UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

TRABAJO DE FIN DE GRADO

IMPLEMENTACIÓN DE COMPORTAMIENTOS SOCIALES PARA ROBOTS MÓVILES

ESCUELA DE INGENIERÍAS INDUSTRIALES

DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA

GRADO EN INGENIERÍA ELECTRÓNICA, ROBÓTICA Y MECATRÓNICA

Autor: Francisco José Anguita Chamorro

Tutor: Javier González Monroy

Cotutor: Cipriano Galindo Andrades

Málaga, Septiembre 2023

Agradecimientos

A mi tutor Javier y a mi cotutor Cipriano, sin su ayuda no podría haber terminado este trabajo.

A todos los amigos que he hecho por el camino y que me han acompañado hasta aquí.

A mi familia y en especial a mis padres, que me animaron a seguir cuando más quise darme la vuelta.

RESUMEN

A lo largo de este proyecto se estudia, desarrolla e implementa una serie de algoritmos que doten a un robot móvil de un comportamiento más cercano a algunos de los estándares sociales más aceptados. El código que dota al robot de dicho comportamiento está escrito en C++ usando ROS.

El proyecto se ha desarrollado en el entorno de simulación CoppeliaSim, un simulador capaz de incluir personas en la escena para poder realizar todas las pruebas. En el proyecto se implementan una serie de nodos agrupados en un paquete que ejecutan los distintos algoritmos que afectan al comportamiento del robot reduciendo su velocidad, haciendo que mantenga las distancias o evitando que cruce entre dos personas que mantienen una conversación.

PALABRAS CLAVE: área proxémica, navegación, mapa de coste, ROS, CoppeliaSim, C++

ABSTRACT

This Project consists on the study, development and implementation of a series of algorithms that allow a mobile robot to behave in a way that is closer to some accepted social standards. The code that provides the robot with this kind of behaviour is written in C++ using ROS.

The Project has been developed in the simulation environment CoppeliaSim, a simulator in which it is possible to include people in the scene to test all the functionality. For the project, certain packages and nodes have been implemented to execute the algorithms that modify the robot usual navigation to reduce its movement speed, keep the distance or prevent it from interrupting two people that may be having a conversation.

KEY WORDS: proxemic area, navigation, costmap, ROS, CoppeliaSim, C++

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

Hoy día la robótica está presente en una gran variedad de entornos, y los continuos avances en tecnología hacen que esta tendencia no pare de crecer.

Pese a la reticencia de algunos sectores de la población, especialmente los más mayores, menos familiarizados con la tecnología [1], a la adopción de robots en ambientes cada vez más cercanos a lo cotidiano, la robótica se está instalando en ámbitos muy variados [2].

La industria es el sector en el que más acostumbramos a encontrar robots en los entornos de trabajo. Los autómatas y robots manipuladores realizan desde hace ya años las tareas repetitivas más fáciles de automatizar. Un ejemplo muy común en la industria son los robots manipuladores KUKA, como el robot KR 1000 titan. Este manipulador ofrece muchas posibilidades como robot de paletizado o en combinación con ejes lineales.

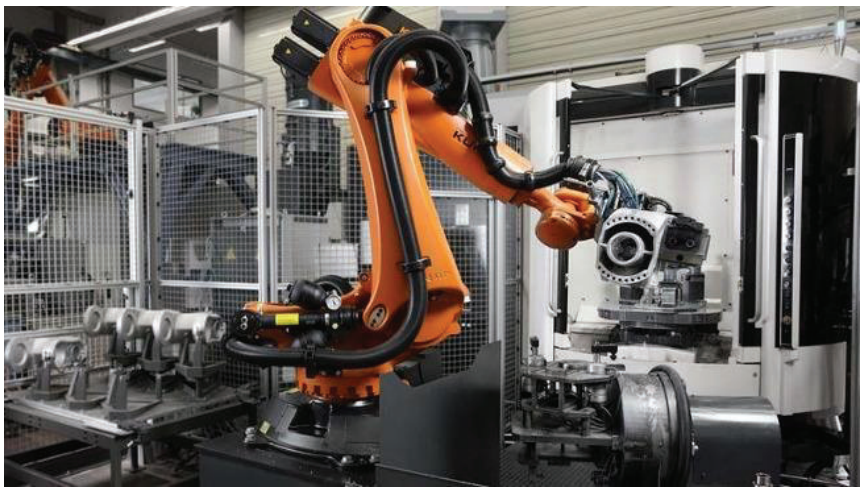

Figura 1. Robot KUKA KR 1000 titan [3]

La medicina es otro campo que nadie extraña ver asociado a la robótica. Desde pequeños robots que sirven de ayuda al movimiento articular en procesos de rehabilitación de pacientes como el Lokomat de Hocoma (ver Figura 2), hasta complejos robots quirúrgicos que realizan tareas en complicadas operaciones en el caso del robot da Vinci (ver Figura 3), los hospitales no paran de incorporar y hacer pruebas con robots de todo tipo [4].

Figura 2. Robot Lokomat de Hocoma [4]

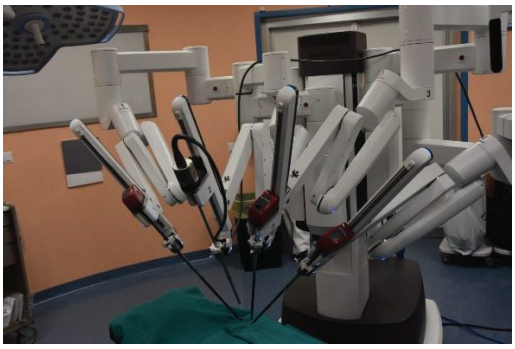

Figura 3. Robot quirúrgico da Vinci [4]

También en los hospitales encontramos robots con un trato aún más cercano con los pacientes, que proporcionan monitorización del estado de esos pacientes, realizan pequeñas tareas e incluso sirven de asistencia social para pacientes mayores que puedan sentirse especialmente solos [5][6].

En hostelería se están incorporando robots móviles que realizan la función de servicio en mesas. Como ejemplo de este tipo de robots tenemos el BellaBot de Pudu Robotics, que transporta comandas a las mesas y reproduce frases para ayudar a los clientes a identificar qué deben coger o si están obstruyendo su paso para preguntarles de forma educada si pueden dejarle pasar [7].

Figura 4. Robot Bellabot de Pudu Robotics [7]

La Robótica Social y la Interacción Humano-Robot surgen de las necesidades que implica la finalidad social de los robots que se programan hoy en día [8]. Estas dos ramas se centran en el estudio de los comportamientos sociales de los seres

humanos para posteriormente replicarlos los más fielmente posible en robots que vayan a trabajar en entornos que los obliguen a mezclarse con otras personas.

La Proxémica estudia cómo percibimos y qué relaciones establecemos a través del espacio y la distancia que ponemos entre nosotros mismos y otras personas o las cosas que nos rodean [9]. Como parte de la semiótica, es decir, del conjunto de signos que usamos para comunicarnos, es importante tener en cuenta los posibles significados que se suelen dar a las distancias interpersonales a la hora de relacionarnos con otras personas.

Del estudio de estas distancias, surge el concepto de 'Área Proxémica' [10], comúnmente conocida como "espacio personal", que cada persona usa para delimitar el espacio a su alrededor que no debe ser invadido para que tenga lugar una interacción social satisfactoria que no incomode al usuario.

En robótica se tiene en cuenta esta área y el estudio de la proxémica a la hora de diseñar algoritmos de navegación o de acercamiento a personas e interacción con ellas [11] para evitar siempre que las acciones del robot resulten invasivas. Acciones comúnmente consideradas de mala educación y que rompen con lo establecido como aceptable en el ámbito social pueden ser acercarse demasiado a una persona, hablar a alguien que está de espaldas, pasar por medio de dos personas que hablan [12], no mirar a una persona cuando habla o interrumpirla mientras está haciéndolo [13]. Este tipo de malas conductas se usan como punto de partida en el desarrollo de proyectos de robótica social.

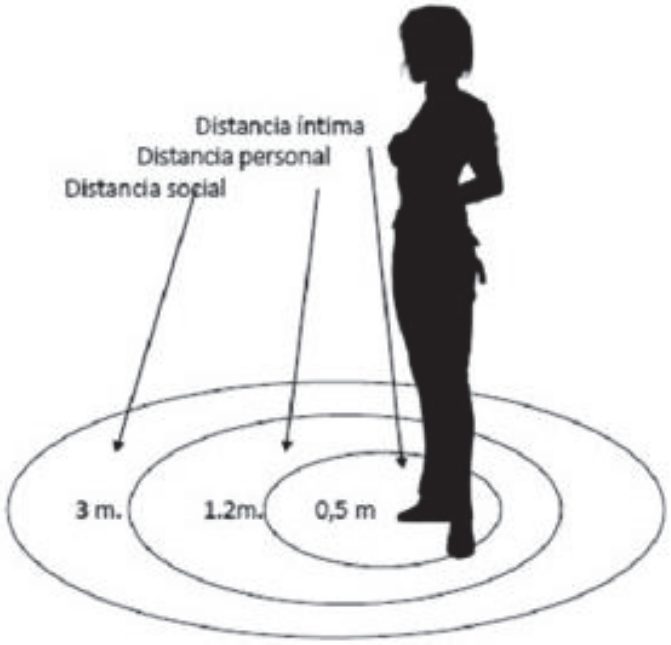

Figura 5. Ejemplo de Área Proxémica [10]

1.1 MOTIVACIÓN

La presencia de robots en ambientes en los que comparten un espacio con personas obliga a considerar durante el diseño y la programación de estas máquinas el cómo su presencia puede afectar al desarrollo de las actividades que las personas que los rodean están llevando a cabo.

Aunque la opinión popular no se opone por completo a la adopción de robots en los diversos entornos en los que se están implantando, sí que manifiestan cierta preocupación en algunos asuntos como seguridad o ética [14].

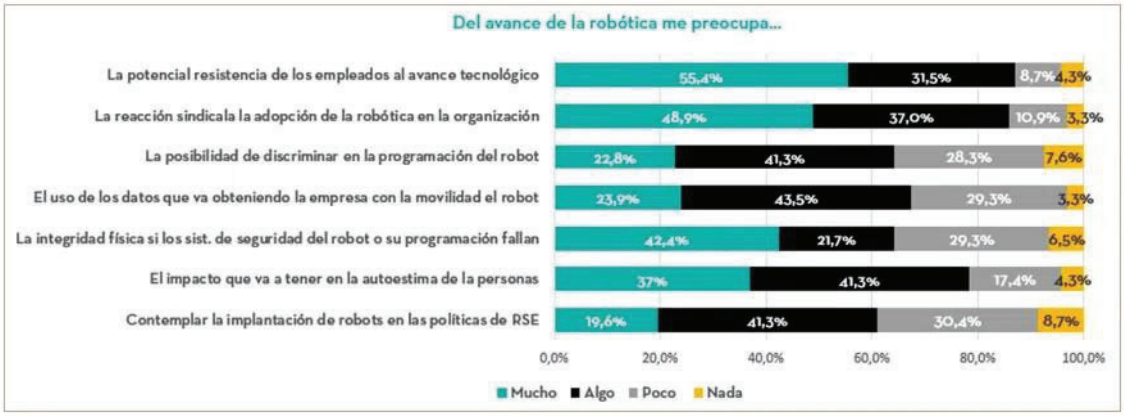

Figura 6. Gráfico de respuestas a la pregunta 'Del avance de la robótica me preocupa…' [14]

Un robot que navegue teniendo en cuenta la presencia de personas y que adapta su comportamiento a las situaciones que puedan darse al moverse en un ambiente social facilitaría su integración en el entorno y despejaría muchas dudas de las personas que se muestran recelosas a la colaboración con robots.

1.2 OBJETIVOS

El objetivo primero de este trabajo es el diseño de una serie de algoritmos que modifiquen el comportamiento básico de un robot cuya navegación está completamente resuelta. Estos algoritmos modifican la navegación autónoma del robot cuando se detecta presencia humana, teniendo en cuenta sus áreas proxémicas y adaptando la navegación para mejorar su integración social.

Este objetivo primario se concretará en la implementación de tres comportamientos principales que se consideran básicos para un robot que se desenvuelve en entornos sociales. Por un lado, se dotará al robot de la capacidad de ajustar dinámicamente su velocidad si detecta la presencia de personas. Por otro lado, se incorporará la detección de posibles interacciones entre personas para

evitar que el robot pase por medio de ellas y provoque una interrupción. Finalmente, se desarrollará un algoritmo que modifique la navegación básica del robot para ceder el paso en el caso de detectar una persona mientras se encuentra en un espacio estrecho como podría ser un pasillo.

El estudio del paquete estándar de navegación para su modificación mediante los algoritmos correspondientes conforma un objetivo secundario de este proyecto.

Para la consecución de estos objetivos generales, se plantean los siguientes objetivos parciales:

- Estudio de los algoritmos de navegación autónoma.
- Análisis y selección de comportamientos sociales a integrar en el robot móvil.
- Diseño de las distintas escenas que simulen las situaciones en las que se pondrá a prueba el comportamiento del robot.
- Implementación de métodos de interacción social sobre la arquitectura robótica.
- Puesta a prueba de la implementación y verificación experimental de los resultados.

1.3 ESTRUCTURA DE LA MEMORIA

La estructura que seguirá la memoria será la siguiente:

En este primer capítulo se ha hecho una introducción a los conceptos generales más relevantes en torno a los cuales se encuadra la temática de este trabajo, así como su propósito principal.

En el capítulo 2 se darán algunos apuntes sobre el estado del arte de la robótica social, cómo se está aplicando y de qué manera se desea aplicar en un futuro próximo.

En el capítulo 3 se enlistarán las distintas herramientas usadas en el desarrollo del proyecto, dando una breve explicación del funcionamiento de las mismas para establecer cuál es el punto de partida del trabajo.

El capítulo 4 expondrá los distintos comportamientos que se han perseguido, así como el desarrollo de los algoritmos y nodos que los implementan y el diseño de los escenarios en los que serán puestos a prueba.

En el capítulo 5 se muestran los resultados de las pruebas realizadas en simulación.

El capítulo 6 establecerá las conclusiones extraídas durante el trabajo y los posibles trabajos futuros que puedan realizarse.

Cierra la memoria un único anexo dedicado a la estructura de un repositorio público de GitHub en el que se puede encontrar todo el código desarrollado para

este trabajo, así como todos los archivos e instrucciones de instalación necesarios para replicar este proyecto en cualquier entorno.

2. ESTADO DEL ARTE

El objetivo de este trabajo es la selección e implementación de comportamientos sociales y su verificación mediante un entorno de simulación que facilite la experimentación en ambientes sociales artificiales. Por ello, se describe en esta sección el estado del arte de estos dos campos: la robótica social y la simulación en robótica.

2.1 ROBÓTICA SOCIAL

Se han mencionado ya algunos de los campos en los que la inclusión continuada de robots ha forzado a introducir en la ecuación de su diseño las variables relacionadas con la coexistencia de personas y máquinas en los entornos de trabajo.

A continuación, se enumeran algunas de las líneas de investigación sobre las que se está estudiando actualmente:

- LA CONFIANZA EN LOS ROBOTS

Con los robots cada vez más presentes en cualquier entorno, es importante tener en cuenta lo que la gente que va a compartir espacio con ellos piensa sobre estas máquinas.

Sobre la capacidad de los robots de llevar a cabo las tareas para las que son diseñados no quedan apenas dudas [15] [16]. Tanto es así, que en el campo de la interacción humano-robot, se ha demostrado empíricamente que las personas tendemos a sobrevalorar las capacidades de un robot cuando se nos presenta. De manera contraproducente, esto resulta en un descenso de la confianza cuando comprobamos que efectivamente el robot no es tan resolutivo en las interacciones como nos habíamos podido imaginar [17].

El interés actual versa más sobre la disposición de las personas a convivir con unas máquinas que empiezan a mezclarse con ellos en el día a día. La impresión que la máquina vaya a tener sobre las personas se tiene en cuenta desde las primeras fases del diseño del robot.

Un primer planteamiento muy común es el de diseñar un robot lo más parecido posible a un ser humano para aumentar así la confianza en el mismo, ya que de esta forma resulta en una figura menos desconocido. Está demostrado que esto es cierto solo hasta cierto punto. Los robots que imitan en gran medida las características humanas pero que siguen siendo muy distinguibles de los humanos suelen generar una menor confianza que los que adoptan formas humanoides menos fieles a nosotros. Esta confianza se recupera cuando el robot se vuelve prácticamente indistinguible de una persona [17].

Este fenómeno que da a la curva de confianza una forma de 'U' a la percepción humana de los robots antropomorfos se conoce como 'Uncanny Valley' (ver Figura 7) y tiene implicaciones muy importantes al abordar el diseño de un robot [18]

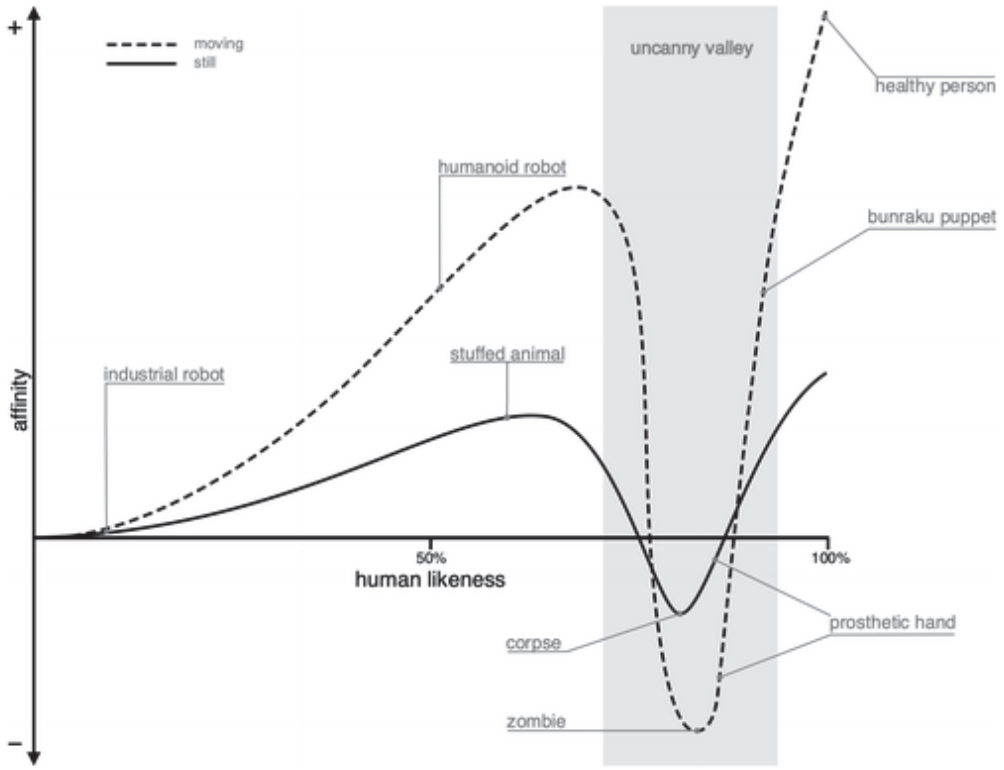

Figura 7. La hipótesis del *Uncanny Valley* [18]

- ASISTENCIA ROBÓTICA EN ÁREAS DE MEDICINA

La medicina es uno de los campos en los que más se está explotando el uso de robots. Los hospitales son lugares que inducen gran estrés, y en ambientes donde los riesgos de infección son elevados, un robot puede presentar grandes ventajas. La pandemia que siguió a la aparición del COVID-19 reforzó la investigación en esta línea para buscar formas de reducir el contacto humano cuando se vuelve potencialmente peligroso.

Numerosos experimentos se han llevado a cabo para poner a prueba los efectos que la robótica de asistencia puede tener en pacientes.

El caso de los pacientes de cáncer es especialmente delicado. Estos pacientes están expuestos a tratamientos generalmente prolongados en el tiempo que les inducen un estado de estrés y depresión que debe ser tratado con delicadeza. Se ha comprobado que tanto la presencia de robots humanoides [19] como robots que imitan a animales y se comportan como mascotas [20] durante los tratamientos ayuda a reducir esos niveles de estrés y malestar en los pacientes más pequeños. En

el caso de los más mayores, aunque los resultados dan lugar a algunas dudas, siguen siendo positivos en la mayor parte de los estudios [21].

Experimentos muy parecidos se han llevado a cabo también con pacientes de casuísticas muy distintas como pueden ser los de autismo [22]. La naturaleza mental del autismo hace que sea muy difícil trazar líneas generales de terapia, pero el uso de un robot ofrece la posibilidad de emular situaciones sociales concretas de las que los pacientes pueden aprender habilidades muy difíciles de conseguir en ambientes sociales que no estén controlados.

- EMOCIONES EN LA INTERACCIÓN HUMANO-ROBOT

Una línea de investigación que además de interés científico tiene gran repercusión mediática entre la gente es la del diseño de robots que se parezcan cada vez más a los seres humanos.

En este sentido, ya se han logrado grandes avances que han permitido la fabricación de robots casi idénticos a los humanos en su forma y en los gestos que realiza [23]. El siguiente paso es dotar a estas máquinas de una característica tan humana y tan impropia de estas máquinas como las emociones.

Figura 8. Robot Erica de Hiroshi Ishiguro Laboratories

La robótica social es un campo multidisciplinar que estudia las reacciones emocionales, cognitivas, sociales y físicas de las personas a las interacciones humano-robot [24]. Los estudios empíricos que se llevan a cabo en esta área versan sobre cómo lograr que los robots expresen emociones, cómo los humanos detectamos esas emociones y el efecto que estas pueden tener sobre nosotros.

Los métodos usados para lograr emociones en los robots se clasifican en estáticos y dinámicos.

Los métodos estáticos parten de una codificación previa de las emociones, usando arquitecturas basadas en scripts predefinidos [25] o espacios emocionales [26].

Los métodos dinámicos usan algoritmos de detección para reconocer emociones en los humanos e imitarlas de la manera más fiel posible [24]. Estos métodos son más efectivos, aunque es difícil llevar a cabo aprendizajes de este tipo dado lo mucho que puede variar la forma de expresar una emoción de una persona a otra.

- PROXÉMCIA EN ROBÓTICA

La incorporación de la proxémica a la robótica es esencial para la coexistencia de humanos y máquinas. En la actualidad se usan distintos métodos para hacer que los robots consideren este factor en su navegación e interacción con las personas.

Uno de estos métodos consiste en el uso de un algoritmo de preferencia proxémica basado en la consideración de 21 puntos situados alrededor del usuario (ver Figura 9)

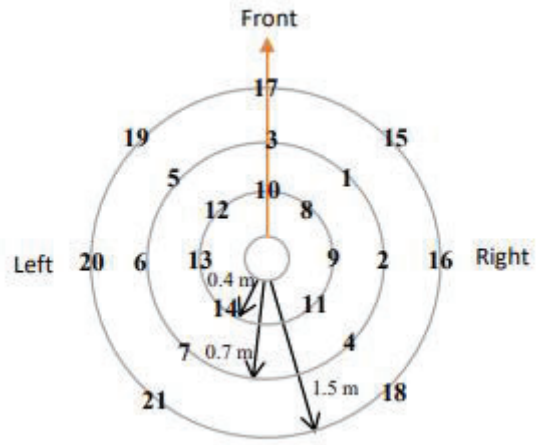

Figura 9. Conjunto de puntos usados para una planificación proxémica de la navegación

Basándose en información contextual como la predisposición de la persona a interactuar con la máquina o el tipo de interacción que se quiere llevar a cabo, el planificador selecciona uno de estos puntos para aproximarse a la persona [27].

La lógica difusa es otro mecanismo usado en la implementación de proxémica [28]. A raíz del estudio de la percepción humana del comfort y de las reacciones de personas que comparten espacio con robots, se han diseñado modelos consistentes en reglas difusas y parámetros que sirven para modificar dinámicamente las áreas proxémicas para que estas sean las más apropiadas en todo momento y se adapten perfectamente a cada situación [29].

Un ejemplo claro que ilustra hasta qué punto está extendida la proxémica en la robótica son las 'social layers' de ROS. ROS, uno de los estándares de robótica más importantes en la actualidad, incluye un conjunto de capas de mapa de coste que añaden esta variable social a la ecuación de la navegación y la planificación y facilita la inclusión de esta característica en cualquier trabajo sobre robótica social.

2.2 SIMULACIÓN EN ROBÓTICA

La implantación de robots requiere de un arduo proceso de diseño y realización de pruebas. Tradicionalmente, la única manera de llevar a cabo pruebas y ajustes era disponer de la propia máquina. Usar una máquina real para pruebas no solo es caro, sino en muchas ocasiones ni siquiera es posible.

Las necesidades investigadoras en este ámbito junto con otras necesidades de índole económica han llevado a un fuerte desarrollo de los simuladores en robótica, tanto de los propios robots como de los entornos en los que estos operan.

Además de las ventajas ya mencionadas, el uso de simuladores permite un proceso de desarrollo mucho más seguro, ya que nunca se está expuesto al espacio de maniobra del robot real; un desarrollo con un abanico de posibilidades más amplio, debido a la multitud de escenarios que se pueden plantear en simulación y que podrían no ser replicables con facilidad en el mundo real; y un desarrollo en el que se generan más datos que pueden ser posteriormente utilizados en otros procesos de aprendizaje autónomo [30].

A continuación (ver Figura 10) se muestran algunos de los simuladores más populares hoy día junto con la frecuencia (número de veces) con la que han sido citados en artículos sobre simulación entre los años 2016 y 2020. Destacan entre ellos los simuladores *Gazebo, MuJoCo* y *CoppeliaSim.*

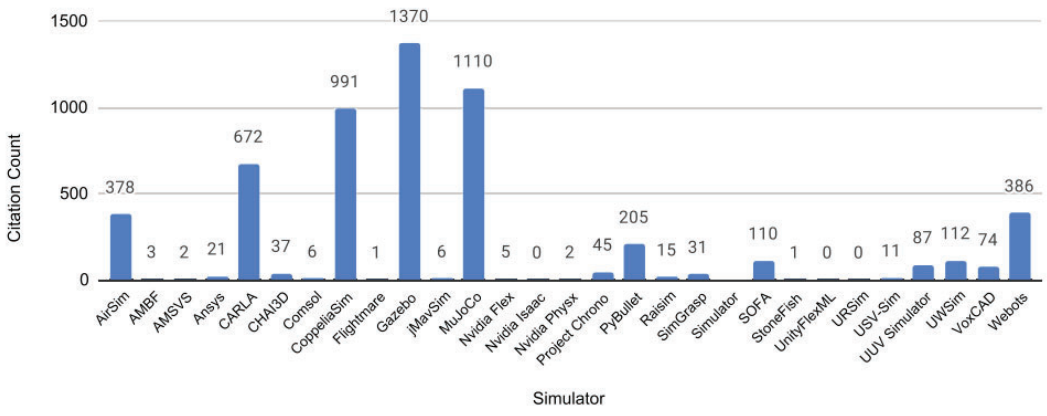

Figura 10. Simuladores más citados en artículos sobre simulación en robótica

Gazebo es un simulador que prácticamente desde su inicio fue incorporado al *framework* de ROS. ROS ('Robotic Operating System') es un estándar de programación para aplicaciones de robótica muy extendido y utilizado actualmente como referente en este campo. El simulador Gazebo, por tanto, goza de desarrollo y mantenimiento por parte de *Open Source Robotics Foundations*, los mismos desarrolladores de ROS [31].

Esta característica, que lo hace totalmente compatible con el citado paradigma de programación en robótica (es, de hecho, el simulador por defecto que se instala junto a ROS), junto con las posibilidades que ofrece para simular en la nube y su potente motor de físicas, lo han convertido en el simulador más usado para trabajar con ROS.

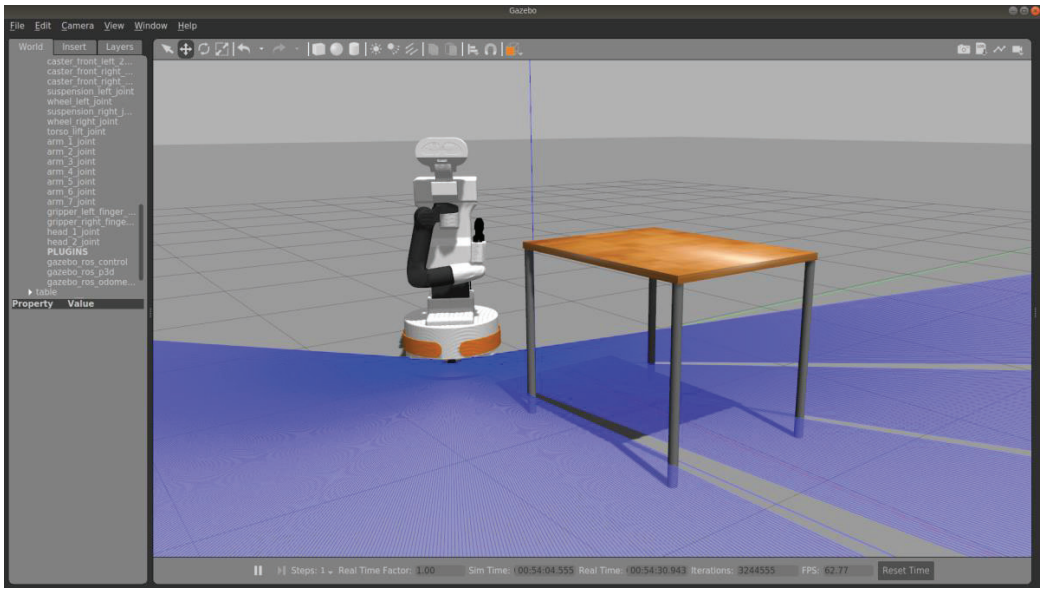

Figura 11. Interfaz de simulación de *Gazebo*

El simulador CoppeliaSim (anteriormente V-REP), pese a no estar tan directamente ligado a ROS, presenta otras ventajas como su flexibilidad, al tener soporte en Windows, Linux y Mac además de la posibilidad de usar 7 lenguajes de programación distintos, o la posibilidad de interactuar con el entorno en tiempo de ejecución [32].

Aunque CoppeliaSim no dispone de muchas de las herramientas y APIs prediseñadas que sí pueden ser usadas con Gazebo, presenta mucha más facilidad a la hora de insertar, editar y manipular modelos tanto al recrear el robot como al rodearlo de un entorno de pruebas. Esto permite, por ejemplo, trabajar con mayor facilidad con modelos que simulen el comportamiento de personas.

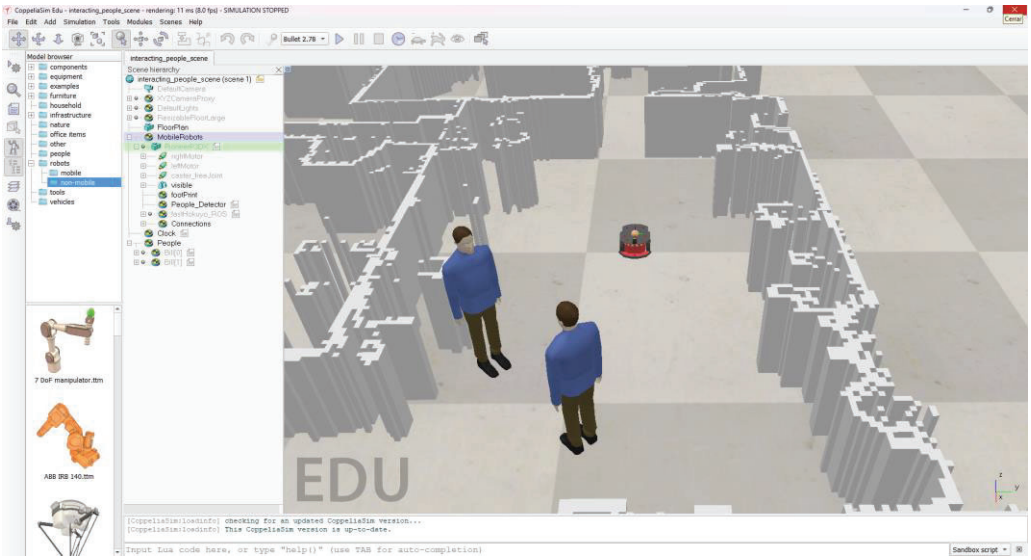

Figura 12. Interfaz de simulación de CoppeliaSim

En la historia más reciente, la simulación en robótica se ha empezado a apoyar también en los motores que se usan en los diseños de videojuegos.

Software como Unity o Unreal Engine tienen una potencia de modelado excepcional al estar específicamente diseñadas para ello, por lo que son cada vez más utilizadas en el diseño de escenarios que serían imposibles de recrear con los simuladores creados para la robótica como los descritos anteriormente.

Como ejemplo para ilustrar el uso de estos motores, en mayo de 2023 un equipo de investigadores en China llevó a cabo un experimento en el que simulaban un Pez Robótico Biomimético que se desplazaba por el agua para optimizar el seguimiento de su trayectoria [33]. En el desarrollo, usaron Unreal Engine para el diseño del entorno dinámico, y AirSim, un sistema desarrollado por Microsoft, para la experimentación con el vehículo autónomo que hacía de pez.

Figura 13. Modelo de Pez Robótico Biomimético

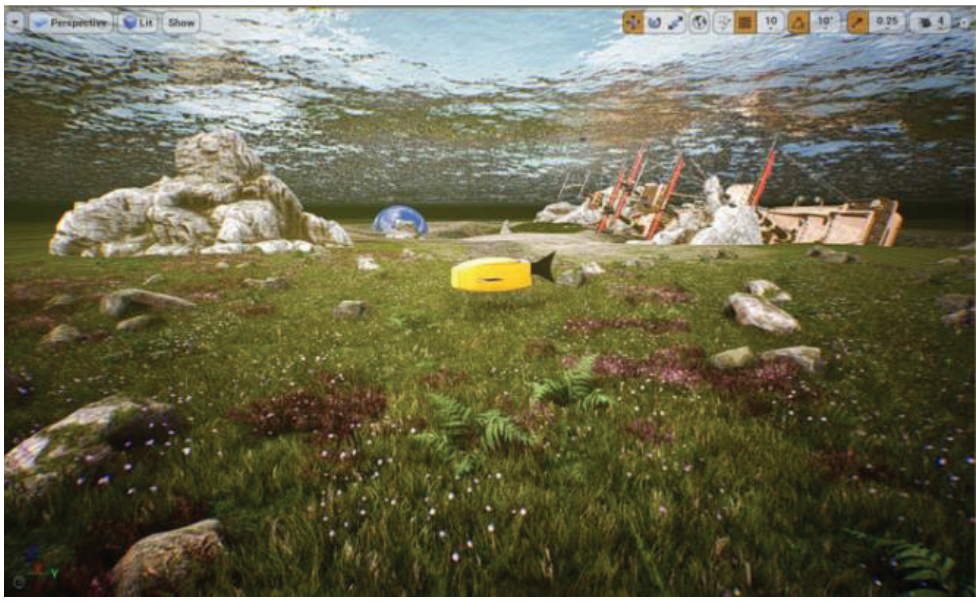

Figura 14. Entorno subacuático diseñado con Unreal Engine

Tras haber explorado las distintas opciones actuales y su disponibilidad, para este trabajo se ha seleccionado el simulador CoppeliaSim. Este simulador es completamente gratuito, y compatible con Linux y Windows al mismo. Además, presenta grandes facilidades a la hora de simular tanto la presencia de personas, como algunas de sus acciones básicas, como también su detección. La experiencia previa en el trabajo con este entorno ha sido otro factor considerado en la elección.

3. HERRAMIENTAS UTILIZADAS

A lo largo de este tercer capítulo se hace un repaso de todas las herramientas que han sido usadas para la realización de este trabajo. En primer lugar, se habla del estándar de programación ROS, en torno al cual gira todo el desarrollo de código implicado en el proyecto. Después, se centra la atención en el paquete de navegación 'move_base', que permite que el robot se mueva. Finalmente, se da una explicación sobre el simulador CoppeliaSim y cómo puede ser configurado para funcionar con ROS.

El objetivo de este capítulo es sentar una base de las tecnologías implicadas en el proyecto sobre la que se argumentarán las decisiones tomadas respecto al diseño, y establecer de forma clara cuál es el punto de inicio del que se partió para desarrollar el trabajo.

3.1 ROS

ROS (*Robot Operating System*) es un conjunto de librerías y herramientas software de código abierto diseñadas para el desarrollo de aplicaciones para la robótica [34].

Desde que se empezara a trabajar en su desarrollo en 2007, ROS se ha ido construyendo con la idea de crear un estándar de programación en robótica, que sirviera para el mayor número de tipos de máquina y para que se pudiera trabajar con el mayor número de lenguajes de programación [35].

Todas estas ventajas, sumadas a las facilidades que ofrece ROS para crear réplicas digitales gracias a las múltiples herramientas de simulación y visualización que pone a nuestra disposición, han llevado a grandes empresas de robótica a la adopción total de este *framework* para el desarrollo de sus aplicaciones.

Estos son también los motivos por los que se ha elegido ROS como la herramienta principal sobre la que desarrollar este trabajo.

- FUNCIONAMIENTO GENERAL DE ROS

ROS funciona mediante un sistema de publicación-suscripción. Lanzar ROS es, en esencia, abrir un broker de mensajería llamado *master* que se encarga de registrar todos los nodos que se van ejecutando, los topics a los que van a publicar información y los topics a los que se suscriben para recibirla.

Dos nodos quedan conectados cuando uno se suscribe al topic en el que publica el otro. Una vez quedan conectados, no es necesario mantener el master abierto para su funcionamiento.

- NODOS Y PAQUETES

Un nodo es la unidad de código ejecutable elemental de ROS. Un nodo es una aplicación que queda registrada en un *master* de ROS y que lleva a cabo una serie de funciones. Por norma general, estas funciones consisten en la recepción de unos datos entradas mediante la suscripción a uno o más *topics* para realizar con ellos determinadas operaciones y producir una salida que envía mediante un proceso de publicación en otro *topic*.

Los nodos se agrupan en paquetes. Un paquete reúne código fuente junto con otros archivos de configuración para conformar el nivel atómico de ROS [36]. Los nodos no pueden funcionar sin un paquete que especifique las condiciones de ese funcionamiento.

Los paquetes de ROS, a su vez, están contenidos en un espacio de trabajo o *workspace*. El *workspace* aporta un entorno en el que poder compilar el código fuente (generalmente, *source*) para producir el código ejecutable.

3.1.1 EL PAQUETE 'MOVE_BASE'

El paquete move_base es el componente principal de la pila de navegación básica que proporciona ROS para las aplicaciones que consisten en desplazar un robot por un entorno. En ese sentido, move_base hace de interfaz para la configuración e interacción con esta pila de navegación [37].

Este paquete contiene una serie de nodos que realizan todas las operaciones necesarias para calcular comandos de actuación para cualquier robot que publique en los topics a los que están suscritos los nodos de move_base la información necesaria para su funcionamiento.

La figura 15 muestra los nodos que se configuran para lanzar move_base junto con la información que necesita del robot en cuestión para poder calcularle los comandos de movimiento

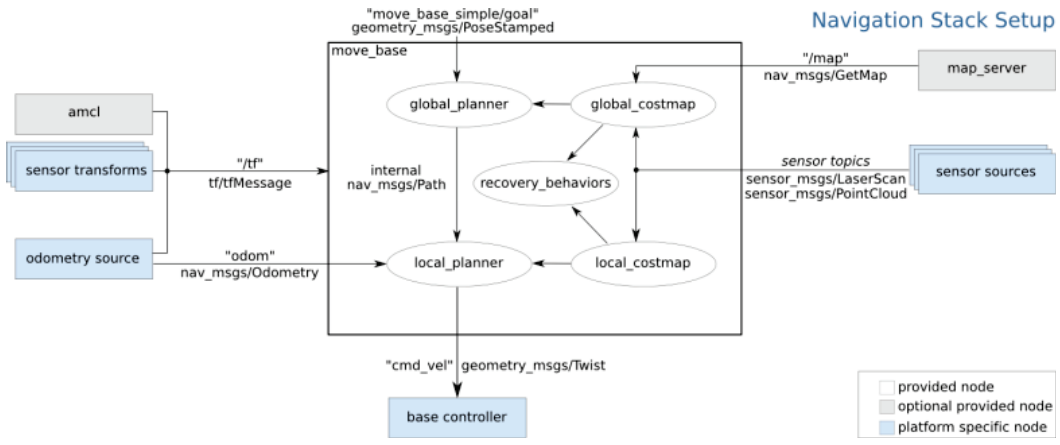

Figura 15. Esquema de funcionamiento del paquete move_base

Los nodos de los paquetes que se lanzan con move_base están suscritos a una serie de topics por los que reciben la información del robot. Estos nodos se pueden dividir en dos grupos, los mapas de coste y los planificadores.

3.1.2 MAPAS DE COSTE Y EL PAQUETE 'COSTMAP2D'

Los mapas de coste son mallas de puntos que se colocan sobre el mapa de un entorno para situar los obstáculos y asignar, en base a la posición de estos obstáculos y a los parámetros con los que se haya configurado, un coste a cada uno de los puntos.

En este proyecto, se ha trabajado con el paquete 'costmap2d' para la implementación de un mapa de coste en dos dimensiones. Este paquete recibe información de una serie de sensores y la usa para construir una malla de ocupación. Sobre esta malla, aplica una inflación para asignar un coste final a cada celda o punto que conforma la malla.

Este mapa asigna a cada celda un valor de coste entre 0 y 255, siendo 0 el coste mínimo y 255 el máximo. Sin embargo, lo que hace realmente el mapa es etiquetar las celdas en función del coste asignado. De esta manera, una celda puede marcarse como ocupada, libre o sin información.

Sobre las celdas que quedan marcadas como ocupadas (es decir, las que conforman un obstáculo) se lleva a cabo un proceso de inflación con el que se propaga el coste desde el obstáculo hasta un radio de inflación que se ajusta al configurar el mapa de coste, decreciendo el coste con la distancia [38]. Teniendo en cuenta el coste asignado por la inflación y el tamaño del robot, se etiquetan de nuevo las celdas con cuatro rangos de coste:

- Coste letal: asignado a las celdas que conforman los obstáculos. De encontrarse el centro del robot en esa celda, supondría una colisión evidente.
- Coste inscrito: asignado a las celdas situadas a una distancia menor que el radio inscrito del robot de un obstáculo. Este coste sigue siendo muy elevado ya que también supondría colisión.
- Coste posible circunscrito: asignado a las celdas situadas a una distancia menor que el radio circunscrito del robot. Si el centro del robot se situara en alguna de estas celdas, dependería de la orientación del robot que estuviera colisionando o no.
- Coste de espacio libre: todos los costes situados en este rango son equivalentes al coste 0, ya que en ningún caso el robot estaría colisionando.

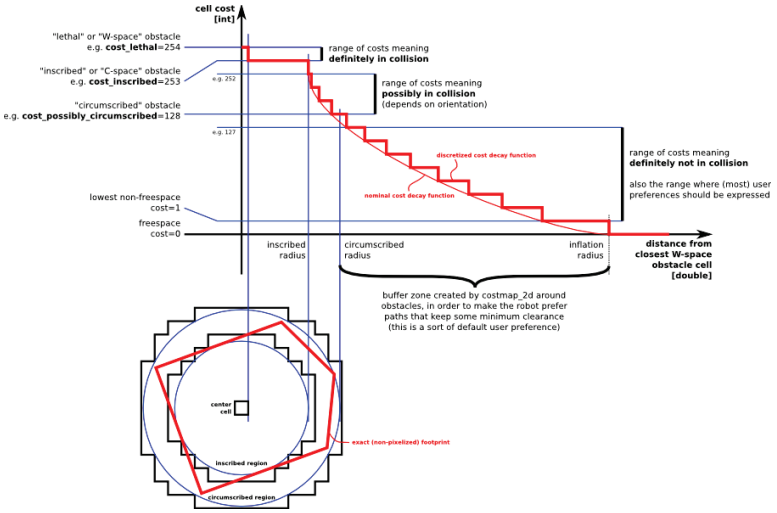

Figura 16. Propagación de coste mediante inflación

- MAPAS GLOBAL Y LOCAL

Al lanzar move_base, en realidad se configuran dos mapas de coste. El mapa de coste global y el mapa local.

El mapa de coste global parte de un mapa para situar las celdas ocupadas de todo el entorno. Es un mapa de coste estático que no varía y tiene en cuenta los obstáculos de dimensión mayor que se entiende que no se van a mover como paredes o muebles de gran tamaño. El nodo global_costmap de la pila de navegación recibe un objeto de tipo mapa y envía información sobre la ubicación de los obstáculos tanto al planificador global como al mapa de coste local.

Figura 17. Mapa de coste global del entorno usado para la simulación

El mapa de coste local es un mapa de coste de dimensión menor que recibe los datos del entorno del robot por medio de sus sensores para elaborar una malla de lo que rodea al robot y que incluye obstáculos que no han sido previstos por el mapa de coste global para enviarla al planificador local.

Figura 18. Ejemplo de mapa de coste local del entorno usado para la simulación

- LAYERED COSTMAPS (MAPAS DE COSTE POR CAPAS)

La complejidad que supone el tener en cuenta las distintas naturalezas de los diferentes obstáculos que podían darse en un mismo entorno da lugar a un gran volumen de operaciones que son necesarias para calcular un mapa de coste, valga la redundancia, muy costoso.

Los mapas de coste por capas conforman un método de separación de los mapas de coste según el tipo de obstáculo que se vaya a encontrar en el entorno [39]. De esta forma, se pueden calcular mallas menos pesadas para un mismo entorno y cuyos pesos se superponen en cada celda para dar lugar al mapa de coste final.

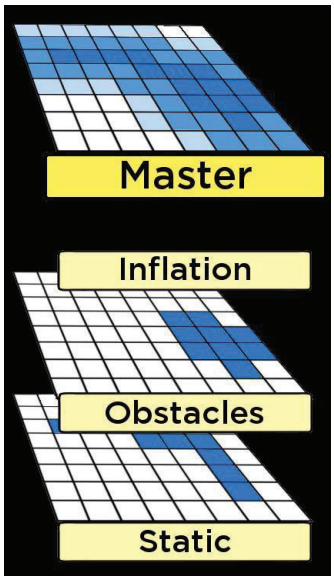

Figura 19. Mapa de coste por capas

El paquete 'costmap2d' calcula un mapa de coste por capas que debe ser configurado por separado para los casos del mapa de coste global y local. Este enfoque permite, por ejemplo, separar la operación de inflación de la de etiquetado de celdas ocupadas en cada uno de los mapas, y añadir otras capas con funcionalidades específicas como son las 'social layers'.

- LAS *SOCIAL LAYERS* DE ROS

ROS ya incluye una implementación de capas de mapa de coste que tienen en cuenta restricciones de índole social como el área proxémica en el cómputo de los costes [40].

Estas capas están contenidas en un paquete de ROS llamado 'social_layers', que incluye tres implementaciones:

- La capa 'socialLayer': esta capa conforma una clase padre con toda la configuración y atributos comunes para las otras dos capas. En esta capa se configura, entre otras cosas, el topic al que se suscriben las capas para obtener la información sobre la ubicación de personas, por defecto '/people'.
- La capa 'proxemicLayer': la capa proxémica hereda de la capa 'social_layer' y sobreescribe los métodos de cálculo de coste para dibujar un área proxémica de costes aumentados en torno a las personas detectadas (ver Figura 20). La capa implementada también tiene en cuenta la posibilidad de que una persona esté en movimiento. Cuando ese es el caso, modifica la forma del área para estirarla en la dirección del movimiento, transformándola en una elipse.

  Esta capa se configura con 5 parámetros que ajustan la forma y el tamaño del área proxémica,

1. ‘amplitude’: determina el tamaño del área proxémica
2. ‘covariance’: determina la forma (excentricidad) del área
3. ‘cutoff’: determina el valor mínimo que debe alcanzar el coste de un punto del área para ser tenido en cuenta en el cómputo del mapa final.
4. ‘factor’: determina en qué medida afecta el hecho de que la persona esté en movimiento al modificar la forma del área
5. ‘enable’: determina si la capa está activa o no

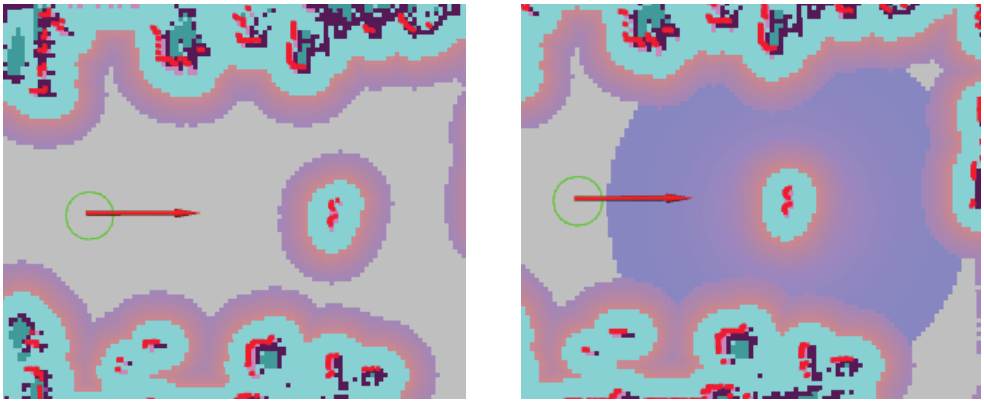

Figura 20. Coste asignado a una persona sin capa proxémica (izq) y con ella (dcha)

- La capa ‘passingLayer’: esta capa hereda igualmente de la capa social e implementa la misma funcionalidad que la capa proxémica, pero añadiendo un coste extra a la derecha de las personas detectadas para asegurarse de que el robot siempre pase por la izquierda de las personas.

3.1.3 PLANIFICADORES

Los planificadores ejecutan los algoritmos de generación de trayectorias y navegación necesarios para producir los comandos de movimiento para el robot.

Para llevar a cabo la operación final del cálculo, el planificador necesita la siguiente información:

- La posición del robot. El robot debe publicar su odometría, la estimación que realiza de su propia ubicación. Esta posición es el punto de inicio de la trayectoria, y es relativa a un sistema de referencia.
- El objetivo del robot. Este es el punto final de la trayectoria y también se da en coordenadas relativas a un sistema de referencia.
- Las transformadas correspondientes. Dado que tanto el punto de inicio como el final de la trayectoria son relativos, el robot deberá publicar las transformadas correspondientes entre su sistema de referencia y los de su entorno.

- Los mapas de coste. El planificador necesita saber por dónde puede pasar el robot y por dónde no de camino a su objetivo para trazar la trayectoria final.

- PLANIFICADORES GLOBAL Y LOCAL

Al igual que sucede con los mapas de coste, move_base divide la planificación en global y local.

El planificador global usa la información relativa al objetivo, el mapa de coste global y las transformadas para calcular una primera trayectoria que solo tiene en cuenta los obstáculos estáticos.

Figura 21. Trayectoria trazada por el planificador global (línea verde)

El planificador local se sirve de la información de los obstáculos dinámicos que le proporciona el mapa de coste local para modificar la trayectoria inicial calculada por el planificador global. El planificador local es quien finalmente publica los comandos de actuación para el robot.

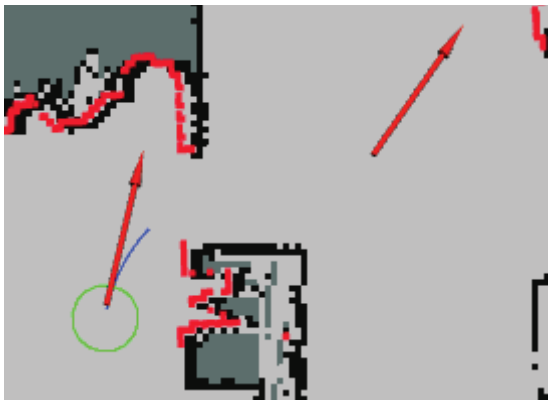

Figura 22. Trayectoria trazada por el planificador local (línea azul)

3.1.4 EL PAQUETE 'DYNAMIC_RECONFIGURE'

El paquete 'dynamic_reconfigure' proporciona una interfaz estándar de ajuste de parámetros de configuración en tiempo de ejecución para los nodos de ROS [41].

Algunos nodos como los mencionados de mapas de coste o planificación se ejecutan con una lista de parámetros a los que se asigna un valor en un archivo de configuración que se carga al lanzar el nodo.

Muchos de estos parámetros, como los límites de velocidad impuestos por el planificador o la amplitud de la inflación de los costes, se configuran como parámetros dinámicos que pueden ser modificados en tiempo de ejecución ejecutando los comandos *dynparam get* y *dynparam set.*

3.2 COPPELIASIM

CoppeliaSim es un simulador de robótica basado en una arquitectura de control distribuida usado para el desarrollo de algoritmos, pruebas de automatizaciones, robótica educativa y monitorización remota entre otras aplicaciones.

La versatilidad de la herramienta, que permite el control de sus objetos y modelos mediante nodos de ROS que pueden estar escritos en C++ o Python, unida a la facilidad con la que se pueden incluir personas a la simulación y emular algunos de sus comportamientos, son las razones por las que se ha decidido usar este entorno para la simulación en este trabajo.

3.2.1 ESCENAS Y SIMULACIÓN

Las escenas de Coppelia son entornos en los que se ubican distintos objetos que se usan para simular las condiciones de funcionamiento del robot.

Los objetos y modelos que se introducen en una escena pueden llevar asociado un script con código que defina alguna de sus propiedades o realice algunas funciones o tareas, como desplazar al objeto por la escena o hacer que publique cierta información [42].

La figura 23 muestra una escena en la que se ha usado un mapa como base para generar una serie de paredes que harán de obstáculo para un robot que deberá navegar por el mapa evitando también las personas que se mueven por el mismo entorno.

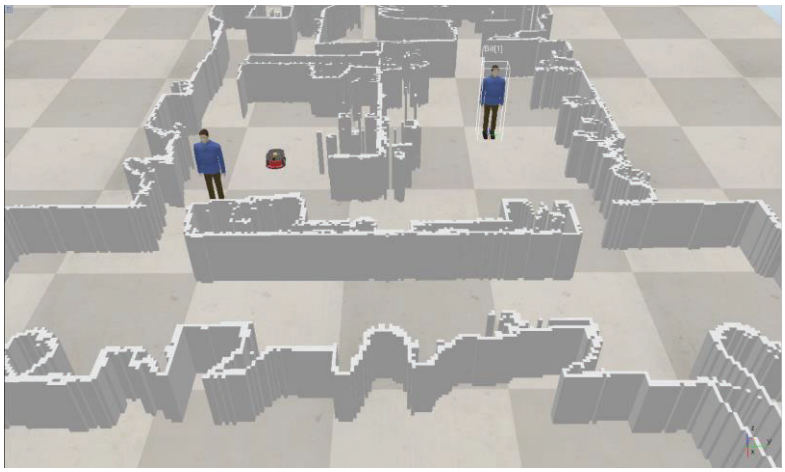

Figura 23. Ejemplo de escena en CoppeliaSim

3.2.2 CONEXIÓN ROS-COPPELIA Y EL PAQUETE 'SIMROS'

Para poder controlar los distintos objetos de Coppelia mediante nodos de ROS, existe en Coppelia el paquete 'simROS'. Este paquete proporciona al simulador toda la funcionalidad necesaria para permitir a los objetos participar del sistema de mensajería por publicación-suscripción de ROS.

Gracias a este paquete y a través de los scripts que se pueden asociar a los objetos, estos pueden publicar y suscribirse a topics como si se tratara de un nodo más. Así, un modelo de un robot situado en una escena podría publicar información relativa a su posición o a los obstáculos que detecta mediante sus sensores al tiempo que se suscribe a otro topic por el que le llegan instrucciones de movimiento.

4. DISEÑO Y DESARROLLO

4.1 CONFIGURACIÓN DEL ENTORNO DE SIMULACIÓN

Para todo el desarrollo, se ha dispuesto de un robot cuya navegación y evasión de obstáculos está completamente resuelta. Se asume también que el robot posee la capacidad de detectar personas. El robot que se ha simulado es un modelo Pioneer de Coppelia. Este robot posee un láser en su parte frontal que realiza un barrido para obtener información de lo que le rodea.

El robot se mueve en un entorno generado a partir de un mapa del laboratorio 2.3.7 de la ETSI Informática de la Universidad de Málaga, ya que este laboratorio es un espacio en el que el robot tendría que lidiar con la presencia constante de gente trabajando mientras se desplaza. En la figura que se presenta a continuación (ver Figura 24) se puede apreciar también cómo está orientado el sistema de referencia correspondiente a la base del robot, con los sentidos positivos de los ejes X, en rojo, Y, en verde, y Z, en azul.

Figura 24. Modelo del robot en simulación y su sistema de referencia

En su script asociado, el modelo del robot ha sido programado para publicar usando el paquete simROS de Coppelia toda la información relativa a su posición, los datos recogidos por el láser y las transformadas necesarias para relacionar su sistema de referencia con el del mapa y estos con el sistema de referencia del láser. Toda la información se publica en los topics a los que se suscriben los nodos de la pila de navegación que se lanzan al configurar move_base y que se muestran en la Figura 15. Estos topics son

- /tf: en este topic se publican todas las transformadas.
- /laser_scan: en este topic se publica la información del láser.

- /odometry, /base_pose_ground_truth: en estos topics se publican la estimación que el robot hace de su posición y su posición real.

Para la detección de personas, se ha simulado un detector que publica en el topic /people_detection un mensaje con la posición y orientación de las personas que entren en el rango de detección simulado.

De esta forma, el robot de la simulación y la pila de navegación de ROS quedan totalmente conectados, de manera que para cualquier objetivo enviado a move_base, se generarán los comandos de velocidad para que el robot siga la trayectoria calculada y alcance su objetivo. El robot recibe estos comandos suscribiéndose al topic /cmd_vel.

4.2 CONFIGURACIÓN DE LA PILA DE NAVEGACIÓN

Se detallan también las configuraciones de la pila de navegación que se han usado junto con la simulación para el desarrollo y la realización de pruebas.

Estas configuraciones se encuentran recogidas en una serie de ficheros en formato 'yaml' (ver Anexo A) que ajustan los parámetros con los que se lanzan los mapas de costes y los planificadores inicialmente.

Con los mapas de coste se han configurado los siguientes elementos:

- **El radio del robot**: que se ha establecido en los 26 cm de radio del modelo del Pioneer usado en simulación (ver Figura 23).
- **La distancia de detección de obstáculos**: aunque el robot detecte un obstáculo con el láser, no lo considerará en el mapa de coste a menos que se encuentre a una distancia inferior a 3.5 metros.
- **Las capas del mapa de coste global**: se ha incluido una capa estática construida a partir del mapa del laboratorio que considera obstáculos fijos como paredes y mesas, y una capa de inflación para esos obstáculos.
- **Las capas del mapa de coste local**: entre las que se cuentan una capa de obstáculos dinámicos que plasma la información obtenida del láser junto con su correspondiente capa de inflación, y la capa social proxémica de ROS. Siguiendo el estudio de la Universidad Rey Juan Carlos junto con la Universidad de León sobre arquitecturas cognitivas usando proxémica [43], se considera que las distancias adecuadas que debe mantener el robot respecto a las personas son de 0.6 m mínimo cuando la persona tiene una actitud positiva hacia el robot, 0.9 m cuando su actitud es neutral, y 1.6 m cuando la actitud es negativa. Considerando que el robot se ha simulado dentro de un laboratorio de robótica, se considera una actitud generalmente neutral hacia él, y se ha configurado la capa proxémica para que el robot mantenga una distancia de al menos 0.9 m siempre que sea posible.

4.3 COMPORTAMIENTOS IMPLEMENTADOS

En el desarrollo del trabajo se han considerado los siguientes comportamientos a implementar:

1. Ajustar de la velocidad en presencia de personas: una alta velocidad de desplazamiento resulta en movimientos bruscos que pueden provocar rechazo en las personas que comparten espacio con el robot.
2. No interrumpir una interacción entre dos personas: es ampliamente considerado como un gesto de mala educación pasar por medio de dos personas que intentan mantener una conversación.
3. Modificar la navegación para ceder el paso en espacios estrechos: las personas siempre tendrán prioridad en el espacio que comparten con el robot, por lo que este debería ser capaz de apartarse y dejar pasar a las personas cuando no se disponga de espacio para que pasen ambos.

4.3.1 ADAPTACIÓN DINÁMICA DE LA VELOCIDAD DE NAVEGACIÓN EN PRESENCIA DE PERSONAS

El primer comportamiento que se ha implementado es el del ajuste de la velocidad del robot en función de la presencia de personas y de su distancia a las mismas.

Para la implementación se ha programado el nodo de ROS 'speed_reconfigure' que se suscribe al topic detector de personas de personas (/people_detection) y al topic de la posición del robot (/base_pose_ground_truth).

El nodo primero comprueba si se han detectado personas evaluando que el tamaño del vector de personas sea mayor de uno. En caso de que se cumpla la condición, recorre dicho vector calculando las distancias del robot a cada una de las personas usando la ecuación (1)

$$distancia = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (1)$$

En la que x2, y2 y z2 son las tres componentes del vector de posición del robot y x1, y1 y z1 corresponden a la posición de la persona detectada.

El algoritmo se queda con la menor de las distancias calculadas para tener en cuenta a la persona más cercana y computa la nueva velocidad aplicando la ecuación (2)

$$velocidad = 0.15 * distancia\_min \quad (2)$$

La distancia máxima a la que el robot es capaz de detectar una persona son 4 metros, de manera que el coeficiente por el que se multiplica la distancia se ha ajustado para que la velocidad máxima a la que se puede desplazar el robot sea de unos 0.6 m/s, siguiendo las pautas descritas en un artículo publicado por la Universidad de Oxford sobre percepción de la velocidad del movimiento en nuestro entorno [44]. Dado que la intención de este nodo no es incidir de manera directa en la navegación, no se ha implementado la posibilidad de que el robot quede detenido por completo en este nodo.

Una vez calculada la nueva velocidad, el nodo llama a la función 'system()' del estándar de C++ para ejecutar un único comando en una sesión de Shell para lanzar el paquete 'dynamic_reconfigure' y establecer el nuevo valor del parámetro 'max_trans_vel' del planificador.

La velocidad del robot no se puede configurar de manera directa, ya que depende de muchos factores que el planificador tiene en cuenta, como la distancia a otros obstáculos y los pesos del mapa de coste. Es por esto que se ha decidido que este nodo actúe sobre la velocidad máxima con la que se configura el planificador.

4.3.2 NO INTERRUMPIR UNA INTERACCIÓN ENTRE PERSONAS

La implementación de este segundo comportamiento dota al robot de la capacidad de discernir si dos personas pueden estar interaccionando de alguna forma y, de esta manera, asegurarse de que su navegación no las interrumpe.

En este caso, se ha programado el nodo 'avoid_interruption', que se suscribe solo al topic de detección de personas /people_detection.

Cuando recibe un mensaje, el algoritmo que ejecuta el nodo comprueba esta vez que se hayan detectado al menos dos personas mirando el tamaño del vector de personas, y vuelve a usar la ecuación (1) pero esta vez para calcular la distancia que las separa entre ellas. Si la distancia calculada es menor que un mínimo configurado en el nodo, pasará a comprobar sus orientaciones.

CoppeliaSim publica la orientación de los objetos en formato de cuaternio. Para una mayor facilidad de interpretación, el nodo convierte el cuaternio de orientación en ángulos de Euler.

Para los cálculos relativos a la orientación de las personas, el nodo toma el tercer ángulo, llamado normalmente 'yaw' por convenio. Este ángulo representa el giro respecto del eje Z, que se corresponde con el eje vertical en el entorno usado para la simulación.

El nodo descompone los ángulos de la orientación de las personas detectadas en sus componentes en los ejes X e Y para poder calcular el producto escalar. A partir del resultado del producto escalar, el nodo decide si existe la posibilidad de que las dos personas se estén mirando. Como se ilustra a continuación (ver Figura 25), dos

vectores que puedan apuntarse el uno al otro siempre darán lugar a un producto escalar negativo.

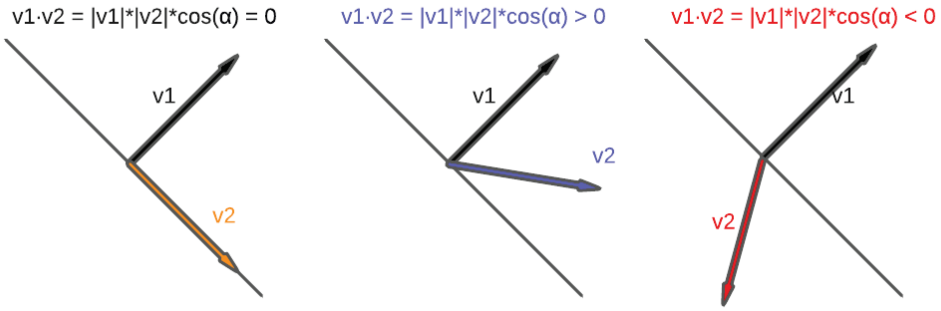

Figura 25. Resultado del producto escalar en función del ángulo que forman los vectores

Sabiendo esto, si el producto escalar calculado por el nodo resulta negativo, considerará que las personas pueden estar interactuando, y calculará, conociendo sus posiciones, el punto medio entre ellas para inflar el coste de la zona intermedia y asegurarse de que el planificador no traza una ruta que pueda interrumpir a estas personas.

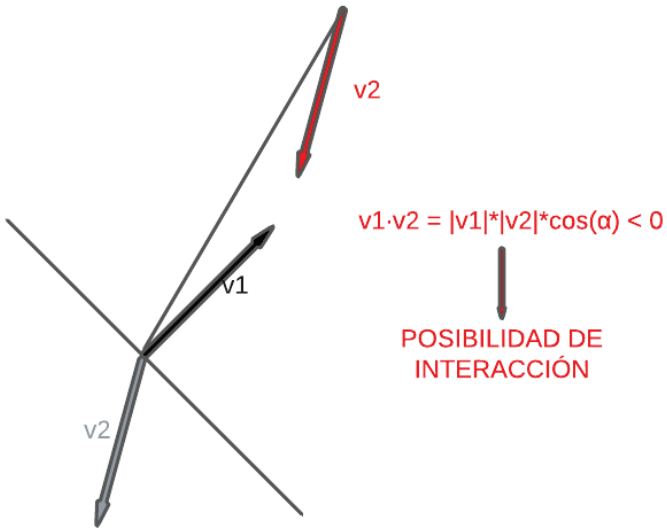

Figura 26. Situación de posible interacción ente personas que se miran

Se ha elaborado un diagrama de flujo (ver Figura 27) que ilustra el funcionamiento del algoritmo que ejecuta el nodo

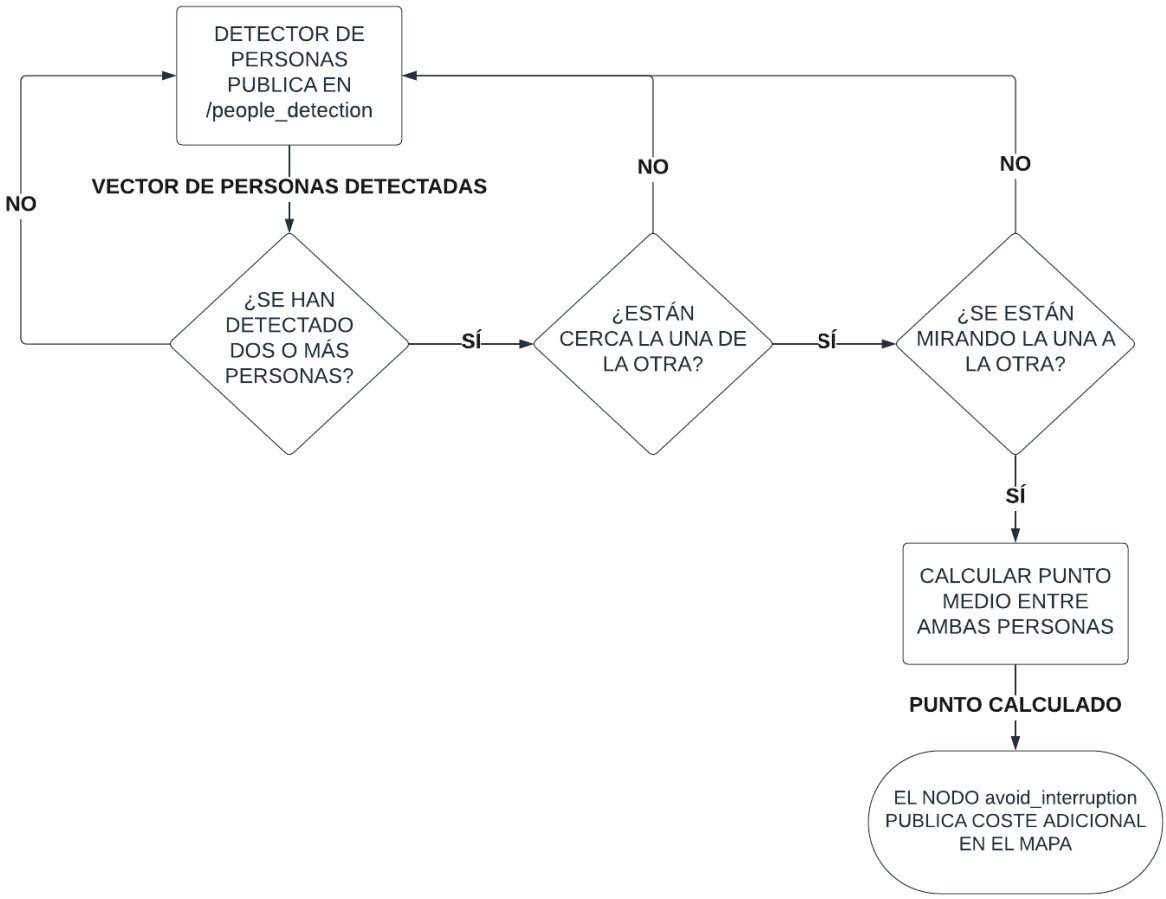

Figura 27. Diagrama de flujo del nodo 'avoid_interruption'

4.3.3 MODIFICAR LA NAVEGACIÓN PARA CEDER EL PASO

La modificación más directa sobre el algoritmo de navegación la realizará el tercer nodo 'narrow_space_bhv'. Su algoritmo comprueba en todo momento si el robot se encuentra en un espacio estrecho y pone en pausa su navegación para ceder el paso si encuentra a una persona en una zona con estas características.

Para lograr esto, el nodo se suscribe, además de al topic de detección de personas (recordamos, /people_detection), al topic en el que el láser del simulador publica sus lecturas de distancia, /laser_scan.

Cuando recibe un mensaje del láser, el nodo calcula la media de distancia medida en un rango dentro del barrido del láser correspondiente a las zonas situadas a la izquierda y derecha del robot.

Teniendo en cuenta que el barrido del láser toma en total 684, y que realiza el barrido de derecha a izquierda, se han usado las medidas correspondientes a las posiciones de la 70 a la 100 del vector de distancias publicadas por el láser para el

lateral derecho del robot, y las medias de las posiciones de la 582 a la 612 para el lado izquierdo.

Si la suma de la media de las distancias de ambos lados no supera un cierto límite, el robot considerará que se encuentra en un espacio estrecho y procederá al cálculo de una posición de seguridad.

Para el cálculo de la posición de seguridad el algoritmo del nodo considera un rango nuevo de medidas que conforman un cono adicional por delante del anterior. Con esas medidas de lo que el robot tiene inmediatamente delante, se vuelve a calcular una media para saber por cuál de los dos lados el robot tiene más espacio.

El algoritmo selecciona el lado con la media de distancia mayor (y por tanto con más espacio), y a esa media le resta el radio del robot más una pequeña distancia extra de margen. Ese número será la coordenada Y de la nueva posición del robot. Para la coordenada X, simplemente se sumará una distancia de 1 metro a la posición del robot. Adelantando un poco la posición se consigue que el robot no tenga que desplazarse de manera totalmente lateral, suavizando la trayectoria hacia la nueva posición. La siguiente figura (ver Figura 28) ilustra de forma esquemática cómo se realiza el cálculo descrito

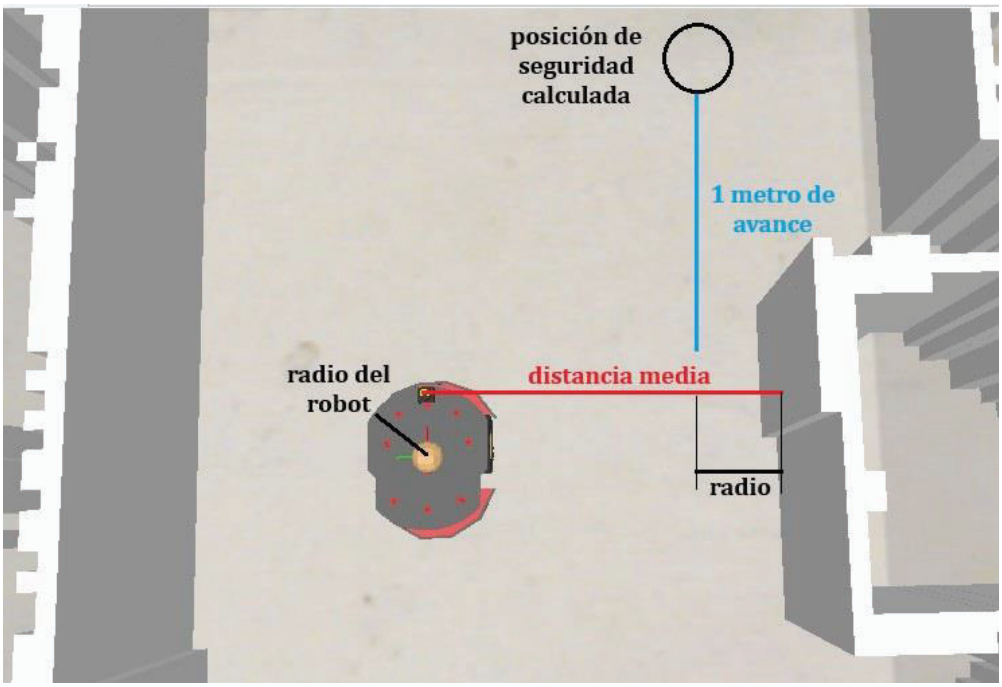

Figura 28. Esquema del cálculo de la posición de seguridad

Esta posición de seguridad se recalcula cada vez que se publica información del láser y el robot decida que está en un espacio estrecho. Si mientras está en un espacio así, el detector de personas publica una detección, el nodo publicará esta posición en el topic /move_base_simple/goal del paquete de navegación.

Esta publicación cancelará la navegación del robot y la sustituirá por el nuevo objetivo. Cuando el robot alcance la nueva posición, al haberse publicado esta última por el topic de objetivo simple de move_base, la ausencia de un nuevo objetivo que perseguir detendrá la navegación hasta que deje de detectarse la persona en cuestión.

Cuando la persona haya pasado y el detector de personas vuelva a publicar un vector vacío, el robot recuperará sus objetivos anteriores y la consiguiente navegación.

Se ha incluido un diagrama de flujo (ver Figura 29) que refleja el funcionamiento del algoritmo descrito.

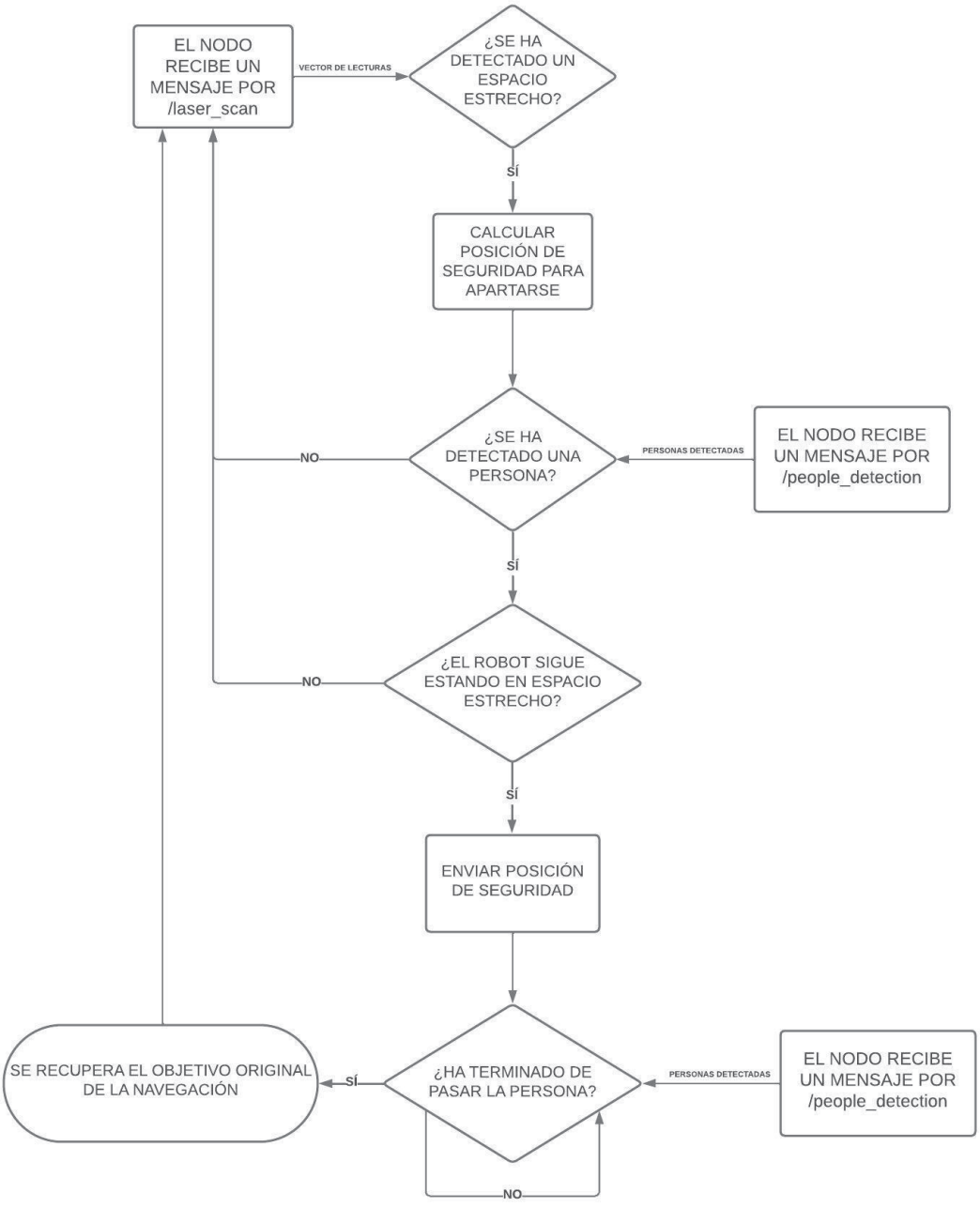

Figura 29. Diagrama de flujo del nodo 'narrow_space_bhv'

5. PRUEBAS Y SIMULACIÓN

Se han realizado una serie de experimentos con el fin de verificar el funcionamiento de los nodos desarrollados.

Para cada nodo se ha diseñado una escena de Coppelia que incluye personas para comprobar que se han logrado los comportamientos deseados.

5.1 ADAPTACIÓN DINÁMICA DE LA VELOCIDAD DE NAVEGACIÓN EN PRESENCIA DE PERSONAS

En esta primera prueba, se han situado dos personas en una de las estancias del mapa de laboratorio (ver Figura 30). El robot atravesará la sala, reduciendo su velocidad al aproximarse a las personas y recuperándola al pasarlas.

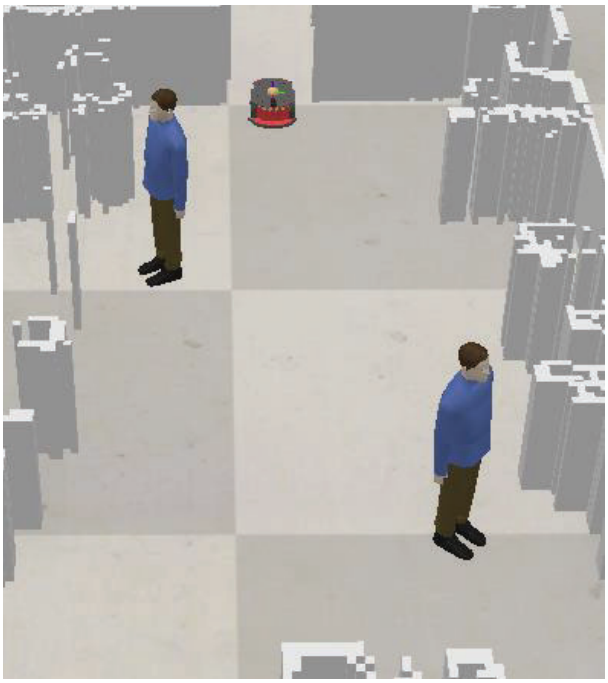

Figura 30. Escena para la prueba del nodo de ajuste de velocidad

El perfil de velocidades del robot a lo largo del recorrido realizado a través de la habitación se muestra a continuación (ver Figura 32)

Figura 31. Recorrido del robot a lo largo de la escena simulada

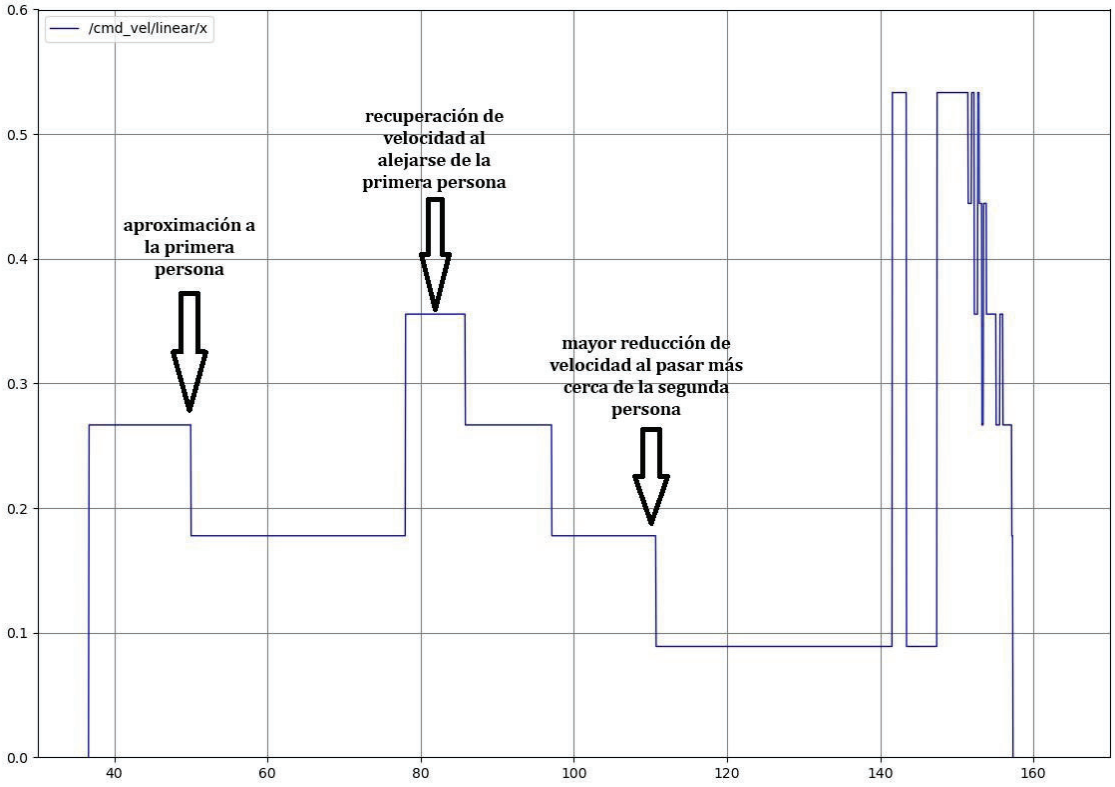

Figura 32. Perfil de velocidades ajustadas por el nodo durante la navegación

En la Figura 30 se comprueba cómo la velocidad del robot ha disminuido hasta poco menos de 0.2 m/s cuando ha pasado junto a la primera persona. Tras la primera disminución, no ha llegado a recuperar toda la velocidad al encontrarse ya a una distancia moderadamente corta de la segunda persona. La reducción a su paso cerca de la segunda persona es mayor ya que pasa a una distancia menor de esta que de la primera persona (ver Figura 31), llegando a ponerse por debajo de los 0.1 m/s.

Después de rebasar a esta última, ha alcanzado su velocidad máxima segundos antes de volver a reducirla al aproximarse al punto objetivo de la navegación. En ningún momento el robot a rebasado los 0.6 m/s de velocidad máxima impuesta por el nodo, ni siquiera en el tramo final.

5.2 NO INTERRUMPIR UNA INTERACCIÓN ENTRE DOS PERSONAS

La escena para la simulación de este caso incluye dos personas enfrentadas que el robot pueda interpretar como que puedan estar interactuando (ver Figura 33).

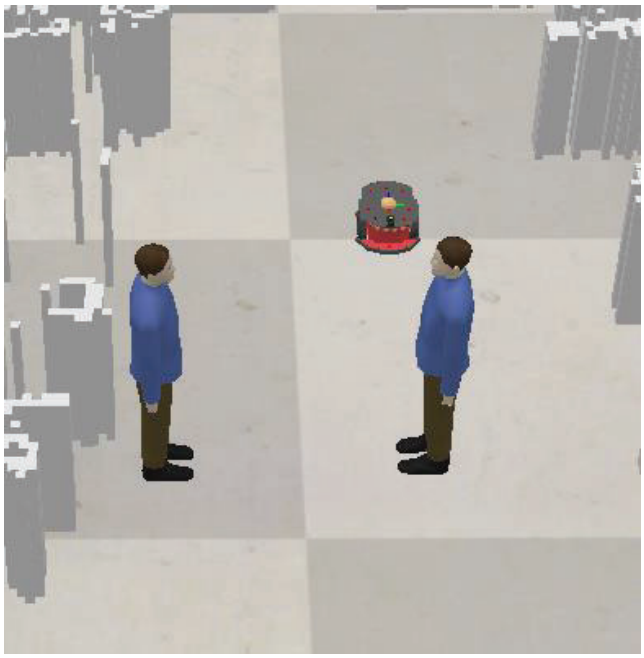

Figura 33. Escena para la prueba del nodo de evasión de interrupciones

Se ha lanzado la simulación en primer lugar sin ejecutar el nodo. El coste que la capa proxémica asigna a las personas con la configuración predeterminada es el siguiente (ver Figura 34)

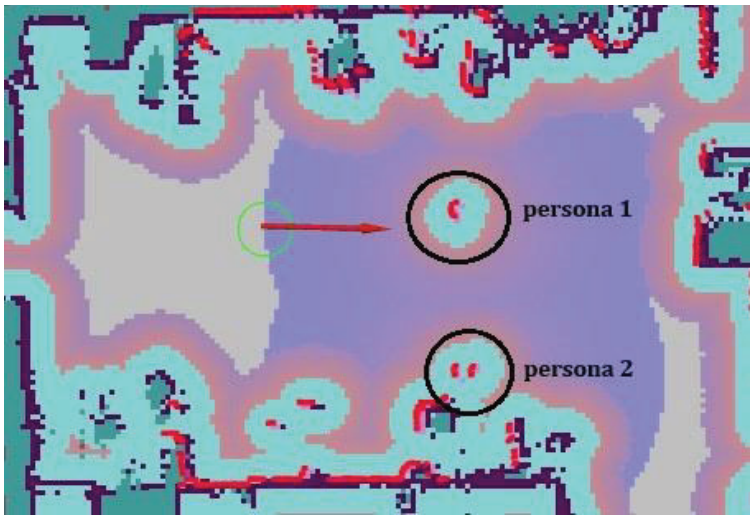

Figura 34. Área de coste de las personas con la configuración predeterminada

Se comprueba que este coste no es suficiente para impedir que el planificador guíe al robot por el espacio intermedio entre las dos personas en su cálculo de la trayectoria óptima (ver Figura 35)

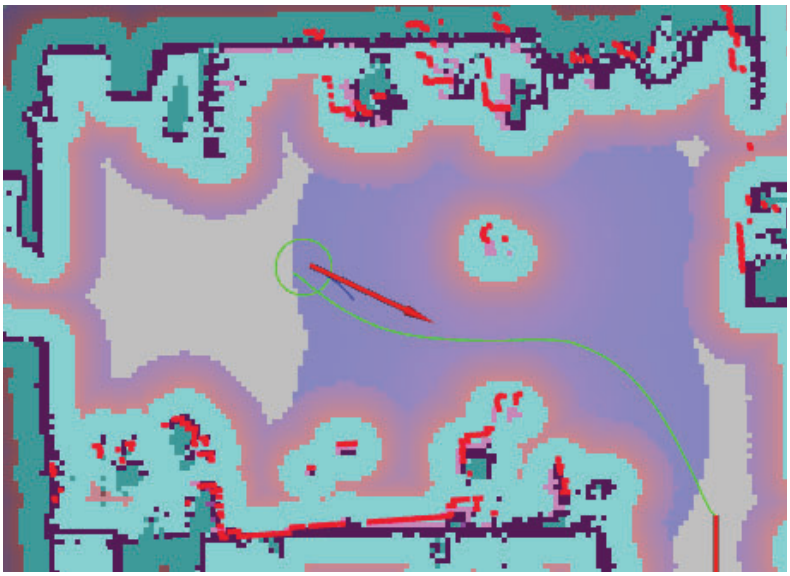

Figura 35. Trayectoria planificada con el coste predeterminado

A continuación, se muestra el mapa resultante del cómputo del coste total teniendo en cuenta el coste adicional introducido por el nodo al detectar que las dos personas podrían estar teniendo una conversación (ver Figura 36)

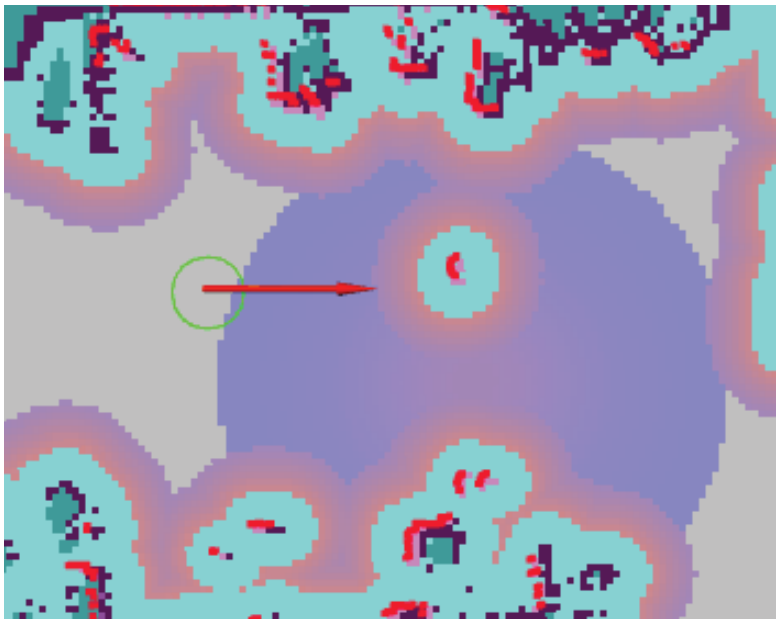

Figura 36. Mapa de costes expandido teniendo en cuenta la interacción entre personas

Esta vez la trayectoria sí evita la zona intermedia y rodea la pareja de personas para alcanzar su objetivo sin interrumpirlas (ver Figura 37)

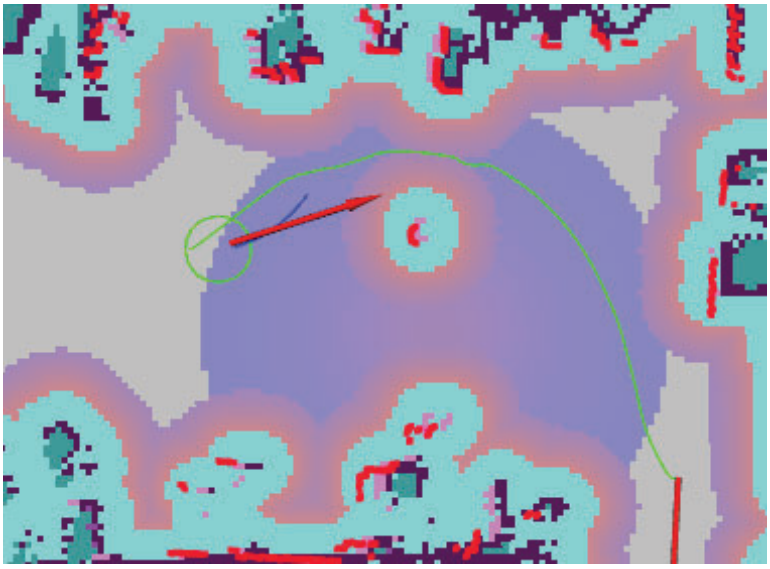

Figura 37. Trayectoria planificada con coste ampliado

5.3 MODIFICAR LA NAVEGACIÓN PARA CEDER EL PASO

Para verificar el funcionamiento de este nodo se ha dispuesto una escena en la estancia correspondiente al pasillo del laboratorio. En este pasillo, una persona camina hacia delante y el robot se encuentra con ella, viéndose obligado a apartarse hasta que la persona haya pasado (ver Figura 38)

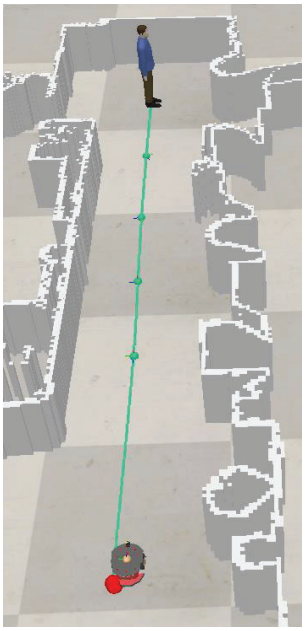

Figura 38. Escena diseñada para las pruebas del nodo

Al robot se le comanda moverse hasta el final del pasillo (ver Figura 39)

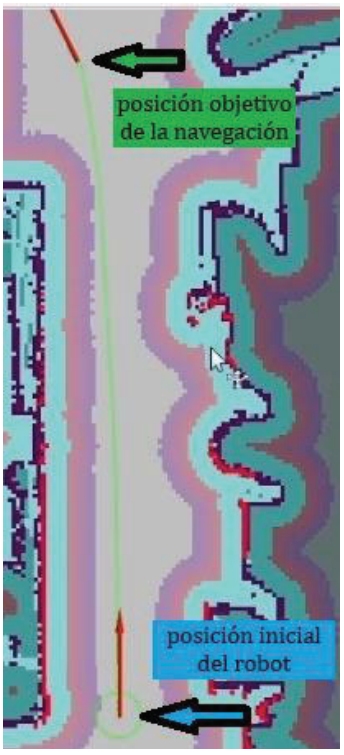

Figura 39. Recorrido inicial del robot por el pasillo

El robot informa de que se encuentra en un espacio reducido en cuanto entra al pasillo, como se puede comprobar por medio del topic de publicación de

información. A la misma vez, calcula una posición de seguridad y la publica por el mismo topic. El nodo, además, guarda el objetivo actual de la navegación para su posterior recuperación. En la siguiente figura (ver Figura 40) se muestra cómo el nodo informa de la detección del espacio estrecho ("NARROW SPACE"), del objetivo de seguridad calculado ("SAFETY GOAL"), y del objetivo guardado para su posterior recuperación ("RECOVERY")

```
data: "NARROW SPACE: YES; SAFETY GOAL: [1.5, -0.798818]; RECOVERY: [0, 0]"
---
data: "NARROW SPACE: YES; SAFETY GOAL: [0, 0]; RECOVERY: [4.02106, 4.54741]"
```

Figura 40. Información publicada por el nodo

Cuando detecta a la persona, el objetivo de seguridad es enviado a la pila de navegación (ver Figura 41) y el robot espera a que pase la persona, es decir, deje de detectarla (ver Figura 42)

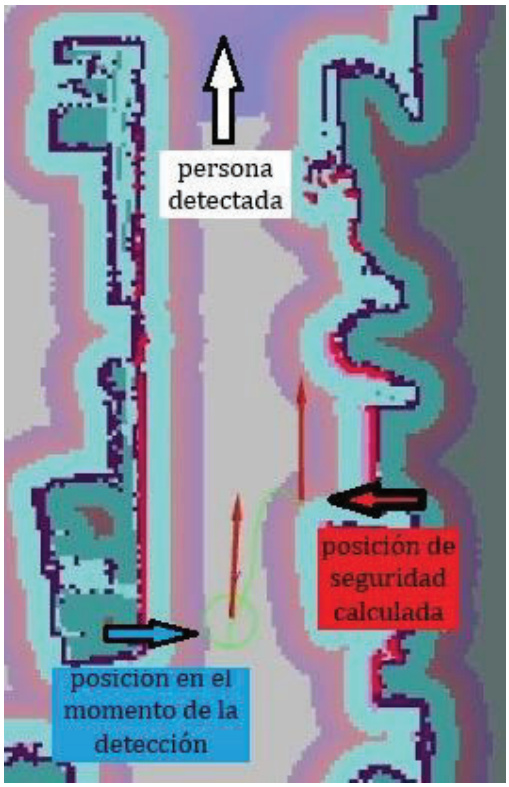

Figura 41. Navegación modificada para ceder el paso a la persona

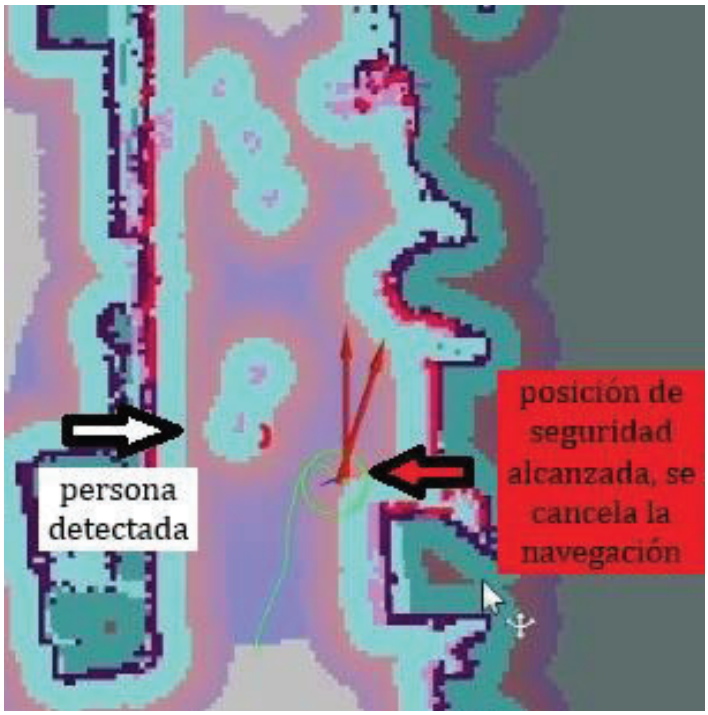

Figura 42. Posición de espera del robot mientras la persona pasa

Una vez la persona ha pasado, el nodo publica el objetivo previo guardado y el robot recupera su navegación original (ver Figura 43).

Figura 43. Recuperación del objetivo de navegación previo a la interrupción

6. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se ha implementado una serie de algoritmos que afectan a la navegación de un robot móvil y se ha logrado que este se comporte de una manera más acorde a estándares sociales.

Estos algoritmos se han implementado en forma de nodos de ROS programados usando el lenguaje C++, y se han reunido en un paquete que puede ser compilado para su uso en cualquier máquina (ver Anexo A).

El paquete incluye nodos que dotan al robot de la capacidad de ajustar su velocidad en función de la distancia a personas detectadas, de alterar sus mapas de coste para impedir el paso entre dos personas que puedan estar conversando o la modificación de su navegación para ceder el paso a personas en lugares estrechos como pasillos.

Estos nodos han sido testeados usando el entorno de simulación CoppeliaSim. Para las pruebas se han diseñado diversas situaciones en las que había personas presentes y se ha simulado una detección de personas que se asume resuelta para cualquier máquina en la que vaya a funcionar el paquete.

Tras las pruebas se ha concluido que los nodos funcionan correctamente, siendo por tanto más fácil convivir con el robot en un entorno del trabajo al haberse reducido las posibilidades de que el robot interfiera en el desarrollo de las actividades de las personas.

Como trabajo futuro se deja la depuración del código que implementa los algoritmos para lograr una mayor velocidad de reacción del robot y por tanto un mejor funcionamiento. Se deja también la posibilidad de añadir otras funcionalidades más cercanas al ámbito de la interacción humano robot como pueden ser el seguimiento de personas si se le da la orden o la posibilidad de comandarle que se acerque a una persona para entregar un objeto o mensaje.

Respecto a los objetivos del proyecto, se consideran cumplidos al haberse podido estudiar el funcionamiento general de la pila de navegación de ROS, desde su configuración previa para funcionar con el robot hasta la publicación final de los comandos que guían al robot hasta su objetivo, y se ha logrado la implementación y prueba de los comportamientos propuestos para este trabajo.

7. BIBLIOGRAFÍA

[1] Wu, Y. H., Wrobel, J., Cornuet, M., Kerhervé, H., Damnée, S., & Rigaud, A. S. (2014). Acceptance of an assistive robot in older adults: a mixed-method study of human–robot interaction over a 1-month period in the Living Lab setting. *Clinical interventions in aging*, 801-811.

[2] M. Moradi, M. Moradi and F. Bayat, "On robot acceptance and adoption a case study," 2018 8th Conference of AI & Robotics and 10th RoboCup Iranopen International Symposium (IRANOPEN), Qazvin, Iran, 2018, pp. 21-25, doi: 10.1109/RIOS.2018.8406626.

[3] *KR 1000 titan – KUKA.* (31/08/2023). https://www.kuka.com/es-es/productos-servicios/sistemas-de-robot/robot-industrial/kr-1000-titan

[4] Morgan, A. A., Abdi, J., Syed, M. A. Q., Kohen, G. E., Barlow, P. L., & Vizcaychipi, M. P. (2022). Robots in Healthcare: A scoping review. *Current Robotics Reports*, 3(4), 271-280. https://doi.org/10.1007/s43154-022-00095-4

[5] Iat, & Iat. (2020b). Robótica social o la relación entre humanos y robots. *IAT*. https://iat.es/tecnologias/robotica/social/

[6] Guntur, S. R., Gorrepati, R. R., & Dirisala, V. R. (2019). Robotics in healthcare: an internet of medical robotic things (IoMRT) perspective. In *Machine learning in bio-signal analysis and diagnostic imaging* (pp. 293-318). Academic Press.

[7] Estapé, J. A. P. (2021). BellaBot, el robot gatuno camarero que sirve las mesas. *Computer Hoy*. https://computerhoy.com/noticias/tecnologia/bellabot-robot-gatuno-camarero-sirve-mesas-899699

[8] Azpiazu Arrieta, G., y Bayón Pérez, J. (2022). Tendencias laborales y el futuro del trabajo por medio de la robotización, digitalización e inteligencia artificial en España. *Razón Crítica*, (12). https://doi.org/10.21789/25007807.1805

[9] Guzmán Martínez, G. (2018). Proxémica: qué es y cómo nos ayuda a entender los espacios. *Psicología y Mente.* https://psicologiaymente.com/social/proxemica

[10] Gnambs, T., & Appel, M. (2019). Are robots becoming unpopular? Changes in attitudes towards autonomous robotic systems in Europe. *Computers in Human Behavior*, *93*, 53-61. https://doi.org/10.1016/j.chb.2018.11.045

[11] L. Takayama and C. Pantofaru, "Influences on proxemic behaviors in human-robot interaction," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 2009, pp. 5495-5502, doi: 10.1109/IROS.2009.5354145.

[12] Garcia-Salguero, M., Monroy, J., Solano, A., & Gonzalez-Jimenez, J. (2019, January). Socially acceptable approach to humans by a mobile robot. In *Proceedings of the 2nd International Conference on Applications of Intelligent Systems* (pp. 1-7).

[13] Isaka, T., Aoki, R., Ohshima, N., & Mukawa, N. (2018, December). Study of socially appropriate robot behaviors in human-robot conversation closure. In *Proceedings of the 30th Australian Conference on Computer-Human Interaction* (pp. 519-523).

[14] Sánchez, L. J. (2018). La presencia de robots en las empresas generará conflictos a corto plazo, según un estudio de Cuatrecasas y Adecco. *Confilegal*. https://confilegal.com/20180223-la-presencia-de-robots-en-las-empresas-generara-conflictos-a-corto-plazo-segun-un-estudio-de-cuatrecasas-y-adecco/

[15] Schaefer, K. E., Chen, J. Y. C., Szalma, J. L., & Hancock, P. A. (2016). A Meta-Analysis of Factors Influencing the Development of Trust in Automation: Implications for Understanding Autonomy in Future Systems. *Human Factors*, *58*(3), 377–400. https://doi.org/10.1177/0018720816634228

[16] Glikson, E., & Woolley, A. W. (2020). Human Trust in Artificial Intelligence: Review of Empirical research. *The Academy of Management Annals*, *14*(2), 627-660. https://doi.org/10.5465/annals.2018.0057

[17] Kok, B. C., & Soh, H. (2020). Trust in robots: challenges and opportunities. *Current Robotics Reports*, *1*(4), 297-309. https://doi.org/10.1007/s43154-020-00029-y

[18] Mara, M., Appel, M., & Gnambs, T. (2022). Human-Like Robots and the Uncanny Valley. *Zeitschrift Fur Psychologie-journal of Psychology*, *230*(1), 33-46. https://doi.org/10.1027/2151-2604/a000486

[19] Alemi, M., Ghanbarzadeh, A., Meghdari, A., & Moghadam, L. J. (2015). Clinical application of a humanoid robot in pediatric cancer interventions. *International Journal of Social Robotics*, *8*(5), 743-759. https://doi.org/10.1007/s12369-015-0294-y

[20] Piatt, J., Nagata, S., Šabanović, S., Cheng, W. L., Bennett, C., Lee, H. R., & Hakken, D. (2016). Companionship with a robot? Therapists' perspectives on socially assistive robots as therapeutic interventions in community mental health for older adults. *American Journal of Recreation Therapy*, *15*(4), 29-39.

[21] Bemelmans, R., Gelderblom, G. J., Jonker, P., & De Witte, L. (2012). Socially Assistive robots in elderly Care: A Systematic Review into Effects and Effectiveness. *Journal of the American Medical Directors Association*, *13*(2), 114-120.e1. https://doi.org/10.1016/j.jamda.2010.10.002

[22] Cabibihan, J., Javed, H., Ang, M. H., & Aljunied, S. M. (2013). Why Robots? A survey on the roles and benefits of social robots in the therapy of children with autism. *International Journal of Social Robotics*, *5*(4), 593-618. https://doi.org/10.1007/s12369-013-0202-2

[23] de Wit, J., Brandse, A., Krahmer, E., & Vogt, P. (2020, March). Varied human-like gestures for social robots: Investigating the effects on children's engagement and

language learning. In *Proceedings of the 2020 ACM/IEEE international conference on human-robot interaction* (pp. 359-367).

[24] Stock-Homburg, R. (2021). Survey of Emotions in Human–Robot Interactions: Perspectives from Robotic Psychology on 20 years of research. *International Journal of Social Robotics*, *14*(2), 389-411. https://doi.org/10.1007/s12369-021-00778-6

[25] Dodd W, Gutierrez R (2005) The role of episodic memory and emotion in a cognitive robot. In: ROMAN 2005. IEEE international workshop on robot and human interactive communication, 2005. IEEE, pp 692–697

[26] Chao-gang W, Jie-yu Z, Yuan-yuan Z (2008) An emotion generation model for interactive virtual robots. In: 2008 international symposium on computational intelligence and design, vol 2. IEEE, pp 238–241

[27] Samarakoon, S. B. P., Muthugala, M. V. J., & Jayasekara, A. B. P. (2022). A Review on Human–Robot Proxemics. *Electronics*, *11*(16), 2490.

[28] Kosiński, T., Obaid, M., Woźniak, P. W., Fjeld, M., & Kucharski, J. (2016, August). A fuzzy data-based model for Human-Robot Proxemics. In *2016 25th IEEE international symposium on robot and human interactive communication (RO-MAN)* (pp. 335-340). IEEE.

[29] Zhou, L., Liu, M., & Liu, J. (2019, October). Dynamic proxemic distances in different soical environments. In *2019 IEEE International Conference on Advanced Robotics and its Social Impacts (ARSO)* (pp. 401-406). IEEE.

[30] Choi, H. S., Crump, C., Duriez, C., Elmquist, A., Hager, G. D., Han, D. K., Hearl, F. J., Hodgins, J. K., Jain, A., Leve, F. A., Li, C., Meier, F., Negrut, D., Righetti, L., Rodriguez, A., Tan, J., & Trinkle, J. (2020). On the use of simulation in robotics: opportunities, challenges, and suggestions for moving forward. *Proceedings of the National Academy of Sciences of the United States of America*, *118*(1). https://doi.org/10.1073/pnas.1907856118

[31] J. Collins, S. Chand, A. Vanderkop and D. Howard, "A Review of Physics Simulators for Robotic Applications," in *IEEE Access*, vol. 9, pp. 51416-51431, 2021, doi: 10.1109/ACCESS.2021.3068769.

[32] Nogueira, L. (2014). Comparative analysis between gazebo and v-rep robotic simulators. *Seminario Interno de Cognicao Artificial-SICA*, *2014*(5), 2.

[33] Wang, M., Wang, K., Zhao, Q., Zheng, X., He, G., & Yu, J. (2023). LQR control and optimization for trajectory tracking of biomimetic robotic fish based on Unreal engine. *Biomimetics*, *8*(2), 236. https://doi.org/10.3390/biomimetics8020236

[34] *ROS: Home*. (31/08/2023). https://www.ros.org/

[35] *What is ROS? | Ubuntu*. (31/08/2023). Ubuntu. https://ubuntu.com/robotics/what-is-ros

[36] Packt, & Packt. (2018). ROS Architecture and Concepts. *Packt Hub*. https://hub.packtpub.com/ros-architecture-and-concepts/

[37] *Move_Base - ROS Wiki*. (31/08/2023). http://wiki.ros.org/move_base

[38] *Costmap_2d - ROS Wiki*. (31/08/2023). http://wiki.ros.org/costmap_2d?distro=noetic

[39] D. V. Lu, D. Hershberger and W. D. Smart, "Layered costmaps for context-sensitive navigation," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, 2014, pp. 709-715, doi: 10.1109/IROS.2014.6942636.

[40] *Social_navigation_layers - ROS Wiki*. (31/08/2023). http://wiki.ros.org/social_navigation_layers

[41] *Dynamic_reconfigure - ROS Wiki*. (31/08/2023). http://wiki.ros.org/dynamic_reconfigure

[42] *Robot Simulator CoppeliaSim: Create, compose, simulate, any robot - Coppelia Robotics*. (s. f.). https://www.coppeliarobotics.com/

[43] Ginés, J., Martín, F., Vargas, D., Rodríguez, F. J., & Matellán, V. (2019). Social navigation in a cognitive architecture using dynamic proxemic zones. *Sensors*, *19*(23), 5189.

[44] Shani-Feinstein, Y., Kyung, E. J., & Goldenberg, J. (2022). Moving, fast or slow: How perceived speed influences mental representation and decision making. *Journal of Consumer Research*, *49*(3), 520-542.

ANEXOS

ANEXO A. ESTRUCTURA DEL REPOSITORIO DE CÓDIGO DEL PROYECTO

Como parte del proyecto, se ha creado un repositorio alojado en la plataforma GitHub. Este repositorio incluye todo el código que ejecutan los nodos para implementar los diferentes algoritmos descritos. Además, incluye instrucciones para la instalación del proyecto, de manera que pueda ser replicado con facilidad en otro equipo. El repositorio puede ser descargado desde la URL https://github.com/keikei014/tfg_social/

El repositorio está organizado en paquetes que agrupan los nodos según el propósito que cumplen. También se incluyen carpetas que no son paquetes y que recogen archivos de configuración y otros archivos auxiliares.

El paquete 'social_pkg' incluye los tres nodos principales diseñados para el proyecto.

El paquete 'utils_pkg' se ha añadido para albergar los nodos auxiliares que no implementan comportamientos sociales pero que son necesarios para el correcto funcionamiento de los que sí lo hacen.

El paquete 'missions_pkg' incluye diversos archivos de configuración que establecen los parámetros de lanzamiento del stack de navegación de ROS. Este paquete también incluye ficheros '.launch' que sirven para ejecutar todos los nodos de navegación junto con los nodos de comportamiento social lanzando un solo comando.

El paquete 'social_navigation_layers' contiene las implementaciones de las distintas capas sociales para mapas de coste que han sido utilizadas en el proyecto.

El paquete 'people' ha sido incluido para poder usar una serie de mensajes preconstruidos para el funcionamiento de las capas de coste sociales.

En la carpeta 'scenes' se pueden encontrar los archivos de mundo de CoppeliaSim con los que se pueden replicar las situaciones y condiciones de simulación mostradas en los ejemplos de este proyecto.

Por último, el fichero 'README.txt' detalla todas las dependencias que precisa la instalación del proyecto, así como la información sobre los paquetes que se encuentra en este anexo e indicaciones sobre las distintas formas de lanzar los nodos.