tareas complejas. Se diferencian de otros robots móviles en su alto nivel de autonomía y su capacidad para realizar tareas complejas.	Se utilizan en una variedad de tareas, desde la manipulación de objetos hasta la realización de tareas de inspección.	Boston Dynamics' Spot
Máquinas de Datos Autónomas (ADM)	Estos robots pueden recopilar y analizar datos de forma autónoma. Se diferencian de otros robots móviles en que su principal tarea es la recopilación y análisis de datos.	A menudo se utilizan en tareas de inspección y monitoreo, así como en la investigación científica.	SeaBED AUV

Solución Robótica de Manipulación de Materiales (RMHS)	Estos robots están diseñados para manipular y mover materiales. Se diferencian de otros robots móviles en que su principal tarea es la manipulación de materiales.	Se utilizan en entornos industriales para tareas como la clasificación, el embalaje y la paletización.	Fetch Robotics Freight500

Tabla 2.4 Otros tipos de robots, basados en su aplicación y funcionalidad, elaboración propia.

En este trabajo de fin de máster, el tipo de robot usado es el siguiente:

Robots Móviles con Brazo Manipulador: Son plataformas móviles conocidos como Manipuladores Móviles que combinan las capacidades de locomoción de los robots móviles con la habilidad de manipular objetos proporcionada por un brazo robótico. El brazo robótico puede realizar tareas como recoger y colocar objetos, montar piezas o incluso realizar operaciones más delicadas como la soldadura. La combinación de estas capacidades permite a estos robots llevar a cabo tareas complejas en entornos dinámicos, [20].

Los Robots Móviles con Brazo Manipulador, también conocidos como Manipuladores Móviles, se componen de varios sistemas principales que trabajan en conjunto para permitir la movilidad y la manipulación de objetos. A continuación, se presenta una descripción general de estos sistemas principales:

Plataforma Móvil: Proporciona la base móvil del robot y permite su movimiento a través de diferentes entornos. Los métodos comunes de locomoción incluyen ruedas y orugas. Este bloque también incorpora los sistemas de navegación y sensores utilizados para la orientación y la detección de obstáculos.

Brazo Robótico: Este es el componente que permite al robot interactuar con su entorno y manipular objetos. Puede constar de varios grados de libertad (DOF), lo que permite movimientos en múltiples direcciones y rotaciones.

End-Effector o Herramienta Final: Este es el dispositivo al final del brazo robótico que interactúa directamente con el entorno. Los end-effectors pueden ser garras, pinzas, herramientas de soldadura, cámaras, sensores, entre otros, dependiendo de la tarea específica del robot.

Sistema de Control: Este es el cerebro del robot que coordina todos los demás sistemas. Toma entradas de los sensores, las procesa y genera las acciones apropiadas para los actuadores del robot.

Fuente de Energía: Esta puede ser una batería recargable, un cable de alimentación o cualquier otra fuente de energía que proporcione la energía necesaria para el funcionamiento del robot.

Sensores: Estos dispositivos recogen información del entorno. Los sensores pueden incluir cámaras, LIDAR, sensores de proximidad, giroscopios, acelerómetros, entre otros.

Línea de tiempo Robot Manipulador Móvil:

Año	Robot	Empresa / Institución de Investigación
1996	Hilare 2bis	LAAS-CNRS, Francia
2000	Jaume	Laboratorio de Inteligencia Robótica, Universidad Jaume I, España.
2004	FAuStO	Universidad de Verona, Italia
2006	Neobotix MM-500	Neobotix GmbH, Alemania
2009	Pequeño ayudante	Departamento de Producción, Universidad de Aalborg, Dinamarca
2012	G-WAM	Robotnik Automation & Barrett Technologies, España y Estados Unidos
2013	UBR-1	Robótica sin límites, Estados Unidos
2013	X-WAM	Robotnik Automation & Barrett Technologies, España y Estados Unidos

2015	CARLOS	AIMEN, España
2015	RB-1	Robotnik Automation & Kinova Robotics, España y Canadá

Tabla 2.5 Línea de tiempo Robot Manipulador Móvil, [20].

Ventajas y desventajas de los Robots Móviles con Brazo Manipulador:

Los Robots Móviles con Brazo Manipulador, o Manipuladores Móviles, aportan un conjunto único de ventajas, pero también presentan algunos desafíos.

Ventajas

Flexibilidad: Estos robots pueden navegar a través de un entorno dinámico, evitar obstáculos y realizar tareas de manipulación. Esto les permite operar en entornos más complejos y cambiantes que los robots estacionarios o los robots móviles sin capacidades de manipulación.

Incremento de productividad: Al ser capaces de moverse y realizar tareas de manipulación, pueden aumentar la eficiencia y la productividad en muchas aplicaciones industriales. Pueden trabajar de forma continua durante largos periodos de tiempo sin descansos, y no se cansan ni se aburren.

Reducción del riesgo de lesiones: Pueden realizar tareas peligrosas, reduciendo así el riesgo de lesiones para los trabajadores humanos.

Automatización de tareas complejas: Con la combinación de movilidad y manipulación, estos robots pueden realizar tareas que anteriormente requerían la intervención humana, como la recogida y colocación de objetos en estantes en un almacén.

Desventajas

Costo: Estos robots pueden ser costosos de comprar y mantener. También puede ser necesario realizar una formación costosa para el personal que va a trabajar con el robot.

Tecnología compleja: La combinación de movilidad y manipulación implica una mayor complejidad tecnológica. Esto puede suponer desafíos en términos de programación, mantenimiento y resolución de problemas.

Limitaciones en el manejo de objetos: Aunque los brazos robóticos pueden manejar una amplia variedad de objetos, pueden tener dificultades con objetos que son especialmente pesados, delicados o de formas inusuales.

Necesidad de un entorno adecuado: Aunque estos robots pueden navegar de forma autónoma, todavía pueden requerir ciertos ajustes en el entorno para operar de manera óptima, como suficiente espacio para moverse, estabilidad del suelo, entre otros.

Interacción con humanos: Si bien estos robots están diseñados para trabajar en entornos donde también hay humanos presentes, la interacción segura sigue siendo un área de investigación activa.

Aplicaciones:

Los Robots Móviles con Brazo Manipulador tienen una amplia gama de aplicaciones potenciales, muchas de las cuales se encuentran en entornos industriales, aunque también se están explorando usos en otros campos.

Almacenamiento y logística: Los robots móviles con brazos manipuladores pueden moverse de forma autónoma por un almacén, seleccionar y recoger productos y luego llevarlos a una ubicación designada para el empaquetado o el envío. Estos robots pueden funcionar de manera más eficiente que los humanos en tareas de repetición y pueden trabajar durante largos periodos de tiempo sin descansos.

Un ejemplo de esta aplicación es el siguiente:

Figura 2.5 Robot móvil RB-Kairos+, Robotnik Automation S.L.

El RB-Kairos+ de Robotnik es un robot móvil diseñado para una fácil integración con los brazos robóticos de la serie e-Series de Universal Robots. Tanto el software como el hardware del RB-Kairos+ están plenamente equipados para acoplar el brazo robótico, transformándolo así en un manipulador móvil. Esta configuración amplía de manera significativa el área de trabajo del cobot, ya que puede moverse y operar en diferentes ubicaciones. Por lo tanto, ofrece un valor añadido significativo para los usuarios actuales de los brazos robóticos URe, extendiendo su funcionalidad más allá de su posición estática original, [21].

Investigación y desarrollo: Los brazos manipuladores montados en robots móviles pueden utilizarse en laboratorios para la automatización de experimentos. Pueden moverse de una estación de trabajo a otra, realizando tareas como la manipulación de muestras, la operación de equipos de laboratorio y la recogida de datos.

Un ejemplo de esta aplicación es el siguiente:

Figura 2.6 Manipulador Móvil XL-GEN, Robotnik Automation S.L.

El manipulador móvil XL-GEN, destinado a investigación y desarrollo, es altamente personalizable, pudiendo operar sin brazo, con un brazo preparado para Kinova, o con un brazo de 6 o 7 grados de libertad que puede equiparse con una pinza de 2 o 3 dedos. A pesar del bajo consumo energético del brazo, el robot puede funcionar hasta por 10 horas. Utiliza una arquitectura de control modular y abierta basada en ROS, compatible con varios sensores como láseres y cámaras. Además, tiene dos posibles configuraciones cinemáticas: omnidireccional con ruedas mecanum, y una configuración de dirección deslizante para interiores y exteriores mediante la sustitución de las ruedas. El XL-GEN se ha diseñado para varias aplicaciones de investigación en interiores, incluyendo la inspección, la medicina y la logística, [22].

Cuidado de la salud: Los robots móviles con brazos manipuladores pueden utilizarse en entornos de atención sanitaria para ayudar con tareas como la limpieza, la distribución de alimentos y medicamentos, y posiblemente incluso procedimientos médicos. Por ejemplo, algunos hospitales ya están utilizando robots para desinfectar habitaciones con luz UV.

Un ejemplo de esta aplicación es el siguiente:

Figura 2.7 El robot Cobi que aplica vacunas sin agujas, Cobionix.

Cobi, desarrollado por una empresa emergente canadiense, es el primer robot autónomo que administra vacunas de manera indolora y sin agujas. Durante la pandemia, los robots han demostrado su utilidad en hospitales para desinfectar, entregar medicamentos y proporcionar compañía a los pacientes. Ahora, Cobi puede abordar desafíos como la vacunación, donde la demanda a menudo supera la capacidad del personal médico. Cobi, capaz de administrar inyecciones intramusculares, cuenta con un brazo robótico equipado con un almacén de viales y una pantalla táctil para interactuar con los pacientes. Este robot puede registrar a los pacientes para recibir su vacuna a través de su panel táctil, facilitando así la distribución de vacunas en zonas con acceso limitado a la atención médica, [23] .

Agricultura: En la agricultura, estos robots pueden ser utilizados para tareas de recolección, plantación y riego. Pueden moverse a través de los campos, identificar plantas individuales, y realizar tareas de manipulación como la recolección de frutas y verduras.

Un ejemplo de esta aplicación es el siguiente:

Figura 2.8 Robot Móvil RB-VOGUI+ DUAL con dos brazos robóticos UR integrados.

El proyecto europeo Bacchus demostró el potencial de los robots en la agricultura, presentando un robot capaz de desplazarse autónomamente en campos de cultivo, detectar y recolectar variedades específicas de uva con precisión y delicadeza. Bacchus, que integra el robot móvil RB-VOGUI+ DUAL de Robotnik con dos brazos robóticos UR, utiliza un sistema de sensores para recopilar datos del entorno, inspeccionar cultivos, y, con la ayuda de pinzas adaptativas fabricadas con técnicas de impresión 3D, recolectar frutas sin dañarlas. Según Ángel Soriano, director del proyecto, este sistema imita la mecánica humana para cosechar, identificando el estado del cultivo y tomando decisiones óptimas de acción. Este desarrollo supone un gran avance en productividad, especialización y sostenibilidad ambiental, revolucionando la agricultura selectiva de alta precisión mediante la introducción de la manipulación móvil autónoma, [24].

Construcción: Los robots móviles con brazos manipuladores pueden utilizarse en sitios de construcción para realizar tareas como el transporte de materiales, la perforación, el atornillado y otras tareas de construcción.

Un ejemplo de esta aplicación es el siguiente:

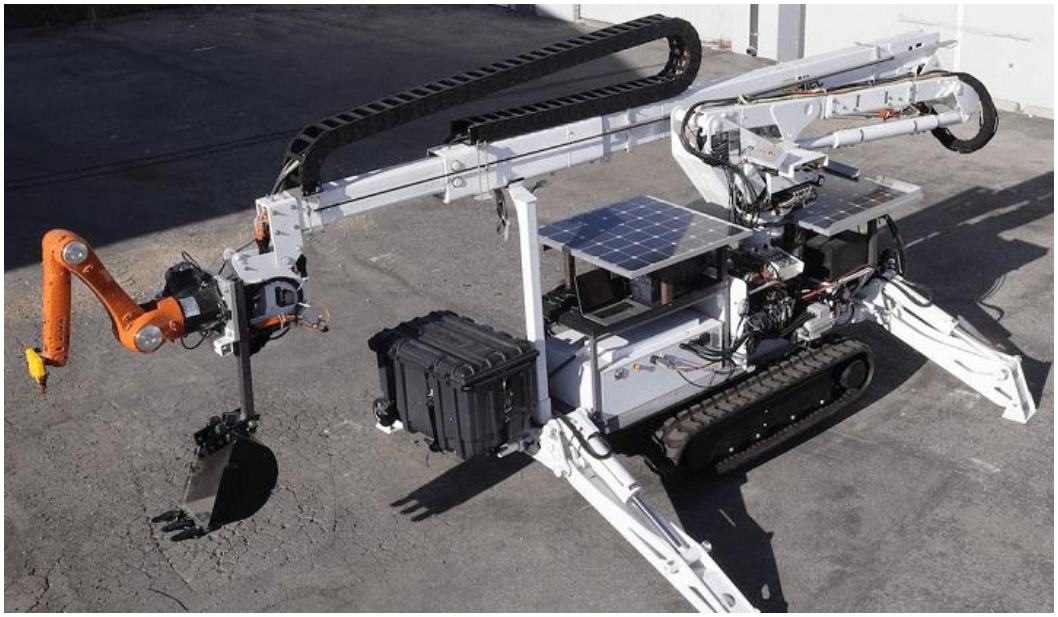


Figura 2.9 Método de impresión 3D para la construcción de grandes estructuras.

El laboratorio Mediated Matter del Instituto de Tecnología de Massachusetts (MIT) desarrolló en el 2019 una técnica de impresión 3D para edificar grandes estructuras, según un artículo publicado en Science Robotics. La metodología emplea un brazo robótico, ubicado en un vehículo, que dirige una boquilla que expulsa espuma expansiva. Con sucesivas aplicaciones, se forman capas de espuma que se endurecen al expandirse, permitiendo que el espacio entre dos muros paralelos se llene con concreto. La investigación evidenció la impresión 3D de una porción de cúpula hemielipsoidal a escala arquitectónica, utilizando la espuma aislante Froth-Pak de Dow Chemical por su tasa de curado y consistencia en la deposición de capas. El sistema de impresión, en su segunda versión, incluye un sistema de elevación aérea AT40GW de Altec 2015 y un brazo robótico eléctrico KUKA AGILUS. Se demostró su eficacia al imprimir una cúpula abierta de 14,6 m de diámetro y 3,7 m de altura en menos de 13,5 horas, [25].

Normas y Certificaciones Esenciales para Robots Móviles con Brazos Manipuladores Colaborativos:

En el mundo de la robótica, existen varias normas ISO y otras regulaciones para garantizar la seguridad y el rendimiento eficaz de los sistemas robóticos, incluyendo robots móviles con brazos manipuladores colaborativos.

Norma	Descripción	Enlace
ISO 10218	Proporciona las directrices y requisitos necesarios para garantizar la seguridad durante los procesos de producción, integración, implementación y mantenimiento de robots y sistemas robóticos industriales.	https://www.iso.org/standard/51330.html
ISO 13482	Establece los requisitos y pautas para el diseño seguro, medidas de protección y uso de robots de cuidado personal, que incluyen robots móviles sirvientes, robots asistentes físicos y robots portadores de personas.	https://www.iso.org/standard/53820.html
ISO/TS 15066	Proporciona información sobre la seguridad de los robots colaborativos, incluyendo una guía para la evaluación de riesgos y la implementación de sistemas de seguridad para robots que trabajan en proximidad o junto a humanos.	https://www.iso.org/standard/62996.html
ANSI/RIA R15.06	Es la norma estadounidense que establece los criterios de seguridad para	https://webstore.ansi.org/standards/ria/ansiriar15062012

	robots industriales y sistemas robóticos, y está armonizada con la ISO 10218.	
EN ISO 13849	Esta norma europea aborda aspectos de seguridad relacionados con el control de sistemas de maquinaria, incluyendo robots.	https://www.tuvsud.com/es-mx/industrias/manufactura/componentes-y-equipos/seguridad-funcional-de-maquinaria-industrial
IEC 62061	Proporciona pautas para el diseño y la integración de sistemas de seguridad relacionados con la maquinaria, incluyendo robots industriales.	https://www.tecnical.cat/apunts-tecnics/cas-seguridad-en-maquinas-apuntes-tecnicos-tecnical-manresa-igualada.pdf

Tabla 2.6 Normas y Certificaciones Esenciales para Robots Móviles con Brazos Manipuladores, elaboración propia.

2.5 Marcas ArUco como referencia visual

Las marcas ArUco son un tipo de referencia visual ampliamente utilizada en la robótica y la visión por computadora. Estas marcas consisten en patrones de cuadrados negros y blancos que se colocan en objetos o superficies para permitir a los robots o sistemas de visión por computadora detectar, reconocer y seguir estas marcas, [26].

El funcionamiento de las marcas ArUco se basa en algoritmos de detección y seguimiento de características visuales. Estos algoritmos analizan las imágenes capturadas por las cámaras o sensores de visión del robot y buscan los patrones de cuadrados negros y blancos característicos de las marcas ArUco.

Estos patrones se definen como una matriz de cuadrados de 7x7. Cada uno de los cuadrados es negro o blanco, pero las dos filas y columnas exteriores son negras. El sistema cuenta con algoritmos implementados en funciones para el reconocimiento de estos patrones. Gracias al uso de la teoría de codificación digital, el sistema logra un bajo nivel de confusión o detección falsa de marcadores. Para una detección exitosa del

marcador, el sistema asume que se conocen el tamaño del marcador y los parámetros de calibración de la cámara, [27].

Figura 2.10 Patrones de marcadores artificiales ArUco.

Una vez que se detectan las marcas ArUco en una imagen, se utilizan algoritmos de reconocimiento para identificar de manera única cada marca y asociarla con una posición en el espacio tridimensional. Esto permite al robot conocer la ubicación y orientación de las marcas en relación con su propio sistema de coordenadas.

Las marcas ArUco se utilizan para una variedad de aplicaciones en la robótica y la visión por computadora. Algunos ejemplos incluyen la navegación autónoma de robots, la calibración de cámaras, la interacción hombre-robot y la localización y seguimiento de objetos.

2.6 Métodos de clasificación de paquetes en centros logísticos

Los métodos de clasificación de paquetes en centros logísticos utilizando robótica se han convertido en una solución eficiente y precisa para optimizar los procesos de clasificación y distribución de paquetes. Estos métodos emplean diversas técnicas y algoritmos para identificar y categorizar los paquetes según sus características, como forma, tamaño, peso o etiquetas, [28].

Algunos de los métodos más comunes utilizados en la clasificación de paquetes en centros logísticos mediante robótica son:

Visión por computadora: La visión por computadora desempeña un papel fundamental en la clasificación de paquetes. Se utilizan cámaras y sensores para capturar imágenes de los paquetes y luego se aplican algoritmos de procesamiento de imágenes para

extraer características relevantes. Estas características se utilizan para determinar la clase o categoría a la que pertenece cada paquete.

Aprendizaje automático: Los algoritmos de aprendizaje automático, como las redes neuronales, se utilizan para entrenar modelos capaces de reconocer patrones y características en las imágenes de los paquetes. Estos modelos pueden aprender a clasificar los paquetes en diferentes categorías basándose en ejemplos previamente etiquetados. Con el tiempo, el modelo se vuelve más preciso y eficiente en la clasificación.

Sensores de peso y dimensiones: Los robots utilizados en los centros logísticos a menudo están equipados con sensores de peso y dimensiones. Estos sensores permiten medir con precisión el peso y tamaño de los paquetes, lo que ayuda en la clasificación y distribución adecuada. Los paquetes se pueden dirigir a diferentes áreas o cajas según su peso y tamaño.

Sistemas de etiquetado y lectura de códigos de barras: Los paquetes suelen tener etiquetas o códigos de barras que contienen información sobre su destino o características. Los robots pueden estar equipados con sistemas de lectura de códigos de barras para identificar y clasificar automáticamente los paquetes en función de esta información. Esto agiliza el proceso de clasificación y minimiza los errores humanos.

Planificación de rutas y brazos robóticos: Los brazos robóticos se utilizan para manipular y mover los paquetes dentro del centro logístico. Se pueden implementar algoritmos de planificación de rutas para optimizar la secuencia de movimientos de los brazos robóticos, minimizando los tiempos de desplazamiento y maximizando la eficiencia en la clasificación.

Integración con sistemas de gestión: Los sistemas de clasificación de paquetes en centros logísticos suelen estar integrados con sistemas de gestión de almacenes y sistemas de seguimiento y trazabilidad. Esta integración permite un flujo de información y coordinación eficiente entre los diferentes procesos logísticos, asegurando una clasificación precisa y una distribución adecuada de los paquetes.

La siguiente tabla recoge a modo de ejemplo empresas que trabajan con robots para la clasificación de paquetes en centros logísticos:

Empresa	Uso	Robot	Tipo	Año de Integración
Amazon	Utiliza robots colaborativos para la clasificación eficiente de paquetes.	Robot Kiva	Colaborativo	2012
FedEx	Implementa robots autónomos para mover y clasificar paquetes en centros logísticos.	Robot Sawyer	Colaborativo	2016
DHL	Utiliza robots robóticos para agilizar el proceso de clasificación y distribución de paquetes.	Robot EffiBOT	Colaborativo	2019
JD.com	Desarrolla sistemas robóticos para reconocimiento y clasificación automatizada de paquetes.	Robot AGV	Industrial	2015
Alibaba Group	Emplea robots colaborativos para la clasificación y transporte de paquetes en centros logísticos.	Robot Hikvision	Colaborativo	2017
UPS	Implementa sistemas automatizados de clasificación de paquetes utilizando robots colaborativos.	Robot UR10e	Colaborativo	2018

Tabla 2.7 Empresas y robots utilizados en la clasificación de paquetes en centros logísticos, elaboración propia.

3. METODOLOGÍA

3.1 Introducción al simulador de robótica CoppeliaSim (anteriormente V-REP)

CoppeliaSim (anteriormente conocido como V-REP, Virtual Robot Experimentation Platform) es un simulador de robótica ampliamente utilizado en la investigación, desarrollo y validación de sistemas robóticos. Proporciona un entorno virtual altamente flexible y realista para modelar y simular robots, ambientes, sensores y objetos físicos.

CoppeliaSim se utiliza con el propósito de diseñar, probar y validar algoritmos de control, estrategias de movimiento, sistemas de percepción y planificación de robots. Permite a los investigadores y desarrolladores crear escenas complejas y realistas, replicando situaciones reales en un entorno virtual controlado.

Este simulador se utiliza en diversas áreas de la robótica, incluyendo la robótica industrial, la robótica móvil, la robótica colaborativa y la robótica de servicio. Es especialmente útil para la simulación de sistemas complejos que involucran múltiples robots interaccionando entre sí y con su entorno.

CoppeliaSim ofrece una amplia gama de funcionalidades y características técnicas avanzadas. Permite la programación de robots virtuales utilizando diferentes lenguajes de programación, como C/C++, Python y MATLAB, lo que brinda flexibilidad a los investigadores para implementar y probar algoritmos personalizados. Además, proporciona una interfaz gráfica intuitiva que facilita la creación y configuración de escenas robóticas complejas, y cuenta con un motor de física que simula de manera precisa las interacciones entre los objetos en el entorno simulado.

En comparación con otros simuladores, como AirSim y Webots, CoppeliaSim se destaca como el simulador óptimo para este TFM. Sobresale particularmente por su ligereza, la simulación realista de sensores y su capacidad para satisfacer adecuadamente los requisitos tanto de los robots como del entorno de trabajo desarrollado a lo largo de este trabajo.

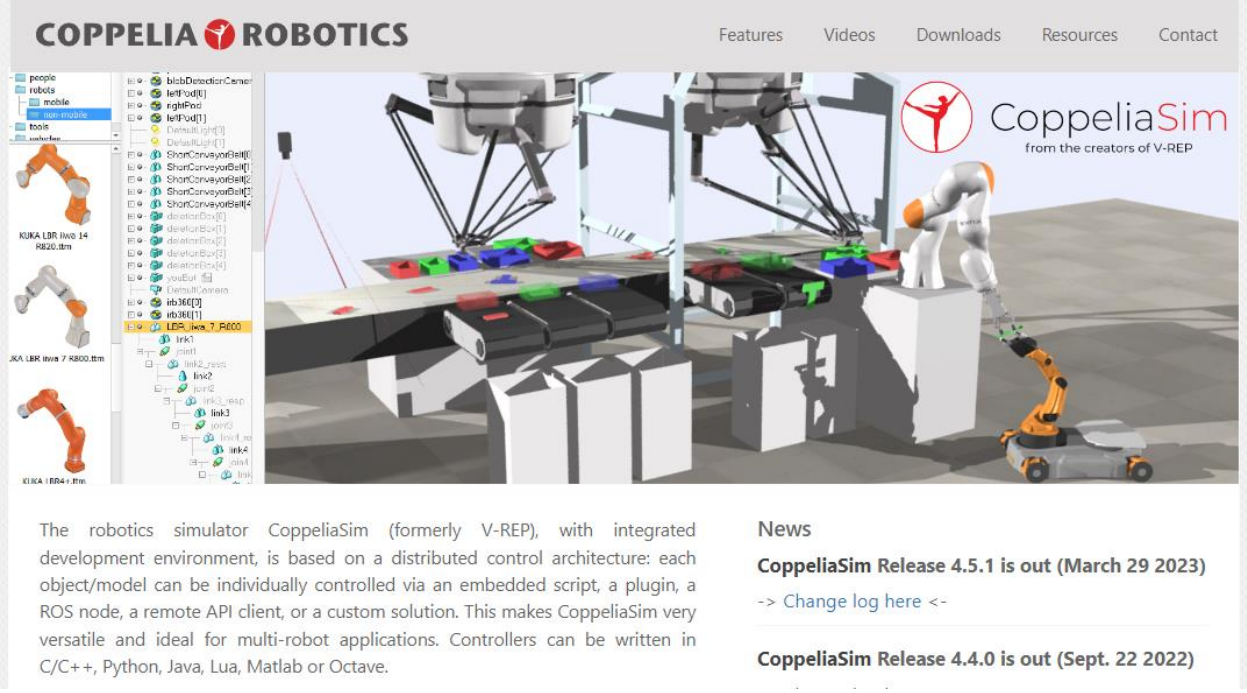


Figura 3.1 The robotics simulator CoppeliaSim, [29].

Además, CoppeliaSim ha sido desarrollado y refinado a lo largo de los años por un equipo de expertos en robótica. Su trayectoria y reconocimiento en la comunidad científica y académica respaldan su idoneidad como herramienta para este proyecto de investigación. La evolución continua de CoppeliaSim ha permitido la integración de tecnologías emergentes, como la visión artificial y los marcadores ArUco, que son componentes clave en el sistema propuesto en este TFM.

3.2 Creación de una nueva estación de trabajo en CoppeliaSim

El simulador CoppeliaSim se descarga según la versión y sistema operativo deseado o requerido desde la sección de descargas, ubicada en el menú principal, del sitio web oficial de CoppeliaSim en https://www.coppeliarobotics.com/coppeliaSim.

Una vez que la instalación esté completa y se haya iniciado correctamente el programa CoppeliaSim, se mostrará la interfaz principal:

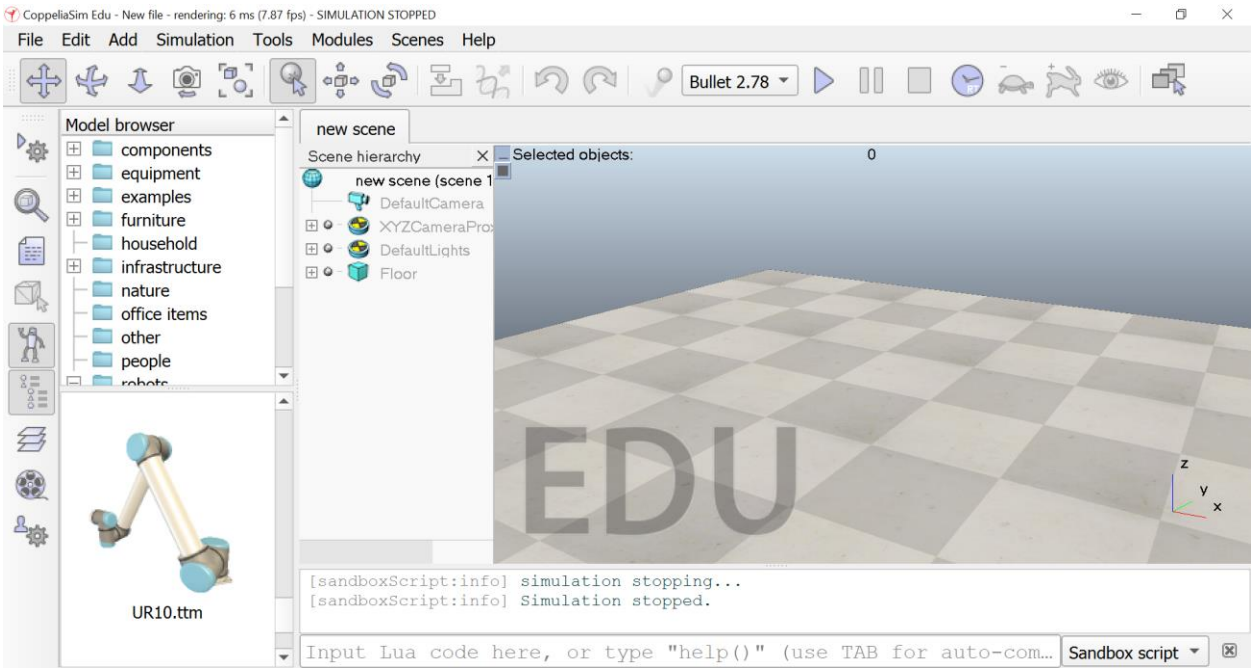


Figura 3.2 Interfaz principal del simulador de robótica CoppeliaSim.

Para crear una nueva estación se hace clic en la pestaña "Archivo" logrando desplegar el menú correspondiente desde donde se selecciona la opción "Nueva Escena" que permite la creación de estaciones en CoppeliaSim:

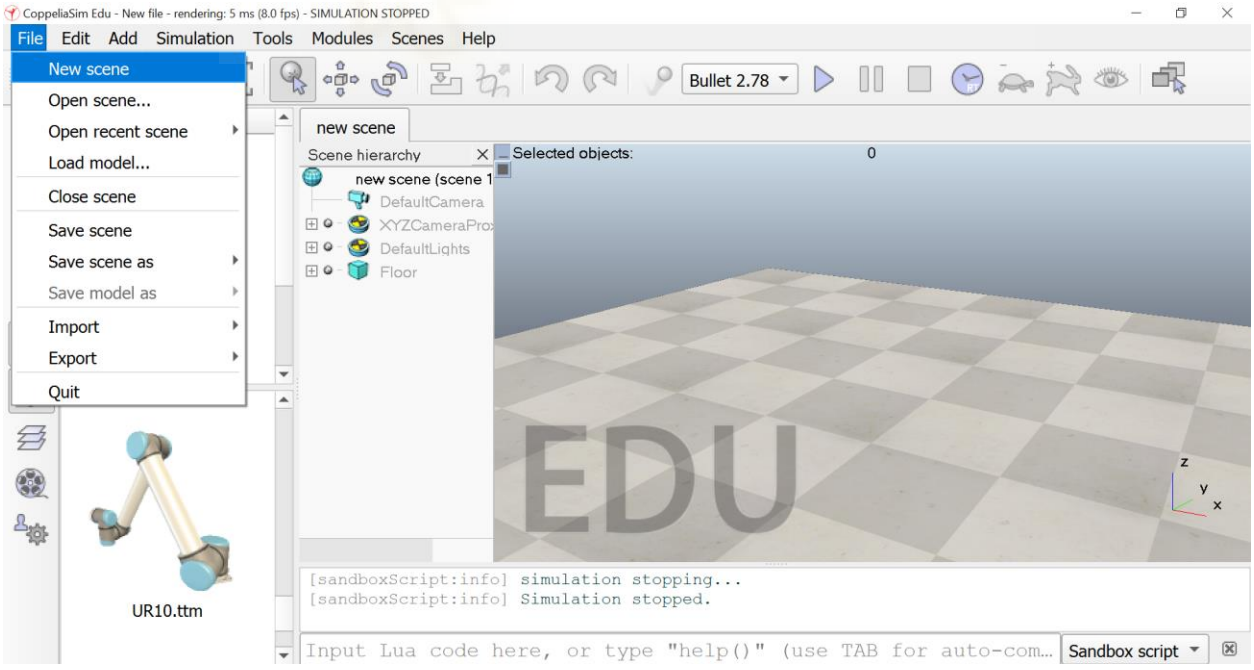


Figura 3.3 Opción "Nueva Escena" para la creación de estaciones en CoppeliaSim.

La interfaz de usuario de CoppeliaSim se presenta como un entorno gráfico intuitivo y completo que facilita la interacción con el simulador. Está diseñada para brindar una experiencia de usuario fluida y eficiente durante el desarrollo y la simulación de robots y sistemas robóticos. La interfaz de usuario cuenta con diversas ventanas y paneles que permiten acceder a las diferentes funcionalidades y herramientas de CoppeliaSim, como la creación y configuración de modelos de robots, la programación de comportamientos, la gestión de sensores y actuadores, la configuración del entorno de simulación y la visualización en tiempo real de los resultados.

A continuación, se ilustra una vista típica de la aplicación CoppeliaSim y sus elementos:

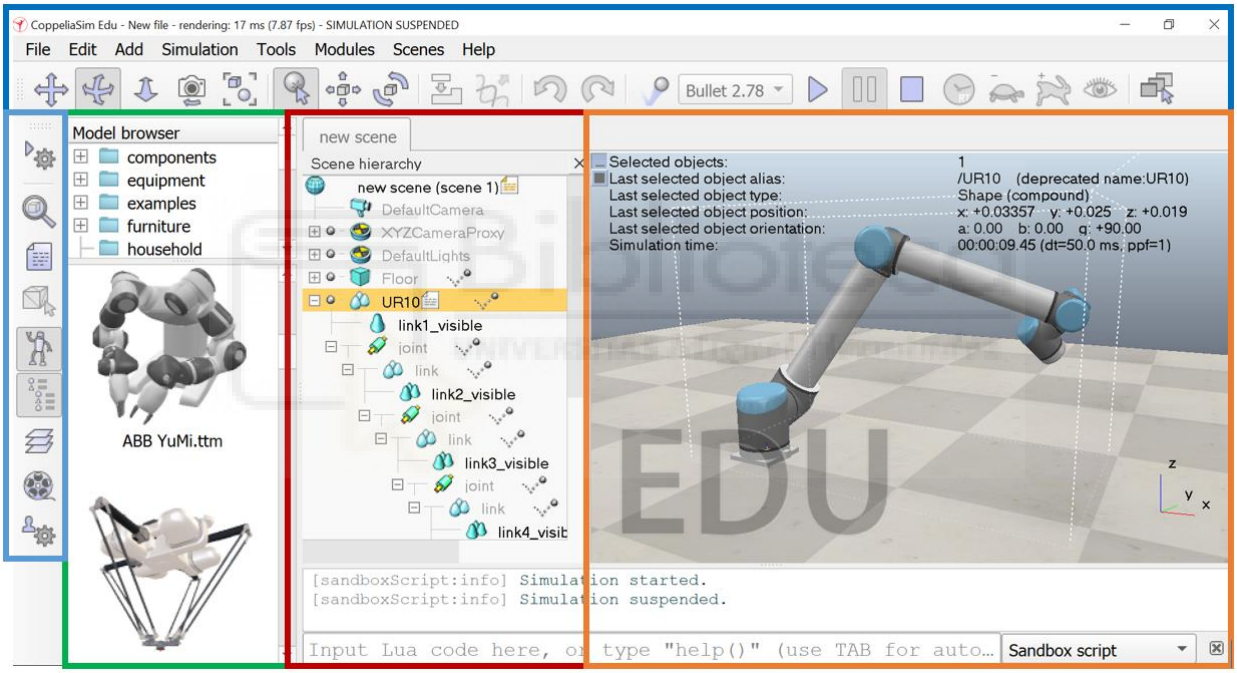


Figura 3.4 Interfaz de usuario de CoppeliaSim.

Barra de aplicaciones: La barra de aplicaciones en CoppeliaSim muestra información relevante sobre la licencia utilizada, el nombre del archivo de la escena actual, el tiempo empleado en el renderizado y el estado de simulación en el modo de edición activo. Además, esta barra permite arrastrar y soltar archivos relacionados con CoppeliaSim en la escena, como archivos de escena ".ttt" y archivos de modelo ".ttm".

Barra de menús: La barra de menús en CoppeliaSim proporciona acceso a casi todas las funcionalidades del simulador. Al hacer clic en los diferentes elementos de la barra de menús, se activarán cuadros de diálogo correspondientes. El contenido de la barra de menús se adapta al contexto y varía según el estado actual del simulador. También es posible acceder a las funciones de la barra de menús de forma alternativa a través de menús emergentes, al hacer doble clic en íconos en la vista de jerarquía de escenas o al hacer clic en botones de la barra de herramientas.

Barras de herramientas: Las barras de herramientas en CoppeliaSim presentan funciones a las que se accede con frecuencia. Estas herramientas incluyen opciones para cambiar el modo de navegación, seleccionar diferentes páginas y más. Algunas funciones se pueden acceder a través de la primera barra de herramientas, mientras que todas las funciones están disponibles en la segunda barra de herramientas. Las barras de herramientas se pueden acoplar o desacoplar según la preferencia del usuario.

Navegador de modelos: El navegador de modelos en CoppeliaSim muestra una lista de los modelos disponibles en la escena de simulación. Permite al usuario explorar y seleccionar modelos específicos para interactuar con ellos. El navegador de modelos es útil para administrar y organizar los elementos presentes en la escena, así como para acceder rápidamente a propiedades y configuraciones específicas de los modelos.

Jerarquía de escenas: La vista de jerarquía de escenas muestra la estructura de la escena de simulación en forma de árbol. Proporciona una representación visual de la jerarquía de objetos, modelos y sus relaciones en la escena. Esta vista permite al usuario examinar y organizar los elementos de la escena, así como realizar acciones como agregar, eliminar o modificar objetos en la jerarquía.

Entorno de simulación: El entorno de simulación en CoppeliaSim es la ventana principal donde se visualiza la escena y se lleva a cabo la simulación. En esta área, se muestra la representación gráfica de los objetos, modelos y entidades presentes en la escena. El entorno de simulación ofrece herramientas de navegación y manipulación para interactuar con los objetos y observar el comportamiento de la simulación en tiempo real.

3.3 Componentes y funcionalidad del sistema automatizado para la navegación y almacenamiento eficiente de paquetes en centros logísticos y su entorno de simulación

3.3.1 Plataforma móvil

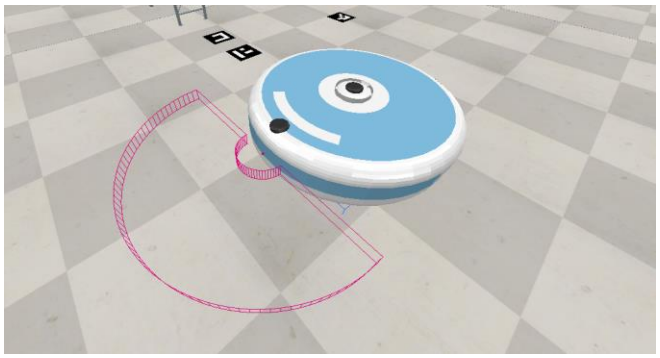


Figura 3.5 Plataforma móvil utilizada, entorno propio en CoppeliaSim.

Diseño y funcionamiento del sistema de locomoción:

La plataforma móvil es el componente encargado de la locomoción del robot y la detección de obstáculos dentro de un entorno de almacén. Está compuesta por tres ruedas, siendo las dos ruedas traseras motorizadas, mientras que la rueda delantera opera como una rueda libre. Este robot emplea un sistema de conducción diferencial, donde las dos ruedas traseras pueden ser conducidas a diferentes velocidades y direcciones, lo que permite una maniobrabilidad suave y precisa. Gracias a este sistema, el robot puede girar sobre su propio eje o seguir rutas curvas de manera efectiva.

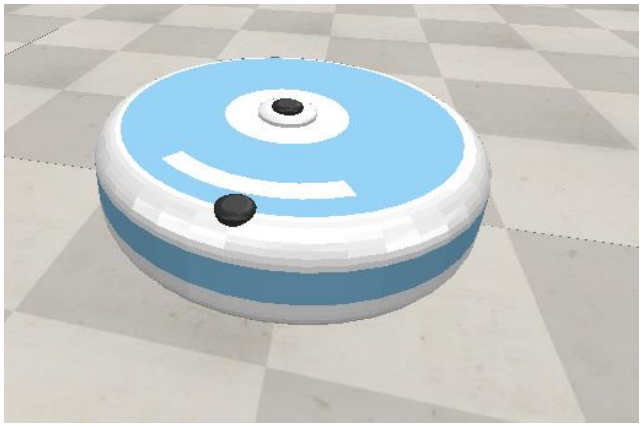


Figura 3.6 Vista superior de la plataforma móvil utilizada.

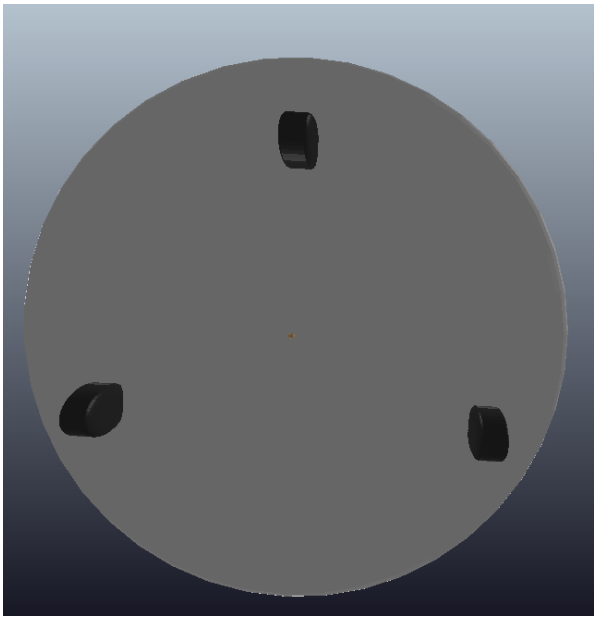


Figura 3.7 Vista superior de la plataforma móvil utilizada.

Sensor de proximidad y navegación segura:

Para garantizar una navegación segura, la plataforma móvil está equipada con un sensor de proximidad posicionado en la parte delantera. Este sensor detecta objetos en el camino del robot y activa una parada automática hasta que el obstáculo sea eliminado.

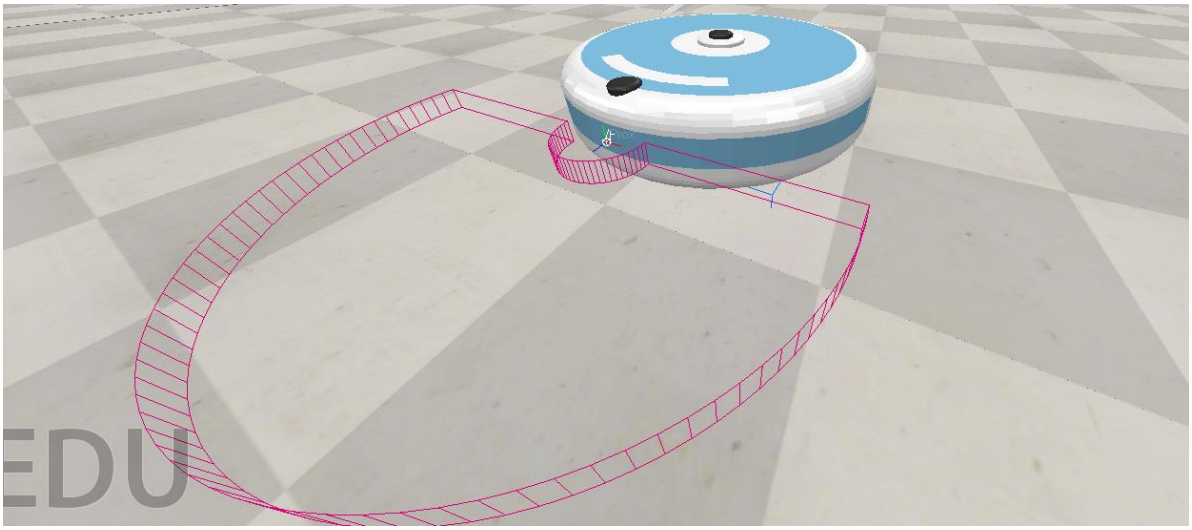


Figura 3.8 Sensor de aproximación para la navegación segura de la plataforma móvil.

Cámara de visión y detección de marcadores ArUco:

Además, se monta una cámara de visión debajo de la plataforma, entre las ruedas. Esta cámara permite al robot identificar los marcadores ArUco colocados en el suelo, ayudando en la localización y navegación dentro del entorno del almacén.

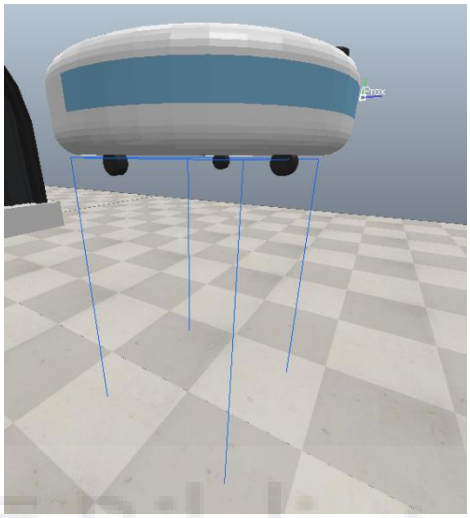


Figura 3.9 Campo de visión de la cámara ubicada entre las ruedas de la plataforma móvil.

3.3.2 Brazo robot colaborativo industrial UR10e

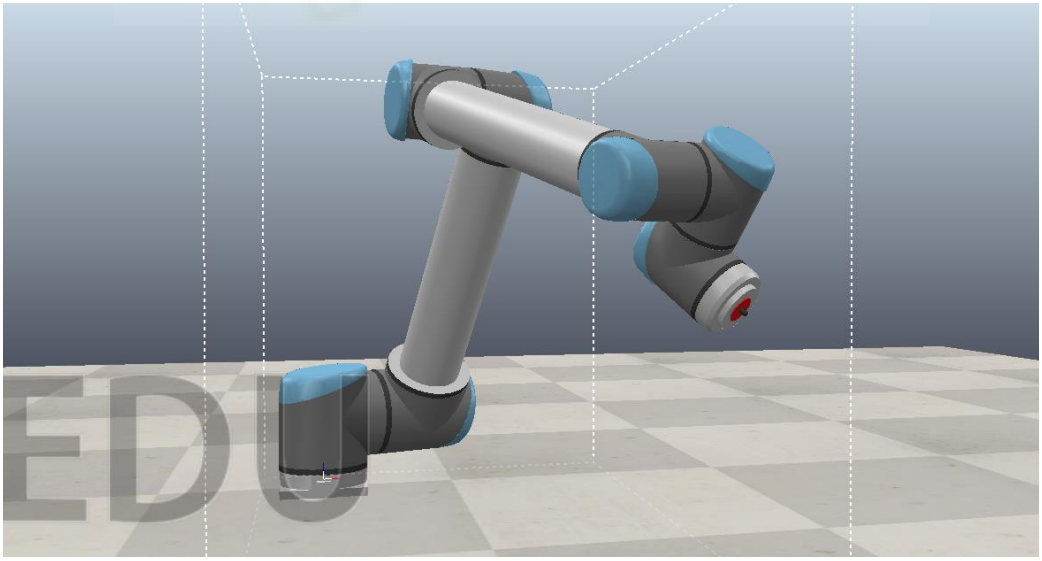


Figura 3.10 Brazo robot colaborativo UR10e de Universal Robot en entorno CoppeliaSim.

Características y capacidades del UR10e:

El brazo robótico UR10e es responsable de las tareas de manipulación y colocación de cajas. Desarrollado por Universal Robots, el UR10e es un brazo robótico versátil y altamente capaz. Ofrece un alcance de hasta 1300 mm y una capacidad de carga de 12.5 kg, lo que lo hace adecuado para manejar una amplia gama de cajas comúnmente encontradas en almacenes. El brazo está equipado con múltiples articulaciones, permitiéndole moverse con precisión y flexibilidad.

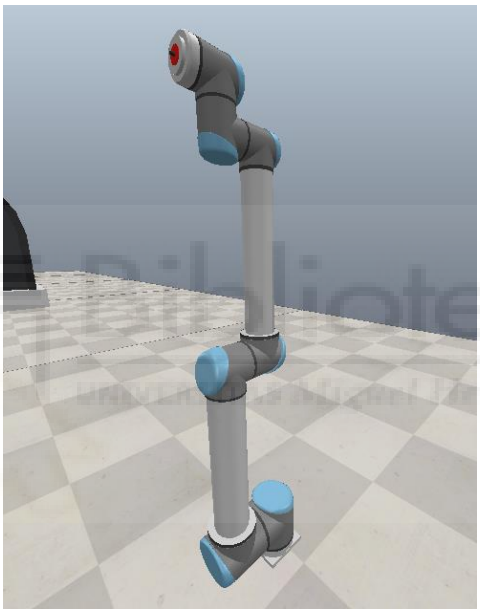


Figura 3.11 Brazo robot colaborativo UR10e en posición inicial (HOME).

Funcionalidad de la ventosa y sensor de proximidad:

En la punta del brazo robótico UR10e, se adjunta una ventosa de vacío para agarrar de forma segura las cajas durante el proceso de almacenamiento. Además, se integra un sensor de proximidad en la punta del brazo para detectar la posición de la caja, permitiendo un agarre preciso.

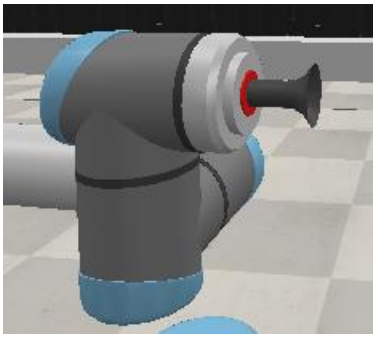


Figura 3.12 Efector final del brazo robot colaborativo UR10e, ventosa de vacío.

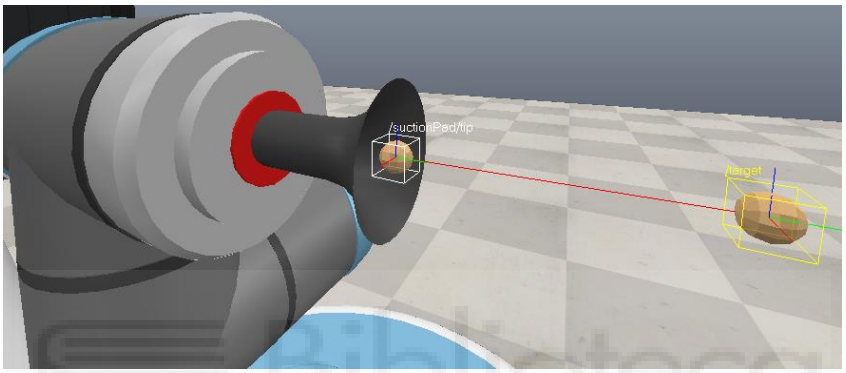


Figura 3.13 Sensor de proximidad para detectar la posición de la caja en el efector final.

3.3.3 Cámara de visión y lectura de marcadores aruco en las cajas

Además, se instala una cámara de visión en la punta del brazo, que facilita la lectura de los marcadores ArUco colocados en las cajas. Esto permite al robot identificar y diferenciar entre varias cajas, asegurando un almacenamiento preciso y eficiente.

Figura 3.14 Marcadores ArUco en las Cajas (paquetes de la simulación).

3.3.4 Mecanismos de control del robot en el entorno de simulación CoppeliaSim

En el escenario de simulación generado a través de CoppeliaSim, el brazo robótico UR10e emplea una modalidad de control basada en cinemática inversa híbrida. Este modelo de control es esencialmente un esquema de solución algorítmica que permite la manipulación precisa de las articulaciones del brazo robótico, de forma que se pueda determinar con exactitud el posicionamiento y movimiento del efector final, es decir, la punta del brazo robótico. De esta manera, en lugar de requerir una programación detallada para cada una de las articulaciones, este método solo necesita que el usuario defina la posición espacial deseada para el efector final.

La principal ventaja de este enfoque es su capacidad para simplificar significativamente el proceso de control del brazo robótico. Con la cinemática inversa híbrida, el sistema puede determinar de manera autónoma los ángulos necesarios para cada articulación a fin de alcanzar la ubicación objetivo-especificada para el efector final. Este método de control elimina la necesidad de codificar manualmente la cinemática inversa, permitiendo así una programación más intuitiva y eficiente de las tareas a realizar por el robot. Asimismo, contribuye a una mayor precisión en las operaciones del UR10e, fundamental en los procesos de manipulación y almacenamiento de paquetes dentro del entorno logístico en el que se ha implementado la solución propuesta en este Trabajo de Fin de Máster.

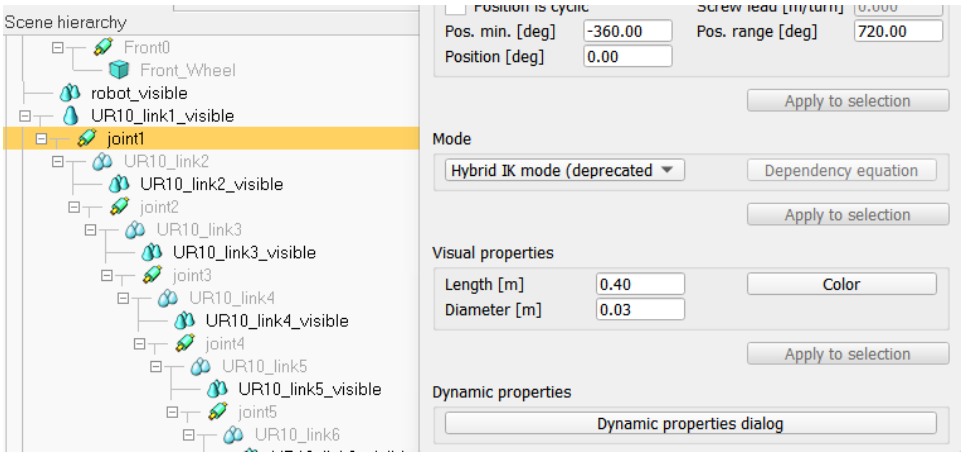


Figura 3.15 Cinemática inversa híbrida en la interfaz gráfica de usuario de CoppeliaSim.

3.3.5 Descripción del entorno de simulación CoppeliaSim

En el entorno de trabajo simulado dentro de CoppeliaSim, el robot móvil con el brazo robótico opera en presencia de dos figuras humanas caminando. Como un robot colaborativo (cobot), está diseñado para trabajar junto a los humanos garantizando su seguridad. El robot está equipado con sensores y algoritmos que le permiten detectar y evitar colisiones con humanos, manteniendo un entorno de trabajo seguro.

Figura 3.16 Escena del entorno de trabajo simulado en CoppeliaSim.

Almacenamiento y localización de cajas:

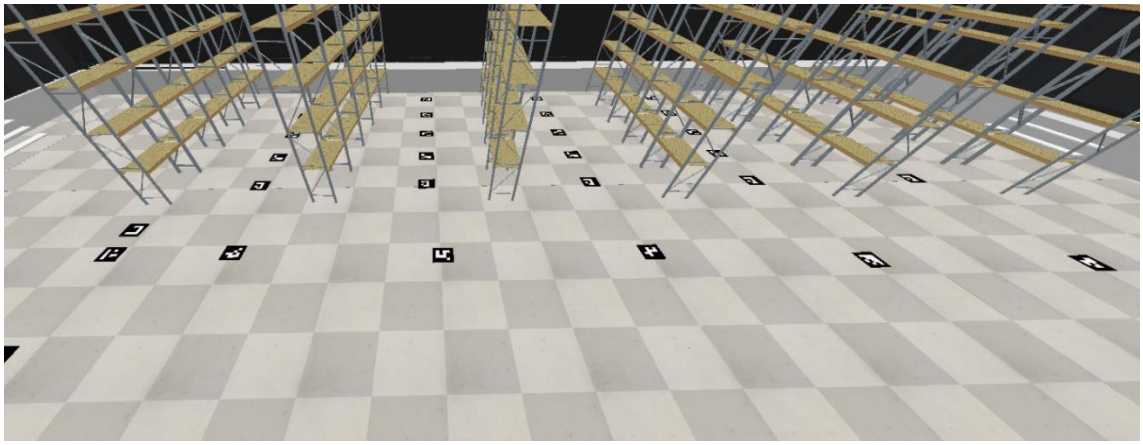


Figura 3.17 Vista de las estanterías y marcadores ArUco dentro del entorno de trabajo simulado en CoppeliaSim.

En la escena del almacén, hay estanterías donde el robot almacenará las cajas que maneja. Para ayudar en la navegación y localización del robot, se colocan múltiples marcadores ArUco en el suelo. Estos marcadores sirven como referencias visuales, guiando al robot mientras se mueve y realiza sus tareas. Las cajas están equipadas con marcadores ArUco, que transmiten información al robot sobre las ubicaciones de almacenamiento designadas.

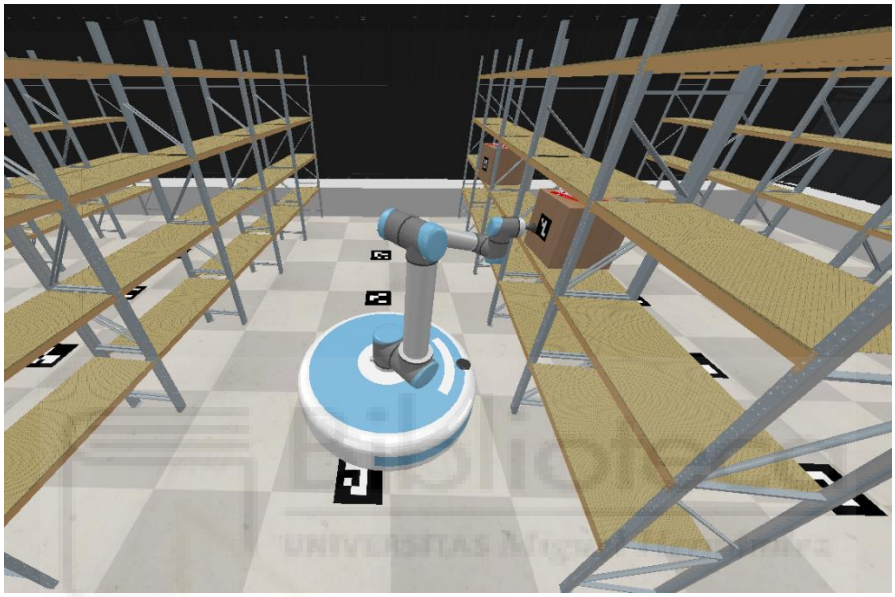


Figura 3.18 Escena del almacenamiento y localización de cajas (paquetes) en CoppeliaSim.

Sistema de cinta transportadora:

Además de las estanterías y marcadores, hay un sistema de cinta transportadora presente en la escena. Esta cinta transportadora genera cajas, cada una de las cuales está fijada con un marcador ArUco. La cinta transportadora mueve las cajas hacia la punta de la cinta hasta que un sensor de proximidad integrado en la cinta detecta la presencia de una caja. Una vez que el sensor de proximidad detecta una caja, la cinta transportadora deja de moverse. El estado del sensor de proximidad ya sea que detecte una caja o no, es conocido por el robot. Basándose en esta información, el robot toma una decisión informada sobre si iniciar el movimiento de su brazo robótico para agarrar la caja.

Figura 3.19 Escena del sistema de cinta transportadora en CoppeliaSim.

Figura 3.20 Escena del sensor de proximidad que detecta la presencia de las cajas para parar la cinta transportadora en CoppeliaSim.

3.3.6 Más vistas del entorno de simulación CoppeliaSim

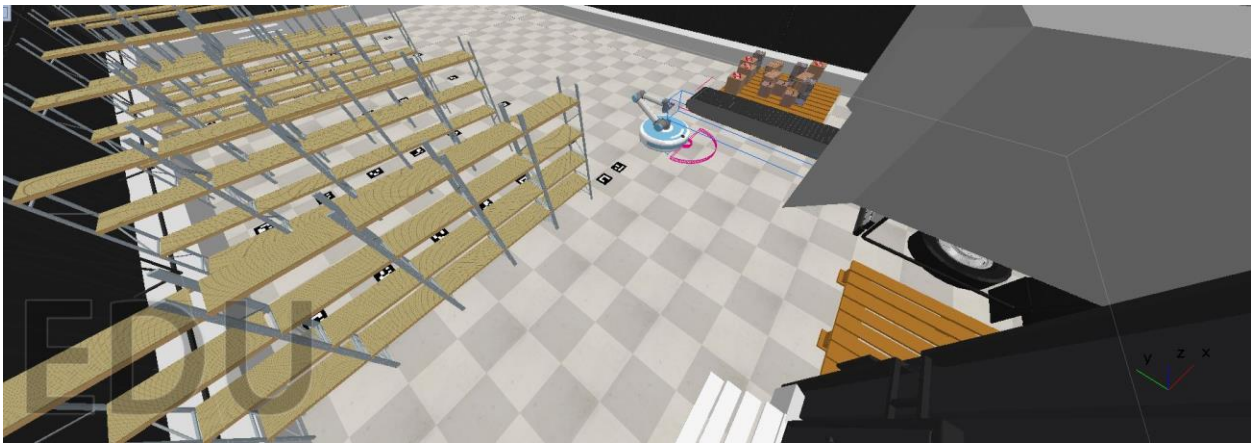


Figura 3.21 Vista superior del almacén, entorno de trabajo desarrollado en CoppeliaSim.

Figura 3.22 Vista lateral-frontal del almacén, entorno de trabajo desarrollado en CoppeliaSim.

Figura 3.23 Vista hacia la cinta transportadora, entorno de trabajo desarrollado en CoppeliaSim.

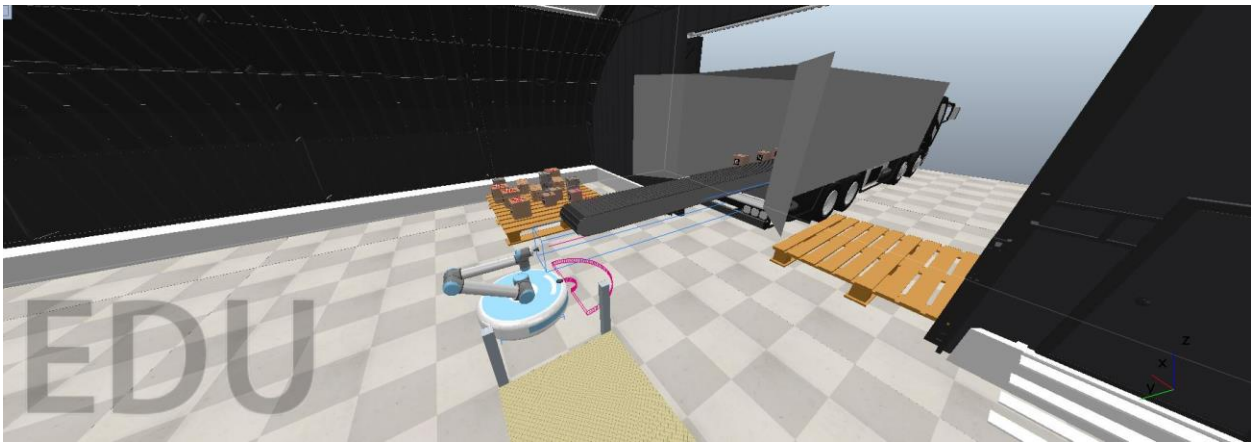


Figura 3.24 Vista superior llegada de paquetes, entorno de trabajo desarrollado en CoppeliaSim.

Figura 3.25 Vista frontal desde afuera llegada de paquetes, entorno de trabajo desarrollado en CoppeliaSim.

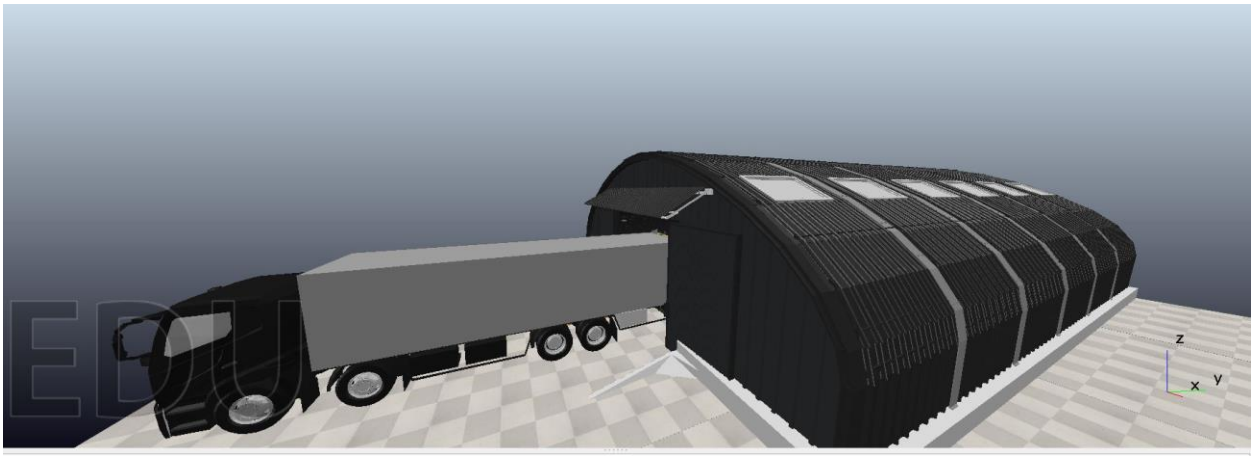


Figura 3.26 Vista de las afuera del almacén, entorno de trabajo desarrollado en CoppeliaSim.

3.4 Herramientas de programación para el sistema planteado

Para este Trabajo de Fin de Máster (TFM), se ha requerido de un entorno de programación robusto y flexible que facilite la codificación, depuración y simulación de los algoritmos necesarios para una correcta interacción entre los componentes hardware y software del sistema propuesto. Dentro de este marco, se ha elegido utilizar Anaconda 3 y el entorno de desarrollo integrado (IDE) Spyder para el desarrollo y ejecución del código en Python que posibilita la simulación y operación del sistema.

Anaconda 3 es una distribución de código abierto de Python y R, que simplifica la computación científica y la ciencia de datos. En este proyecto, Anaconda 3 se ha seleccionado por dos razones principales.

En primer lugar, proporciona una forma conveniente de gestionar paquetes y dependencias en Python, algo esencial dada la naturaleza modular del software del proyecto, que integra diversas bibliotecas para la simulación, el control del robot y la visión por computadora.

En segundo lugar, facilita el establecimiento de entornos virtuales, permitiendo crear un entorno de desarrollo aislado y reproducible, lo cual resulta vital para garantizar la consistencia y fiabilidad del proyecto.

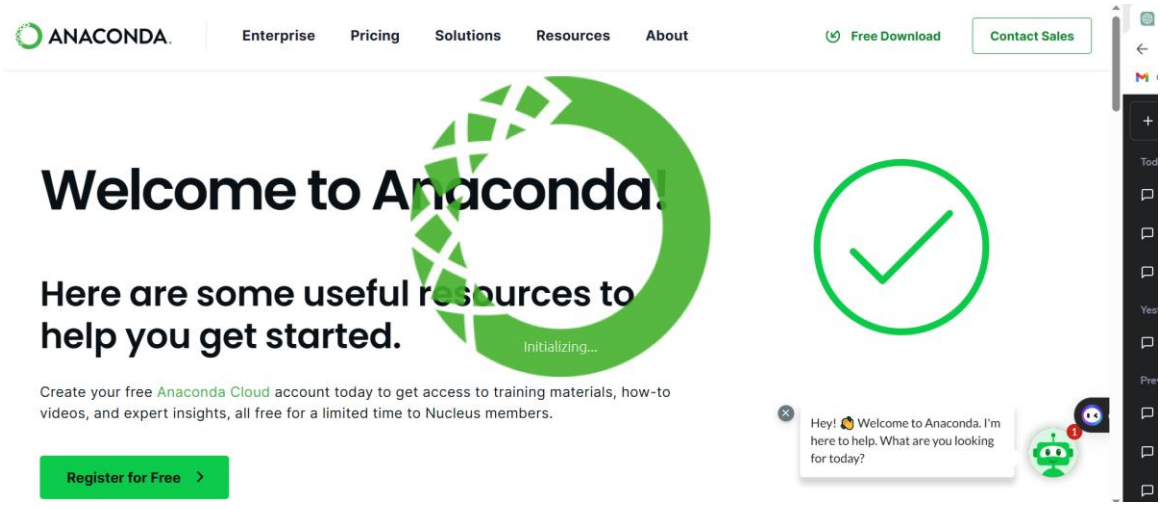


Figura 3.27 Pantalla de inicio de Anaconda: distribución de los lenguajes de programación.

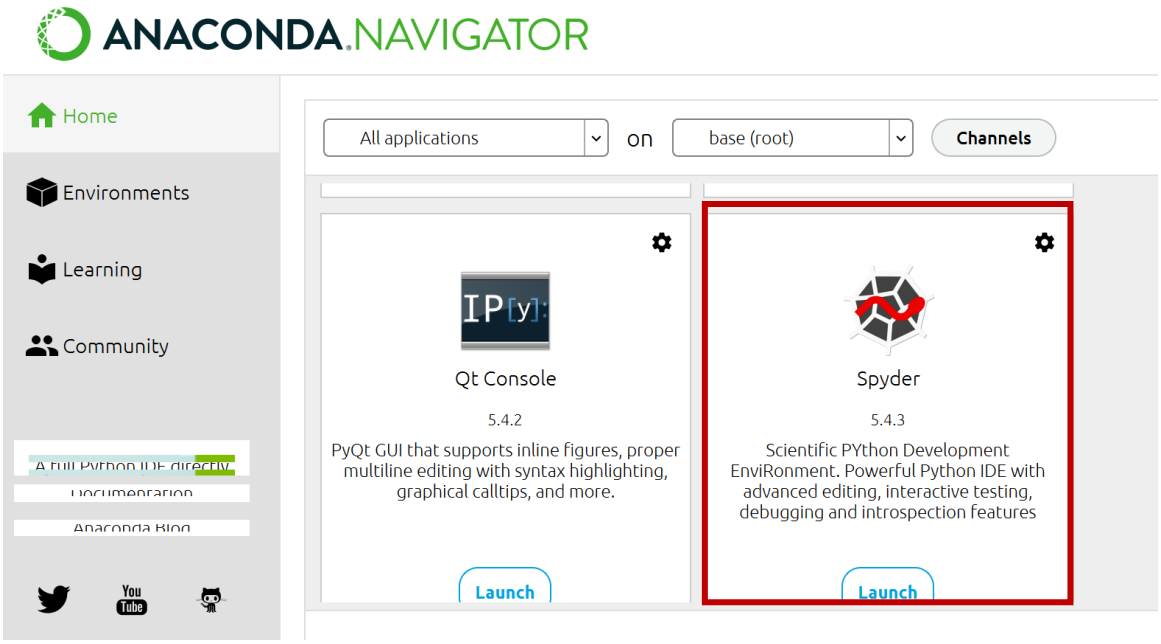


Figura 3.28 Galería de lenguajes de programación disponibles en Anaconda.

Por su parte, Spyder es un entorno de desarrollo integrado (IDE) que forma parte de la distribución Anaconda y es especialmente diseñado para la ciencia de datos en Python. Spyder se ha utilizado para desarrollar, probar y depurar el código Python responsable de la simulación y control del sistema. Spyder ofrece una serie de características que lo hacen particularmente útil para este proyecto.

Por ejemplo, su explorador de variables permite una fácil visualización y manipulación de las variables en memoria durante la ejecución del programa. Además, su potente depurador permite encontrar y corregir errores de manera eficiente. También incluye un editor de código con resaltado de sintaxis y autocompletado de código, lo que facilita la escritura de código.

Por último, su integración con las herramientas de computación científica y análisis de datos disponibles en Anaconda, como NumPy, SciPy y Matplotlib, ha sido esencial para el procesamiento y visualización de datos en este proyecto.

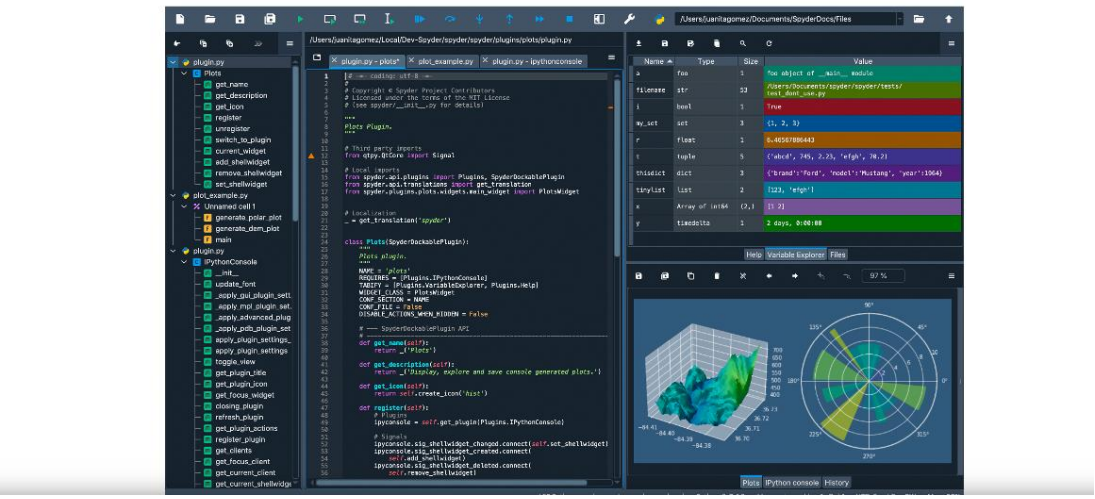


Figura 3.29 Interfaz gráfica de Spyder: entorno de desarrollo para la programación en el lenguaje Python.

Por tanto, el uso de Anaconda 3 y Spyder fue determinante en el desarrollo exitoso del proyecto, permitiendo la creación de un código Python sólido y eficiente para la simulación y control del sistema logístico propuesto, incluyendo la integración y operación del Robot Colaborativo UR10e, el Robot Móvil, los marcadores ArUco y el entorno de simulación en CoppeliaSim.

3.5 Integración con la API Remota de CoppeliaSim

Finalizando la revisión de las herramientas de programación utilizadas en este Trabajo de Fin de Máster, la API Remota Heredada (Legacy Remote API) de CoppeliaSim ha sido un componente vital. Este conjunto de funciones posibilita la comunicación fluida entre el código Python, administrado mediante Anaconda y Spyder IDE, y la simulación en CoppeliaSim. Este enlace bidireccional es fundamental para el control y la interacción con el Robot Colaborativo UR10e, la plataforma móvil y los marcadores ArUco a través del código Python.

Las funciones de la API Remota Heredada brindan la posibilidad de interactuar con la escena de simulación en tiempo real, permitiendo la lectura de sensores, el control de

motores y la manipulación de objetos en la escena. Esta capacidad de interactuar y modificar la simulación en tiempo real ha sido esencial para la validación y optimización del sistema propuesto en este proyecto.

De esta manera, se logra una conexión perfecta entre el lenguaje de programación Python, la distribución de Anaconda 3, el IDE Spyder y la API Remota de CoppeliaSim, culminando en la creación de un sistema de almacenamiento y transporte de paquetes eficiente y optimizado.

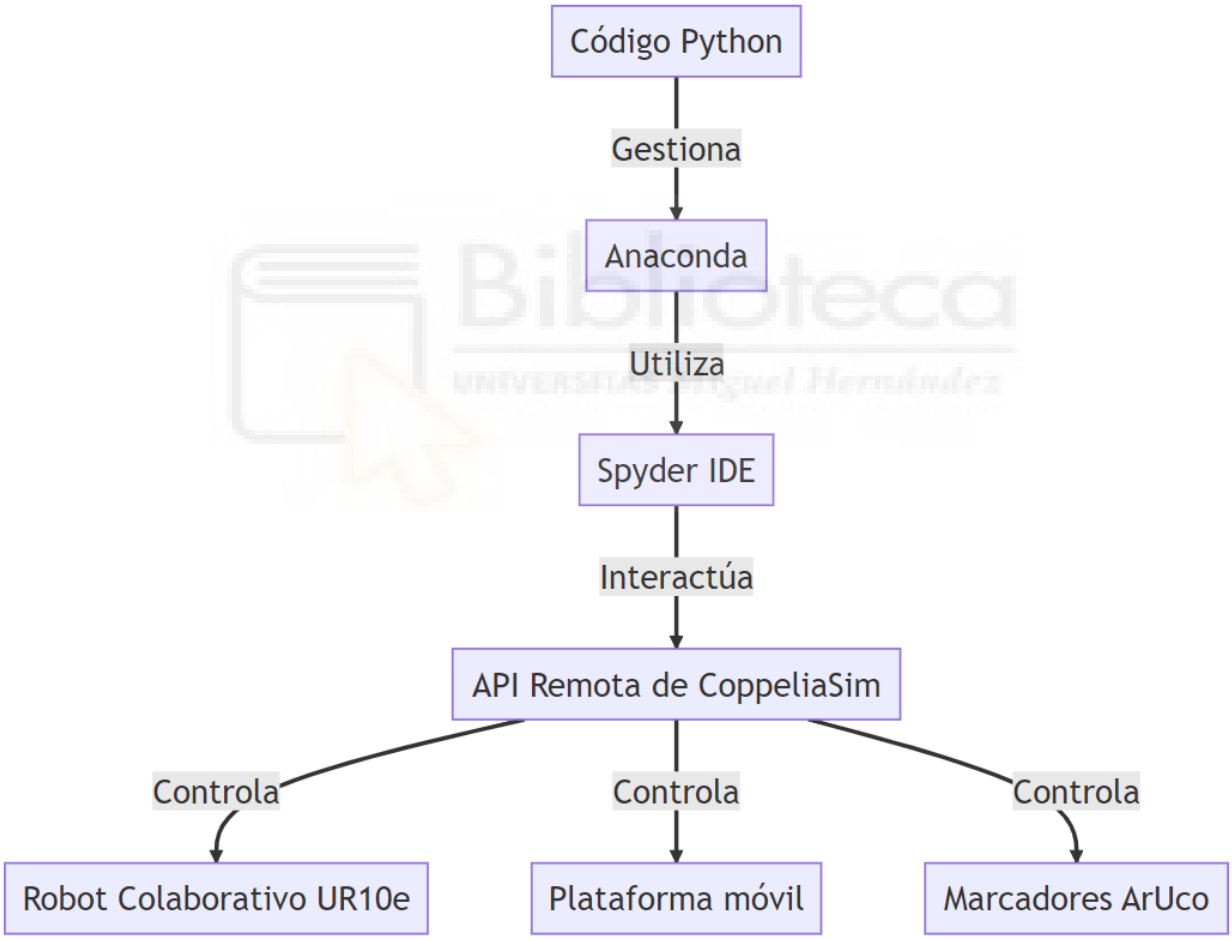


Figura 3.30 Diagrama en bloques que refleja la integración y flujo de comunicación entre las herramientas utilizadas en este TFM, elaboración propia.

4. IMPLEMENTACIÓN Y SIMULACIÓN DEL SISTEMA AUTOMATIZADO PARA LA NAVEGACIÓN Y ALMACENAMIENTO EFICIENTE DE PAQUETES EN CENTROS LOGÍSTICOS UTILIZANDO EL ROBOT COLABORATIVO UR10e DE UNIVERSAL ROBOTS, EN CONJUNTO CON UN ROBOT MÓVIL

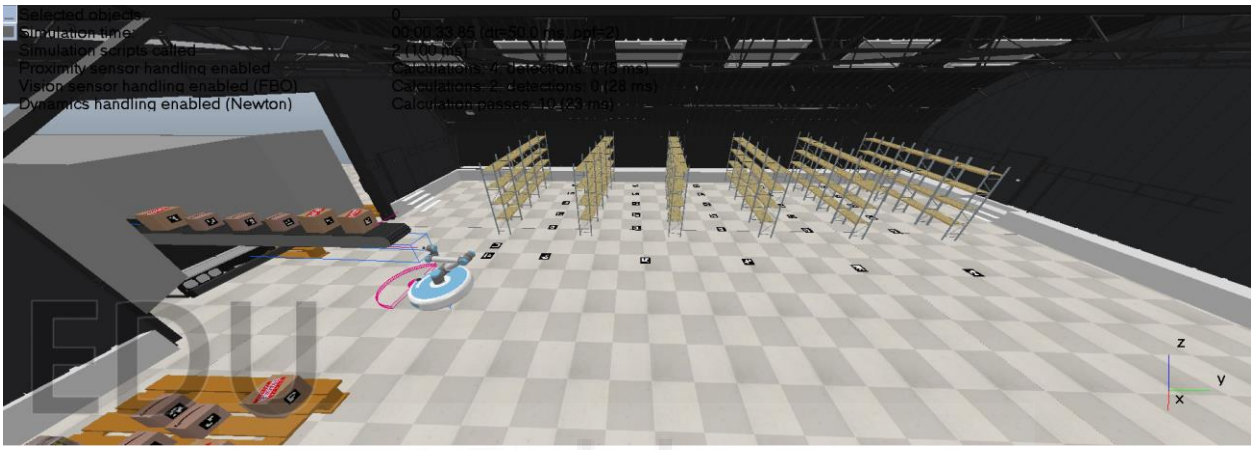


Figura 4.1 Iniciación del entorno de trabajo generado a través de CoppeliaSim.

4.1 Necesidades y limitaciones de la estación de trabajo

En el entorno de trabajo creado para la implementación del sistema automatizado de navegación y almacenamiento de paquetes en centros logísticos, utilizando el robot colaborativo UR10e de Universal Robots y un robot móvil, requiere la adecuación de una serie de requisitos y enfrenta ciertas limitaciones que deben ser tomadas en consideración durante el proceso de implementación y simulación.

Necesidades del entorno de trabajo simulado en CoppeliaSim:

Espacio: Para llevar a cabo la operación de navegación y almacenamiento de paquetes, la estación de trabajo necesita un espacio suficientemente amplio para permitir el desplazamiento libre del robot colaborativo y el robot móvil. Este espacio también debe incluir áreas para el almacenamiento de los paquetes, un lugar para la cinta transportadora y zonas de tránsito para el robot.

Cinta transportadora: La cinta transportadora es fundamental en la operación ya que es donde se colocarán los paquetes para que sean recogidos por el robot colaborativo

UR10e. Esta cinta debe ser lo suficientemente robusta y fiable para manejar la carga de trabajo prevista.

Marcadores ArUco: Los marcadores ArUco son esenciales para guiar al robot hacia las filas, estantes y niveles correctos para la colocación precisa de los paquetes. Deben estar correctamente posicionados y ser de fácil lectura para el sistema de visión del robot.

Sistema de estanterías: Un sistema de estanterías robusto y bien organizado es vital para el almacenamiento eficiente de los paquetes. Las filas, estantes y niveles deben estar claramente marcados y deben ser fácilmente accesibles para el robot.

Limitaciones del entorno de trabajo simulado en CoppeliaSim:

Capacidad de carga: La capacidad de carga del robot colaborativo UR10e de Universal Robots podría ser una limitación, ya que puede restringir el tamaño y el peso de los paquetes que se pueden manejar en la estación de trabajo.

Limitaciones de la cinta transportadora: La cinta transportadora puede tener limitaciones en cuanto a la velocidad de operación y la cantidad de paquetes que puede manejar a la vez, lo cual puede afectar la eficiencia general del sistema.

Limitaciones de visión: La eficiencia de los marcadores ArUco puede verse afectada si la visión del robot se ve obstruida o si los marcadores están mal colocados, lo que podría dar lugar a errores en la navegación y la ubicación de los paquetes.

Limitaciones del entorno: La presencia de obstáculos físicos, las condiciones de iluminación variables y la interferencia de otras operaciones en el entorno de trabajo pueden afectar la eficiencia y precisión del sistema automatizado.

4.2 Flujo de trabajo y secuencia de acciones

Flujo de trabajo del sistema desarrollado en este trabajo de fin de máster:

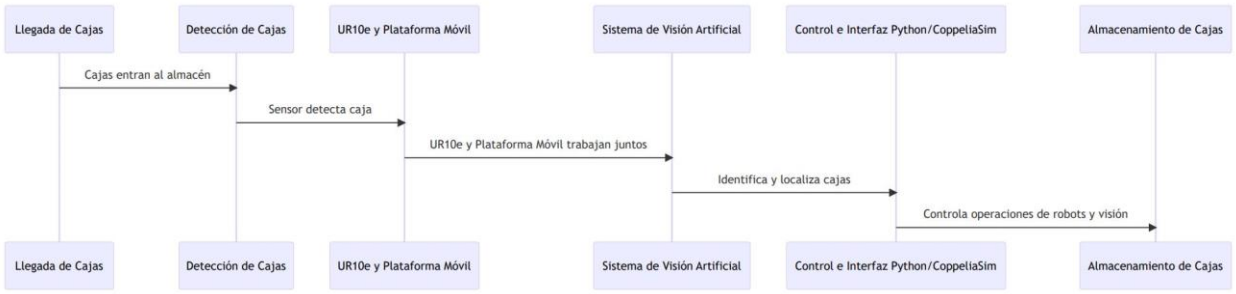


Figura 4.2 Diagrama de secuencia del sistema desarrollado en este TFM.

Bloque 1 - Generación de Cajas: Este es el punto de partida en el diagrama, representando el sistema de la cinta transportadora que genera las cajas.

Bloque 2 - Detección de Cajas: Aquí tendríamos un bloque que representa el sensor de proximidad en la cinta transportadora que detecta la presencia de una caja.

Bloque 3 - UR10e y Robot Móvil: Este bloque simboliza el Robot Colaborativo UR10e y el Robot Móvil (plataforma móvil) en una sola unidad, ya que trabajan conjuntamente.

Bloque 4 - Sistema de Visión Artificial: Este bloque representa el sistema de visión artificial que utiliza los marcadores ArUco para identificar y localizar las cajas y también para orientar al robot móvil.

Bloque 5 - Control e Interfaz Python/CoppeliaSim: Aquí se representaría la interfaz de programación en Python y el ambiente de simulación en CoppeliaSim que controlan las operaciones de los robots y la visión artificial.

Bloque 6 - Almacenamiento de Cajas: Este sería el punto final del diagrama, representando los estantes donde las cajas son almacenadas.

Secuencia de acciones de la simulación del sistema:

En este punto se mostrará el mecanismo de manejo secuencial de cajas en la simulación, un sistema diseñado para permitir que un robot manipule de manera eficiente y sistemática las cajas en un entorno controlado. Se describirá detalladamente el flujo de

trabajo y las acciones que el robot realiza, desde la detección de una caja en la cinta transportadora hasta su ubicación precisa en los estantes designados. Además, se explorará la importancia de los marcadores ArUco en la navegación y ubicación de las cajas, brindando información clave para guiar al robot hacia las filas, estantes y niveles correctos. Este mecanismo garantiza una manipulación confiable y una colocación exacta de las cajas, optimizando la eficiencia del proceso y asegurando una gestión sistemática de las mismas.

- Inicialmente, el robot se posiciona en el marcador ArUco número 11 y espera instrucciones adicionales.

- Cuando el sensor de proximidad en la cinta transportadora detecta un objeto, el robot inicia el movimiento de su brazo robótico para agarrar la caja.

- Antes de agarrar la caja, la cámara en la punta del brazo robótico lee el marcador ArUco adjunto a la caja, que consta de tres cifras numéricas.

- Después de leer el marcador ArUco, el robot ejecuta una serie de movimientos según la secuencia siguiente:

1. Gira a la derecha y avanza hasta detectar el marcador ArUco número 10.
2. Hace un giro a la izquierda y continúa avanzando hasta detectar la primera cifra del número en el marcador ArUco.
3. Gira a la derecha y avanza hasta detectar la segunda cifra del número en el marcador ArUco adjunto a la caja.
4. Gira a la derecha y posiciona la caja según el nivel indicado por la tercera cifra del marcador ArUco.
5. Gira a la derecha y avanza hasta detectar la primera cifra del marcador ArUco en la caja.
6. Gira a la derecha y continúa avanzando hasta detectar el marcador ArUco número 10 o 13.
7. Hace un giro a la izquierda y avanza hasta detectar el marcador ArUco número 15.

- Luego, el robot repite el proceso, permitiendo que maneje la siguiente caja en la cinta transportadora.

Navegación y ubicación de las cajas:

- La interpretación de las tres cifras en los marcadores ArUco es esencial para la navegación y ubicación de las cajas por parte del robot.

- La primera cifra en el marcador ArUco indica la fila de los estantes a la que el robot debe navegar.

- La segunda cifra especifica el número del estante dentro de la fila designada.

- La tercera cifra guía al robot hacia el nivel apropiado en el estante donde debe colocar la caja.

Por ejemplo, si el robot detecta el número 953, navegará a la fila 9, luego al estante 5 y finalmente colocará la caja en el tercer nivel del estante. En otro caso, si el marcador ArUco en la caja muestra el número 617, el robot se moverá hacia la fila 6, posicionará la caja en el estante 1 y en el séptimo nivel del estante.

La lectura y seguimiento de los marcadores ArUco permiten al robot navegar de manera precisa y ubicar las cajas en las ubicaciones adecuadas dentro del sistema de estanterías.

4.3 Código y control del robot móvil y brazo colaborativo en el entorno de trabajo

Instalación y configuración del entorno de desarrollo

Para comenzar, se procedió a instalar el entorno de desarrollo integrado (IDE) Spyder (anaconda3) Python desde su página principal. Luego, se instaló la biblioteca cv2 mediante el siguiente comando ejecutado en la consola: "pip install opencv-python". Posteriormente, en el código, se importaron diversas bibliotecas necesarias, como sim, simConst, matplotlib.pyplot, time, cv2, ArUco, numpy, math, pi de math, Image de PIL, imutils y array. Es importante destacar que los archivos sim.py, simConst.py y

remoteApi.dll deben estar ubicados en la misma carpeta que el archivo warehouse.py para su correcto funcionamiento.

Descripción de la clase Bot y sus métodos

La clase Bot representa al robot en la simulación. Esta clase tiene varios atributos que almacenan los nombres o identificadores de diferentes objetos en el entorno de simulación, como articulaciones, sensores, cámaras y objetos. La clase Bot tiene un constructor (init method) que inicializa el robot conectándose a CoppeliaSim utilizando la dirección IP y el número de puerto proporcionados. Se establece la conexión del cliente, se obtienen los identificadores de objetos para varios componentes del robot y se realizan algunas inicializaciones.

Además, la clase Bot contiene varios métodos que definen el comportamiento y las acciones del robot. Algunos de los métodos destacados son:

move(pos): Mueve la punta del robot a una posición especificada mediante el establecimiento de la posición objetivo de la punta del robot.

aruco(idcam): Utiliza un algoritmo de detección de marcadores ArUco para identificar y devolver el ID del marcador ArUco detectado por una cámara específica en la simulación.

get_orientation(): Obtiene la orientación de un objeto de referencia en la simulación.

get_image(idcam): Obtiene una imagen de una cámara especificada en la simulación y la devuelve como un arreglo de NumPy.

turn_to_orientation(target_orientation): Rota el robot para alcanzar una orientación objetivo-específica utilizando un sistema de control PID para una precisión en la rotación.

move_forward(target_orientation, a): Mueve el robot hacia adelante manteniendo una orientación específica hasta alcanzar un marcador ArUco con el ID a. Se utiliza un sistema de control PID para que el robot se mueva en línea recta.

grab(): Activa el sensor de proximidad del robot para detectar objetos cercanos y realiza una acción de agarre sobre el objeto detectado.

go(a, b): Mueve el robot a una fila (a) y estantería (b) específicas utilizando una combinación de rotaciones y movimientos hacia adelante.

back(a, b): Mueve el robot de regreso a su posición inicial desde una fila (a) y estantería (b) especificadas.

sep(number): Separa un número de tres dígitos en dígitos individuales y los devuelve como una lista.

put(a): Realiza una acción de colocación basada en el valor de a, el cual determina la estantería de destino.

execute(): Es el bucle de ejecución principal del robot, donde realiza continuamente las acciones necesarias para manejar las cajas en la cinta transportadora.

La función main() crea una instancia de la clase Bot y llama a su método execute() para iniciar la ejecución del robot en la simulación. El código utiliza la API remota (Remote API Legacy) para controlar el robot en CoppeliaSim. Esta API permite que programas externos se comuniquen y controlen simulaciones en ejecución en CoppeliaSim.

Interfaz con CoppeliaSim usando la API remota y comandos claves utilizados en el código

El código utiliza la API remota heredada (Remote API Legacy) para controlar el robot en CoppeliaSim. La API remota permite que programas externos se comuniquen y controlen simulaciones en ejecución en CoppeliaSim, [30].

A continuación, se explica brevemente algunos de los comandos clave utilizados en el código:

sim.simxFinish(-1): Este comando finaliza cualquier conexión existente con CoppeliaSim.

self.clientID = sim.simxStart(self.ip, self.port, True, True, 5000, 5): Este comando establece una nueva conexión con CoppeliaSim utilizando la dirección IP (self.ip) y el número de puerto (self.port) especificados. Retorna un ID de cliente que se utiliza para la comunicación posterior con la simulación.

sim.simxGetObjectHandle(self.clientID, object_name, sim.simx_opmode_oneshot_wait): Este comando obtiene el identificador de un objeto específico en la simulación. El objeto se identifica por su nombre (object_name) y la función espera hasta que se obtenga correctamente el identificador.

sim.simxSetObjectPosition(self.clientID, object_handle, relative_to, position, sim.simx_opmode_oneshot): Este comando establece la posición de un objeto en la simulación. Toma el ID del cliente (self.clientID), el identificador del objeto (object_handle), el marco de referencia relativo al cual se especifica la posición (relative_to) y la posición deseada (position). El modo sim.simx_opmode_oneshot especifica que la operación se debe realizar solo una vez.

sim.simxGetObjectOrientation(self.clientID, object_handle, relative_to, sim.simx_opmode_streaming): Este comando obtiene la orientación de un objeto en la simulación. Toma el ID del cliente (self.clientID), el identificador del objeto (object_handle), el marco de referencia relativo del cual se obtiene la orientación (relative_to) y el modo sim.simx_opmode_streaming para transmitir continuamente los datos de orientación.

sim.simxSetJointTargetVelocity(self.clientID, joint_handle, velocity, sim.simx_opmode_oneshot): Este comando establece la velocidad objetivo de una articulación en la simulación. Toma el ID del cliente (self.clientID), el identificador de la articulación (joint_handle), la velocidad deseada (velocity) y el modo sim.simx_opmode_oneshot para realizar la operación una vez.

sim.simxReadProximitySensor(self.clientID, sensor_handle, sim.simx_opmode_streaming): Este comando lee el estado de un sensor de proximidad en la simulación. Toma el ID del cliente (self.clientID), el identificador del

sensor (sensor_handle) y el modo sim.simx_opmode_streaming para transmitir continuamente los datos del sensor.

sim.simxSetObjectParent(self.clientID, child_handle, parent_handle, True, sim.simx_opmode_oneshot): Este comando establece la relación padre-hijo entre dos objetos en la simulación. Toma el ID del cliente (self.clientID), el identificador del objeto hijo (child_handle), el identificador del objeto padre (parent_handle), un valor booleano que indica si el hijo debe mantener su posición absoluta con respecto al padre (True) y el modo sim.simx_opmode_oneshot para realizar la operación una vez.

sim.simxGetVisionSensorImage(self.clientID, sensor_handle, 0, sim.simx_opmode_streaming): Este comando obtiene los datos de imagen de un sensor de visión en la simulación. Toma el ID del cliente (self.clientID), el identificador del sensor (sensor_handle), un especificador de formato de imagen (0 para RGB) y el modo sim.simx_opmode_streaming para transmitir continuamente los datos de la imagen.

5. RESULTADOS

5.1 Simulación de la estación de trabajo conexión Python – CoppeliaSim

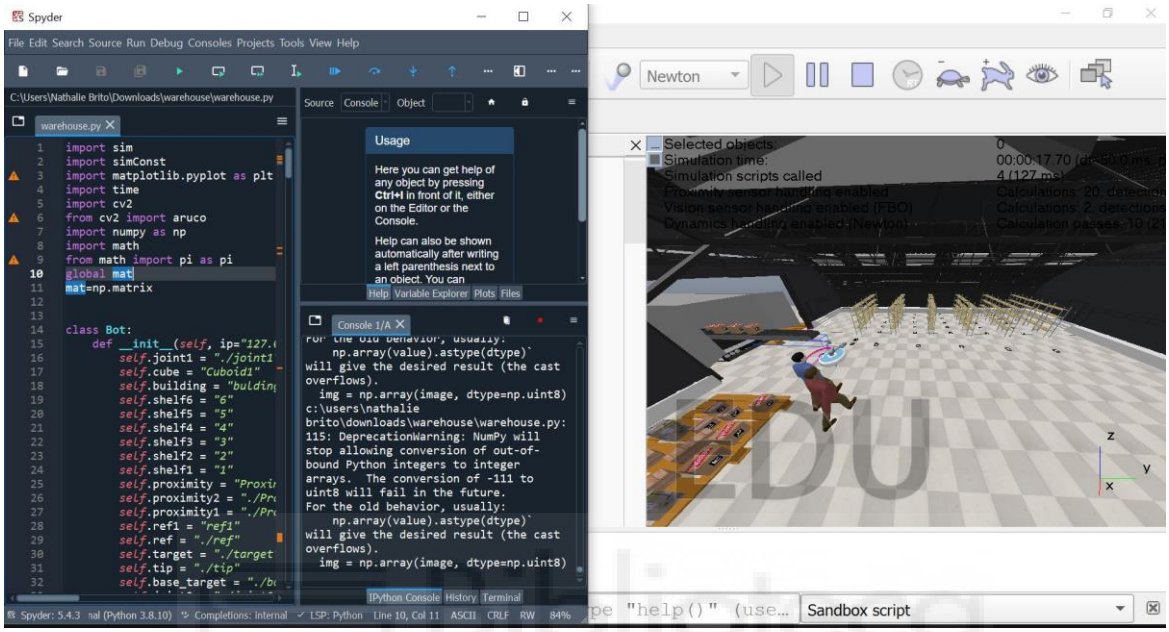


Figura 5.1 Vista de la iniciación del sistema, play en CoppeliaSim y play en Spyder IDE (Python).

Para validar el diseño de la estación de trabajo y asegurar su funcionamiento eficiente, se realizó una simulación en el entorno CoppeliaSim. A través de la conexión con Python, se pudo implementar y evaluar la totalidad del sistema, incluyendo el Robot Colaborativo UR10e, el Robot Móvil y los marcadores ArUco.

El Robot Colaborativo UR10e demostró ser altamente eficiente para las tareas de manipulación y almacenamiento de paquetes. Su diseño robusto y versátil le permitió manejar una variedad de paquetes de diferentes tamaños y formas, optimizando así el uso del espacio de almacenamiento.

La plataforma móvil demostró su capacidad para moverse de manera segura y precisa dentro del centro logístico. Mediante el uso de marcadores ArUco para trazar su ruta de navegación, el robot móvil pudo evitar obstáculos y llegar a sus destinos de manera confiable.

Los marcadores ArUco demostraron ser una herramienta eficaz para la identificación y localización precisa de paquetes. A través de la visión artificial, el sistema pudo reconocer y rastrear los paquetes en tiempo real, lo que minimizó la posibilidad de errores.

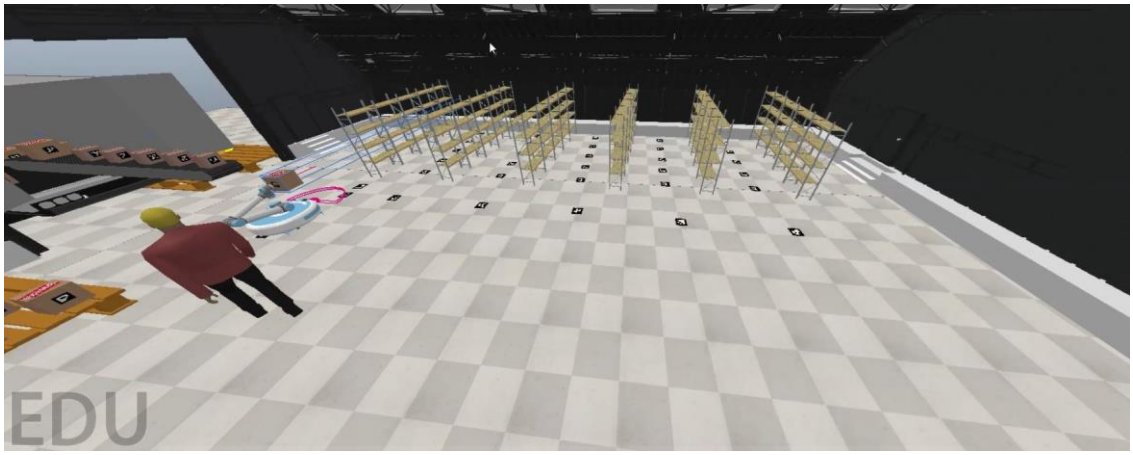


Figura 5.2 Escena 1 del entorno de trabajo en CoppeliaSim en funcionamiento.

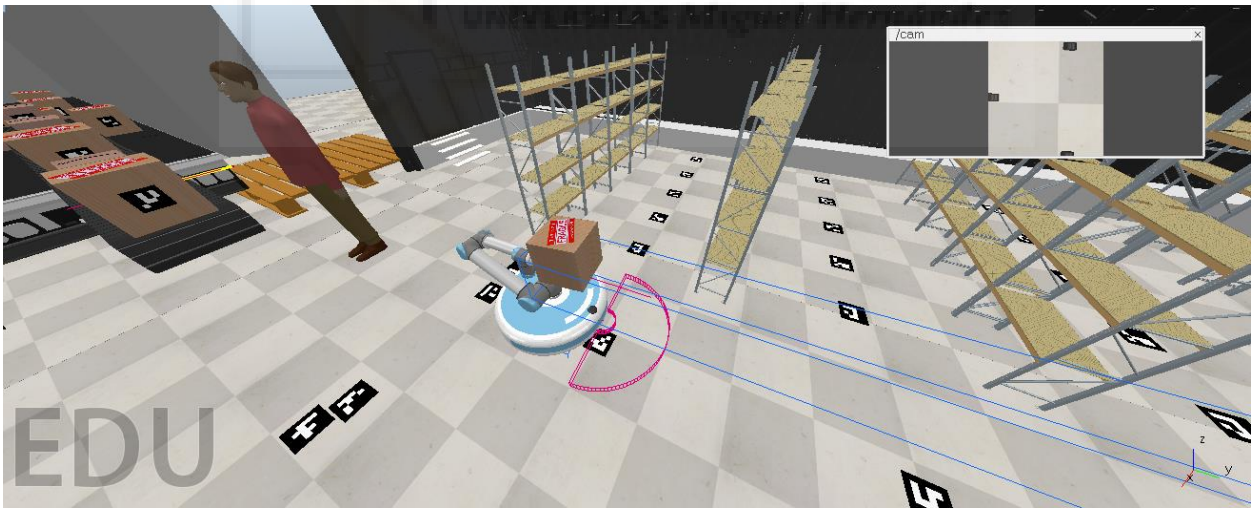


Figura 5.3 Escena 2 del entorno de trabajo en CoppeliaSim en funcionamiento.

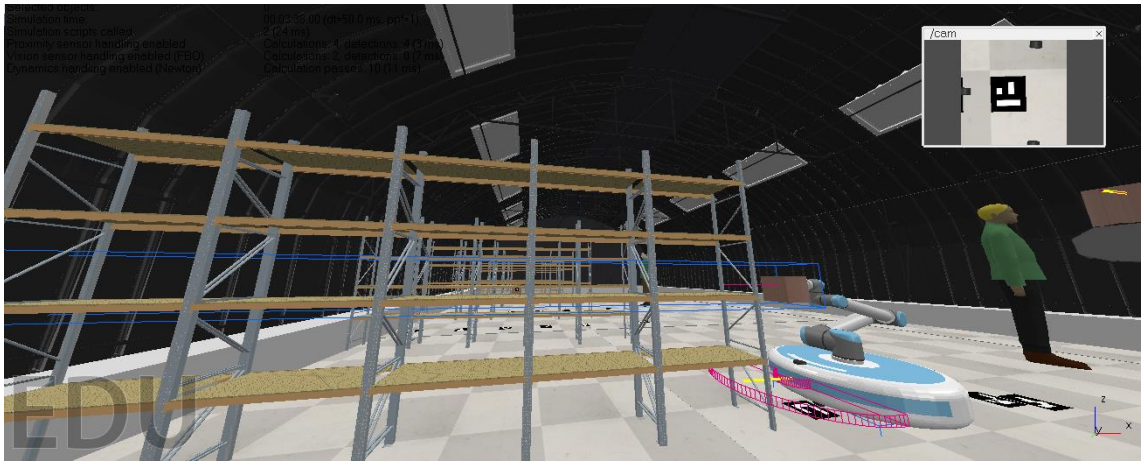


Figura 5.4 Escena 3 del entorno de trabajo en CoppeliaSim en funcionamiento.

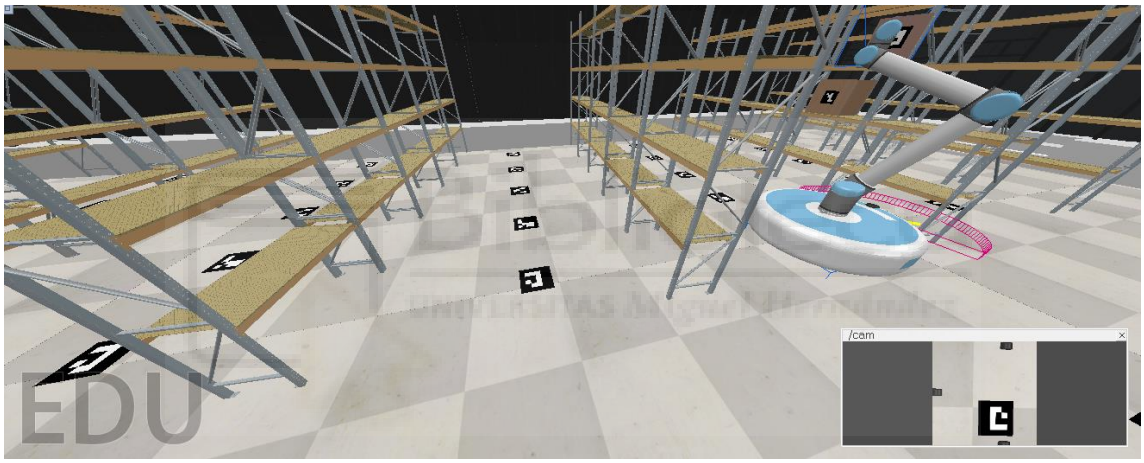


Figura 5.5 Escena 4 del entorno de trabajo en CoppeliaSim en funcionamiento.

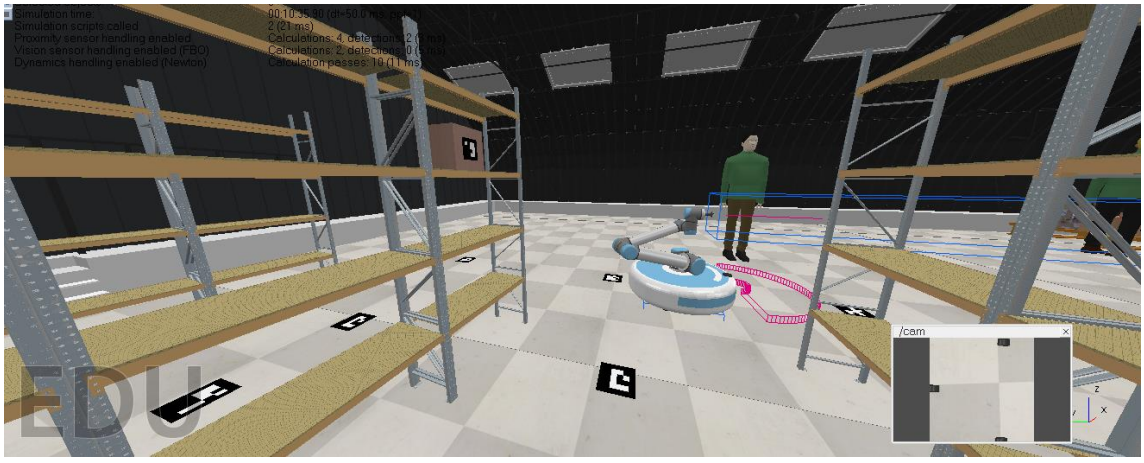


Figura 5.6 Escena 5 del entorno de trabajo en CoppeliaSim en funcionamiento.

5.2 Evaluación y análisis de los resultados obtenidos en la simulación

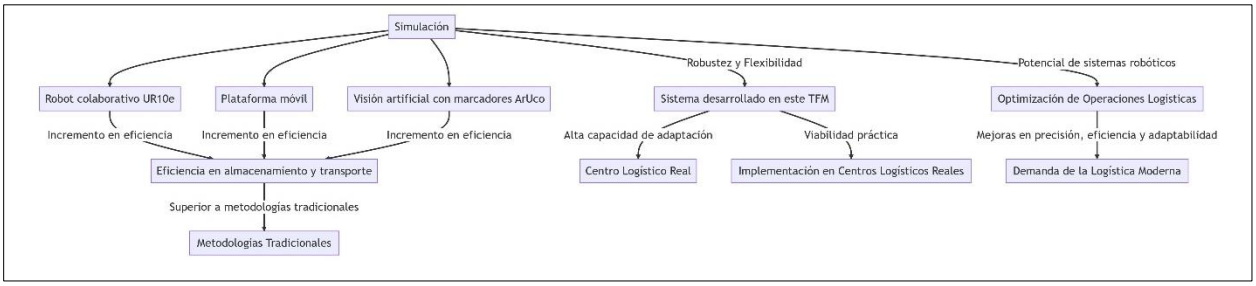


Figura 5.7 Análisis de los resultados obtenidos en la simulación.

El análisis y evaluación de las simulaciones llevaron a la comprobación de que la eficiencia y precisión de las tareas ejecutadas dentro del ambiente simulado del centro logístico, presentan un rendimiento excepcional. La incorporación y operación conjunta del robot colaborativo UR10e, la plataforma móvil autónoma y el uso de visión artificial con marcadores ArUco, ha generado un incremento significativo en la eficiencia en el almacenamiento y transporte de paquetes, superando considerablemente las metodologías tradicionales.

Los resultados de la simulación destacan, además, la robustez y la flexibilidad del sistema propuesto. Pese a los desafíos y la variabilidad que se puede presentar en un entorno de centro logístico real, el sistema ha demostrado una alta capacidad de adaptación y una efectiva respuesta ante diversas situaciones. Esto sugiere que la solución integrada propuesta en este Trabajo de Fin de Máster no solo tiene una sólida base teórica, sino que también presenta viabilidad práctica y un alto grado de promesa para su implementación en centros logísticos reales.

Por tanto, los resultados obtenidos brindan una evidencia convincente del potencial que los sistemas robóticos colaborativos y autónomos poseen para optimizar las operaciones logísticas. La implementación de este tipo de sistema puede dar lugar a mejoras significativas en términos de precisión, eficiencia y adaptabilidad, satisfaciendo las demandas crecientes de la logística moderna y proporcionando soluciones a los desafíos que este campo afronta actualmente.

6. CONCLUSIONES Y LÍNEAS FUTURAS DE TRABAJO

6.1 Conclusiones

Mediante este Trabajo de Fin de Máster se ha demostrado la viabilidad y las ventajas de integrar el robot colaborativo UR10e de Universal Robots, una plataforma móvil y la visión artificial con marcadores ArUco en un entorno logístico. La eficiencia y precisión alcanzadas en los procesos de manipulación y almacenamiento de paquetes, gracias a esta propuesta, evidencian la potencialidad de los sistemas robóticos colaborativos en el sector logístico.

A través de la simulación en CoppeliaSim, hemos validado la eficacia del sistema robótico y la utilidad de los marcadores ArUco tanto para la localización precisa de paquetes como para la navegación segura y precisa de la plataforma móvil. La implementación de la visión artificial ha permitido una identificación y manipulación eficaz de los paquetes, minimizando errores y optimizando el uso del espacio.

La utilización de un ambiente de programación flexible y robusto como Anaconda 3, junto con el entorno de desarrollo integrado (IDE) Spyder, ha facilitado la codificación, depuración y simulación de los algoritmos requeridos, reafirmando la importancia de estas herramientas en el desarrollo de proyectos robóticos de alta complejidad.

El desarrollo de este TFM ha supuesto un desafío y, al mismo tiempo, una oportunidad para profundizar en el ámbito de la robótica aplicada a la logística. El proyecto ha evidenciado el impacto que la tecnología robótica puede tener en la optimización de procesos logísticos, abriendo una vía de investigación y desarrollo que puede conducir a soluciones efectivas y eficientes para el almacenamiento y transporte de paquetes en centros logísticos.

Los resultados obtenidos son prometedores y refuerzan la idea de que la integración de robots colaborativos, robots móviles autónomos y visión artificial en un entorno logístico es una estrategia viable y efectiva para mejorar la eficiencia y precisión de los procesos logísticos. Sin embargo, aunque el sistema ha demostrado un alto nivel de rendimiento en el entorno simulado, es importante considerar la posibilidad de llevar a cabo pruebas

adicionales en un entorno real para validar su eficacia y fiabilidad en condiciones más prácticas y variadas.

Finalmente, este Trabajo de Fin de Máster ha cumplido su propósito de contribuir a la investigación en robótica aplicada a la logística y ha sentado las bases para el desarrollo de soluciones más efectivas y eficientes para los centros logísticos. Asimismo, ha fortalecido mi desarrollo profesional y especialización en el área de la robótica, contribuyendo a mi formación como docente universitaria.

6.2 Líneas futuras de trabajo

Mirando La investigación y desarrollo llevados a cabo en este Trabajo de Fin de Máster han abierto varias posibilidades interesantes para el trabajo futuro. Entre ellas, destacan las siguientes:

Implementación en entornos reales: Aunque este proyecto ha demostrado la eficacia de nuestro sistema en un entorno simulado, la validación definitiva vendrá de su implementación y prueba en un entorno real. Esto permitiría analizar y mejorar el rendimiento del sistema ante situaciones no previstas en la simulación, ajustándose a las variabilidades y exigencias de un centro logístico real.

Optimización del rendimiento del sistema: Aunque los resultados obtenidos son prometedores, siempre existe espacio para la mejora y la optimización. Sería beneficioso explorar algoritmos más eficientes para el control de los robots, la identificación y seguimiento de marcadores ArUco, y la optimización del uso del espacio de almacenamiento.

Explorar la utilización de Inteligencia Artificial: Se podría investigar la aplicación de técnicas de Inteligencia Artificial (IA) y aprendizaje automático para mejorar la autonomía y eficiencia del sistema. Esto podría incluir la predicción de la demanda de almacenamiento y transporte, la optimización de rutas, o el aprendizaje de nuevos comportamientos por parte de los robots para mejorar su adaptabilidad.

Integración de más tipos de robots: Finalmente, una línea de investigación interesante sería la integración de diferentes tipos de robots para tareas especializadas, creando un sistema de robótica colaborativa más diverso y flexible.

A través de estos posibles avances, se puede continuar ampliando y mejorando el impacto de la robótica en la logística, trabajando en la creación de sistemas cada vez más eficientes y fiables. La investigación en esta área seguirá siendo crucial para impulsar el desarrollo de soluciones que puedan hacer frente a los crecientes desafíos de la logística en el siglo XXI.

7. BIBLIOGRAFÍA

[1] A. Keshvarparast, D. Battini, O. Battaia y A. Pirayesh, «Robots colaborativos en sistemas de fabricación y ensamblaje: revisión de la literatura y agenda de investigación futura,» *Journal of Intelligent Manufacturing,* 2023.

[2] A. Hameed1, A. Ordys, J. Możaryn y A. Sibilska-Mroziewicz, «Diseño y métodos de sistemas de control para robots colaborativos: revisión,» *Sección de Computación e Inteligencia Artificial,* vol. 13, nº 1, 2023.

[3] E. Robotics, «Blog EDS Robotics,» 07 01 2021. [En línea]. Available: https://www.edsrobotics.com/blog/diferencias-robots-vs-cobots/. [Último acceso: 28 06 2023].

[4] H. I. Christensen, «Collaborative Robots: From the Lab to the Marketplace,» *Science Robotics,* 2015.

[5] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki y S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementations, The MIT Press, 2005.

[6] T. Fong, «Collaborative Robots: The Future of Human-Robot Interaction,» Springer, 2018.

[7] M. S. a. S. K. Martin Hägele, «Collaborative Robots for Assembly: A Survey,» *IEEE Transactions on Automation Science and Engineering,* 2020.

[8] V. Böhm, «Collaborative Robots: From Sensors to Planning and Control,» Springer International Publishing, Suiza, 2020.

[9] T. p. l. Industria, «Tecnología para la Industria: Aplicaciones y Ventajas de los Robots Colaborativos en la Industria: Mejora la eficiencia y seguridad.,» 07 06 2023. [En línea]. Available: https://tecnologiaparalaindustria.com/aplicaciones-y-ventajas-de-los-robots-colaborativos-en-la-industria-mejora-la-eficiencia-y-seguridad/. [Último acceso: 28 06 2023].

[10] S. u. l. R. e. l. 4. infoPLC, «Norma ISO para Robots Colaborativos,» 17 02 2016. [En línea]. Available: https://www.infoplc.net/actualidad-industrial/item/103207-iso-norma-ts15066-robots-colaborativos. [Último acceso: 28 06 2023].

[11] I. -. I. O. f. Standardization, «ISO/TS 15066:2016 - Robots and robotic devices — Collaborative robots,» 02 2016. [En línea]. Available: https://www.iso.org/standard/62996.html.

[12] M. &. S. e. S. A. e. W. Fanny Platbrood (Product Manager Industrial Safety, SAFE ROBOTICS: SEGURIDAD EN SISTEMAS DE ROBOTS, Alemania, 2018.

[13] Y. Chen, F. Sun y L. Zhou, «Visual Perception Techniques in Collaborative Robots: A Comprehensive Review.,» *IEEE Transactions on Industrial Informatics,* vol. 17, nº 3, pp. 1697-1712, 2021.

[14] R. Szeliski, «Computer vision: algorithms and applications.,» *Springer Science & Business Media.,* 2010.

[15] Y. LeCun, Y. Bengio y G. Hinton, «Deep learning.,» *Nature,* vol. 521, nº 7553, pp. 436-444., 2015.

[16] J. Borenstein, H. Everett, L. Feng y D. Wehe, «Mobile Robot Positioning: Sensors and Techniques,» *Journal of Robotic Systems,* vol. 14, nº 4, pp. 231 - 249, 1997.

[17] V. B. Sotelo, J. G. Sánchez y R. S. Ortigoza, «Robots Móviles: Evolución y Estado del Arte,» *Polibits, Disponible en: http://www.redalyc.org/articulo.oa?id=402640448003,* nº 35, pp. 12-17, 2007.

[18] «Vehículos de guiado autónomo (AGV) en aplicaciones industriales: una revisión,» *Revista Politécnica, Disponible en: ResearchGate, https://www.researchgate.net/publication/334181954_Vehiculos_de_guiado_autonomo_AGV_en_aplicaciones_industriales_una_revision,* vol. 15, nº 28, pp. 117-137, 2019.

[19] Robotnik, «Blog de la Empresa: Robotnik Automation S.L.,» Robotnik, 11 08 2021. [En línea]. Available: https://robotnik.eu/es/que-es-un-robot-movil-autonomo-amr-lo-que-aportan-nuestros-robot-moviles-a-tu-empresa/. [Último acceso: 25 Junio 2023].

[20] HiSoUR, «HiSoUR: ARTE CULTURA HISTORIA,» HiSoUR, [En línea]. Available: https://www.hisour.com/es/mobile-manipulator-42888/. [Último acceso: 25 Junio 2023].

[21] Interempresas.net, «Interempresas.net (Robótica industrial),» Redacción Interempresas, 03 11 2020. [En línea]. Available: https://www.interempresas.net/Robotica-industrial/Articulos/317732-RB-Kairos-primer-robot-movil-totalmente-preparado-integrar-brazos-Universal-Robots-Series.html. [Último acceso: 25 06 2023].

[22] Robotnik, «Página Web de Robotnik Automation S.L.,» Empresa: Robotnik Automation S.L., [En línea]. Available: https://robotnik.eu/es/productos/manipuladores-moviles/manipuladores-xl-gen/. [Último acceso: 25 06 2023].

[23] M. S. Romero, «El Español, Omicrono,» 15 11 2021. [En línea]. Available: https://www.elespanol.com/omicrono/tecnologia/20211115/cobi-primer-vacunas-autonoma-indolora-sin-agujas/625687836_0.html. [Último acceso: 30 06 2023].

[24] J. PELEGRÍ, «Universal Robots Blog,» 16 05 2022. [En línea]. Available: https://www.universal-robots.com/es/blog/robots-para-agricultura/. [Último acceso: 30 06 2023].

[25] IMPRESORAS3D.COM, «Noticias Impresoras3D.com,» 11 01 2018. [En línea]. Available: https://www.impresoras3d.com/asi-robot-3d-imprime-edificio-14-horas/. [Último acceso: 30 06 2023].

[26] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas y M. Marín-Jiménez, «Automatic generation and detection of highly reliable fiducial markers under occlusion.,» *Pattern Recognition,* vol. 47, nº 6, pp. 2280-2292, 2014.

[27] A. Babinec, L. Jurišica, P. Hubinský y F. Duchoň, «Visual Localization of Mobile Robot Using Artificial Markers,» *Procedia Engineering (ResearchGate),* vol. 96, pp. 1-9, 2014.

[28] Y. Wang, L. He, X. Zhou, Z. Zhu y Y. Wang, «Robotic Package Sorting System Based on Deep Learning and RGB-D Sensors.,» *IEEE Access,* vol. 9, pp. 35222-35233, 2021.

[29] CoppeliaSim, «Coppelia Robotics,» [En línea]. Available: https://www.coppeliarobotics.com/. [Último acceso: 20 06 2023].

[30] CoppeliaRobotics, «Legacy remote API functions (Python),» [En línea]. Available: https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsPython.htm. [Último acceso: 30 05 2023].

[31] CFZ-Cobots, «CFZ Cobots: Distribuidor Oficial Universal Robots,» 11 2021. [En línea]. Available: https://cfzcobots.com/wp-content/uploads/2021/11/ur10e-tech-spec-12.5kg.pdf. [Último acceso: 24 05 2023].

[32] O. Khatib, «Real-time obstacle avoidance for manipulators and mobile robots.,» *International Journal of Robotics Research,* vol. 4, nº 1, pp. 90-98, 1985.

8. ANEXOS

8.1 Glosario general

Robótica: La robótica es una disciplina que combina la ingeniería, la informática y la ciencia para diseñar, construir y operar robots. Los robots son máquinas programables capaces de realizar tareas de forma autónoma o semiautónoma, y su aplicación abarca diversos campos como la industria, la medicina, la exploración espacial entre otros.

Robot colaborativo: Un robot colaborativo es un tipo de robot diseñado para interactuar y trabajar de forma segura junto con los seres humanos en un entorno compartido. Por tal razón, se caracteriza por su capacidad para realizar tareas en colaboración con los humanos, ofreciendo flexibilidad y adaptabilidad en la ejecución de diversas actividades.

UR10e: El UR10e es un robot colaborativo fabricado por Universal Robots. Es conocido por su diseño ergonómico y su capacidad para realizar movimientos precisos y seguros. Este robot se utiliza en una amplia gama de aplicaciones, desde la manipulación de objetos hasta la soldadura y el montaje.

Robot Móvil: Es una máquina autónoma o semiautónoma que tiene la capacidad de moverse y navegar de forma independiente en un ambiente, utilizando diversas formas de locomoción. Estos robots pueden estar equipados con una variedad de sensores y sistemas de procesamiento de datos para percibir y responder a su entorno, permitiéndoles realizar tareas en una variedad de contextos, desde ambientes domésticos hasta industriales y de exploración. En este trabajo, la plataforma móvil tiene la función de transportar al robot colaborativo UR10e a diferentes ubicaciones dentro del centro logístico.

Visión Artificial: Es una tecnología que se ocupa de extraer, analizar y comprender información útil de imágenes para tomar decisiones y ejecutar acciones. En el contexto de este trabajo de fin de máster, la visión artificial se utiliza para que los robots puedan "ver" su entorno y tomar decisiones basadas en la identificación y el reconocimiento de objetos y marcadores.

Marcadores ArUco: Los marcadores ArUco son una forma de código bidimensional que se utiliza en aplicaciones de visión artificial para la detección y seguimiento de objetos. Estos marcadores suelen tener forma cuadrada y están compuestos por patrones de alta frecuencia que permiten su identificación y localización precisa. En este trabajo, sirven para la identificación de paquetes y la traza de rutas para el Robot Móvil.

Python: Python es un lenguaje de programación de alto nivel ampliamente utilizado en el campo de la inteligencia artificial y la robótica. Es conocido por su simplicidad y legibilidad, lo que facilita el desarrollo de algoritmos y aplicaciones complejas. Python se ha empleado en este proyecto para el desarrollo de los algoritmos de control y visión.

CoppeliaSim: CoppeliaSim es un simulador de robótica ampliamente utilizado que permite modelar, simular y visualizar escenas y robots en entornos virtuales. Proporciona una interfaz gráfica intuitiva y herramientas de programación para el desarrollo y prueba de algoritmos de control y planificación de robots.

Clasificación de paquetes: La clasificación de paquetes se refiere al proceso de categorizar y organizar los paquetes de acuerdo con ciertos criterios predefinidos. En el contexto de esta tesis, se utiliza la clasificación de paquetes para determinar la ubicación y destino de cada paquete en un centro logístico.

Centros logísticos: Los centros logísticos son instalaciones que se encargan del almacenamiento, distribución y gestión de productos y mercancías. Son lugares estratégicos donde se lleva a cabo el procesamiento y control de los flujos de materiales para satisfacer la demanda del mercado de manera eficiente.

Simulación: La simulación consiste en la recreación de un sistema o proceso en un entorno virtual o computacional. En el contexto de este trabajo de fin de máster, se emplea para probar y validar el sistema automatizado propuesto en un entorno seguro y controlado.

Automatización: La automatización se refiere a la utilización de tecnología y sistemas para realizar tareas de forma automática, sin intervención humana directa. En este trabajo de fin de máster, la automatización se aplica al proceso de clasificación y almacenamiento de paquetes en los centros logísticos mediante el uso de robots.

Eficiencia de Almacenamiento: Capacidad de optimizar el uso del espacio de almacenamiento, mejorando así la productividad y reduciendo los costos operacionales. En este trabajo, se busca incrementar la eficiencia de almacenamiento en los centros logísticos a través de la automatización del proceso de clasificación y almacenamiento de paquetes.

Navegación de Robots: Capacidad de un robot para moverse de manera autónoma por un entorno físico. Este aspecto es crucial en este trabajo de fin de máster, ya que el Robot Móvil necesita navegar de forma segura y eficiente dentro del centro logístico.

8.2 Ficha técnica del robot colaborativo UR10e

UR10e Ficha técnica

Rendimiento

Consumo de energía	Aprox. 350W para un programa típico		
Operación de colaboración	17 funciones de seguridad ajustables avanzadas, incluyendo supervisión del codo Control remoto de acuerdo con ISO 10218		
Certificaciones	EN ISO 13849-1, Cat.3, PL d, y EN ISO 10218-1		
Sensor F/P - Fuerza, X-Y-Z	Sensor F/P - Par, X-Y-Z		
Rango	100 N	Rango	10 Nm
Resolución	2,0 N	Resolución	0,02 Nm
Precisión	5,5 N	Precisión	0,60 Nm
Rango de temperatura ambiente	0–50°C		
Humedad	90%RH (sin condensación)		

Especificación

Carga útil	12.5kg / 27.5lbs
Alcance	1300mm / 51,2in
Grados de libertad	6 articulaciones giratorias
Programación	Interfaz gráfica del usuario PolyScope con pantalla táctil de 12" con soporte

Movimiento

Repetibilidad de posición	+/-0,05mm con carga, según ISO 9283	
Movim. del eje del brazo robot.	Radio de acción	Velocidad máxima
Base	± 360°	± 120°/s
Hombro	± 360°	± 120°/s
Codo	± 360°	± 180°/s
Muñeca 1	± 360°	± 180°/s
Muñeca 2	± 360°	± 180°/s
Muñeca 3	± 360°	± 180°/s
Velocidad típica de TCP		1 m/s / 39,4 in/s

Funciones

Clasificación IP	IP54	
Clase ISO Sala limpia	5	
Ruido	Menos de 65dB(A)	
Montaje del robot	Cualquier orientación	
Puertos de E/S en herramienta	Entrada digital Salida digital Entrada analógica Salida analógica Interface UART (9.6k-5Mbps)	2 2 2 0
E/S de fuente de alimentación en herramienta	12V/24V 600mA continuos, 2A porcortos	periodos

Características físicas

Huella	Ø 190 mm
Materiales	Aluminio, Plásticos de PP, Acero
Tipo de conector para herramienta del robot	M8 | M8 8-pin
Long. cable del brazo robótico	6m / 236in
Peso incluyendo cable	33,5 kg / 73,9lbs

Caja de control

Funciones

Clasificación IP	IP44	
Clase ISO Sala limpia	6	
Rango de temperatura ambiente	0–50°C	
Puertos de E/S	Entrada digital Salida digital Entrada analógica Salida analógica Control a 500 Hz, 4 entradas digitales en cuadratura de alta velocidad dedicadas	16 16 2 2
E/S de fuente de alimentación	24 V 2A	
Comunicación	Frecuencia de control: 500 Hz ModbusTCP: Frecuencia de señal de 500 Hz ProfiNet y EthernetIP: Frecuencia deseñal de 500 Hz Puertos USB: 1 USB 2.0, 1 USB 3.0	
Fuente de alimentación	100-240VAC, 47-440Hz	
Humedad	90%RH (sin condensación)	

Características físicas

Tamaño de la caja de control (anch. x alt. x prof.)	475mm × 423mm × 268mm / 18,7in × 16,7in × 10,6in
Peso	Máx. 13,6kg / 30,0lbs
Materiales	Acero

Consola de programación

Funciones

Clasificación IP	IP54
Humedad	90%RH (sin condensación)
Resolución de pantalla	1280 x 800 píxeles

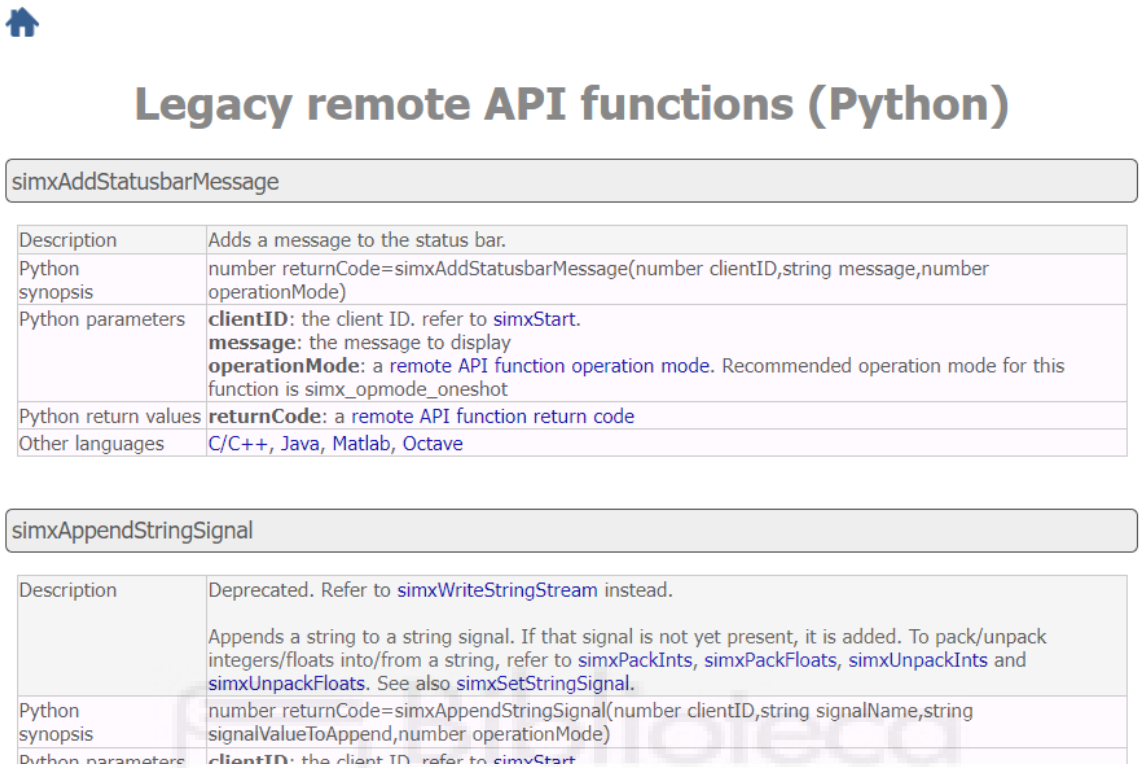
Características físicas

Materiales	Plástico
Peso incluyendo 1m de cable de TP	1,6kg / 3,5lbs
Longitud del cable	4,5m / 177,17in

UNIVERSAL ROBOTS
universal-robots.com

Figura 8.1 Ficha técnica del robot colaborativo UR10e, [31].

8.3 API Remota Heredada (Legacy Remote API) de CoppeliaSim



Legacy remote API functions (Python)

simxAddStatusbarMessage

Description	Adds a message to the status bar.
Python synopsis	number returnCode=simxAddStatusbarMessage(number clientID,string message,number operationMode)
Python parameters	**clientID**: the client ID. refer to simxStart. **message**: the message to display **operationMode**: a remote API function operation mode. Recommended operation mode for this function is simx_opmode_oneshot
Python return values	**returnCode**: a remote API function return code
Other languages	C/C++, Java, Matlab, Octave

simxAppendStringSignal

Description	Deprecated. Refer to simxWriteStringStream instead. Appends a string to a string signal. If that signal is not yet present, it is added. To pack/unpack integers/floats into/from a string, refer to simxPackInts, simxPackFloats, simxUnpackInts and simxUnpackFloats. See also simxSetStringSignal.
Python synopsis	number returnCode=simxAppendStringSignal(number clientID,string signalName,string signalValueToAppend,number operationMode)
Python parameters	**clientID**: the client ID. refer to simxStart

Figura 8.2 API Remota Heredada (Legacy Remote API) de CoppeliaSim. https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsPython.htm

8.4 Código en lenguaje de programación Python utilizado para el sistema de este presente TFM

```
import sim
import simConst
import matplotlib.pyplot as plt
import time
import cv2
from cv2 import aruco
import numpy as np
import math
from math import pi as pi
global mat
mat=np.matrix


class Bot:
```

```
    def __init__(self, ip="127.0.0.1", port=19997):
        self.joint1 = "./joint1"
        self.cube = "Cuboid1"
        self.building = "bulding_2"
        self.shelf6 = "6"
        self.shelf5 = "5"
        self.shelf4 = "4"
        self.shelf3 = "3"
        self.shelf2 = "2"
        self.shelf1 = "1"
        self.proximity = "Proximity_sensor"
        self.proximity2 = "./Prox"
        self.proximity1 = "./Proximity"
        self.ref1 = "ref1"
        self.ref = "./ref"
        self.target = "./target"
        self.tip = "./tip"
        self.base_target = "./base_target"
        self.joint2 = "./joint2"
        self.joint3 = "./joint3"
        self.joint4 = "./joint4"
        self.joint5 = "./joint5"
        self.joint6 = "./joint6"
        self.motorLeft = "./Left"
        self.motorRight = "./Right"
        self.camera = "./cam"
        self.camera1 = "./camera"
        self.s = "./Sensor"
        self.l = "./LoopClosureDummy1"
        self.l2 = "./LoopClosureDummy2"
        self.suctionPadLink = "./Link"
        self.Body = "./Body"
        self.ip = ip
        self.port = port
        print ('Connecting')
        sim.simxFinish(-1)
        self.clientID=sim.simxStart(self.ip,self.port,True,True,5000,5)
        self.activate = True
        if(self.clientID == -1):
            print("Not connected")
            self.activate = False
        _, self.motorRight = sim.simxGetObjectHandle(self.clientID,
self.motorRight, sim.simx_opmode_oneshot_wait)
        _, self.motorLeft = sim.simxGetObjectHandle(self.clientID,
self.motorLeft, sim.simx_opmode_oneshot_wait)
```

```
        _, self.camera = sim.simxGetObjectHandle(self.clientID, self.camera,
sim.simx_opmode_oneshot_wait)
        _, self.camera1 = sim.simxGetObjectHandle(self.clientID, self.camera1,
sim.simx_opmode_oneshot_wait)
        _, self.ref = sim.simxGetObjectHandle(self.clientID, self.ref,
sim.simx_opmode_oneshot_wait)
        _, self.ref1 = sim.simxGetObjectHandle(self.clientID, self.ref1,
sim.simx_opmode_oneshot_wait)
        _, self.proximity = sim.simxGetObjectHandle(self.clientID,
self.proximity, sim.simx_opmode_oneshot_wait)
        _, self.proximity1 = sim.simxGetObjectHandle(self.clientID,
self.proximity1, sim.simx_opmode_oneshot_wait)
        _, self.proximity2 = sim.simxGetObjectHandle(self.clientID,
self.proximity2, sim.simx_opmode_oneshot_wait)
        _, self.joint1 = sim.simxGetObjectHandle(self.clientID, self.joint1,
sim.simx_opmode_oneshot_wait)
        _, self.joint2 = sim.simxGetObjectHandle(self.clientID, self.joint2,
sim.simx_opmode_oneshot_wait)
        _, self.joint3 = sim.simxGetObjectHandle(self.clientID, self.joint3,
sim.simx_opmode_oneshot_wait)
        _, self.joint4 = sim.simxGetObjectHandle(self.clientID, self.joint4,
sim.simx_opmode_oneshot_wait)
        _, self.joint5 = sim.simxGetObjectHandle(self.clientID, self.joint5,
sim.simx_opmode_oneshot_wait)
        _, self.joint6 = sim.simxGetObjectHandle(self.clientID, self.joint6,
sim.simx_opmode_oneshot_wait)
        _, self.cube = sim.simxGetObjectHandle(self.clientID, self.cube,
sim.simx_opmode_oneshot_wait)
        _, self.building = sim.simxGetObjectHandle(self.clientID, self.building,
sim.simx_opmode_oneshot_wait)
        _, self.shelf6 = sim.simxGetObjectHandle(self.clientID, self.shelf6,
sim.simx_opmode_oneshot_wait)
        _, self.shelf5 = sim.simxGetObjectHandle(self.clientID, self.shelf5,
sim.simx_opmode_oneshot_wait)
        _, self.shelf4 = sim.simxGetObjectHandle(self.clientID, self.shelf4,
sim.simx_opmode_oneshot_wait)
        _, self.shelf3 = sim.simxGetObjectHandle(self.clientID, self.shelf3,
sim.simx_opmode_oneshot_wait)
        _, self.shelf2 = sim.simxGetObjectHandle(self.clientID, self.shelf2,
sim.simx_opmode_oneshot_wait)
        _, self.shelf1 = sim.simxGetObjectHandle(self.clientID, self.shelf1,
sim.simx_opmode_oneshot_wait)
        _, self.target = sim.simxGetObjectHandle(self.clientID, self.target,
sim.simx_opmode_oneshot_wait)
```

```
        _, self.tip = sim.simxGetObjectHandle(self.clientID, self.tip,
sim.simx_opmode_oneshot_wait)
        _, self.base_target = sim.simxGetObjectHandle(self.clientID,
self.base_target, sim.simx_opmode_oneshot_wait)
        _, self.s = sim.simxGetObjectHandle(self.clientID, self.s,
sim.simx_opmode_oneshot_wait)
        _, self.l = sim.simxGetObjectHandle(self.clientID, self.l,
sim.simx_opmode_oneshot_wait)
        _, self.l2 = sim.simxGetObjectHandle(self.clientID, self.l2,
sim.simx_opmode_oneshot_wait)
        _, self.suctionPadLink = sim.simxGetObjectHandle(self.clientID,
self.suctionPadLink, sim.simx_opmode_oneshot_wait)
        _, self.Body = sim.simxGetObjectHandle(self.clientID, self.Body,
sim.simx_opmode_oneshot_wait)
        _,self.robot=sim.simxGetObjectHandle(self.clientID,"robot",
sim.simx_opmode_oneshot_wait)
        self.px=[]
        self.py=[]
        time.sleep(1)
    def move(self,pos):
        x=pos[0]
        y=pos[1]
        z=pos[2]
        n=10
        x1=x/n
        y1=y/n
        z1=z/n
        for i in range(1,n+1):
            sim.simxSetObjectPosition(self.clientID, self.target, self.tip,
[x1,y1,z1], sim.simx_opmode_oneshot)
            time.sleep(0.2)
    def aruco(self,idcam):
        while True:
            _, resolution, image = sim.simxGetVisionSensorImage(self.clientID,
idcam, 0, sim. simx_opmode_streaming)
            #print(resolution)
            dictionary =
cv2.aruco.getPredefinedDictionary(cv2.aruco.DICT_4X4_1000)
            parameters =  cv2.aruco.DetectorParameters()
            detector = cv2.aruco.ArucoDetector(dictionary, parameters)
            # Check if the resolution list has the expected length
            if len(resolution) == 2:
                width, height = resolution

                # Check if the image dimensions are valid
```

```
                if width > 0 and height > 0:
                    img = np.array(image, dtype=np.uint8)
                    img.resize([width, height, 3])  # Resize the image array to
match the resolution
                    img = np.fliplr(img)
                    # Convert the image to grayscale for marker detection
                    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                    # Detect ArUco markers
                    corners, ids, rejected = cv2.aruco.detectMarkers(gray,
dictionary, parameters=parameters)
                    #print(ids)
                    #cv2.imshow('Original Image', gray)
                    #cv2.waitKey(0)
                    #cv2.destroyAllWindows()

                    if ids is not None:
                        #print (ids)
                        return ids[0][0]
                    if ids is None:
                        return 100
                else:
                    print("Invalid image dimensions.")
            #else:
                #print("Invalid resolution data.")
    def get_orientation(self):
        _,x=sim.simxGetObjectOrientation(self.clientID, self.ref, self.ref1,
sim.simx_opmode_streaming)
        degrees_table = [angle * (180 / math.pi) for angle in x]
        return degrees_table[2]
    def get_image(self, idcam):
        if(self.activate == False):
            return
        while True :
            _, resolution, image=sim.simxGetVisionSensorImage(self.clientID,
idcam, 0, sim.simx_opmode_streaming)
            print(resolution)
        if(len(resolution) == 0):
            print("No Front Cam")
            return None
        img = np.array(image, dtype = np.uint8)
        img.resize([resolution[0], resolution[1], 3])
        img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
        return img
    def turn_to_orientation(self, target_orientation):
        while True:
```

```
            current_orientation = self.get_orientation()
            error = target_orientation - current_orientation
            dt=0.0005
            # Apply PID control
            control_signal = self.Kp *abs(error) + self.Ki * self.integral*dt +
self.Kd *( (abs(error) - self.last_error)/dt)
            self.last_error = abs(error)
            self.integral += abs(error)

            # Rotate the robot based on the control signal
            if error > 0:
                sim.simxSetJointTargetVelocity(self.clientID, self.motorRight,
control_signal, sim.simx_opmode_oneshot)
                sim.simxSetJointTargetVelocity(self.clientID, self.motorLeft, -
control_signal, sim.simx_opmode_oneshot)
            elif error < 0:
                sim.simxSetJointTargetVelocity(self.clientID, self.motorRight, -
control_signal, sim.simx_opmode_oneshot)
                sim.simxSetJointTargetVelocity(self.clientID, self.motorLeft,
control_signal, sim.simx_opmode_oneshot)

            # Check if the target orientation is reached within a tolerance
            if abs(error)<1:
                break

            time.sleep(0.1)

    def move_forward(self, target_orientation,a):
        while True:
            current_orientation = self.get_orientation()
            error = target_orientation - current_orientation
            if error > 180:
                error -= 360
            # Adjust error for orientation to the left
            elif error < -180:
                error += 360
            elif -180<error<180:
                error += 0

            print(error)
            dt=0.001
            # Apply PID control for forward movement
            forward_error = error  # Error for forward movement (can be distance-
related)
```

```
            forward_control_signal = (self.forward_Kp * forward_error+
self.forward_Ki * (self.forward_integral*dt)+ self.forward_Kd*((forward_error-
self.forward_last_error)/dt))
            self.forward_last_error = forward_error
            self.forward_integral += forward_error
            sim.simxSetJointTargetVelocity(self.clientID,
self.motorRight,10+forward_control_signal, sim.simx_opmode_oneshot)
            sim.simxSetJointTargetVelocity(self.clientID, self.motorLeft,10-
forward_control_signal, sim.simx_opmode_oneshot)
            _,state,point,objet,surface=sim.simxReadProximitySensor(self.clientID
, self.proximity2, sim.simx_opmode_streaming)
            if state == True and objet!=self.building and objet!=self.shelf5 and
objet!=self.shelf4 and objet!=self.shelf3 and objet!=self.shelf2 and
objet!=self.shelf1 and objet!=self.shelf6:
                sim.simxSetJointTargetVelocity(self.clientID, self.motorRight,0,
sim.simx_opmode_oneshot)
                sim.simxSetJointTargetVelocity(self.clientID, self.motorLeft,0,
sim.simx_opmode_oneshot)
            if self.aruco(self.camera) == a:
                break
            time.sleep(0.001)
    def move_forward1(self, target_orientation,a):
        while True:
            current_orientation = self.get_orientation()
            error = target_orientation - current_orientation
            if error > 180:
                error -= 360
            # Adjust error for orientation to the left
            elif error < -180:
                error += 360
            elif -180<error<180:
                error += 0
            print(error)
            dt=0.001
            # Apply PID control for forward movement
            forward_error = error  # Error for forward movement (can be distance-
related)
            forward_control_signal = (self.forward_Kp * forward_error+
self.forward_Ki * (self.forward_integral*dt)+ self.forward_Kd*((forward_error-
self.forward_last_error)/dt))
            self.forward_last_error = forward_error
            self.forward_integral += forward_error
            sim.simxSetJointTargetVelocity(self.clientID,
self.motorRight,10+forward_control_signal, sim.simx_opmode_oneshot)
```

```
            sim.simxSetJointTargetVelocity(self.clientID, self.motorLeft,10-
forward_control_signal, sim.simx_opmode_oneshot)
            if self.aruco(self.camera) == 13:
                break
            if self.aruco(self.camera) == a:
                break
            time.sleep(0.001)
    def grab(self):
        _,state,point,objet,surface=sim.simxReadProximitySensor(self.clientID,
self.proximity, sim.simx_opmode_streaming)
        parent=-1
        point1 = [None] * 3
        point1[0]=0
        point1[1]=0
        point1[2]=0
        while point1[2]==0:
            _1,state1,point1,objet1,surface1=sim.simxReadProximitySensor(self.cli
entID, self.proximity1, sim.simx_opmode_streaming)
        #print(point1)
        self.move([0,point1[2],0])
        sim.simxSetObjectParent(self.clientID, objet, self.Body, True,
sim.simx_opmode_oneshot)
        sim.simxSetObjectIntParameter(self.clientID,
objet,simConst.sim_shapeintparam_static ,1,sim.simx_opmode_oneshot)
        self.move([0,-0.41324421763420105,0])
        self.move([0,0,-0.45])
    def go(self,a,b):
        self.turn_to_orientation(-90)
        self.move_forward(-90,10)
        self.turn_to_orientation(-180)
        self.move_forward(180,a)
        self.turn_to_orientation(-90)
        self.move_forward(-90,b)
        self.turn_to_orientation(-180)
        sim.simxSetJointTargetVelocity(self.clientID, self.motorRight, 0,
sim.simx_opmode_oneshot)
        sim.simxSetJointTargetVelocity(self.clientID, self.motorLeft, 0,
sim.simx_opmode_oneshot)
    def back(self,a,b):
        self.turn_to_orientation(90)
        if b==1:
            self.move_forward(110,a)
        if b==2:
            self.move_forward(100,a)
        if b==3:
```

```
            self.move_forward(95,a)
        if b==4:
            self.move_forward(94,a)
        if b==5:
                self.move_forward(93.5,a)
        self.turn_to_orientation(0)
        self.move_forward1(0,10)
        self.turn_to_orientation(90)
        self.move_forward(80,15)
        self.turn_to_orientation(0)
        sim.simxSetJointTargetVelocity(self.clientID, self.motorRight, 0,
sim.simx_opmode_oneshot)
        sim.simxSetJointTargetVelocity(self.clientID, self.motorLeft, 0,
sim.simx_opmode_oneshot)
    def ini(self):
        self.move([+0.16088,+0.17805,+0.5388])
    def sep(self,number):
        # Convert the number to a string
        number_str = str(number)
        # Pad the number with leading zeros if it has less than 3 digits
        number_str = number_str.zfill(3)
        # Separate each digit
        a = int(number_str[0])
        b = int(number_str[1])
        c = int(number_str[2])
        return [a,b,c]

    def put(self,a):
        if a==1:
            _,state1,point1,objet1,surface1=sim.simxReadProximitySensor(self.clie
ntID, self.proximity1, sim.simx_opmode_streaming)
            initial_position = [0.16088245809078217, 0.1780521422624588,
0.5388044118881226]
            target_position = [-0.17, 0.76, 0.259998083114624]
            step_size = 0.1
            num_steps = int(math.ceil(max(abs(target_position[0] -
initial_position[0]),
                                          abs(target_position[1] -
initial_position[1]),
                                          abs(target_position[2] -
initial_position[2]))) / step_size)
            waypoints = []
            for i in range(num_steps + 1):
                x = initial_position[0] + (target_position[0] -
initial_position[0]) * i / num_steps
```

```
                y = initial_position[1] + (target_position[1] -
initial_position[1]) * i / num_steps
                z = initial_position[2] + (target_position[2] -
initial_position[2]) * i / num_steps
                waypoints.append([x, y, z])  # Assuming orientation is not
changed
            for i in range(0,11):
                sim.simxSetObjectPosition(self.clientID, self.target,
self.base_target, waypoints[i], sim.simx_opmode_oneshot)
                time.sleep(0.2)
            sim.simxSetObjectIntParameter(self.clientID,
objet1,simConst.sim_shapeintparam_static ,0,sim.simx_opmode_oneshot)
            sim.simxSetObjectParent(self.clientID, objet1, -1, True,
sim.simx_opmode_oneshot)
            target_position = [0.16088245809078217, 0.1780521422624588,
0.5388044118881226]
            initial_position = [-0.17, 0.76, 0.259998083114624]
            step_size = 0.1
            num_steps = int(math.ceil(max(abs(target_position[0] -
initial_position[0]),
                                          abs(target_position[1] -
initial_position[1]),
                                          abs(target_position[2] -
initial_position[2])))) / step_size)
            waypoints = []
            for i in range(num_steps + 1):
                x = initial_position[0] + (target_position[0] -
initial_position[0]) * i / num_steps
                y = initial_position[1] + (target_position[1] -
initial_position[1]) * i / num_steps
                z = initial_position[2] + (target_position[2] -
initial_position[2]) * i / num_steps
                waypoints.append([x, y, z])  # Assuming orientation is not
changed
            for i in range(0,num_steps + 1):
                sim.simxSetObjectPosition(self.clientID, self.target,
self.base_target, waypoints[i], sim.simx_opmode_oneshot)
                time.sleep(0.2)
            sim.simxSetObjectOrientation(self.clientID, self.target,
self.base_target, [0,0,0], sim.simx_opmode_oneshot)
        if a==2:
            _,state1,point1,objet1,surface1=sim.simxReadProximitySensor(self.clie
ntID, self.proximity1, sim.simx_opmode_streaming)
            initial_position = [0.16088245809078217, 0.1780521422624588,
0.5388044118881226]
```

```
            target_position = [-0.2609400749206543, 0.6896467208862305,
0.6333509087562561]
            step_size = 0.1
            num_steps = int(math.ceil(max(abs(target_position[0] -
initial_position[0]),
                                          abs(target_position[1] -
initial_position[1]),
                                          abs(target_position[2] -
initial_position[2])))) / step_size)
            waypoints = []
            for i in range(num_steps + 1):
                x = initial_position[0] + (target_position[0] -
initial_position[0]) * i / num_steps
                y = initial_position[1] + (target_position[1] -
initial_position[1]) * i / num_steps
                z = initial_position[2] + (target_position[2] -
initial_position[2]) * i / num_steps
                waypoints.append([x, y, z])  # Assuming orientation is not
changed
            for i in range(0,num_steps + 1):
                sim.simxSetObjectPosition(self.clientID, self.target,
self.base_target, waypoints[i], sim.simx_opmode_oneshot)
                time.sleep(0.2)
            sim.simxSetObjectIntParameter(self.clientID,
objet1,simConst.sim_shapeintparam_static ,0,sim.simx_opmode_oneshot)
            sim.simxSetObjectParent(self.clientID, objet1, -1, True,
sim.simx_opmode_oneshot)
            target_position = [0.16088245809078217, 0.1780521422624588,
0.5388044118881226]
            initial_position = [-0.2609400749206543, 0.6896467208862305,
0.6333509087562561]
            step_size = 0.1
            num_steps = int(math.ceil(max(abs(target_position[0] -
initial_position[0]),
                                          abs(target_position[1] -
initial_position[1]),
                                          abs(target_position[2] -
initial_position[2])))) / step_size)
            waypoints = []
            for i in range(num_steps + 1):
                x = initial_position[0] + (target_position[0] -
initial_position[0]) * i / num_steps
                y = initial_position[1] + (target_position[1] -
initial_position[1]) * i / num_steps
```

```
                z = initial_position[2] + (target_position[2] -
initial_position[2]) * i / num_steps
                waypoints.append([x, y, z])  # Assuming orientation is not
changed
            for i in range(0,num_steps + 1):
                sim.simxSetObjectPosition(self.clientID, self.target,
self.base_target, waypoints[i], sim.simx_opmode_oneshot)
                time.sleep(0.2)
            sim.simxSetObjectOrientation(self.clientID, self.target,
self.base_target, [0,0,0], sim.simx_opmode_oneshot)
        if a==3:
            _,state1,point1,objet1,surface1=sim.simxReadProximitySensor(self.clie
ntID, self.proximity1, sim.simx_opmode_streaming)
            initial_position = [0.16088245809078217, 0.1780521422624588,
0.5388044118881226]
            target_position = [-0.3860071897506714, 0.1780521422624588,
1.020534634590149]
            step_size = 0.1
            num_steps = int(math.ceil(max(abs(target_position[0] -
initial_position[0]),
                                          abs(target_position[1] -
initial_position[1]),
                                          abs(target_position[2] -
initial_position[2])))) / step_size)
            waypoints = []
            for i in range(num_steps + 1):
                x = initial_position[0] + (target_position[0] -
initial_position[0]) * i / num_steps
                y = initial_position[1] + (target_position[1] -
initial_position[1]) * i / num_steps
                z = initial_position[2] + (target_position[2] -
initial_position[2]) * i / num_steps
                waypoints.append([x, y, z])  # Assuming orientation is not
changed
            for i in range(0,num_steps + 1):
                sim.simxSetObjectPosition(self.clientID, self.target,
self.base_target, waypoints[i], sim.simx_opmode_oneshot)
                time.sleep(0.2)
            initial_position = [-0.3860071897506714, 0.1780521422624588,
1.020534634590149]
            target_position = [-0.3860071897506714, 0.7067471146583557,
1.020534634590149]
            step_size = 0.1
            num_steps = int(math.ceil(max(abs(target_position[0] -
initial_position[0]),
```

```
                                        abs(target_position[1] -
initial_position[1]),
                                        abs(target_position[2] -
initial_position[2]))) / step_size)
            waypoints = []
            for i in range(num_steps + 1):
                x = initial_position[0] + (target_position[0] -
initial_position[0]) * i / num_steps
                y = initial_position[1] + (target_position[1] -
initial_position[1]) * i / num_steps
                z = initial_position[2] + (target_position[2] -
initial_position[2]) * i / num_steps
                waypoints.append([x, y, z])  # Assuming orientation is not
changed
            for i in range(0,num_steps + 1):
                sim.simxSetObjectPosition(self.clientID, self.target,
self.base_target, waypoints[i], sim.simx_opmode_oneshot)
                time.sleep(0.2)
            sim.simxSetObjectIntParameter(self.clientID,
objet1,simConst.sim_shapeintparam_static ,0,sim.simx_opmode_oneshot)
            sim.simxSetObjectParent(self.clientID, objet1, -1, True,
sim.simx_opmode_oneshot)
            target_position = [0.16088245809078217, 0.1780521422624588,
0.5388044118881226]
            initial_position = [-0.3860071897506714, 0.7067471146583557,
1.020534634590149]
            step_size = 0.1
            num_steps = int(math.ceil(max(abs(target_position[0] -
initial_position[0]),
                                        abs(target_position[1] -
initial_position[1]),
                                        abs(target_position[2] -
initial_position[2]))) / step_size)
            waypoints = []
            for i in range(num_steps + 1):
                x = initial_position[0] + (target_position[0] -
initial_position[0]) * i / num_steps
                y = initial_position[1] + (target_position[1] -
initial_position[1]) * i / num_steps
                z = initial_position[2] + (target_position[2] -
initial_position[2]) * i / num_steps
                waypoints.append([x, y, z])  # Assuming orientation is not
changed
            for i in range(0,num_steps + 1):
```

```
                sim.simxSetObjectPosition(self.clientID, self.target,
self.base_target, waypoints[i], sim.simx_opmode_oneshot)
                time.sleep(0.2)
            sim.simxSetObjectOrientation(self.clientID, self.target,
self.base_target, [0,0,0], sim.simx_opmode_oneshot)
    def execute(self):
        if(self.activate == False):
            return
        vac=self.suctionPadLink
        parent = vac #26 27 28 29
        infiniteStrength= True
        maxPullForce=3
        maxShearForce=1
        maxPeelTorque=0.1
        enabled= True
        keepInPlace= True
        # Initialize the PID parameters
        dt=0.0005
        self.Kp = 0.06  # Proportional gain
        self.Ki = 0.01  # Integral gain
        self.Kd = 0.0000 # Derivative gain
        self.forward_Kp=0.8
        self.forward_Ki=10#1.5
        self.forward_Kd=0.00000
        self.last_error = 0.0
        self.integral = 0.0
        self.forward_last_error = 0
        self.forward_integral = 0

        print("connected")
        sim.simxSetObjectIntParameter(self.clientID, self.cube,
simConst.sim_shapeintparam_static, 0, sim.simx_opmode_oneshot)
        sim.simxSetObjectParent(self.clientID, self.cube,-1, True,
sim.simx_opmode_oneshot)
        sim.simxSetJointTargetVelocity(self.clientID, self.motorRight, 0,
sim.simx_opmode_oneshot)
        sim.simxSetJointTargetVelocity(self.clientID, self.motorLeft, 0,
sim.simx_opmode_oneshot)
        while True:
            x=self.aruco(self.camera)
            if x ==11 or x==15:
                _,state,point,objet,surface=sim.simxReadProximitySensor(self.clie
ntID, self.proximity, sim.simx_opmode_streaming)
                if state==1:
                    self.move([0,0,0.45]) #move the tip of tha arm up upwards
```

```
                    y=self.aruco(self.camera1)
                    z=self.sep(y)
                    self.grab()
                    self.go(z[0],z[1])
                    self.put(z[2])
                    self.back(z[0],z[1])
        sim.simxFinish(self.clientID)


def main():
    ex = Bot()
    ex.execute()


if __name__ == '__main__':
    main()
```