

**UNIVERSIDAD POLITÉCNICA DE MADRID**

**Escuela Técnica Superior de Ingenieros Industriales**

**Área de conocimiento: Automática y Electrónica**



**Trabajo Fin de Grado**

**Control Fuzzy de un robot social para navegación en presencia de humanos**

Autor: **José Miguel Alberca Sánchez**

Tutor: **Fernando Matía Espada**

*Ph.D*

Tutor: **Biel Piero Eloy Alvarado Vásquez**  
*MSc*

**2018**



---

## Agradecimientos

---

En primer lugar, querría agradecer a mis padres la oportunidad de cumplir un sueño. El camino ha sido duro pero siempre he tenido su apoyo. No soy capaz de encontrar palabras suficientes para compensar el esfuerzo y los sacrificios que han realizado para que yo pueda encontrarme en esta situación y convertirme en el hombre que soy hoy en día.

También querría dar las gracias a Carmen, mi compañera en la vida, mi mejor amiga. Gracias por poner mi mundo patas arriba, hacerme ver la vida de una manera totalmente distinta, por tu apoyo, por tu cariño, y por todo lo que significas para mí.

A mis amigos, los que han estado siempre, y a los que han llegado en Madrid. Muchas de las experiencias buenas que han sido parte del camino han sido junto a vosotros.

A mis tutores, Biel y Fernando. Dos profesionales increíbles, dos expertos en programación que han conseguido que aprenda más de lo que pude imaginar. Gracias por permitirme formar parte de un proyecto tan bonito como el de DORIS.

Por último, a Doris, fuente de penas y alegrías durante unos cuantos meses.



---

## **Resumen**

---

El desarrollo de robots sociales es un sector de la robótica que gana importancia con el paso de los años gracias a los avances tecnológicos generados. Las funciones de los robots sociales están relacionadas con trabajos en los que existe interacción con gente de tal forma que dicha interacción sea consistente con la psicología social humana. Por tanto estos robots tienen aplicaciones basadas en el trabajo en espacios públicos, y usos domésticos.

El desarrollo de este tipo de robots supone un esfuerzo en distintas áreas de la robótica como es el desarrollo de inteligencia artificial, la capacidad de interactuar a través de sonido, imagen, o ambas, o la habilidad de que su navegación sea autónoma.

Este último punto es en el que se centra el presente proyecto. Doris es un robot del Centro de Automática y Robótica diseñado para trabajar como guía en museos y otras actividades en espacios públicos como ferias, teatros, etc. Obviamente, en un robot que trabaja en un entorno como éste, es necesario que el robot pueda navegar entre distintas partes dentro del espacio.

Aunque Doris cuenta con un sistema de mapas para definir a qué punto tiene que ir, o dónde se encuentra, para evitar acumulación de errores, y evitar la necesidad de rutas preestablecidas, se utilizan sensores externos, como cámaras, láser, etc. para que Doris prescinda de la información previa del entorno y se guíe a través de la información que recibe del láser, en el caso del presente proyecto.

Uno de los entornos en los que Doris debe trabajar es en el de pasillos, zonas de tránsito en salas, en las que en el futuro se deberá implementar un controlador para la navegación dentro de la sala. Los objetivos principales del proyecto consisten en conseguir que el robot avance por el centro del pasillo siempre que sea posible solo a través de los datos recibidos por el láser, esquivar objetos que aparezcan frontalmente, y filtrar obstáculos dinámicos que no suponen un impedimento para el avance del robot, como por ejemplo la apertura de una puerta, pero que si no se filtraran provocarían una desviación en la trayectoria de Doris.

Para realizar este cometido se creará un controlador borroso. Este controlador basa su funcionamiento en el hecho de que en los conjuntos borrosos, los elementos no pertenecen de forma binaria a un conjunto, como ocurre en la lógica clásica, sino que los elementos tienen un grado de pertenencia a cada conjunto. Este tipo de controladores permiten a los robots resolver problemas de una manera más humana. Los seres humanos están acostum-

brados a la resolución de problemas en los que las condiciones no son estrictas, ya que es fácil para nosotros generar soluciones flexibles a través del uso de palabras como “algo”, o “muy”. Sin embargo, si se utiliza la lógica clásica, un robot necesita datos exactos para poder funcionar, pero el uso de lógica borrosa solventa esto, a través de la conversión de los datos obtenidos en lenguaje a través de etiquetas lingüísticas, un conjunto de reglas que definen el funcionamiento y la asignación de pesos a dichas etiquetas. Realizando un proceso de desborrosificación, se obtienen los valores numéricos necesarios para controlar las variables de salida del sistema.

En el caso de Doris, se decide que las variables a controlar sean la velocidad lineal, y la velocidad angular, para decidir la trayectoria de Doris durante todo el trayecto. En cuanto a las variables de entrada al sistema borroso, como las mediciones de Doris abarcan 180° al frente del robot, se dividen las mediciones en tres sectores de 60° buscando diferenciar lo que pasa a los lados del robot, y delante de él.

Para obtener un valor de entrada al sistema borroso, se dividen esos sectores angulares en cinco sectores radiales, y se obtiene la media de las mediciones en el sector radial más cercano con un número significativo de puntos dentro de dicho sector. Cuando el controlador se instancia un número suficiente de veces en el que la diferencia entre las distancias medidas por los sectores izquierdo y derecho están por debajo de cierto umbral se considera que el robot está estabilizado. Si el robot está estabilizado y la diferencia vuelve a superar el umbral, el controlador empieza a comprobar si realmente existe un cambio en las medidas del entorno, o esa diferencia se debe por ejemplo al paso de una persona junto a Doris.

Una vez que se han filtrado los datos de entrada, se introducen los datos en el sistema borroso, en el que tanto las funciones de pertenencia de las entradas como de las salidas están formadas cinco etiquetas lingüísticas posibles. Las etiquetas lingüísticas de las variables de entrada analizan la distancia en términos como “Muy Cerca”, o “Lejos”, las etiquetas de la velocidad líneal son del tipo “Muy rápido”, o “Hacia Atrás”, y las de la velocidad angular indican expresiones como “Rápido a la izquierda”, o “Despacio a la derecha”. Todas las posibles combinaciones de estas etiquetas en función de las distancias medidas, crean un árbol de reglas con  $5^3$  posibilidades, con reglas del tipo “IF DISTANCIA IZQUIERDA IS ... AND DISTANCIA FRONTAL IS ... AND DISTANCIA DERECHA IS ... THEN VELOCIDAD LINEAL IS ... AND VELOCIDAD ANGULAR IS ...”, como las variables pertenecerán a más de un conjunto, se activarán varias reglas, a las que habrá que dar un peso en función del grado de pertenencia de la variable a dicho conjunto. Realizando el proceso inverso, se obtendrán los valores finales de velocidad angular y lineal.

El objetivo del trabajo es crear una clase que contenga el sistema borroso para posibilitar la navegación en el entorno de pasillo, y las funciones auxiliares al sistema para conseguir los datos de entrada y facilitar los datos de salida a la clase que maneja la navegación.

En cuanto a la fase de pruebas del proyecto, deberá comprobarse la viabilidad del sistema borroso, y ajustar el sistema y las variables auxiliares a las necesidades del entorno de pruebas elegido.

---

# Índice general

---

<b>1. Introducción</b>	<b>1</b>
1.1. Marco del trabajo . . . . .	1
1.2. Motivación del proyecto . . . . .	1
1.3. Justificación . . . . .	1
1.4. Objetivo . . . . .	2
1.5. Estructura del documento . . . . .	2
<b>2. Estado del arte</b>	<b>5</b>
2.1. Robótica móvil y navegación autónoma . . . . .	5
2.2. Evolución de la robótica móvil . . . . .	7
2.3. Robótica social . . . . .	8
2.3.1. Historia de los robots sociales . . . . .	8
2.3.2. Robots sociales integración social: conceptos y definiciones . . . . .	9
2.3.3. Robots Socialmente Interactivos . . . . .	9
2.3.4. Aplicaciones . . . . .	10
2.4. Estado del arte de la robótica social . . . . .	13
2.4.1. Robótica Social fuera del DISAM . . . . .	13
2.4.2. Robótica Social en el DISAM . . . . .	16
<b>3. Doris, el robot social chica</b>	<b>19</b>
3.0.1. Arquitectura de hardware . . . . .	19
3.0.2. Arquitectura de software . . . . .	20
3.1. Algoritmos de localización . . . . .	21
3.1.1. Localización visual . . . . .	21
3.1.2. Localización LRF . . . . .	23
3.2. Filtro de Kalman e integración . . . . .	24
3.2.1. Filtro de Kalman . . . . .	24
3.2.2. Fusión del sensor con la odometría . . . . .	25
<b>4. Lógica Fuzzy</b>	<b>27</b>
4.1. Orígenes del pensamiento borroso . . . . .	27
4.2. Lógica y tipos de lógica . . . . .	30
4.2.1. Lógica de proposiciones . . . . .	31
4.2.2. Lógica de predicados . . . . .	32
4.2.3. Lógica borrosa . . . . .	33

4.3.	Controladores borrosos . . . . .	34
4.3.1.	Borrosificación . . . . .	34
4.3.2.	Aplicación de reglas . . . . .	35
4.3.3.	Conclusión . . . . .	36
4.3.4.	Desborrosificación . . . . .	36
4.4.	Construcción del controlador borroso . . . . .	37
4.4.1.	Parámetros de configuración . . . . .	37
4.4.2.	Construcción de la tabla de reglas . . . . .	37
4.4.3.	Parámetros de ajuste . . . . .	38
4.4.4.	Ventajas e inconvenientes . . . . .	38
4.5.	Modelado borroso . . . . .	38
4.5.1.	Modelo de Mamdani . . . . .	39
4.6.	Estado del arte en navegación con lógica fuzzy . . . . .	39
4.6.1.	Navegación con redes neuronales . . . . .	39
4.6.2.	Navegación con lógica borrosa . . . . .	40
<b>5.</b>	<b>Implementación del controlador</b>	<b>43</b>
5.1.	Plataformas de desarrollo y ejecución . . . . .	43
5.2.	Bibliotecas empleadas . . . . .	44
5.2.1.	Biblioteca FuzzyLite . . . . .	44
5.3.	Controlador teórico . . . . .	45
5.4.	Estructura de clases . . . . .	52
5.4.1.	La clase RNAActionGoto . . . . .	52
5.5.	La clase RobotNode . . . . .	54
5.6.	La clase RNHallwayController . . . . .	56
5.6.1.	Atributos de RNHallwayController . . . . .	56
5.6.2.	Métodos de RNHallwayController . . . . .	57
5.7.	Estados del controlador . . . . .	59
5.8.	Reglas fuzzy . . . . .	63
<b>6.</b>	<b>Pruebas físicas y resultados</b>	<b>67</b>
6.1.	Entorno de pruebas . . . . .	67
6.2.	Ajustes previos . . . . .	68
6.2.1.	Ajuste del láser . . . . .	68
6.2.2.	Ajustes de las funciones de pertenencia de entrada . . . . .	69
6.3.	Pruebas dinámicas . . . . .	70
6.3.1.	Ajuste de las funciones de pertenencia de salida . . . . .	70
6.3.2.	Comprobación del funcionamiento del filtrado de datos . . . . .	72
<b>7.</b>	<b>Conclusiones y líneas de investigación futuras</b>	<b>75</b>
7.1.	Conclusiones . . . . .	75
7.2.	Líneas de investigación futura . . . . .	76
<b>8.</b>	<b>Anexos</b>	<b>77</b>
8.1.	Anexo I: Estructura de Descomposición del Proyecto . . . . .	77
8.2.	Anexo II: Diagrama de Gantt . . . . .	79
8.3.	Anexo III: Presupuesto del proyecto . . . . .	81
	<b>Bibliografía</b>	<b>86</b>

# **Introducción**

---

En el primer capítulo se procede a realizar una pequeña introducción al trabajo de fin de grado a presentar. Esta introducción pretende explicar la motivación del proyecto, los objetivos y el alcance del mismo, así como la estructura seguida para conseguir dichos objetivos.

## **1.1. Marco del trabajo**

El presente trabajo se ha llevado a cabo en el Centro de Automática y Robótica del CSIC (Centro Superior de Investigaciones Científicas) y la UPM (Universidad Politécnica de Madrid), centro dedicado a la investigación en los campos de la Ingeniería de Control, Percepción Artificial y Robótica.

## **1.2. Motivación del proyecto**

Este trabajo de fin de grado pretende abordar una de las problemáticas asociadas a la navegación de robots móviles en entornos dinámicos. Estos robots trabajan con información previa a través de mapas de su entorno, pero estos entornos son cambiantes, y pueden aparecer nuevos objetos en el entorno que no han sido concebidos en el mapa, o alguno de los ya existentes puede haber cambiado su posición. Para resolver este problema, en la actualidad se utilizan distintas técnicas basadas en recibir información en tiempo real y procesarla a través de distintos algoritmos de tal manera que el robot pueda reaccionar a estos cambios en su entorno de forma autónoma y sea capaz de planificar su trayectoria de manera más eficaz, así como evitar colisiones que comprometan al robot.

## **1.3. Justificación**

El enfoque del trabajo se centra en la creación de un controlador borroso, que permita al robot navegar en un entorno de pasillo.

La creación de este tipo de controladores es de gran interés en la industria y en la investigación, puesto que permiten dar solución a problemas con un alto nivel de incertidumbre, los cuales son solucionados más fácilmente por humanos que por máquinas. Por tanto, estos controladores son fundamentales para el uso de robots que trabajan en entornos sociales, ya que el entorno cambiante en salas y pasillos provoca alteraciones constantemente, invalidando así la información obtenida a priori por el robot, y la velocidad con la que cambia el entorno propicia que sea más beneficioso trabajar con incertidumbre.

La UPM, y en concreto el Centro de Automática y Robótica (CAR) ha diseñado y construido un robot que tiene por objetivo trabajar en entornos dinámicos cerrados para interactuar con gente en espacios públicos como museos, teatros, ferias, etc.

## 1.4. Objetivo

Se trata de un trabajo esencialmente experimental cuyo propósito es la creación de un controlador fuzzy (difuso) enfocado a la navegación en un cierto pasillo de un robot móvil a través de mediciones láser acerca de la distancia a las paredes. Para ello se creará una clase controlador en el entorno de programación C++ que debe ser compatible con las clases y funciones ya creadas para el movimiento del robot DORIS, y se utilizará en pasillos de la ETSII con pasillos previamente mapeados para realizar pruebas.

El principal interés de este proyecto es resolver de forma automática la navegación entre distintas salas y establecer una base para la navegación completa de DORIS en entornos dinámicos, ya que los principios del controlador serán fácilmente extrapolables a funciones para navegación en salas en las que haya que esquivar objetos no mapeados previamente.

Se busca que la metodología sea fácilmente reproducible para posibilitar la adición de futuras funcionalidades al controlador o la creación de otros controladores. Por tanto, los métodos utilizados tienen que poseer la capacidad de ser funcionales con otros tipos de mediciones como odometría y poder responder a otras variables de salida distintas de la velocidad.

## 1.5. Estructura del documento

La memoria de este Trabajo de Fin de Grado se encuentra dividido en siete capítulos principales. Se propone un resumen ejecutivo para sintetizar los aspectos más significativos, la estructura del documento es la siguiente

**Capítulo 1: Introducción** Incluye una breve introducción al trabajo. En este capítulo se encuentran el marco del trabajo, la motivación del proyecto, la justificación, los objetivos y el alcance del proyecto.

**Capítulo 2: Estado del arte** Se aborda el estado en el que se encuentran las investigaciones actuales con lógica difusa, y las relacionadas con navegación.

**Capítulo 3: Doris, el robot social chica** En el que se comentarán los aspectos básicos del robot Doris, y la fusión de sensores a través del Filtro Extendido de Kalman.

**Capítulo 4: Lógica Fuzzy** Para la comprensión de los métodos utilizados para conseguir los objetivos es necesario dedicar un capítulo a entender la lógica borrosa y su comparación con las lógicas clásicas, así como del funcionamiento de un PID borroso.

**Capítulo 5: Implementación del controlador** Se explican en profundidad las ideas desarrolladas, desde el funcionamiento general hasta la implementación de funciones específicas.

**Capítulo 6: Pruebas físicas y resultados** Se describen las pruebas realizadas, así como los resultados y las conclusiones sobre la viabilidad del desarrollo.

**Capítulo 7: Conclusiones y futuras líneas de investigación** Incluye un breve análisis del trabajo y las posibles vías futuras de desarrollo a partir del proyecto.

**Capítulo 8: Anexos** Se data el presupuesto del proyecto, así como los tiempos y organización en el desarrollo del proyecto.



---

## Estado del arte

---

### 2.1. Robótica móvil y navegación autónoma

Como indica Ollero[41], la robótica móvil es un campo de investigación que ocupa importantes líneas de trabajo en laboratorios y centros técnicos de todo el mundo. Su desarrollo supone la integración de numerosas disciplinas entre las que se encuentran la automática, la electrónica, la ingeniería mecánica, la informática, la inteligencia artificial, la estadística, etc.

Para ser considerado como tal, un robot móvil precisa de un sistema de locomoción que le permita desplazarse. Para ello existen numerosas posibilidades en función de las necesidades del robot, existen robots terrestres que andan, reptan, o se mueven a través de ruedas, y también robots aéreos o submarinos.

Dentro de los robots móviles, los robots autónomos son aquellos que no necesitan de la intervención humana para mantener su capacidad de navegación y su orientación dentro del espacio. Según Smithers[55], la idea principal del concepto de autonomía se obtiene de su etimología: *auto* (propio) y *nomos* (ley o regla). Los sistemas automáticos se autorregulan pero no son capaces de generar las leyes que deben tratar de seguir sus reguladores. A diferencia de ellos, los sistemas autónomos han de poder determinar sus estrategias, o leyes, de conducta y modificar sus pautas de comportamiento al mismo tiempo que operan en el entorno. Es obvio por tanto, que la autonomía añade requisitos sobre el concepto de automatismo.

El enfoque más utilizado en la literatura para estudiar la navegación autónoma consiste en:

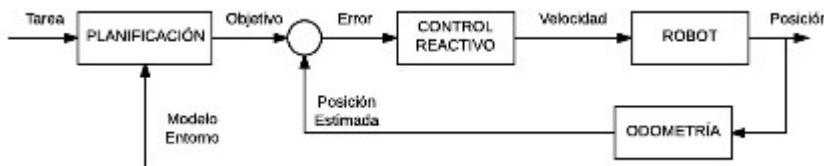
1. Localización: ¿dónde estoy?
2. Planificación de tareas: ¿hacia dónde quiero ir?
3. Planificación del Movimiento: ¿cómo puedo llegar allí?

Los robots autónomos se justifican ante la dificultad o imposibilidad de intervención humana, bien directa o teleoperada, por lo que han de ser capaces de adquirir información sobre el entorno por sí mismos, así que deben estar dotados de sensores que proporcionen al robot las medidas y datos necesarios sobre dicho entorno. Si bien es cierto que

los robots suelen contar con sensores propioceptivos que proporcionan medidas sobre la posición o la velocidad del robot, como pueden ser por ejemplo los encoders de las ruedas, sin embargo, este tipo de sensores, aunque trabajan bien a corto plazo, provocan grandes errores en cuanto a las mediciones a medida que aumenta la distancia recorrida, por esto, es necesario disponer de sensores estereoeceptivos que proporcionan estas mismas medidas de manera externa al robot. Los sensores más utilizados en robótica móvil son los de ultrasonidos, infrarrojos, los láseres y las cámaras. Además, como es obvio, el robot autónomo ha de ser capaz de procesar la información, y ejecutar los algoritmos que la utilizan, por lo tanto deberá estar dotado de computadores o microprocesadores.

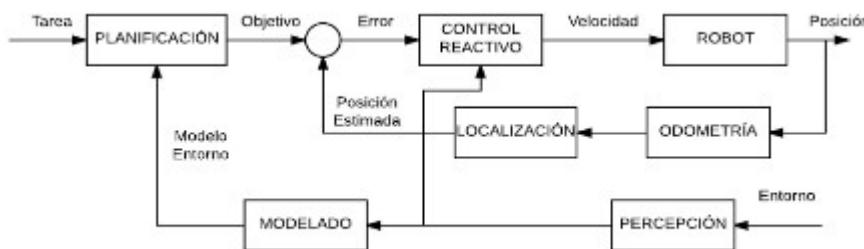
A la navegación que prescinde de la planificación de rutas, se la ha denominado navegación reactiva. En este tipo de navegación se prescinde de la información previa del entorno y se guía el robot únicamente a partir de la información de los sensores en el momento del desplazamiento. Al realizarse la navegación reactiva en línea, es posible, dependiendo del algoritmo utilizado, que el coste computacional de este tipo de navegación sea mayor. Además, se ha de contar con la necesidad de procesar los datos rápidamente para responder a situaciones con obstáculos, tanto para su detección, como para poder esquivarlos.

Dentro de la arquitectura de control de un robot móvil estándar, encontramos cuatro módulos básicos: Planificación, Control, Percepción y Modelado. Asumiendo que los modelos del entorno y las medidas odométricas del robot son perfectos, un esquema básico de control y planificación sería el mostrado en la figura 2.1



**Figura 2.1** Esquema de control básico

La inclusión de un módulo de Percepción, mediante sensores estereoeceptivos, completa la información necesaria para una correcta navegación. El uso de sensores estereoeceptivos obliga a tener en cuenta una serie de marcas ya sean naturales o artificiales, que permitan corregir los errores odométricos que se producen. El uso de un módulo de modelado permite corregir el modelo del entorno con la nueva información que proporciona el módulo de Percepción, como se muestra en la figura 2.2



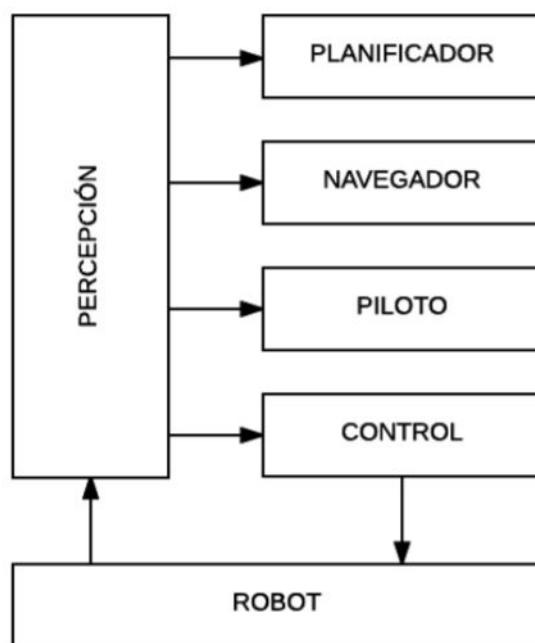
**Figura 2.2** Esquema de control completo

En el área de la arquitectura de control de un robot móvil, para que el robot consiga

el grado de autonomía que se desee, se debe contar con unos módulos imprescindibles:

- Planificador: Qué debe hacer el robot.
- Navegador: Cuándo debe hacerlo.
- Percepción: Por qué lo hace.
- Piloto: Quién lo hace.
- Control: Cómo lo hace.

Este proyecto está centrado en trabajar en uno de los módulos de control, como se verá en apartados posteriores.



**Figura 2.3** Arquitectura de control

## 2.2. Evolución de la robótica móvil

Durante todos estos años desde el surgimiento de la robótica ha habido una gran evolución en lo que a tecnología empleada se refiere. A continuación se muestra una lista de robots que reflejan dicha evolución:

- SHAKEY [39] (años 60). Se considera el primer robot serio. Dispone de cámara, sensor táctil y telémetro.
- Jet Propulsion Laboratory AMR (años 70). Desarrollado por la NASA, disponía de un sistema de visión y era capaz de realizar modelado y planificación.
- HILARE (1977-1983). Posee la misma estructura que SHAKEY pero incorpora un sistema de visión 3D, ultrasonidos e infrarrojos.

- VAD-1 (años 80). Fue el primer prototipo desarrollado por el DISAM.
- ROBUTER (años 90). Sistema de locomoción diferencial. 24 sensores de ultrasónidos. Dos sistemas de visión: activa con luz estructurada y pasiva con cámara pan-tilt.
- BLACKY [49] (1994-2001). Locomoción por giro síncrono. 24 ultrasonidos. Láser rotatorio.
- URBANO [50] (2001-Actualidad). Sistema de locomoción por giro síncrono. Telémetro láser. Comunicaciones Ethernet wireless.
- DORIS (2014-Actualidad). Sistema de locomoción diferencial. Telémetro láser. Comunicaciones Ethernet wireless. Robot sobre el que se articula el presente proyecto.

## 2.3. Robótica social

Un robot social es aquel que interactúa y se comunica con las personas (de forma sencilla y agradable) siguiendo comportamientos, patrones y normas sociales. Para eso, además de tener apariencia agradable, se necesita que disponga de habilidades que se ubican dentro del dominio de la llamada inteligencia social. Se debe tener en cuenta que la socialización con las personas es un tema difícil, ya que los robots y los humanos no comparten un lenguaje común ni perciben el mundo de la misma forma.

También es importante que el robot exhiba una cierta "personalidad" distintiva. Hay varias razones para creer que si un robot tuviera una personalidad convincente, la gente estaría más dispuesta a interactuar y establecer algún tipo de relación con él. En particular, la personalidad del robot puede proporcionar una realimentación útil, ofreciendo a los usuarios una manera de modelar y entender su conducta [20].

### 2.3.1. Historia de los robots sociales

Desde el surgimiento de los robots inspirados biológicamente, la comunidad investigadora se ha interesado por la posibilidad de que los robots actúasen entre sí. En el momento que apareció la Inteligencia Artificial, los investigadores empezaron a aplicar principios como la comunicación indirecta entre individuos a través de modificaciones en el entorno compartido para lograr un comportamiento colectivo de los robots.

En [7], Dautenhahn y Billard propusieron lo siguiente: Los robots sociales son agentes que forman parte de un grupo heterogéneo: una sociedad de robots y humanos. Son capaces de reconocerse entre sí y participar en interacciones sociales, perciben e interpretan el mundo en términos de su propia experiencia, y se comunican de forma explícita y aprenden unos de otros.

### 2.3.2. Robots sociales integración social: conceptos y definiciones

En [2], Breazeal define cuatro clases de robots sociales en términos de lo bien que el robot puede soportar el modelo social adscrito; y la complejidad del escenario de interacción que pueden soportar:

- Robots Socialmente Sugerentes: Robots que se basan en la tendencia humana a antropomorfizar y sacar provecho de los sentimientos evocados cuando los seres humanos crían, cuida, o participan con su creación”.
- Robots de Interfaz Social: Robots que proporcionan una interfaz ”natural.<sup>em</sup>pleando señales sociales o modalidades de comunicación similares a las humanas.
- Socialmente Receptivos: Robots socialmente pasivos pero que se benefician de la interacción. Por ejemplo, habilidades de aprendizaje por imitación.
- Sociables: Participan de forma activa con los seres humanos con el fin de satisfacer objetivos sociales internos (unidades, emociones, etc.)

Esta lista puede ser ampliada con:

- Robots Situados Socialmente: Robots rodeados por un entorno social que perciben y al que reaccionan [8]. Socialmente robots situados deben ser capaces de distinguir entre otros agentes sociales y diversos objetos en el entorno.
- Robots Socialmente Integrados: Robots que están:
  - Situados en un entorno social e interactúan con otros y con humanos.
  - Estructuralmente acoplados con el entorno.
  - Al menos parcialmente, conscientes de las estructuras de interacción humanas.
- Robots Socialmente Inteligentes: Robots que muestran aspectos de la inteligencia social humana, basados en modelos profundos de la cognición humana [9][10]

### 2.3.3. Robots Socialmente Interactivos

Los robots socialmente interactivos pueden usarse para infinidad de propósitos: desde realizar ensayos de investigación, funcionar como juguetes, actuar como herramientas educativas, ayudas terapéuticas, etc. El supuesto común es que los humanos prefieren interactuar con las máquinas de la misma forma que lo hacen con el resto de las personas.

Los robots socialmente interactivos trabajan como compañeros, lo que significa que necesitan mostrar un cierto grado de adaptabilidad y flexibilidad para mantener la interacción con una amplia gama de seres humanos. Pueden tener diferentes formas y funciones, que van desde robots cuyo único propósito es involucrar a la gente en las interacciones sociales, a los robots que están diseñados para cumplir con las normas sociales con el fin de realizar una serie de tareas en entornos habitados por humanos [3][40][44][51].

Algunos robots usan modelos profundos de interacción humana que fomentan la interacción social. Otros muestran su competencia social simplemente reaccionando a comportamientos humanos, dejando que los humanos atribuyan al robot de estados mentales y emociones [11][12][17][43].

Este tipo de robots es importante para los aspectos en los que deben mostrar habilidades de interacción, ya sea porque se necesitan ese tipo de habilidades para resolver tareas específicas, o porque la función principal del robot sea interactuar socialmente con las personas. Una discusión de los campos de aplicación, espacios de diseño, y las habilidades sociales deseables para los robots se da en [13][14][23].

Aunque los robots socialmente interactivos ya han sido utilizados con éxito, aún queda mucho trabajo que hacer para aumentar su eficacia. Por ejemplo, para que sean aceptados como compañeros “naturales, necesitan habilidades sociales más sofisticadas, tales como la capacidad de reconocer el contexto social. Además, necesitan soportar una amplia gama de usuarios: distintos géneros, distintos antecedentes culturales y sociales, distintas edades, etc. En muchas aplicaciones actuales, los robots sociales sólo interactúan a corto plazo (por ejemplo, tour en un museo) y pueden permitirse el lujo de tratar a todos los seres humanos de la misma manera. Pero, cuando un robot se convierte en parte de la vida de una persona, el robot deberá ser capaz de tratarlo como un individuo distinto.

### 2.3.4. Aplicaciones

#### Robots de servicio

Dentro de las muchas aplicaciones de los robots sociales, en este estudio se centrará la atención en los robots de servicio, ya que Doris pertenece a esta categoría.

Los robots son funcionales cuando aportan servicios concretos a la humanidad. Por ejemplo, en las fábricas automovilísticas los robots de soldadura han incrementado significativamente el rendimiento de las líneas de montaje, proporcionando soldaduras muy finas y precisas.

Hay varias razones por las que un robot orientado a realizar una tarea puede encontrar beneficios en la interacción social. Probablemente el más evidente sea la facilidad de uso. En un caso donde el robot tenga que interactuar con varias personas, la integración social como el diálogo y la gesticulación ayudan a que el robot sea más fácil de usar por los principiantes y resulte más eficiente para los expertos.

Un buen número de robots sociales han sido diseñados para misiones públicas específicas. Estos robots han demostrado que con el comportamiento correcto, los robots móviles pueden atraer y participar con el público satisfactoriamente en entornos poblados.

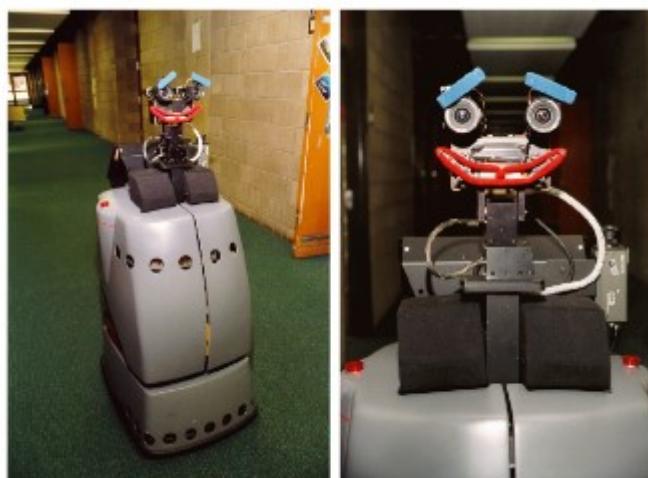
Un ejemplo de este tipo de robots es la serie Robox, instalados en la Switzerland Expo 2002[54]. Durante la exposición, los robots tenían dos objetivos: el primero, presentarse como demostraciones de la tecnología robótica; y el segundo, funcionar como guía a través de habitaciones llenas de exposiciones sobre robótica. El diseño del robot refleja su carácter social. El uso de cejas y ojos permite realizar expresiones y comunicar intenciones, como por ejemplo la dirección de desplazamiento.



**Figura 2.4** RoboX en la Switzerland Expo 2002

Trabajos anteriores de otras investigaciones, demostraron que ese aspecto puede tener una influencia cuantitativa en la eficacia del robot realizando visitas guiadas al público. En [52] se describen las diferencias entre MINERVA (robot guía del Smithsonian Institutions Museum of American History)[59][60] y RHINO (robot anterior que realiza tours en el German Science Museum)[22][4][5]. En el mismo documento, se especifican tres características fundamentales para el éxito de los robots que deben mostrar interacciones espontáneas en espacios públicos:

- El robot debe tener un punto de atención evidente para el ser humano. P. ej.: cara robótica o animada.
- El robot debe mostrar su estado emocional a los visitantes como forma de transmitir eficientemente sus intenciones. P. ej.: mostrar frustración por encontrarse bloqueado por los turistas.
- El robot debe tener la capacidad de adaptar los parámetros de interacción basándose en experiencias pasadas.



**Figura 2.5** Robot Minerva

La serie de robots Mobot Museum Robot, compuesta de cuatro robots es una base para las observaciones acerca de la evolución de la robótica social para público [62]. Los objetivos de esta serie de robots eran: desplegar una serie de robots en los espacios del museo que fueran capaces de interactuar satisfactoriamente entre ellos, los humanos y los objetos expuestos.



**Figura 2.6** La serie de robots Mobot

## Otras aplicaciones

Existen muchas otras aplicaciones en las que se está investigando en cuanto a robótica social. Algunas de estas aplicaciones son:

- Entretenimiento: generalmente se certifica que en comparación con los robots industriales y de investigación, los robots juguete son tecnológicamente insatisfactorios.
- Terapia: robots dedicados a terapias de rehabilitación como sillas de ruedas robóticas que permiten recuperar parte de la movilidad perdida durante, por ejemplo, una lesión medular[45], o que ayudan a pacientes que han sufrido pérdida de control motriz por culpa de un ictus[32], serán considerados robots sociales, y no máquinas, si existe interacción autónoma con el paciente.
- Robótica antropomorfa: Una de las motivaciones para la investigación deriva de un deseo de emular, lo más fielmente posible, la interacción humana y natural. En [24][25], por ejemplo, se plantea la premisa de que los humanos deben ser capaces de interactuar con los robots de la misma forma que lo harían con otros humanos.
- Educación: El papel de la robótica en la educación es un aspecto que aumenta notablemente con el paso de los años y los avances que se logran. Desde hace años la comunidad educativa se ha animado a incorporar actividades robóticas orientadas a la pedagogía.

## 2.4. Estado del arte de la robótica social

Dentro del ámbito de la robótica social existen múltiples campos de estudio en los que poder aplicar estas técnicas. En este caso, para el desarrollo de este proyecto el campo más importante es el uso de los robots sociales como guías en visitas a museos, ferias, eventos, exposiciones, etc.

En este apartado se va a realizar un estudio de las diferentes plataformas que existen o han existido empleadas para este uso. Si bien, cabe la posibilidad de que algún caso no se vea reflejado en el texto.

### 2.4.1. Robótica Social fuera del DISAM

Probablemente el primer caso fue Rhino [4][5], desarrollado en la Universidad de Bonn (Alemania).



**Figura 2.7** Rhino realizando demostraciones. Detalle de Rhino.

A Rhino le siguió Minerva [59][60], desarrollada conjuntamente entre la Carnegie Mellon University (Pittsburg, EUA) y la Universidad de Bonn.



**Figura 2.8** Minerva - Cara de Minerva - Minerva realizando demostraciones

Otro caso que se puede nombrar es la serie Mobot [62][23]. La serie estaba compuesta de cuatro robots cuyo objetivo era mejorar las técnicas existentes en materia de robótica

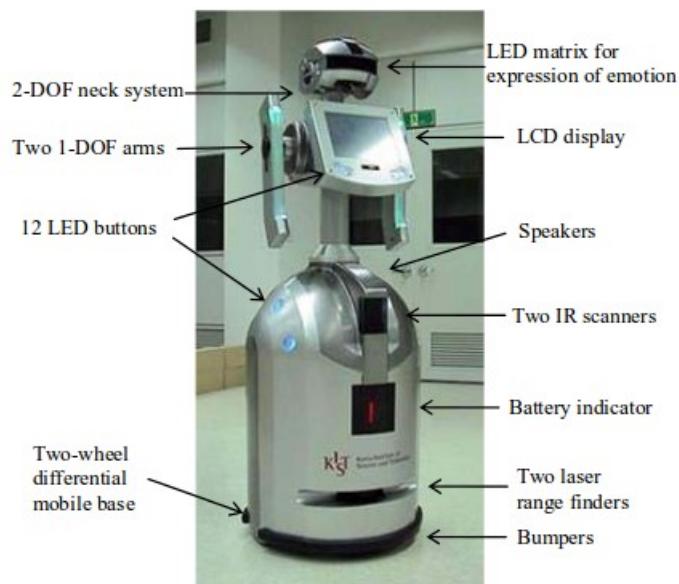
social.

RoboX [54] era una plataforma móvil completamente autónoma con múltiples capacidades de interacción.



**Figura 2.9** RoboX realizando desmotraciones - Detalle de RoboX

Otro robot guía desarrollado centrándose en la interacción humano-robot y la navegación autónoma fue Jinny [30]. Se mostró al público en 2003 en el Korea Science Festival y en centro de exposiciones Hyundai Heavy Industries. En 2005 en la Universidad de Freiburg (Alemania) se desarrolló Alpha [1].



**Figura 2.10** Robot Jinny



**Figura 2.11** Alpha en funcionamiento

Robovie [53][26] fue un robot desarrollado en Japón entre varios centros de investigación. Fue utilizado como base de experimentos para su uso como guía en museo utilizando un sistema de localización mediante la tecnología RFID. Robotinho [18] fue diseñado originalmente para jugar al fútbol en la RoboCup Humanoid League TeenSize.



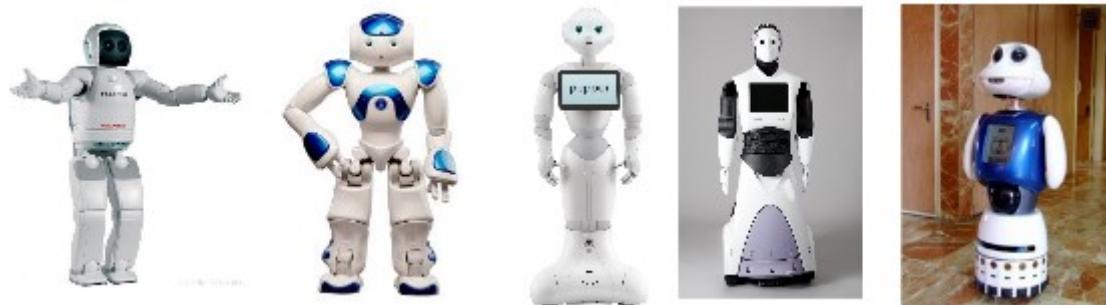
**Figura 2.12** Robotinho



**Figura 2.13** Robovie

Aparte de los anteriormente mencionados, más enfocados a las tareas de realizar visitas guiadas en museos, existen múltiples robots sociales que realizan otro tipo de servicios. Por nombrar algunos, los más destacados podrían ser Asimo (Honda), Nao (Aldebaran),

Pepper (Aldebaran), REEM (PAL Robotics), Maggie (UC3M).



**Figura 2.14** Asimo - Nao - Pepper - REEM - Maggie

#### 2.4.2. Robótica Social en el DISAM

El primer caso de estudio será Blacky. Blacky surgió del objetivo de crear un sistema completamente autónomo con grandes capacidades de interacción humano-robot, y capaz de navegar con seguridad en entornos tan complejos como pueden ser museos o exposiciones.

Blacky se mostró en público por primera vez en Indumática 2001, una feria organizada en la propia universidad. Más tarde se llevaron a cabo otras dos puestas en funcionamiento. Todo el sistema se construyó partiendo de cero, e intentando considerar todos los aspectos fundamentales de la robótica móvil: se pretendía implementar un controlador de bajo nivel para evitar las colisiones, comportamientos reactivos, llevaba un Filtro de Kalman Extendido (EKF) para la localización autónoma y la navegación. Un sistema de sintetización de voz se utilizaba como ayuda a la navegación y para interactuar con las personas presentes.



**Figura 2.15** Blacky

El control del movimiento de Blacky se dividía en dos partes:

- Por un lado estaba el controlador de bajo nivel que servía de interfaz inteligente con el hardware del robot.
- Y por el otro, una gran colección de comportamientos reactivos, patrones simples de movimiento. De la gran cantidad de comportamientos que se programaron, finalmente solo un par de ellos fueron finalmente implementados en la presentación oficial en la feria:
  - “Follow corridor”, trataba de moverse a través de un pasillo siguiendo una dirección predefinida pudiendo realizar movimientos laterales para evitar obstáculos.
  - “Intelligent escape”, realizaba una serie de movimientos semialeatorios buscando espacios libres, esto hacía que pareciesen movimientos autónomos inteligentes mientras hablaba con los humanos

Para conseguir un correcto funcionamiento durante el desempeño de sus tareas era fundamental una buena estimación de la posición, ya que si no podría descontrolarse en sus movimientos y moverse por sitios inadecuados. Para solucionar los problemas de la localización se recurrió al conocido algoritmo del Filtro de Kalman Extendido (EKF).

Para la interacción con los humanos, se recurrió a las síntesis de voz mediante las librerías IBM TTS-ViaVOICE. Cualquier frase escrita por el supervisor podía ser sintetizada por el ordenador a bordo. Esto permitía sincronizar las presentaciones orales con los movimientos del robot.

En definitiva, Blacky se puso en funcionamiento exitosamente en 3 eventos: Indumática 2001 (UPM), Cybertech 2001 (UPM) y Madrid for the Science II (IFEMA 2001). El controlador de bajo nivel para la evasión de colisiones fue de gran utilidad para un desarrollo correcto y fiable de la navegación en entornos abarrotados. Aunque el sistema de localización autónoma del robot parecía tener un funcionamiento aceptable, fallaba en múltiples ocasiones. El carácter amigable de Blacky y su personalidad generaron gran atención y alcanzaron un éxito notable.

La segunda generación de robots sociales del DISAM trajo a escena a URBANO. El objetivo principal de URBANO iba más allá de conseguir una plataforma robótica para evaluar los algoritmos de localización y navegación simultánea desarrollados en el propio departamento. Se deseaba conseguir una plataforma que sirviese para presentar los avances conseguidos al público, para ayudar a obtener financiación para futuros proyectos y lo más importante, atraer a la gente y futuros estudiantes a enrolarse en la investigación del grupo.



**Figura 2.16 URBANO**

Debido al éxito y la atracción que despertaba, desde diferentes instituciones y compañías privadas se solicitó el alquiler de URBANO para su uso durante varios días en diferentes eventos. Pero debido al carácter público y no lucrativo de la Universidad no se pudo llevar a cabo esa posibilidad.

Los trabajos realizados durante todos estos años con Blacky y URBANO han llevado al departamento al desarrollo de la tercera plataforma robótica diseñada para interacción social y orientada al uso como guía en museos. Esta es la conocida como Doris, la cual es la protagonista del proyecto que se muestra en este documento.

## Doris, el robot social chica

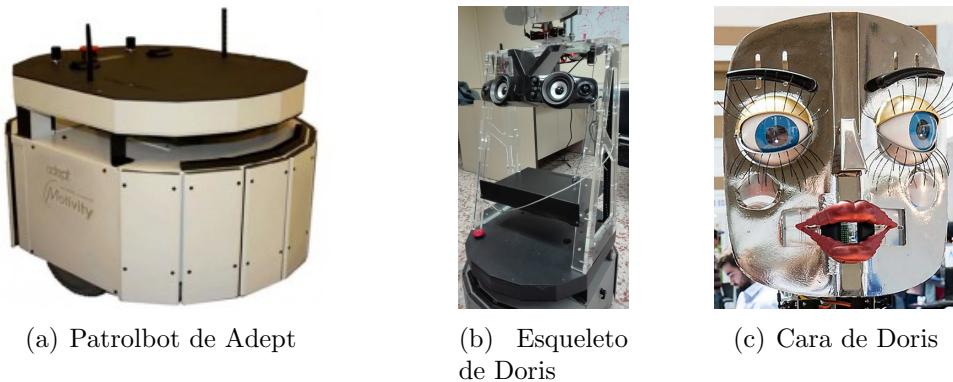
Doris es un robot móvil concebido para trabajar en entornos dinámicos de interior para interactuar con gente. Ha supuesto el trabajo de los pasados años en el CAR de la Universidad Politécnica de Madrid, representando la mejora de Blacky y Urbano, los dos robots previos en los que el grupo estuvo trabajando.

Como se comentó antes, siendo la mejora de Urbano, se espera que Doris trabaje como un sistema para interactuar con gente en espacios públicos como museos, teatros, ferias, etc.

### 3.0.1. Arquitectura de hardware

El hardware está compuesto por tres partes:

- La plataforma, diseñada por Adept MobileRobots, es un robot móvil de dirección diferencial, el cual tiene sensores ya construidos (bumpers, sonars y láser), todos conectados por un puerto serial al ordenador. El ordenador tiene puertos USB, ethernet y conexión WIFI. La figura 4.1.a muestra la plataforma
- El esqueleto, unido a la plataforma móvil, hecho de metacrilato, sostiene unos speakers, antenas RFID, un HUB USB, un switch PoE de ethernet, un lector RF y la cabeza del robot. El toso está diseñado para representar apariencia humana En el futuro, se le añadirán brazos, para desarrollar tareas de pick and place.
- La cabeza, unida al esqueleto, con 20 DoF capaces de mostrar diferentes expresiones ordenadas por el usuario del robot. El robot tiene capacidad de expresión oral, con sincronización entre la boca y el sonido emitido. Esta combinación se llama visema, mostrada en la figura 4.1.c

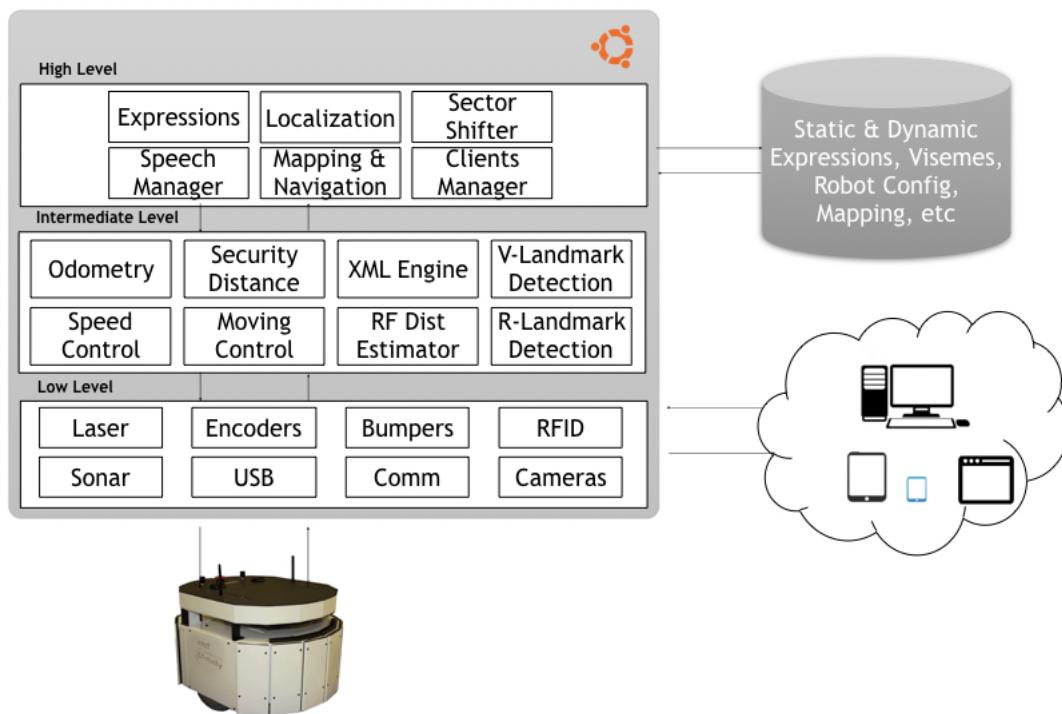


**Figura 3.1** Hardware de Doris

La información reunida por todos los sensores se usa para la localización en interiores. La operación de unión de sensores se detallará más adelante.

### 3.0.2. Arquitectura de software

El software trabaja como una aplicación Cliente-Servidor, esto significa que hay una comunicación directa entre Doris (servidor) y el que está controlándola (cliente). Esta aplicación de control puede realizarse vía web, escritorio o por dispositivo móvil. La aplicación servidor está dividida en tres niveles detallados en la figura 3.2. Cada bloque en la figura detalla una tarea, siendo ejecutada asíncronamente en un hilo diferente, lo que significa que en un nivel más alto de arquitectura, existe un gestor de tareas.



**Figura 3.2** Arquitectura de software de Doris

- Bajo nivel: Comienza las conexiones con diferentes dispositivos instalados en la plataforma (encoders, bumpers, sonars, cámaras, láser, etc.), también con dispositivos

ethernet y USB, así que este nivel permite leer y escribir datos proporcionados por niveles superiores, por ejemplo, la posición del robot, la velocidad a la que el robot debe moverse, etc.

- Nivel intermedio: En este nivel, los datos enviados por la primera capa son procesados, para ser usados por el alto nivel posteriormente, por ejemplo los datos de los encoders se traducen en coordenadas polares diferenciales para odometría, y entonces se convierten en una posición orientada

También incluye el control de movimiento proporcionado por la capa superior, por ejemplo, el control de velocidad al alcanzar la posición deseada, la revisión constante de la distancia de seguridad establecida para la telemetría láser, etc.

Una de las características más importantes en esta capa es la detección de puntos destacados, esto incluye aquellos detectados por el láser, por la cámara y por las antenas RF, proporcionando la posición de cada una para cada tipo. Finalmente también suministra un manejo de archivos XML para el mapeado y la configuración de Doris.

- Alto nivel: en este nivel, se generan las instrucciones para los movimientos de Doris. La tarea de localización tiene lugar en esta capa desde los datos suministrados en el nivel intermedio gracias al uso de puntos destacados. Adicionalmente, la lectura y escritura desde y hacia las bases de datos tienen lugar en este nivel: esto incluye la configuración del robot, el mapeado geométrico, los esquemas de visema, y el posicionamiento de la cara.

Este proyecto está basado en trabajar en el nivel intermedio de la arquitectura software. Dentro del proyecto general de Doris, en el alto nivel, ya se ha desarrollado la tecnología de mapeado, sin embargo, todavía debe crearse, como futura línea de investigación, un supervisor que maneje las instrucciones en función del controlador que sea necesario en cada momento. Para ello, este supervisor de navegación, en caso de encontrarse en el entorno de pasillo, llamará al software de Moving Conrol, dentro del cual, debe estar el controlador desarrollado, que a su vez utilizará el láser para leer los datos proporcionados por el mismo.

## 3.1. Algoritmos de localización

Para evitar utilizar algoritmos SLAM, por su alto coste operacional, se propone como método para que Doris sepa donde está, empezar primero encontrando la posición con el sistema RFID, entonces la orientación con la cámara omnidireccional, explicado en la sección 3.1.1 y se finaliza refinando la estimación a través del rango del láser, comentado en la sección 3.1.2.

### 3.1.1. Localización visual

En la plataforma del robot hay instalada una cámara omnidireccional. Este tipo de cámaras permiten un ángulo de visión de 360 grados, esto significa que dependiendo del tamaño de la habitación, podría ser vista completamente por la cámara. El modelo usado es la cámara hemisférica C25 de Mobotix, como se muestra en la figura 3.3. Ésta

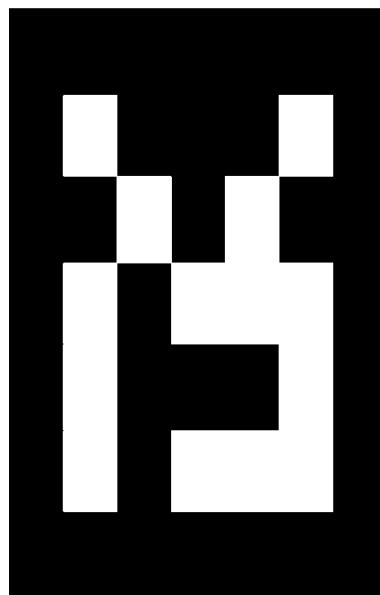
es una cámara de techo con un diámetro de 12 cm. y un peso de 200g. Esto incluye un sensor de día o noche sensible a la luz de 6MP, está alimentado y se comunica vía ethernet.



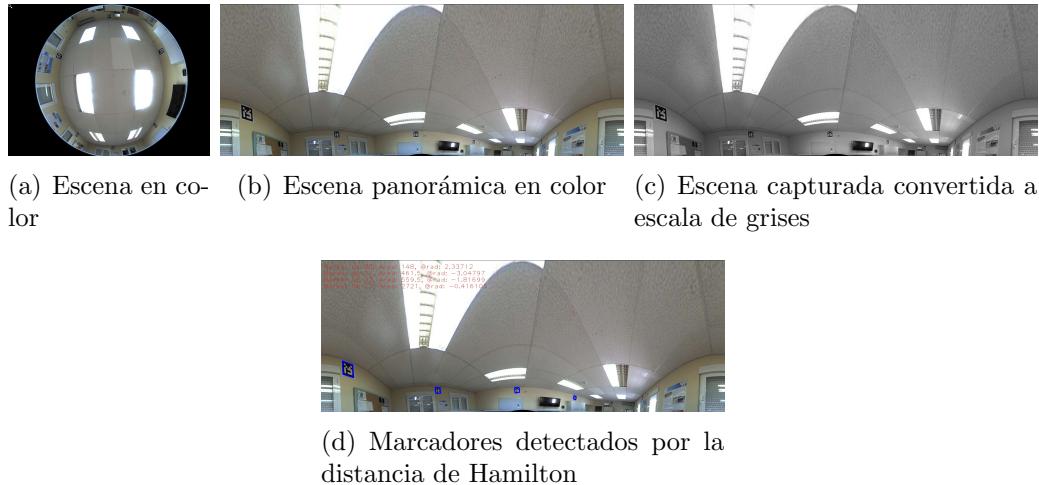
**Figura 3.3** Cámara omnidireccional C25 de Mobotix

La cámara ofrece diferentes tipos de modos de display con diferente calidad y resolución, en este caso, solo se utiliza la vista completa con una resolución de 1280x960 pixels.

La meta principal es reconocer y estimar la posición, ya sea en coordenadas polares o en cartesianas, de ciertos marcadores que siguen ciertos criterios. El marcador puede colocarse como el de la figura , impreso en un papel de tamaño A3, con una etiqueta consistente en una matriz binaria de pixeles, conteniendo un código que puede direccionar a cierto tipo de información. Esta etiqueta es robusta contra la rotación y las vistas en perspectiva. Para simplificar, los marcadores están localizados sobre la pared a 2.5m del suelo, y considerando que la altura del robot ronda los 1.45m.



**Figura 3.4** Punto clave utilizado



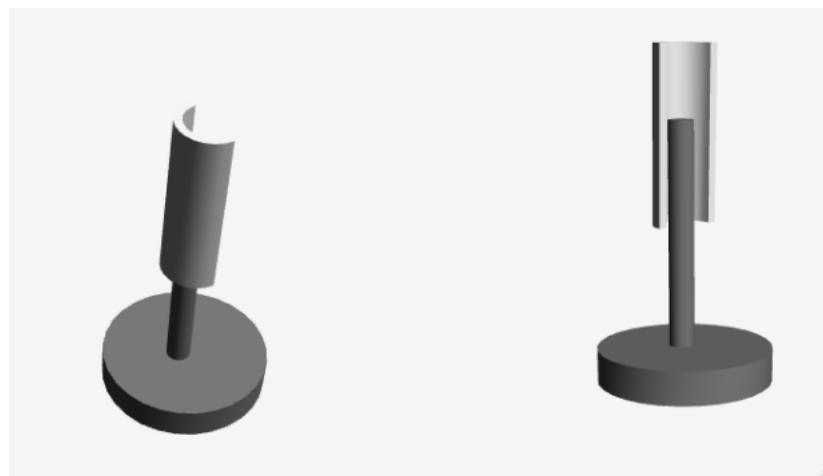
**Figura 3.5** Imágenes obtenidas en el proceso de tratamiento

### 3.1.2. Localización LRF

La tecnología laser es ampliamente utilizada en el campo de la robótica, especialmente para el diseño de mapas utilizando algoritmos SLAM. En Doris es utilizada para obtener las entradas de los controladores de navegación del robot, así como para la detección de puntos clave con algunas especificaciones. Estos puntos clave, que han sido diseñados ad hoc tienen forma circular, y están dotados de placas frontales reflectivas.

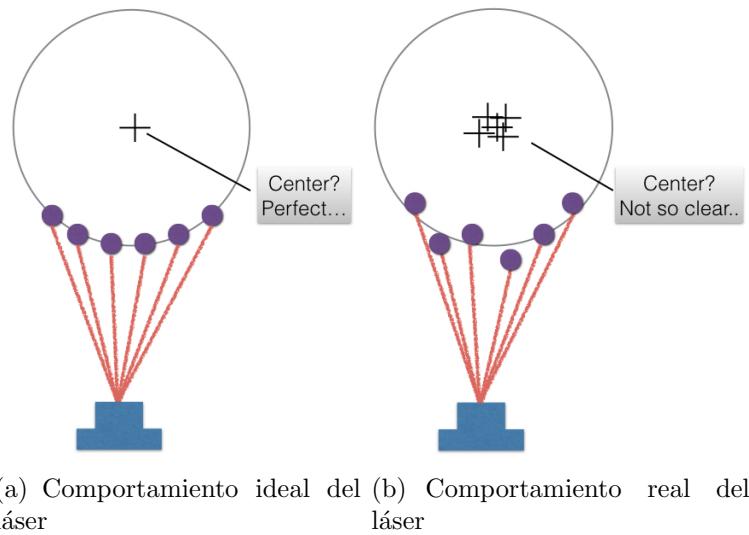
El laser tiene la funcionalidad de añadir ciertas propiedades a cada medida, por ejemplo la intensidad de reflexión cuando el haz golpea la superficie reflexiva. Sacando ventaja de esto, cada punto clave fue dotado con su respectivo material reflexivo.

La razón de su forma circular es porque un punto clave plano con el mismo material reflexivo fue probado antes, y el resultado fue que el laser solo puede detectarlo cuando el haz golpea justo enfrente del punto clave, en otro caso, el haz rebotaría hacia otra dirección, no volviendo hacia el laser. La figura muestra el prototipo de los puntos clave diseñados.



**Figura 3.6** Marcadores reflectantes diseñados por el grupo

Para localizar el centro del punto clave reflexivo con máxima precisión, se asume que cada punto clave es un cilindro perfecto con un radio conocido ( $r = 0,045m$ ). Como el laser SICK tiene la propiedad de devolver la posición en un ángulo de aquellos puntos que tienen reflectividad, se asume que esos puntos pertenecen al punto clave, y con el radio, es muy fácil obtener la posición del centro. Esto puede conseguirse aplicando la Ley de los senos y para la corrección del error, se utiliza una estimación iterativa de mínimos cuadrados. Así que la antes mencionada Ley de senos se utiliza como modelo de observación.



**Figura 3.7** Representación de las medidas obtenidas por el láser.

## 3.2. Filtro de Kalman e integración

Cada sensor descrito anteriormente luce bien funcionando por su cuenta, pero uno de los propósitos de Doris es trabajar en sinergia con su odometría a través de la fusión de sensores. Un método para lograrlo es utilizar el comúnmente conocido Filtro de Kalman. Como se comentó anteriormente, el vector de estado de la odometría del robot se define como:

$$\mathbf{x}(k) = \begin{bmatrix} x(k) \\ y(k) \\ \theta(k) \end{bmatrix} \quad (3.1)$$

La idea principal es minimizar el error de odometría que puede ser causado por acumulación de errores en encoders, ruedas que deslizan, etc., y el Filtro de Kalman puede ser adaptado para conseguir esta meta.

### 3.2.1. Filtro de Kalman

Como se propone en [28], consiste en un conjunto de ecuaciones matemáticas para proponer una solución efectiva para el problema de mínimos cuadrados, consiguiendo robustez y potencia gracias al hecho de estar basado en datos pasados, presentes y futuros. Resumiendo, es un algoritmo de mínimos cuadrados recursivo para sistemas dinámicos.

El algoritmo aborda el problema de la estimación de estado de un proceso discreto lineal a través de conjuntos de datos que pueden tener valores ruidosos.

Doris es un sistema no lineal y por esta razón debe utilizarse el Filtro Extendido de Kalman en vez de el Filtro de Kalman original que solo funciona en sistemas lineales.

### 3.2.2. Fusión del sensor con la odometría

En este punto, todos los sensores deben fusionarse en el EKF para ayudar a la odometría a conseguir un mejor posicionamiento del robot. Esto se consigue modificando algunas ecuaciones en el filtro donde la fusión del sensor tiene lugar. Esto resultará en una matriz  $2 * n$  donde  $n$  es el número de puntos clave en la habitación.

La base de datos es un fichero XML que tiene una sección que contiene información sobre los puntos clave. Esta se representa como:

```
<landmark id="2" type="laser" xpos="280" ypos="65"/>
```

donde el elemento `landmark` es parte de un conjunto mayor llamado `landmarks`. Como puede verse, el elemento contiene un atributo `type` que puede tomar tres valores distintos. Estos valores hacen referencia al tipo de sensor del punto clave, y puede ser: "`laser`" o "`camera`".

Los valores posicionales de cada punto clave se utilizarán en el proyecto para ayudar a establecer la posición inicial de Doris de manera fiable y consistente, y también para ser el punto final de las pruebas y poder medir la precisión con la que se alcanza el punto final.



## Lógica Fuzzy

La lógica fuzzy (o lógica borrosa) es una técnica muy utilizada para resolver problemas asociados a la inteligencia artificial, con la que se pretende resolver problemas inabordables por los métodos clásicos.

Esta técnica fue formalizada y desarrollada por el investigador soviético Lotfi Zadeh en 1965 en su libro *Fuzzy Sets*[63].

La lógica fuzzy, o lógica borrosa, está basada en que los seres humanos tienen la capacidad de poder tomar decisiones ante situaciones con altos niveles de incertidumbre y en ocasiones pobemente definidas, por lo que el control borroso trata de implantar en el computador, que es intrínsecamente numérico, las estrategias de control de los operadores de proceso, expresadas normalmente en términos lingüísticos, y por tanto imprecisos. El nexo de unión entre estos dos mundos, el impreciso y el numérico, se basa en la lógica borrosa.

El objetivo de la lógica borrosa es el de la formalización del razonamiento con incertidumbre en máquinas. Para ello, intenta abordar problemas definidos en términos lingüísticos, con datos expresados en términos cualitativos.

### 4.1. Orígenes del pensamiento borroso

En la lógica clásica, cualquier enunciado o proposición puede tomar un valor lógico verdadero o falso

$$\begin{aligned} p \wedge \bar{p} &\text{ siempre es falso} \\ p \vee \bar{p} &\text{ siempre es verdadero} \\ (p \Rightarrow q) \wedge p &\text{ por tanto } q \\ (p \Rightarrow q) \wedge \bar{q} &\text{ por tanto } \bar{p} \\ &\text{etc.} \end{aligned}$$

Según la Teoría de los Conjuntos Borrosos de Zadeh, los valores lógicos son conjuntos borrosos y se corresponden a términos lingüísticos del tipo “a medias”, “casi”, “poco”, “mucho”, etc. Esto permite plantear el problema a abordar en los mismos términos en los que lo haría un experto humano, ateniéndose a la imprecisión antes comentada.

En teoría clásica de conjuntos, los elementos del dominio (universo) pertenecen, o no, a un determinado conjunto, sin embargo, cuando hablamos de conjuntos borrosos, todos los elementos del dominio pertenecen a todos los conjuntos borrosos con un grado de pertenencia entre 0 y 1, que queda determinado por la función de pertenencia característica de cada conjunto.

A continuación se realiza un estudio de la base teórica de la lógica borrosa, a través de los trabajos de [15][16][29][31][38][61][64].

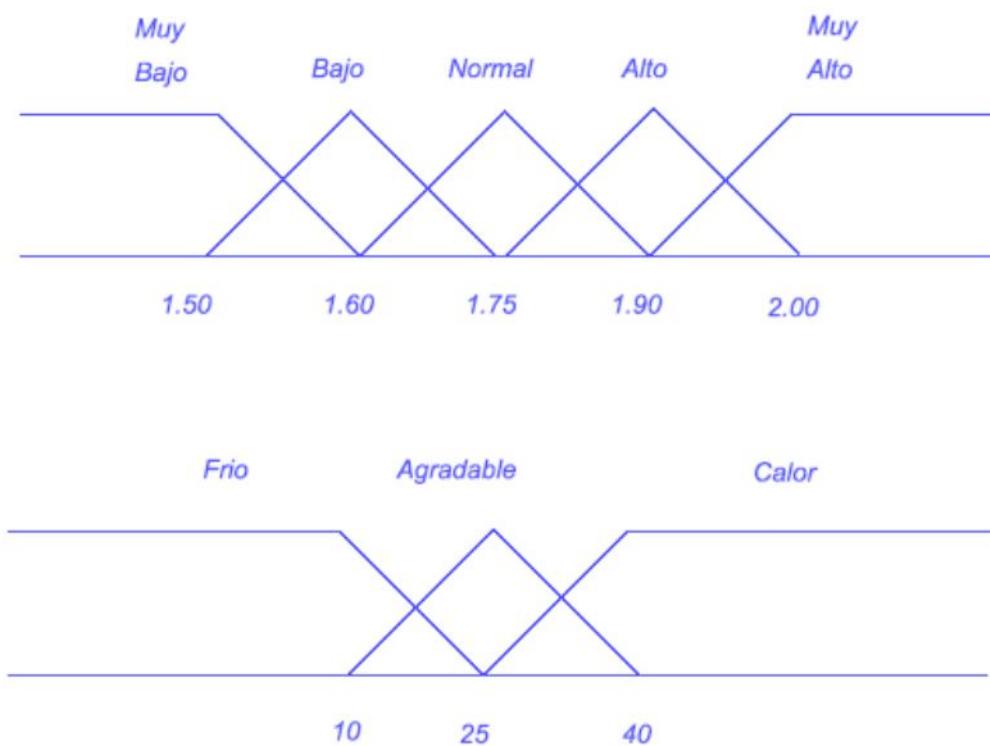
Según [27], se define que un subconjunto borroso  $A$  de un dominio  $X = \{x\}$  es un conjunto de pares ordenados

$$A = \{(x; \mu_A(x)) \mid x \in X\}$$

donde

$$\mu_A : X \rightarrow [0, 1]$$

es la función de pertenencia característica de  $A$ . Esta expresión indica el grado de probabilidad de que éste sea cierto.



**Figura 4.1** Ejemplo de conjuntos en lógica difusa

Las operaciones básicas de los subconjuntos borrosos son las siguientes:

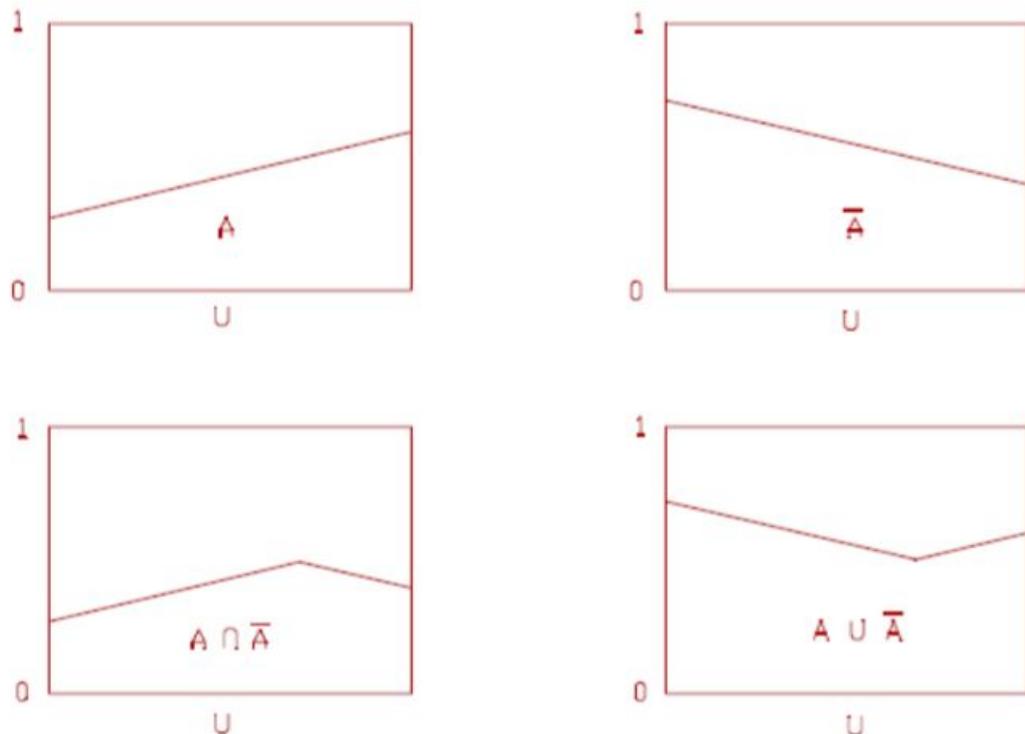
Igualdad	$A = B \Leftrightarrow \forall x \in X \mu_A(x) = \mu_B(x)$
Inclusión	$A \subseteq B \Leftrightarrow \forall x \in X \mu_A(x) \leq \mu_B(x)$
Unión	$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x)) \forall x \in X$
Intersección	$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x)) \forall x \in X$
Complemento	$A = \bar{B} \Leftrightarrow \forall x \in X \mu_A(x) = 1 - \mu_B(x)$

Se pueden definir otras operaciones de unión e intersección, ya que en vez de max se puede utilizar la suma acotada por la unidad, y en vez de min también se utiliza producto. También hay operaciones adicionales a las vistas como son producto, potenciación, distancia, etc.

En definitiva, los conjuntos clásicos se pueden considerar un caso particular de los conjuntos borrosos, en los que la función de pertenencia toma exclusivamente valores 0 ó 1. Podemos definir el conjunto vacío  $\Phi$  como aquel cuya función de pertenencia es constante e igual a cero, y el conjunto completo (el dominio  $X$ ) como el que tiene función de pertenencia constante e igual a uno. Es fácil comprobar entonces que al tratar con conjuntos borrosos la operación de complemento no da, en general, un conjunto disjunto ni complementario en el sentido clásico:

$$A \cap \bar{A} \neq \Phi \quad (4.1)$$

$$A \cup \bar{A} \neq X \quad (4.2)$$



**Figura 4.2** Conjuntos complementarios y disjuntos en lógica difusa

Por último, existen operaciones de relación en conjuntos borrosos:

- Dados dos subconjuntos borrosos  $A$  y  $B$  de dominios  $X$  e  $Y$  respectivamente, se define producto cartesiano como

$$A \times B \{((x, y); \mu_{A \times B}(x, y) = \min(\mu_A(x), \mu_B(y)) \forall x \in X \forall y \in Y\} \quad (4.3)$$

- Se define entonces relación borrosa como cualquier subconjunto del producto cartesiano.

$$ARB = \{((x, y); \mu_R(x, y) \leq \mu_{A \times B}(x, y)) \forall x \in X \forall y \in Y\} \quad (4.4)$$

- Dados tres subconjuntos  $A$ ,  $B$  y  $C$  de universos  $X$ ,  $Y$  y  $Z$  entre los que se tienen definidas las siguientes relaciones

$$AR1B = \{(x, y); \mu_{R1}(x, y)\} \quad (4.5)$$

$$BR2C = \{(y, z); \mu_{R2}(y, z)\} \quad (4.6)$$

se define la relación compuesta de R1 y R2 como

$$A(R1 \bullet R2)C \{((x, z); \max[\min(\mu_{R1}(x, y), \mu_{R2}(y, z)) \forall y \in Y]\} \quad (4.7)$$

## 4.2. Lógica y tipos de lógica

Se define la Lógica como la ciencia que estudia las condiciones formales de validez de una inferencia y, en general, de una argumentación cualquiera.

En [36] se indica que la lógica ha de tener una sintaxis, formada por:

- Variables: enunciados elementales.
- Conectivas: elementos que pueden unir a las variables.
- Sentencias: enunciados compuestos de variables y conectivas.
- Axiomas: sentencias básicas pertenecientes a la lógica que estamos definiendo.
- Reglas operativas: permiten derivar una sentencia de otra.
- Teoremas: sentencias que mediante demostración, esto es, aplicación consecutiva de reglas operativas, se pueden obtener a partir de los axiomas.
- Tesis: sentencia que es o bien un axioma o bien un teorema.

También ha de tener una semántica, formada por los siguientes elementos:

- Conjunto de valores semánticos: posibles interpretaciones de una variable o de una sentencia, con un mínimo de dos elementos.
- Operaciones semánticas: operaciones con los elementos de la lógica, de modo que a cada variable le corresponda un valor semántico y a cada conectiva una operación.
- Tautología: cuando dentro del conjunto de valores semánticos está el valor '1' o 'verdadero', se define como tautología como aquella sentencia que para cualquier interpretación de sus variables la interpretación de la sentencia siempre es verdadera.
- Contradicción: cuando la interpretación es siempre 'falsa'

Se dice de la semántica que es completa cuando toda tautología es una tesis, y que es consistente cuando toda tesis es una tautología.



**Figura 4.3** Tipos de lógica

#### 4.2.1. Lógica de proposiciones

- Como explica [27] la lógica de proposiciones, las variables proposicionales son meros enunciados declarativos, por ejemplo: "hoy es martes".
- Las conectivas son:

$\neg$	negación
$\wedge$	(y) conjunción
$\vee$	(o) disyunción
$\Rightarrow$	condicional
$\Leftrightarrow$	bicondicional

- Axiomas: existen varias posibilidades; la más conocida es la siguiente

$$\begin{aligned}(p \vee p) &\Rightarrow p \\ q &\Rightarrow (p \vee q) \\ (p \vee q) &\Rightarrow (q \vee p) \\ (p \Rightarrow q) &\Rightarrow [(r \vee p) \Rightarrow (r \vee q)]\end{aligned}$$

- Reglas operativas: la regla fundamental es la operación de sustitución por la que una variable proposicional se sustituye por una sentencia. Se podrían definir otras reglas como la de unión, que indica que si  $A$  y  $B$  son tesis entonces  $A \vee B$  es tesis, o la de separación, según la cual si  $A$  es tesis y  $A \wedge B$  es tesis entonces  $B$  es tesis
- Valores semánticos: verdadero, falso o bien 1,0.
- Operaciones semánticas: Son las operaciones conocidas del álgebra de Boole

$$\begin{array}{ll} \neg 0 = 1 & \neg 1 = 0 \\ 0 \wedge 0 = 0 & 0 \wedge 1 = 0 \quad 1 \wedge 0 = 0 \quad 1 \wedge 1 = 1 \\ 0 \vee 0 = 0 & 0 \vee 1 = 1 \quad 1 \vee 1 = 0 \quad 1 \vee 1 = 1 \\ (0 \Rightarrow 0) = 1 & (0 \Rightarrow 1) = 1 \quad (1 \Rightarrow 0) = 0 \quad (1 \Rightarrow 1) = 1 \\ (0 \Leftrightarrow 0) = 1 & (0 \Leftrightarrow 1) = 0 \quad (1 \Leftrightarrow 0) = 0 \quad (1 \Leftrightarrow 1) = 1 \end{array}$$

- La lógica de proposiciones es completa, consistente y cumple los siguientes teoremas

Tercio excluso:  $\neg p \vee p$

Ley de Morgan:  $\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$

#### 4.2.2. Lógica de predicados

Como se indica en [36], este tipo de lógica formaliza el concepto de propiedad (“Alberto es un ser vivo”) y el de relación (“Juan vive en Madrid”)

- Variables:

- Colectivos (“los animales”, “los seres vivos”), con la notación usual  $x, y, z$  para los miembros genéricos del colectivo, denominadas variables propiamente dichas.
- Miembros (“Juan”, “Madrid”), con la notación usual  $a, b, c$ , denominadas constantes.

- Conectivas:

- Las de la lógica de proposiciones.
- Propiedad: da lugar a un predicado monádico. Ej: “Alberto es un ser vivo”:  $S(a)$
- Relación: da lugar a un predicado poliádico. Ej: “Juan vive en Madrid”:  $V(j, m)$
- Cuantificador universal  $\forall$  que formaliza la idea de “para todo elemento ... se verifica ...”

- Cuantificador existencial  $\exists$  que formaliza la idea de “existe un elemento ... tal que ...”
- Axiomas (además de los de la lógica de proposiciones):

$$\begin{aligned}\forall x(P(x)) &\Rightarrow P(a) \\ \forall x(p \Rightarrow P(x)) &\Rightarrow (p \Rightarrow \forall xP(x))\end{aligned}$$

- Valores semánticos: Al igual que en el caso de la lógica de proposiciones, toman valores del tipo verdadero, falso o bien 1, 0
- Operaciones semánticas: Además de las de la lógica de proposiciones, existen operaciones para interpretar el cuantificador  $\forall$  y el cuantificador  $\exists$

$$I(\forall x(P(x)))I(P(a) \wedge P(b) \wedge P(c)\dots)$$

$$I(\exists x(P(x)))I(P(a) \vee P(b) \vee P(c)\dots)$$

- La lógica de proposiciones es completa y consistente.

### 4.2.3. Lógica borrosa

En el caso de la lógica borrosa, la sintaxis mantiene el mismo aspecto de la lógica de predicados en cuanto a variables, constantes, conectivas y cuantificadores. Sin embargo la semántica está basada en el concepto de borrosidad que se formaliza en la teoría de conjuntos borrosos.

Los valores semánticos de la lógica fuzzy obligan a que para predicado haya que definir los correspondientes subconjuntos borrosos. El universo de discurso (dominio) pasa a ser un conjunto de posibles valores particulares que pueden tomar las variables que intervienen en el predicado, y aparecen las etiquetas lingüísticas, que son los valores semánticos correspondientes a un predicado. A la capacidad de discernir entre dos términos lingüísticos se la conoce como granularidad. Por último, en la semántica fuzzy cada término lingüístico corresponde a un subconjunto borroso que lleva asociada una función de pertenencia. Esta representa el grado de asociación de un valor numérico  $\underline{x}$  con ese término.

Dado que la interpretación de un predicado es un conjunto borroso, las operaciones semánticas para interpretar una sentencia serán las correspondientes a los conjuntos borrosos

$$\begin{aligned}I(\neg A) &= \bar{I}(A) \\ I(A \wedge B) &= I(A) \cap I(B) \\ I(A \vee B) &= I(A) \cup I(B)\end{aligned}$$

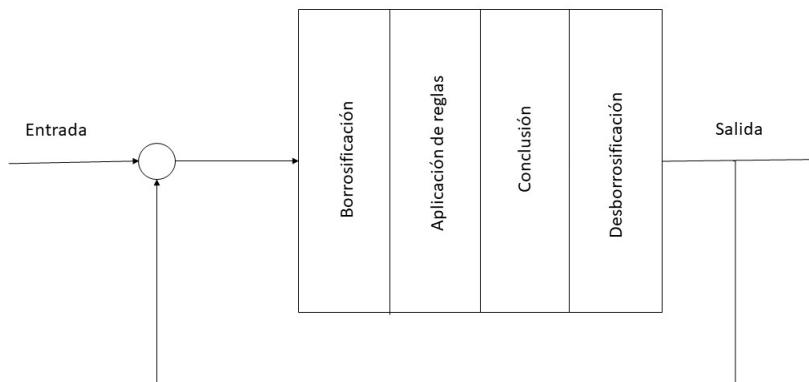
En el caso de la lógica fuzzy, tautologías de la lógica de proposiciones como  $\neg A \vee A$  no pueden aplicarse ya que no se obtendrá un valor de “verdadero o “falso, sino uno que esté dentro de un rango determinado por las funciones de pertenencia. En conclusión, la lógica borrosa no será ni completa ni consistente.

### 4.3. Controladores borrosos

Los controladores borrosos realizan, al igual que otros tipos de controladores como los PID, estabiliza las salidas de un sistema de tal manera que se mejore la respuesta, tanto temporalmente como sustancialmente. En el caso de los controladores borrosos, estos están formados por los elementos de la figura 4.4

- Borrosificación
- Aplicación de reglas
- Conclusión
- Desborrosificación

El funcionamiento del controlador borroso estará basado en la aplicación de las reglas definidas, por ejemplo “Si el error es positivo pequeño Y el cambio del error es positivo pequeño ENTONCES la acción de control es positiva grande”. En caso de no utilizar realimentación, como es el caso del presente proyecto, la entrada no será el error, sino que será directamente la información recibida. Para que el controlador pueda actuar a partir de reglas es necesario realizar un proceso de borrosificación, que transformará los valores numéricos de las variables físicas sobre las que se debe actuar, en valores borrosos. De igual forma, estas reglas deben aportar un valor numérico de control, por lo que habrá que realizar una desborrosificación una vez que se obtenga el conjunto borroso resultante de la inferencia borrosa.



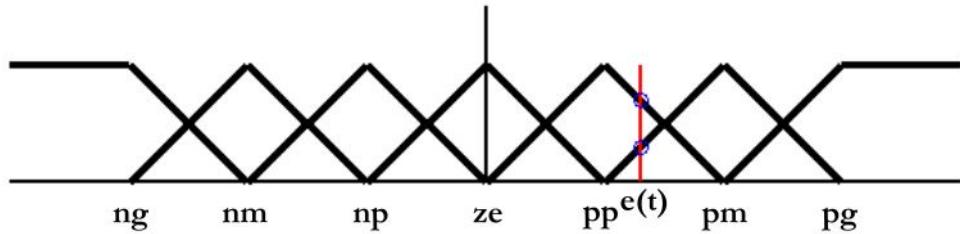
**Figura 4.4** Controlador fuzzy

#### 4.3.1. Borrosificación

Como se indicaba anteriormente, la borrosificación permite obtener, a partir de los valores deterministas de la entrada (o del error), sus valores borrosos equivalentes. Para ello es preciso tener definidos el universo de discurso, las etiquetas lingüísticas y la función

de pertenencia asociada a cada una de ellas.

La borrosificación consiste en calcular el grado de pertenencia de las variables de entrada a cada una de las etiquetas lingüísticas mediante las funciones de pertenencia. Este será un número comprendido entre 0 y 1 para cada etiqueta.



**Figura 4.5** Cálculo del grado de pertenencia de una variable

Las funciones de transferencia pueden adoptar muchas formas distintas, en función del problema convendrá más utilizar un tipo de funciones u otras. Algunas de estas formas son simples, como las funciones triangulares, o trapezoidales, mientras que otras son complejas, como distribuciones gaussianas.

### 4.3.2. Aplicación de reglas

El planteamiento de las reglas, se hace a través de reglas del tipo “if ... then ...”, el número de reglas debe abarcar todas las posibilidades del sistema, por tanto el número de reglas propuestas variará en función del número de variables de entrada al sistema, y del número de etiquetas lingüísticas en las que estén divididas las funciones de pertenencia. El conjunto de reglas se suele representar a través de una tabla, como por ejemplo:

		CE				
		ng	np	ze	pp	pg
E	ng	ng	ng	ng	np	np
	np	ng	np	np	np	ze
	ze	np	np	ze	pp	pp
	pp	ze	pp	pp	pp	pg
	pg	pp	pp	pg	pg	pg

**Cuadro 4.1** Ejemplo de matriz de reglas

Según esta tabla de reglas, para cada etiqueta lingüística tenemos un grado de pertenencia de E y otro de CE, como por ejemplo:

- IF E = ZE AND CE = ZE THEN CU = ZE
- IF E = PG AND CE = NP THEN CU = PP

Para la selección de reglas se construye una tabla adicional o matriz de inferencia que representa el peso que tendrá cada una de las reglas en la conclusión final. En el caso

de tener por ejemplo una variable de entrada en el controlador, al pertenecer el valor a dos conjuntos simultáneamente, se activarán dos de las reglas que hayan sido descritas, en el caso de tener dos variables, se activarán cuatro reglas de manera que se cubran las cuatro simultaneidades que están ocurriendo. Al ser reglas del tipo “IF E = ... AND CE = ... THEN ....”<sup>el</sup> grado de cumplimiento de la premisa será el menor de cada una de sus condiciones (o el producto) y se toma este grado como peso en la conclusión final. Se pueden eliminar reglas con bajo peso.

$$m_{ij} = \min(\mu_i(e(t)), \mu_j(ce(t))) \quad (4.8)$$

### 4.3.3. Conclusión

La acción de control que concluye cada regla es también una etiqueta lingüística de la variable de control (conjunto borroso) con función de pertenencia producto del peso por la función de pertenencia primitiva.

$$\mu'_{ij}(u(t)) = m_{ij} \cdot \mu_k(u(t)) \quad (4.9)$$

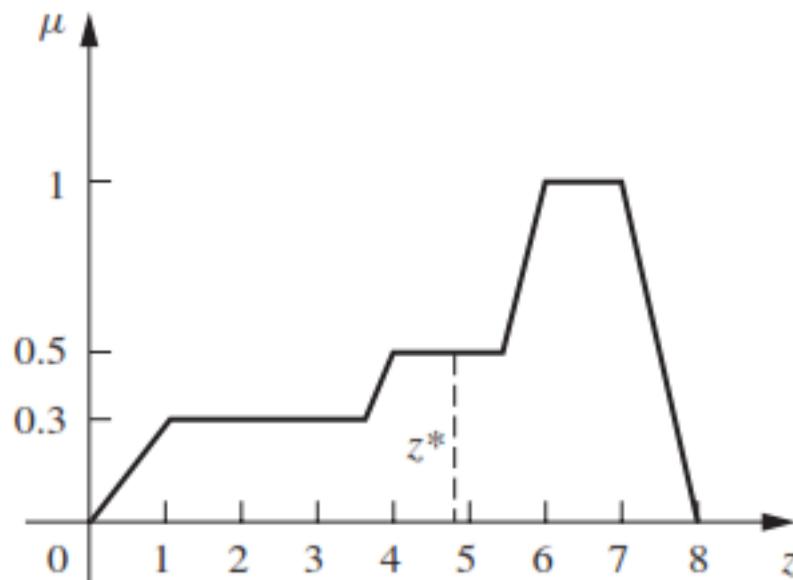
A partir del consecuente de cada regla y del valor del antecedente, y del valor del antecedente obtenido en la aplicación de reglas, se aplica un operador borroso de implicación, obteniendo un nuevo conjunto borroso. Dos de los operadores de implicación más utilizados son el mínimo, que trunca la función de pertenencia de la variable de salida, y el producto, que en este caso la escala.

### 4.3.4. Desborrosificación

Una vez que se tienen los conjuntos borrosos de salida de cada una de las reglas aplicadas, para obtener el valor numérico de la conclusión, éstas se tienen que combinar en un único conjunto borroso utilizando un operador de agregación borrosa como puede ser el máximo, la suma o el or probabilístico.

Con la función de pertenencia agregada, para desborrosificar y obtener un número como salida del sistema, uno de los métodos más utilizados consiste en calcular el centro de gravedad del área de la curva suma, en este caso no importa la forma de las funciones de pertenencia de las etiquetas lingüísticas de la función de control. Únicamente su centro de gravedad y su área, que normalmente será unitaria.

$$cdg = \frac{\sum_i \sum_j c_{ij} m_{ij}}{\sum_i \sum_j m_{ij}} \quad (4.10)$$



**Figura 4.6** Cálculo del centro de gravedad en la desborrosificación

## 4.4. Construcción del controlador borroso

En [36], se definen los parámetros de configuración necesarios para poder crear un controlador borroso funcional sin errores.

### 4.4.1. Parámetros de configuración

- Número de variables de salida.
- Número de variables de entrada.
- Errores y cambios en el error a considerar de entre todos los posibles de las variables de entrada (variables útiles), en caso de utilizar un controlador con realimentación.
- Número de términos lingüísticos a emplear para cada variable útil de entrada.
- Número de términos lingüísticos a emplear para cada variable de salida.
- La tabla de reglas (su dimensión viene definida por la elección de los parámetros anteriores).

### 4.4.2. Construcción de la tabla de reglas

La tabla de reglas debe cumplir tres principios básicos para que un correcto funcionamiento. En primer lugar, como ya se comentó en el apartado 4.3.2, las reglas han de cubrir todas las combinaciones posibles de entradas al controlador. Además, como es obvio, para una situación dada debe existir una única regla, no es posible que varias reglas abarquen

la misma situación coexistiendo. Por último, el conjunto de reglas debe ser robusto frente a perturbaciones en las entradas.

#### 4.4.3. Parámetros de ajuste

Cuando se crean las funciones de pertenencia es necesario definir por el operario los parámetros que las definen, y posteriormente esa definición debe ser ajustada para conseguir el resultado óptimo del controlador.

Por un lado, hay que definir los parámetros característicos de las funciones de pertenencia de todas las funciones de pertenencia, tanto de entrada como de salida, dónde se encuentran los centros de cada elemento de dicha función, o qué parte del conjunto universo abarcan dichos elementos.

También es posible que exista la necesidad de reajustar las salidas del conjunto de reglas, si la primera aproximación de las etiquetas lingüísticas de la salida provoca una respuesta lenta o negativa para la solución del problema.

#### 4.4.4. Ventajas e inconvenientes

Los controladores borrosos no suponen una mejora definitiva de otro tipo de controladores como los PID clásicos, pero aportan otro punto de vista que permite solucionar aquellos problemas en los que los PID no son tan eficientes.

En cuanto a las ventajas, el hecho más importante es que no es necesario un modelo preciso del sistema a controlar. También es una ventaja el hecho de que se implementan fácilmente los conocimientos del operador humano, al tener reglas expresadas en términos lingüísticos. En cuanto a las características técnicas, resulta posible alcanzar con facilidad las especificaciones de tiempo y transitorio fijadas para el controlador, además es poco sensible a cambios de los parámetros del sistema a controlar, ya que no es lineal. Por último, este tipo de controladores permiten contemplar situaciones excepcionales del estado del proceso, gracias a su forma de representar el conocimiento.

Por otra parte, algunos de los inconvenientes más notorios de los controladores fuzzy es que resulta imprescindible la presencia de un experto que suministre el conocimiento necesario, o el hecho de que una modificación en los parámetros del controlador obliga a una revisión de todo el conjunto de reglas para detectar la aparición de nuevas inconsistencias o tendencias hacia la inestabilidad.

### 4.5. Modelado borroso

La obtención del modelo borroso de un sistema dinámico puede realizarse con varios tipos de modelos como son el de Mamdani[35], el de Takagi-Sugeno[57] o el de Tsukamoto, a continuación se hará una breve introducción del funcionamiento teórico del modelo de Mamdani, ya que será el utilizado en el proyecto.

### 4.5.1. Modelo de Mamdani

Supuesta una función

$$f : R^n \rightarrow R \quad y = f(x_1, x_2, \dots, x_n)$$

se modela como conjunto de reglas de la forma

$$S^{(i_1 \dots i_n)}: \text{if } x_1 \text{ is } M_1^{i_1} \text{ and } x_2 \text{ is } M_2^{i_2} \text{ and } \dots x_n \text{ is } M_n^{i_n} \text{ then } \hat{y} \text{ is } M_y^{(i_1 \dots i_n)}$$

o en función del centro de gravedad

$$S^{(i_1 \dots i_n)}: \text{if } x_1 \text{ is } M_1^{i_1} \text{ and } x_2 \text{ is } M_2^{i_2} \text{ and } \dots x_n \text{ is } M_n^{i_n} \text{ then } \hat{y} = c^{(i_1 \dots i_n)}$$

Si  $\mu_j^{i_j}(x^{(j)})$  son las distintas funciones de pertenencia, entonces se define

$$w^{(i_1 \dots i_n)}(x) = \prod_{j=1}^n \mu_j^{i_j}(x^{(j)}) \quad (4.11)$$

y entonces

$$\hat{y} = \frac{\sum_{i_1=1}^{r_1} \dots \sum_{i_n=1}^{r_n} w^{(i_1 \dots i_n)}(x) c^{(i_1 \dots i_n)}}{\sum_{i_1=1}^{r_1} \dots \sum_{i_n=1}^{r_n} w^{(i_1 \dots i_n)}(x)} \quad (4.12)$$

## 4.6. Estado del arte en navegación con lógica fuzzy

Como ya se ha comentado anteriormente, la navegación reactiva es un área importante de investigación tecnológica, debido a la cantidad de dificultades que supone el trabajo en entornos dinámicos desconocidos, siendo la navegación con lógica fuzzy una de las que tiene mayor número de proyectos asociados.

### 4.6.1. Navegación con redes neuronales

Proyectos como [47], que recibe información a través de una cámara no calibrada, o [37], que la recibe a partir de ultrasonidos, han estudiado la navegación reactiva a través de redes neuronales, basada en la similitud con el funcionamiento neuronal humano, por lo que exploran la navegación y la posibilidad de esquivar obstáculos estáticos a través de la repetición de pruebas sucesivas de las cuales el robot aprende, con la técnica llamada “deep learning”. Por otra parte, en el proyecto [6] se consigue la evasión de obstáculos dinámicos en exterior, recibiendo datos de la señal GPS del robot.

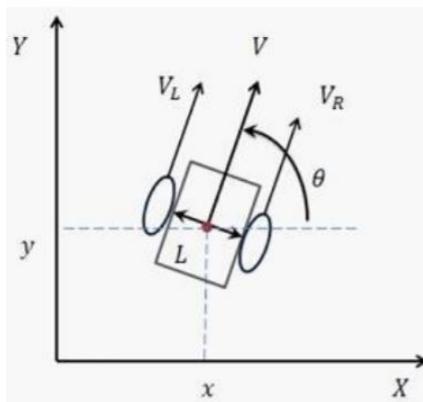
De manera distinta, en [21], el control neuronal se sirve de un algoritmo de planning que crea puntos intermedios anteriores a la meta para reconducir al robot y conseguir que llegue a la meta tomando un pequeño desvío.

#### 4.6.2. Navegación con lógica borrosa

En cuanto a la navegación y la evasión de obstáculos utilizando la lógica borrosa, la aproximación al problema se aborda desde muchos puntos de vista dependiendo en gran medida de la forma en la que se recibe y se transmite información sobre el entorno al robot.

Algunos de los primeros proyectos de desarrollo de navegación borrosa son por ejemplo el de Reignier[48], en el que se utiliza un controlador borroso para aportar conocimiento inicial a controlador basado en redes neuronales. Más adelante, en 1995, Tanaka y Sano realizan una simulación a través de un modelo de coche simulado para determinar si es posible estabilizar una trayectoria predefinida a través de un controlador fuzzy [58].

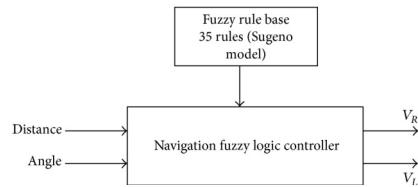
Más recientemente, Li [34] realiza un trabajo sobre evasión de obstáculos estáticos en espacios interiores en el que el sistema fuzzy está formado por tres entradas, las medidas obtenidas por tres sensores de ultrasonidos colocados en el robot, dos en sus laterales y otro en el frontal. Las variables controladas son las velocidades independientes de las dos ruedas que forman el robot, como puede verse en la figura 4.7.



**Figura 4.7** Modelo cinemático del robot de Li

Por su parte, [19] crean en 2013 un sistema de navegación y evasión de obstáculos en entornos basado en la recepción de datos de sensores de ultrasonidos y dos controladores fuzzy, uno de ellos dedicado a la navegación cuando y la detección de obstáculos, y otro a la evasión de obstáculos, con la particularidad de que en este caso, en el controlador dedicado a la navegación, las entradas del sistema son la distancia y el ángulo relativos a la meta. Esta configuración de entradas y salidas en el controlador fuzzy de navegación es utilizado también en [42]. En [33] se estudian varias configuraciones de entradas y salidas, obteniendo información de un vector de sensores de ultrasonido, y un sensor infrarrojo para detectar la meta. El trabajo de Srdjan y Zeljko [56] estudia la evasión de uno o más estímulos fijos, aunque bastante alejado de las otras propuestas, ya que utiliza un controlador basado en imanes virtuales fuzzy.

Se puede concluir que la construcción de controladores fuzzy basados en las mediciones láser y con salidas en la velocidad lineal y angular en entornos semidesconocidos, es una posibilidad con mucho margen de mejora en cuanto a la evasión de obstáculos dinámicos, ya que la mayoría de la literatura está basada en la evasión de obstáculos estáticos.



**Figura 4.8** Esquema de control de un controlador fuzzy de navegación. [42]



# **Implementación del controlador**

---

Como ya se ha mencionado, el trabajo expuesto a continuación tiene por objetivo la creación de un controlador para robots móviles que permita que estos sean capaces de mantenerse centrados en un entorno de pasillo, evitando posibles obstáculos, y siendo capaces de atender a situaciones como la apertura de una puerta, o cambios de anchura, ángulo en el propio pasillo. Para el desarrollo del código se utilizará una biblioteca con diversos elementos de lógica difusa con la que poder crear el controlador de navegación de pasillo, cuyas variables de control serán las velocidades lineal y angular.

## **5.1. Plataformas de desarrollo y ejecución**

En la realización de este proyecto se ha utilizado un PC propiedad del CAR de la UPM y el ordenador portátil personal del alumno, debido a la comodidad a la hora de realizar las distintas pruebas físicas con Doris. El PC del departamento utiliza, como es habitual en el grupo de robótica móvil, el Sistema Operativo Linux, mientras que el ordenador del alumno utiliza el Sistema Operativo Windows 10. En ambos ordenadores se ha desarrollado el proyecto a través del lenguaje de programación C++ sobre el entorno de desarrollo Sublime Text, software de código abierto.

Con ello se consigue la compatibilidad con otros trabajos del grupo y se tiene una buena portabilidad del desarrollo. Algunas de las ventajas del lenguaje C++ son su buena capacidad de cálculo, el hecho de que es un lenguaje orientado a objetos y que se trata de un lenguaje de alto nivel con flexibilidad y potencia expresiva. Todo ello permite reducir el tamaño y la complejidad del código. Además, al tratarse de un lenguaje compilado, presenta una buena eficiencia en tiempos de ejecución frente a los lenguajes interpretados. También cabe destacar que gran parte de los entornos de programación distribuidos por los fabricantes de robots móviles están escritos en este lenguaje. Por supuesto, también es compatible con plataformas de desarrollo libre.

El hecho de utilizar Sublime Text para el desarrollo del código, y la no necesidad de trabajar con un entorno desarrollo visual, genera gran versatilidad y permite generar y compilar código en todas las plataformas.

## 5.2. Bibliotecas empleadas

Para el desarrollo del software del proyecto se ha hecho uso de la biblioteca fuzzy-lite para desarrollar todos los métodos necesarios para conseguir tratar los datos como etiquetas lingüísticas.

### 5.2.1. Biblioteca FuzzyLite

La biblioteca FuzzyLite[46] es una biblioteca de código abierto multiplataforma, ya que puede trabajar en Windows, GNU/Linux, Mac, iOS. Además, es posible utilizarla tanto en programación con C++, como Java o Android.

La biblioteca FuzzyLite contiene los siguientes elementos:

- Controladores: Mamdani, Takagi-Sugeno, Larsen, Tsukamoto, etc.

Distintos tipos de controladores para realizar la inferencia borrosa. En este proyecto se utiliza el método de Mamdani.

- Términos lingüísticos: Básicos como triángulos, trapezoides, rectángulos, o complejos como producto gaussiano, o una función.

Permiten crear funciones de pertenencia de los conjuntos borrosos, donde cada valor representa el grado en el que el elemento pertenece al conjunto borroso.

- Métodos de activación: General, proporcional, el más alto, el más bajo, etc.

Indica el grado de activación de cada regla, para proporcionar la función de pertenencia a desborrosificar.

- T-Normas para la conjunción y la implicación: Producto algebraico, mínimo, etc.

Estos operadores representan la intersección de dos conjuntos borrosos.

- S-Normas para la disyunción y la agregación: Mínimo, suma algebraica, etc.

Estos operadores representan la unión de dos conjuntos borrosos.

- Desborrosificadores: Pueden ser integrales como a través de la centroide, la media de los máximos, etc. o ponderados a través de la media o de la suma.

Convierten los conjuntos borrosos de salida en valores numéricos exactos.

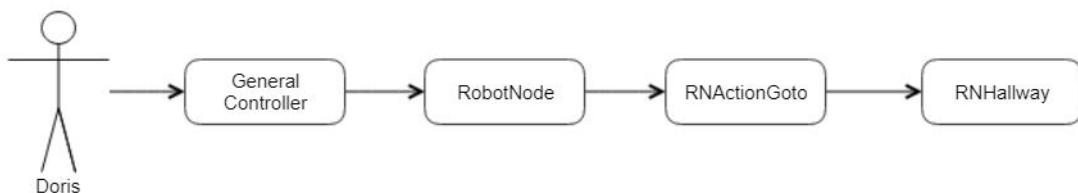
- Conectores: “algo”, “no”, “muy”, etc.

Permiten realizar distintas acciones dentro del bloque de reglas.

### 5.3. Controlador teórico

El objetivo del controlador borroso del presente proyecto es mantener centrada a Doris en un entorno de pasillo que se ve afectado por objetos dinámicos que varían el entorno, o por objetos que no están previamente mapeados, pero que no deben impedir el cumplimiento del objetivo de Doris, que no es otro que el de cruzar de un lado a otro del pasillo de forma segura y eficiente.

Por tanto, el primer paso, obviamente es la necesidad de recibir una señal que indique a Doris que se va a encontrar dentro de un pasillo, permitiendo el buen posicionamiento inicial. Esta señal será negativa una vez que se haya sobrepasado el pasillo, alternando con el control de navegación general en salas, quedando tanto el control de navegación en salas como un supervisor de los distintos controladores de navegación como algunas de las futuras líneas de investigación posible.



**Figura 5.1** Diagrama de casos de uso del controlador RNHallwayController

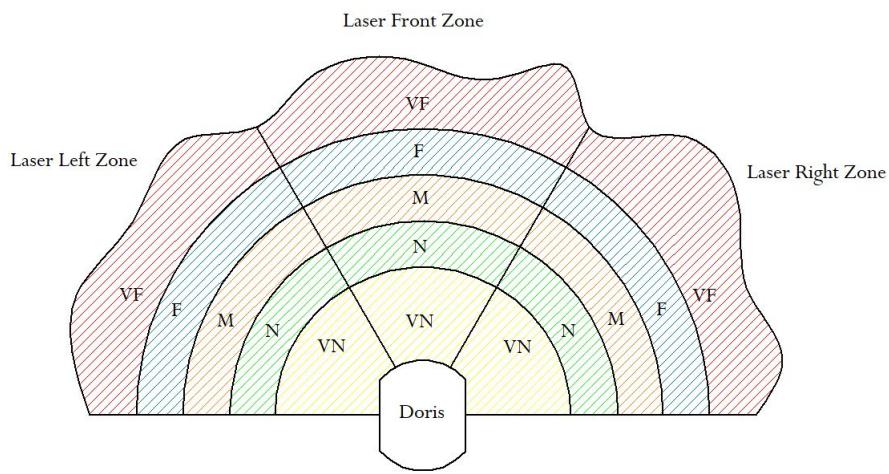
Una vez que se avisa a Doris del hecho de estar dentro de un pasillo, el controlador RNHallwayController() realizará iteraciones sucesivas de manera continua en el tiempo hasta que sobresepa la zona del mapa que marca el fin del pasillo, o llegue hasta el punto de destino que se le indique dentro del pasillo.

El controlador borroso estará formado por un sistema fuzzy de cinco variables, tres de entrada y dos de salida a controlar, como puede verse en la figura 5.2. Las variables de control serán la velocidad angular y la velocidad lineal del robot, con el objetivo de mantener el robot equidistante a las paredes en todo momento.



**Figura 5.2** Esquema del controlador RNHallwayController

Por otra parte, Doris toma 361 mediciones láser divididas entre sí por  $0,5^\circ$ , por lo que se decide dividir las 361 medidas en tres sectores, que abarcan  $60^\circ$  cada uno. Los sectores izquierdo y derecho, tendrán 120 mediciones del rango del láser, mientras que la zona central, tendrá 121 mediciones distintas. Una vez que se hayan dividido las 361 mediciones, el objetivo es conseguir una aproximación de a qué distancia está la pared, o a qué distancia está el objeto significativo más cercano.



**Figura 5.3** Esquema de la sectorización de las mediciones realizadas con el láser

Con el doble propósito de evitar medidas falsas debidas a polvo, o cualquier otro tipo de interferencia, y la idea de ganar en precisión discriminando medidas no significativas, se añade otra sectorización radial, que divide la distancia de las mediciones en cinco posibles zonas, considerando que la última llega al infinito, como se muestra en la figura 5.3.

Si un número  $n$  de medidas entra dentro de la zona más cercana, se realizará la media de las medidas dentro de ese sector (y se descartarán el resto de mediciones), ese valor de la media, será la entrada al sistema fuzzy. En caso de no haber suficientes medidas dentro del sector radial, por ejemplo, si hubiera solo una de las 120 medidas dentro del sector radial, con toda probabilidad no habrá realmente un objeto que pueda colisionar con Doris. Si en definitiva, no hay un número significativo de medidas dentro de esa zona, se comprobarán la siguiente zona más cercana, así, hasta en el caso más extremo, llegar a la zona más lejana.

Tanto las distancias que separan los distintos sectores radiales, como el número de mediciones necesarias para considerar que la zona está activa, serán motivo de ajuste en el apartado de pruebas, de tal forma, que como se explicó en el apartado 4.4.4 es necesaria la presencia de un experto humano, capaz de suministrar unos datos coherentes en función de las necesidades específicas de funcionamiento.

El siguiente paso es entender cuáles son los valores de las distancias del sector izquierdo, frontal y derecho que sirven como entrada que sirven como entrada al sistema fuzzy. En el caso de tomar simplemente como valores de entrada las medias calculadas en la última iteración, el controlador funcionaría en cierto grado, es decir, si se toma siempre en cuenta el último valor de las mediciones, cualquier medida discrepante provocaría una alteración en el controlador, por ejemplo, el hecho de que una persona se situara paralela a Doris, en principio no debería provocar un cambio en el rumbo, ya que sería una situación momentánea (sí debería haber cambio en el rumbo si la persona se mantuviera en la zona de influencia del láser), sin embargo, con esta idea, la reacción de Doris sería la de esquivar a la persona, ya que detectaría mediciones mucho más cercanas en uno de sus lados y trataría de buscar el camino libre. De igual manera, el hecho de encontrar una puerta abierta durante el trayecto, provocaría una grave desviación del robot, llegando incluso a colisionar con la puerta, son todos estos casos los que el controlador debe gestionar de alguna manera.

Como primera aproximación se piensa en no trabajar con las medias de la última iteración, sino con el promedio de las últimas  $n$  entradas al sistema de tal manera que una o varias entradas que den señales falsas no afectaran al rumbo de Doris, siendo el número de iteraciones  $n$  del sistema, una variable a ajustar en el capítulo 6.

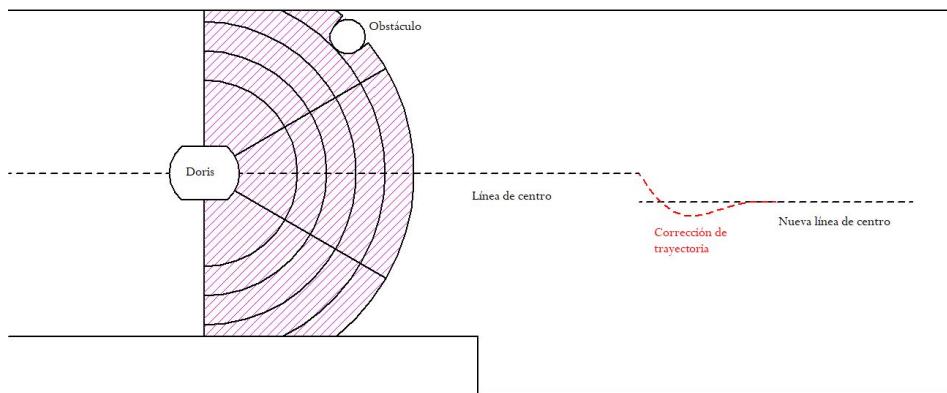
En caso de que la diferencia entre la última medición y el promedio de las últimas  $n$  mediciones supusieran una diferencia significativa, se activaría un sistema de comprobación, que indicará si esa diferencia de distancia se mantiene durante las siguientes  $n$  iteraciones. Si, efectivamente, la comprobación resultase negativa, se habrían obviado las mediciones de la comprobación, sin embargo, si la comprobación fuera positiva, se consideraría que la anchura del pasillo se había visto efectivamente alterada.

Al realizar pruebas, se comprobó que el hecho de trabajar con una media, y no exactamente con el último valor, en un controlador sin realimentación, provocaba serios problemas de precisión y de cabeceo en el movimiento, así como una respuesta del sistema mucho más lenta y con errores en ciertas posiciones.

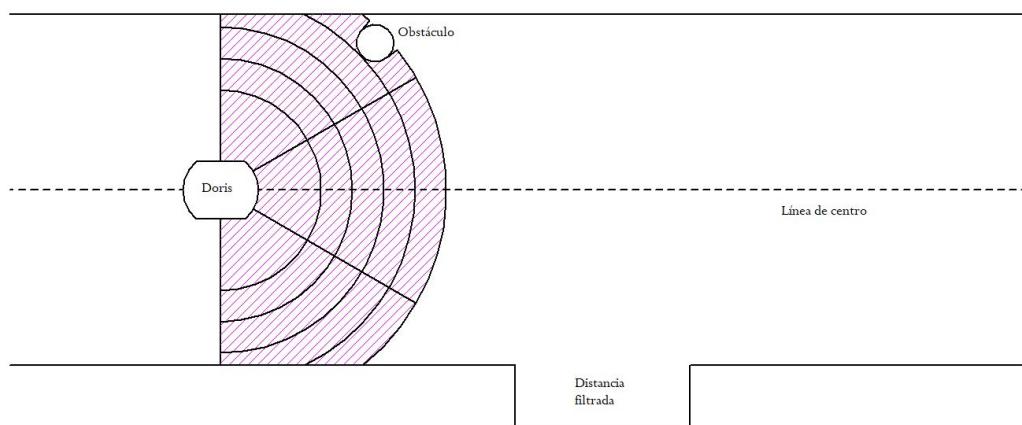
Por tanto, se prefiere volver a la idea inicial en el que las medias aceptadas de los

tres sectores son la entrada al sistema fuzzy. Como se pudo comprobar posteriormente, la navegación era mucho más precisa, y la respuesta mucho más rápida. Para solventar el problema de los objetos dinámicos se decide crear un filtro que funciona de la siguiente forma:

1. Se crean dos vectores, uno con valores “aceptados” otro con valores “rechazados”. Si el valor absoluto de la diferencia es superior al valor  $k$ , entonces se considera que dicho valor es rechazado. Si la diferencia entre la distancia medida en el sector izquierdo y la medida en el sector derecho es inferior a un valor  $k$ , o si el robot no se ha estabilizado todavía, se considera que el valor es aceptado.
2. Un bucle comprueba si en alguna de las últimas  $i$  mediciones aceptadas la diferencia es mayor que la distancia límite, si esta condición es cierta, se considera que el sistema no está estabilizado todavía.
3. Si el sistema no es estable, todos los cambios en las medias de las entradas, conseguirán que haya un cambio en el rumbo de Doris, así se consigue por ejemplo que si el controlador empieza a actuar y Doris empieza desde un punto no centrado en el pasillo, pueda conseguir la posición correcta.
4. Sin embargo, si el robot se ha estabilizado en una posición central y sin embargo, recibe una señal de que la diferencia entre el sector izquierdo y el derecho es mayor que el valor  $k$  anteriormente comentado, el filtro comienza a actuar, y la media de cada uno de los tres últimas entradas pasa al vector tridimensional de valores descartados. Entonces pueden ocurrir dos casos:
  - Si la diferencia se mantiene durante  $n$  iteraciones, entonces es obvio que el pasillo habrá cambiado, por lo que se debe vaciar el vector de valores aceptados, y copiar el de los valores rechazados en el primero, ya que estos valores deberían haber sido aceptados todo el tiempo anterior.
  - Sin embargo, si la diferencia no se mantiene durante las  $n$  iteraciones, se borrará el vector de valores descartados, y se supondrá que era una “falsa” diferencia, como podría ser el cruce de una persona con el robot.
5. Mientras se está realizando la comprobación, la velocidad angular será nula, para mantener la dirección de la última iteración anterior al filtro.
5. El valor de las  $n$  iteraciones del filtro, de las últimas  $i$  mediciones aceptadas, y de la distancia límite  $k$  deberán ser objeto de ajuste en el apartado



(a) Esquema de funcionamiento cuando existe un cambio en el pasillo

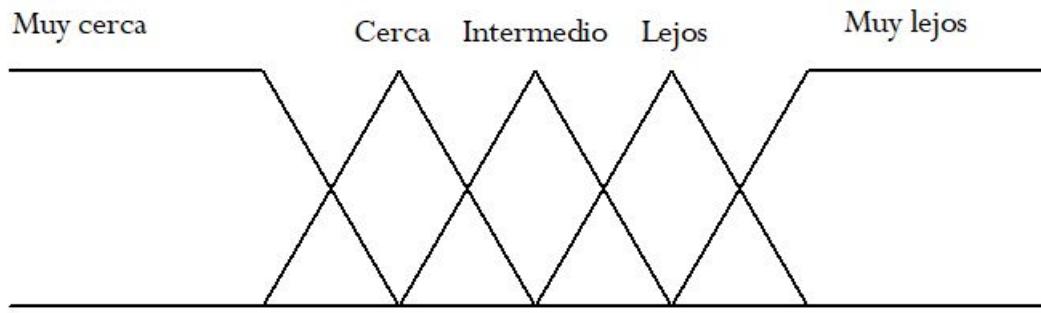


(b) Esquema de funcionamiento cuando el filtro actúa

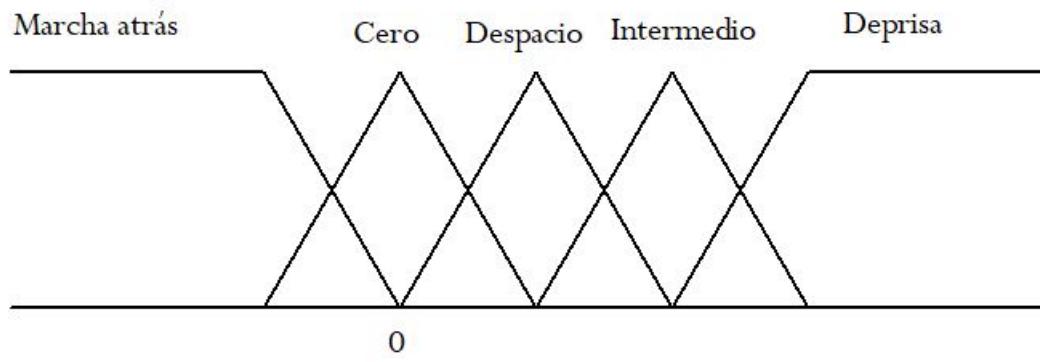
**Figura 5.4** Esquema de navegación de pasillo de Doris

Una vez que se han escogido las entradas al sistema y las salidas, debe crearse un sistema borroso con el que obtener las velocidades lineales y angulares que permitan la correcta navegación. Para conseguir esto, habrá que definir unas funciones de pertenencia tanto para las entradas como para las salidas, estas funciones de pertenencia tendrán componentes con forma triangular, pero sus valores deberán ser ajustados, en apartados posteriores. Además, su rango de acción estará entre  $-\infty$  e  $\infty$  para evitar la posibilidad de valores límite.

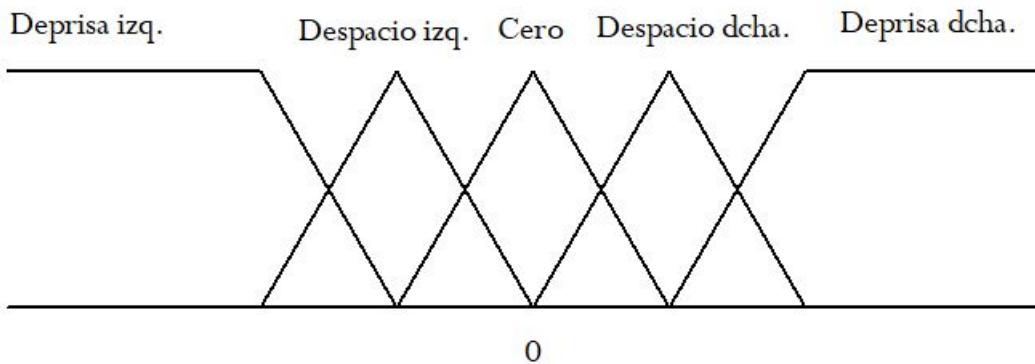
Como puede verse en las figuras 5.5, 5.6 y 5.7 se decide que tanto las funciones de pertenencia de las variables de entrada al sistema, como las funciones de pertenencia de las variables de salida estarán compuestas por cinco etiquetas lingüísticas distintas. En la función de pertenencia de la velocidad lineal, el cero se encontrará desplazado a la izquierda, ya que esto permitirá mayor flexibilidad a la hora de dar valores de velocidad a Doris, y la marcha atrás solo se utiliza en caso de sobrepasar la zona objetivo, o encontrarse extremadamente cerca de un objeto.



**Figura 5.5** Función de pertenencia tipo de las variables de entrada



**Figura 5.6** Función de pertenencia tipo de la velocidad lineal



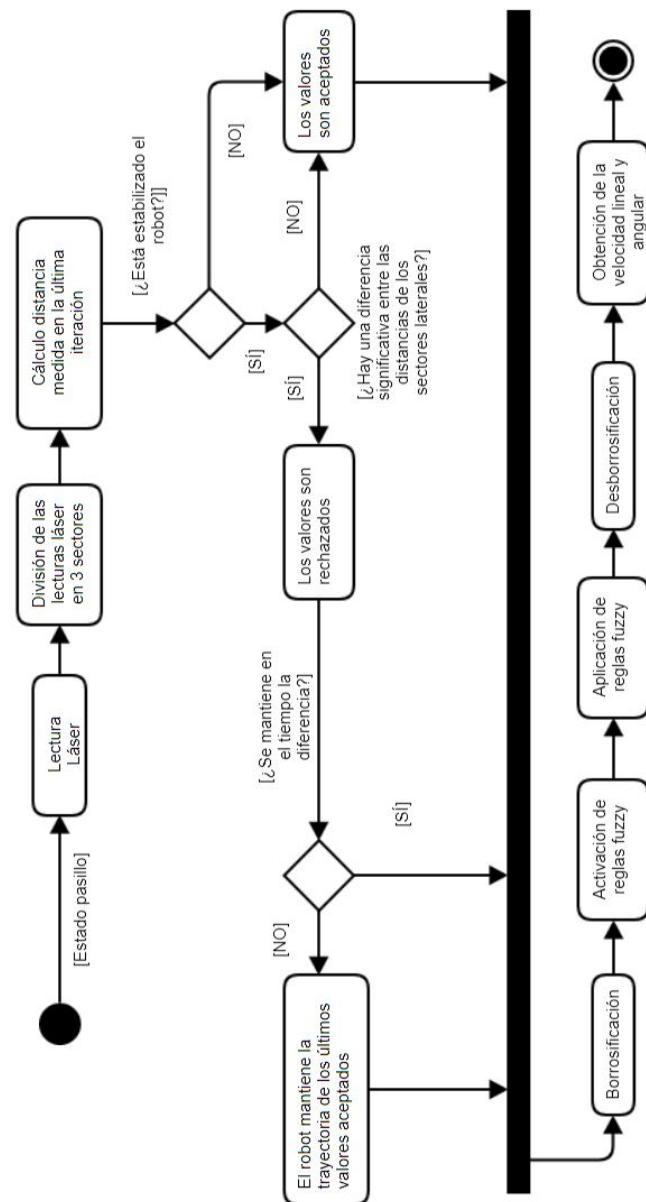
**Figura 5.7** Función de pertenencia tipo de la velocidad angular

En el capítulo 4 se vio que en lógica difusa, un objeto puede pertenecer a dos conjuntos simultáneamente, por tanto, para ver en qué porcentaje pertenece cada variable a cada etiqueta lingüística, se utilizará la suma algebraica.

Como se vio en el apartado 4.3.1, al haber tres entradas y dos salidas, las reglas serán del tipo “**IF ... AND ... AND ... THEN ... AND ...**”. Para calcular la conjunción de las tres etiquetas de entrada, se utilizará el producto algebraico, al igual que para calcular

la activación de las reglas. Una vez que se han calculado las etiquetas lingüísticas para los valores de salida, se realiza una disjunción por suma algebraica, y se procede al último paso, desborrosificar los valores de salida a través de una centroide.

El árbol de reglas intenta seguir dos principios básicos, la velocidad lineal ha de ser más alta cuanto más lejos se esté de las paredes, por un lado por la seguridad del robot, y por otro lado porque cuanto más cerca se esté de las paredes, habrá interés en tener velocidades bajas para ganar en maniobrabilidad. De manera contraria, la velocidad angular deberá ser más alta cuanto más cerca se esté de las paredes, para que el robot gire y consiga esquivar la pared. Por otra parte, si las distancias medidas por el robot en los lados son similares, y el espacio frontal está libre, interesaría que la velocidad angular sea nula, ya que eso significará que está siguiendo la dirección correcta.



**Figura 5.8** Diagrama de actividades

## 5.4. Estructura de clases

En la figura 5.9 puede verse el diagrama UML de las clases que conforman el sistema. Aunque solo se ha implementado la clase `RNHallwayController`, se han realizado ciertas modificaciones a las clases `RNAActionGoto` y `RobotNode` de tal manera que puedan relacionarse de forma correcta con la clase creada en este proyecto. A continuación se explica el uso que se ha hecho de estas clases.

### 5.4.1. La clase RNAActionGoto

La clase `RNAActionGoto` se encarga principalmente de gestionar la navegación de Doris, es la encargada de suministrar al robot la información acerca de dónde se encuentra la meta, si se ha alcanzado o no, las velocidades lineales y angulares proporcionadas por los distintos controladores, etc.

Es la clase que indica si es necesario el uso del controlador `RNHallwayController`. Sus métodos son los siguientes:

```
haveAchievedGoal  
  
bool RNAActionGoto::haveAchievedGoal(void)  
haveCanceledGoal
```

```
bool RNAActionGoto::haveCanceledGoal(void)
```

Estas dos clases indican si se ha alcanzado la meta, y si se ha cancelado el punto de destino respectivamente.

```
cancelGoal  
  
void RNAActionGoto::cancelGoal(void)
```

Esta clase permite cancelar el punto de destino.

```
setGoal  
  
void RNAActionGoto::setGoal(ArPose goal)
```

Crea un nuevo punto de destino dentro del mapa que Doris tiene que alcanzar.

```
getLinearSpeed  
  
double RNAActionGoto::getLinearSpeed(void)  
  
setLinearSpeed  
  
void RNAActionGoto::setLinearSpeed(double speed)
```

Estos métodos permiten establecer la velocidad lineal de Doris y comprobarla en cualquier momento.

```
getAngularSpeed

double RNActionGoto::getAngularSpeed(void)

setAngularSpeed

void RNActionGoto::setAngularSpeed(double speed)
```

De igual manera, establecen la velocidad angular del robot, y permiten comprobarla.

```
getMinimumDistance

double RNActionGoto::getMinimumDistance(void)

setMinimumDistance

void RNActionGoto::setMinimumDistance(double distance)
```

Indica la distancia mínima en línea recta hasta el punto de destino de Doris.

```
getMinimumAngle

double RNActionGoto::getMinimumAngle(void)

setMinimumAngle

void RNActionGoto::setMinimumAngle(double angle)
```

En este caso, se calcula el mínimo ángulo de giro para orientar a Doris en dirección al punto de destino.

```
getGoal

double RNActionGoto::getGoal(void)
```

Devuelve el valor del punto en el que se encuentra la meta.

```
fire

ArActionDesired* RNActionGoto::fire(ArActionDesired current))
```

La función fire crea un objeto del tipo **ArActionDesired**, en el que se decide en función del estado del robot, si el robot está dirigiéndose a la meta, la función calcula indica a Doris las velocidaes lineales y angulares a las que debe moverse. Si el estado indica que se ha alcanzado la meta, se devuelven valores nulos de velocidad y se reinician los

controladores que indican las velocidades necesarias para que la navegación exitosa.

`current` indica el estado actual de Doris.

## 5.5. La clase RobotNode

La clase RobotNode realiza varias tareas de diversos ámbitos, como comprobar el estado de los motores, o bloquear y desbloquear el robot. Muchas de estas funciones no son utilizadas directamente para la realización del proyecto, por lo que solo se va a comentar aquellas que están íntimamente relacionadas con el desarrollo del código, que serán aquellas que realizan funciones con el láser.

`isLaserReady`

```
bool RobotNode::isLaserReady()
```

Indica si el láser está listo para ser usado.

`setLaserReady`

```
void RobotNode::setLaserReady(bool ready)
```

Prepara el láser para ser usado al encender el robot.

`connected`

```
void RobotNode::connected(void)
```

`connectionFailed`

```
void RobotNode::connectionFailed(void)
```

`disconnected`

```
void RobotNode::disconnected(void)
```

`connectionLost`

```
void RobotNode::connectionLost(void)
```

Estas funciones indican respectivamente si el robot se ha conectado exitosamente, si dicha conexión ha fallado, si se ha desconectado el robot, o si se ha perdido la conexión momentáneamente.

`disconnect`

```
void RobotNode::disconnect()
```

Para los motores y deja de correr las distintas funciones.

#### `getRobotPosition`

```
void RobotNode::getRobotPosition()
```

Esta función consigue la posición y la velocidad de Doris dentro del mapa

#### `stopRobot`

```
void RobotNode::stopRobot
```

Detiene el movimiento del robot.

#### `moveAtSpeed`

```
void RobotNode::moveAtSpeed(double linearVelocity,
double angularVelocity)
```

Esta función indica al robot a qué velocidad debe moverse, en caso de tener que ir hacia un punto de destino.

#### `gotoPosition`

```
void RobotNode::gotoPosition(double x, double y, double theta,
bool isHallway, double transSpeed, double rotSpeed)
```

Esta función sirve para hacer que Doris vaya al punto de destino marcado en el mapa.

`x` marca la distancia a moverse en el eje de abscisas, `y` la distancia a moverse en el eje de ordenadas, `theta` el ángulo necesario al punto de destino, `isHallway` indica si está activo el controlador de navegación en pasillo, en el futuro se creará otro controlador para navegación general, `transSpeed` marca la velocidad de translación y `rotSpeed` la velocidad de rotación.

#### `setPosition`

```
void RobotNode::setPosition(double x, double y, double theta)
```

Esta función establece el punto que tiene que alcanzara Doris, por tanto en este caso los parámetros `x`, `y` y `theta` indican valores totales hasta la meta y no los restantes, como en la función anterior.

#### `onLaserScanCompleted`

```
void RobotNode::onLaserScanCompleted(LaserScan* data)
```

En caso de que el escaneo realizado con el láser se complete, se guardan los datos de las mediciones en un objeto de la clase `LaserScan`.

`getLaserScan`

```
LaserScan* RobotNode::getLaserScan()
```

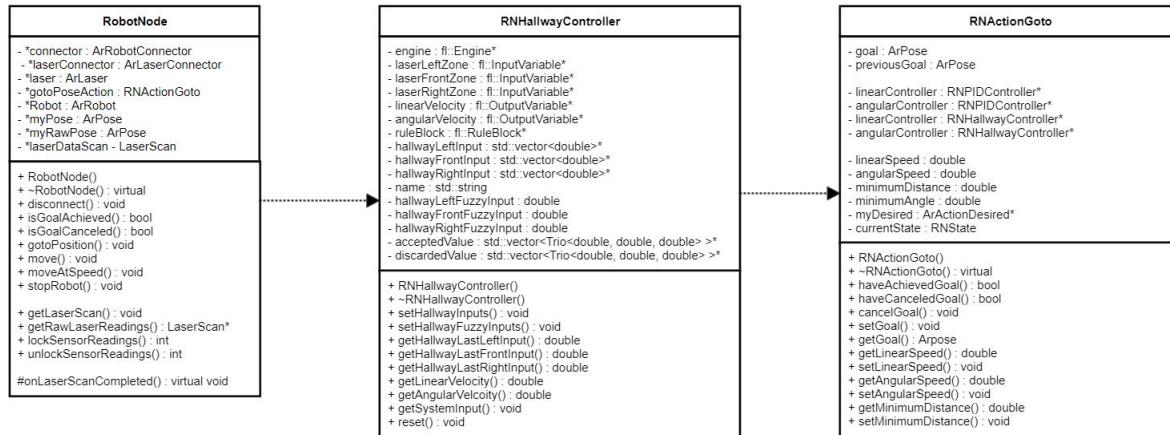
Devuelve un objeto de la clase `LaserScan` en el que se guardan todos los valores de las mediciones realizadas por el láser, tanto de rangos como de intensidades.

## 5.6. La clase RNHallwayController

Como todas las clases en C++, la clase `RNHallwayController` estará formada por atributos y métodos que se describen a continuación

### 5.6.1. Atributos de RNHallwayController

La clase `RNHallwayController` está formada únicamente por argumentos de tipo `private`, es decir, solo son accesibles desde la propia clase y no por otras clases que pudieran estar relacionadas con ella. Las variables propias de la clase son:



**Figura 5.9** Diagrama de clases del controlador RNHallwayController

- `fl::Engine* engine`
- `fl::InputVariable* laserLeftZone`
- `fl::InputVariable* laserFrontZone`
- `fl::InputVariable* laserRightZone`
- `fl::OutputVariable* linearVelocity`
- `fl::OutputVariable* angularVelocity`
- `fl::RuleBlock* ruleBlock`
  
- `std::string name`

- `std::vector<double>* hallwayLeftInput`
- `std::vector<double>* hallwayFrontInput`
- `std::vector<double>* hallwayRightInput`
  
- `std::double hallwayLeftFuzzyInput`
- `std::double hallwayFrontFuzzyInput`
- `std::double hallwayRightFuzzyInput`
  
- `std::vector<Trio<double, double, double> >* acceptedValue`
- `std::vector<Trio<double, double, double> >* discardedValue`

Dentro del campo de datos de la clase, por un lado se pueden encontrar variables con una cabecera de lógica difusa, estas son variables relacionadas con la librería dedicada a lógica borrosa, fuzzylite. Esta librería permite realizar todas las acciones necesarias para la borrosificación y la desborrosificación de las variables. Por tanto, nuestra clase necesita que los objetos tengan tres variables difusas de entrada, que serán `laserLeftZone`, `laserFrontZone` y `laserRightZone`, de la misma forma que se necesitan dos variables difusas de salida, `linearVelocity` y `angularVelocity`. Además se necesita una variable adicional en la que se almacena el bloque de reglas completo, bloque que se desarrollará en apartados posteriores. Por último, se utiliza una variable tipo `Engine` que almacena todas las condiciones del fuzzy para recibir una orden que ejecuta todo el sistema borroso.

En cuanto a los valores con la cabecera standard, por un lado se tienen 3 vectores `hallwayLeftInput`, `hallwayFrontInput` y `hallwayRightInput` en los que se guardan 120 de los 360 mediciones del láser, para poder calcular posteriormente las entradas al sistema fuzzy. Estas entradas al sistema fuzzy se guardan en las variables `hallwayLeftFuzzyInput`, `hallwayFrontFuzzyInput` y `hallwayRightFuzzyInput`. Por último, se crean dos vectores tridimensionales, `acceptedValue` y `discardedValue` en los que se guardan los valores aceptados o descartados por el filtro en cada una de las mediciones anteriores.

### 5.6.2. Métodos de RNHallwayController

En cuanto a los métodos, la clase está formada por las siguientes funciones:

`RNHallwayController(const char* name)`

Dentro del constructor se inicializan las variables necesarias para realizar la borrosificación y desborrosificación, creándose un sistema que se explicará en el siguiente apartado, se crean las etiquetas lingüísticas, las funciones de pertenencia asociadas a cada una en las que se calcularán los grados de pertenencia de las variables que entren y salgan del sistema fuzzy. Se inicializa en definitiva, todo el motor de inferencia de lógica difusa. En este caso las funciones de pertenencia serán triangulares como ya se indicó con anterioridad.

`~RNHallwayController()`

```
getSystemInput
void getSystemInput(const LaserScan* data, double* linearVelocity,
double angularVelocity*)
```

Se encarga de crear un objeto `LaserScan` y guardar las mediciones láser en 3 vectores distintos a través de la función `setHallwayInputs()`, en la cual se han borrado previamente los valores de la anterior medición.

`data` es un objeto que guarda los datos sobre intensidad y rango de las mediciones láser, `linearVelocity` y `angularVelocity` devuelven a la clase `RNACTIONGOTO` los valores de velocidad lineal y angular por referencia hasta la siguiente iteración del controlador.

```
setHallwayInputs()
void setHallwayInputs(const LaserScan* data)

setHallwayFuzzyInputs
```

Una vez que se obtienen las medias de las distancias a las paredes, la función `setHallwayFuzzyInputs()` se encarga de comprobar si ha existido un cambio en el parádigma del pasillo, y de establecer la entrada numérica definitiva al sistema fuzzy, cuyo funcionamiento se abordará en el apartado 5.7.

```
void setHallwayFuzzyInputs(double lastLeftInput, double lastFrontInput,
double lastRightInput)

getHallwayLastLeftInput
double getHallwayLastLeftInput(void)

getHallwayLastFrontInput
double getHallwayLastFrontInput(void)

getHallwayLastRightInput
double getHallwayLastRightInput(void)
```

Estos tres métodos son los encargados de calcular la media de las mediciones en cada uno de los tres sectores, como se indicó en la sección 5.3.

```
getLinearVelocity
double getLinearVelocity(void)

getAngularVelocity
double getAngularVelocity(void)
```

Estas dos funciones permiten conocer el valor obtenido de las velocidades lineales y angulares una vez que se han desborrosificado sus valores.

#### **reset**

```
void reset(void)
```

La función **reset** reinicia el controlador.

## 5.7. Estados del controlador

La idea es que el controlador sea robusto, pero flexible. Tiene que ser capaz de responder a cambios en la anchura del pasillo, pero no tiene que verse afectado por el tránsito de gente a su lado, o por la apertura de una puerta. Para conseguir esto, debe existir un filtro que identifique si las últimas mediciones son una excepción o existe un cambio real en el entorno.

El controlador borroso se activa, cuando la clase **RNACTIONGOTO** indica que se está dentro de un pasillo, desde ese momento hasta que se indique que se ha llegado al punto destino o que se ha abandonado el pasillo, la clase del controlador estará actuando de manera recursiva.

Una vez que la clase **RNACTIONGOTO** indica de la necesidad de actuación del controlador **RNHallwayController**, dicho controlador estará actuando de manera continua realizando las mismas funciones de forma recursiva.

La función con la que se activa la clase **RNHallwayController** es **getSystemInput**. Esta función es la encargada de gestionar todas las funciones auxiliares llamadas dentro del controlador para generar los datos de velocidad lineal y auxiliar necesarios.

En primer lugar, se necesita obtener las futuras entradas del sistema, con lo que es necesario crear un objeto de la clase **LaserScan**, clase heredada de **RobotNode**, en la cual se encuentran las funciones que permiten guardar los valores de la distancia entre otros, de los objetos del tipo **LaserScan**.

El sistema de medición láser de Doris abarca 180°, aportando 361 mediciones separadas 0,5° entre sí. Estas mediciones se obtienen a partir de la función **laserScanData** de la clase **LaserScan**. Se divide el conjunto de medidas en tres subconjuntos de 120°, considerando el ángulo 0 como el primer haz de luz a la izquierda del robot, para utilizarlos como entrada del sistema fuzzy. Como debe utilizarse un único valor para cada variable de entrada, se realiza una sectorización buscando obtener las variables **LaserLeftZone**, **LaserFrontZone** y **LaserRightZone**. Cada una de estas variables se calcula de la siguiente manera:

1. Se toman cinco sectores radiales que sirven como división de las cinco posibles etiquetas lingüísticas de cada función de pertenencia del sistema fuzzy. Estas etiquetas son **VN** que equivale a la zona “Very Near”, o “Muy Cerca”, **N** que equivale a la zona “Near”, o “Cerca”, **M** que equivale a la zona “Medium”, o “Media”, **F** a la

zona “Far”, o “Lejos”, y **VF** a la zona “Very Far”, o “muy lejos”. Estos valores pueden variarse en función del entorno en el que vaya a trabajar, por ejemplo, si Doris va a trabajar generalmente en pasillos muy estrechos, adaptar los valores a la anchura del pasillo permitiría ganar en maniobrabilidad.

2. A través de las funciones equivalentes `getHallwayLastLeftInput`, `getHallwayLastFrontInput` y `getHallwayLastRightInput` se asignan las mediciones a los distintos sectores radiales, sabiendo exactamente cuántas caen dentro de cada zona. Esto cumple dos funciones: por un lado, se dota de preferencia para la entrada del sistema fuzzy al sector más cercano y se descartan los demás datos para los cálculos, ya que empeorarían la precisión de la medición. Por otro lado, permite establecer un filtro en el que si no existe un número suficiente de puntos dentro del sector más cercano, se considera que esas mediciones son ruido y se pasa al siguiente sector, en este caso, aunque el número de mediciones necesarias puede variar, se consideran 10 mediciones mínimas necesarias en el sector, por tanto, por ejemplo, si en el sector **VN** solo hay 3 puntos, y en el sector **N** hay 20 puntos, se utilizará el sector N como última entrada. El valor obtenido de las funciones será la media de los puntos del primer sector válido, y dicho valor será la entrada al sistema fuzzy si es un valor aceptado.

Una vez que se ha obtenido los valores `lastLeftInput`, `lastFrontInput` y `lastRightInput`, la función `getSystemInput()` llama a la función `setHallwayFuzzyInputs()` en la que entrará en acción el filtro que indica si las condiciones del pasillo cambian realmente o se está ante un obstáculo lateral. Para ello en esta función se crean las variables enteras `doorCheckIterations`, que indica el número de iteraciones necesarias para que el cambio en las distancias no sea considerado una falsa señal, `doorCheckDistance` que marca el umbral por encima del cual se considera que existe un cambio importante en las distancias, y la variable booleana `isStable` que indica si el robot se encuentra estable en el centro del pasillo, o está ubicándose todavía.

Si la diferencia medida entre `lastLeftInput` y `lastRightInput` es inferior a `doorCheckDistance` o la variable `isStable` es falsa, los valores de `lastLeftInput`, `lastFrontInput` y `lastRightInput` se pasarán al vector de valores aceptados `acceptedValue`.

El primer paso a realizar por el filtro es comprobar si el sistema es estable, para lo cual, comprueba si la diferencia entre los últimos valores aceptados de `lastLeftInput` y `lastRightInput` es menor que `doorCheckDistance`. Si esto fuera falso, obviamente el robot no estaría estable, por lo que estos valores se pasarán al sistema borroso para corregir la posición. Si no existiera una diferencia significativa, las entradas al sistema fuzzy `hallwayLeftFuzzyInput`, `hallwayFrontFuzzyInput` y `hallwayRightFuzzyInput` también serían los tres elementos del vector `acceptedValue`

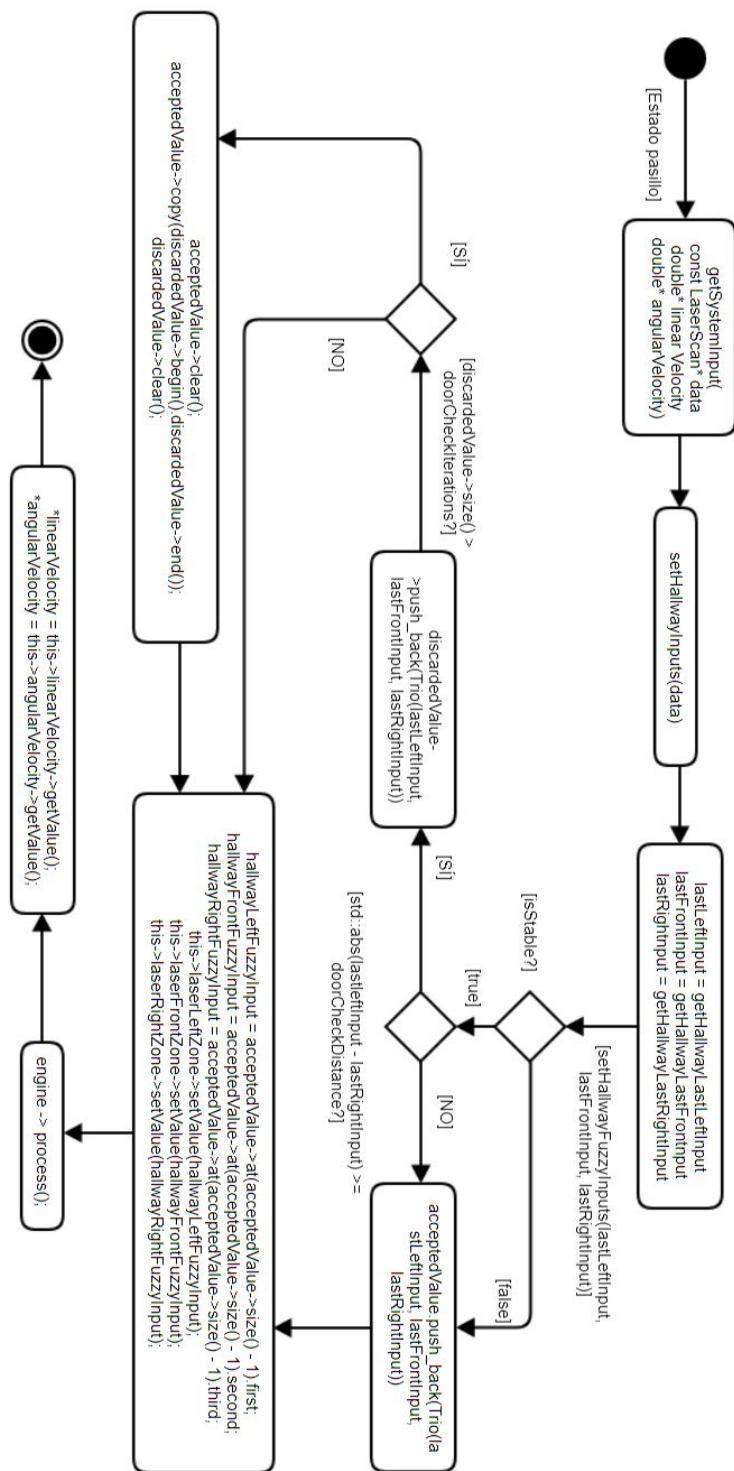
¿Pero qué ocurre si el sistema sí es estable, y aun así se mide una diferencia significativa? Entonces, estos valores pasan al vector `discardedValue` y mientras se realiza la comprobación, los valores de entrada al sistema fuzzy serán los de la última entrada aceptada, para mantener el rumbo constante. Si la diferencia se mantiene durante un número de iteraciones superior a `doorCheckIterations`, se eliminará el vector de valores

aceptados, y se copiará el de valores rechazados en el de valores aceptados, con lo que esta última entrada será la nueva primera entrada al sistema fuzzy.

Ahora que se tienen las tres entradas del sistema fuzzy, `hallwayLeftFuzzyInput`, `hallwayFrontFuzzyInput` y `hallwayRightFuzzyInput`, estos tres valores son sirven como entrada del sistema fuzzy, el cual se inicia a través de la función `engine->process()`. Para que este método funcione, es necesario declarar antes todos los elementos necesarios de la librería fuzzylite en el constructor de la clase.

- Se crea un objeto `Engine`.
  
  
  
  
  
  
- Se crean tres objetos `InputVariable`, es decir, tres objetos para las tres variables de entrada, que corresponden a las medidas de los tres sectores anteriormente explicados. Además de crearse los objetos y habilitarse a través de la función `setEnabled()`, debe indicarse su rango de validez a través de la función `setRange()`, y deben añadirse todos los términos de la función de pertenencia con la función `addTerm()` indicando el nombre de la etiqueta lingüística, y en qué puntos se encuentran los vértices de dicho término. Una vez que se han completado todas estas necesidades, se añade cada variable de entrada al engine gracias a la función `addInputVariable()`.
  
  
  
  
  
  
- De igual manera, se han creado previamente dos objetos `OutputVariable` para la velocidad lineal y la velocidad angular. Además de las funciones anteriores, habrá que indicar el método de agregación utilizado, en nuestro caso, suma algebraica, con la función `setAggregation()` y el método de desborrosificación, por la centroide, con `SetDefuzzifier()`.
  
  
  
  
  
  
- Para el bloque de reglas se crea un objeto de la clase `RuleBlock`, en el que hay que indicar cómo se realiza la conjunción, con `setConjunction()`, a través del producto algebraico, la disjunción, con `setDisjunction()`, en este caso por suma algebraica, y la implicación, con `setImplication()`, que ocurre a través de producto algebraico. Con la función `addRule()` se añaden strings en los que se encuentran todas las reglas necesarias, 125 en este proyecto. Por último, se cargan las reglas en el engine gracias a la función `addRuleBlock()`

En cuanto a los valores de salida, una vez que se obtienen las salidas del sistema fuzzy que gestiona el controlador borroso, desde la función `getSystemInput()`, `linearVelocity` y `angularVelocity` se devuelven a la clase `RNActionGoto` por referencia.



**Figura 5.10** Diagrama de estados del controlador RNHallwayController

## 5.8. Reglas fuzzy

A continuación se muestran las tablas que muestran el conjunto de reglas lógicas del controlador borroso. Se cuenta con un conjunto de 125 reglas del tipo “IF `laserLeftZone IS ... AND laserFrontZone IS ... AND laserRightZone IS ... THEN linearVelocity IS ... AND angularVelocity IS ...`. Estas 125 reglas están escritas en cuanto a los principios expuestos en la sección 5.3. Por dificultad de representar matrices tridimensionales, se decide mostrar las tablas con las etiquetas de `laserFrontZone` y `laserRightZone` en función de la etiqueta lingüística de `laserLeftZone`.

**Cuadro 5.1** Tabla de reglas de la velocidad lineal cuando `laserLeftZone = VF`

Laser Left Zone = VF		Laser Front Zone				
		VF	F	M	N	VN
Laser Right Zone	VF	Fast Forward	Medium Forward	Slow Forward	Zero	Backwards
	F	Medium Forward	Medium Forward	Slow Forward	Zero	Backwards
	M	Slow Forward	Slow Forward	Slow Forward	Zero	Backwards
	N	Zero	Zero	Zero	Zero	Backwards
	VN	Backwards	Backwards	Backwards	Backwards	Backwards

**Cuadro 5.2** Tabla de reglas de la velocidad lineal cuando `laserLeftZone = F`

Laser Left Zone = F		Laser Front Zone				
		VF	F	M	N	VN
Laser Right Zone	VF	Medium Forward	Medium Forward	Slow Forward	Zero	Backwards
	F	Medium Forward	Medium Forward	Slow Forward	Zero	Backwards
	M	Slow Forward	Slow Forward	Slow Forward	Zero	Backwards
	N	Zero	Zero	Zero	Zero	Backwards
	VN	Backwards	Backwards	Backwards	Backwards	Backwards

**Cuadro 5.3** Tabla de reglas de la velocidad lineal cuando `laserLeftZone = M`

Laser Left Zone = M		Laser Front Zone				
		VF	F	M	N	VN
Laser Right Zone	VF	Slow Forward	Slow Forward	Slow Forward	Zero	Backwards
	F	Slow Forward	Slow Forward	Slow Forward	Zero	Backwards
	M	Slow Forward	Slow Forward	Slow Forward	Zero	Backwards
	N	Zero	Zero	Zero	Zero	Backwards
	VN	Backwards	Backwards	Backwards	Backwards	Backwards

**Cuadro 5.4** Tabla de reglas de la velocidad lineal cuando laserLeftZone = N

Laser Left Zone = N		Laser Front Zone				
		VF	F	M	N	VN
Laser Right Zone	VF	Zero	Zero	Zero	Zero	Backwards
	F	Zero	Zero	Zero	Zero	Backwards
	M	Zero	Zero	Zero	Zero	Backwards
	N	Zero	Zero	Zero	Zero	Backwards
	VN	Backwards	Backwards	Backwards	Backwards	Backwards

**Cuadro 5.5** Tabla de reglas de la velocidad lineal cuando laserLeftZone = VN

Laser Left Zone = VN		Laser Front Zone				
		VF	F	M	N	VN
Laser Right Zone	VF	Backwards	Backwards	Backwards	Backwards	Backwards
	F	Backwards	Backwards	Backwards	Backwards	Backwards
	M	Backwards	Backwards	Backwards	Backwards	Backwards
	N	Backwards	Backwards	Backwards	Backwards	Backwards
	VN	Backwards	Backwards	Backwards	Backwards	Backwards

**Cuadro 5.6** Tabla de reglas de la velocidad angular cuando laserLeftZone = VF

Laser Left Zone = VF		Laser Front Zone				
		VF	F	M	N	VN
Laser Right Zone	VF	Zero	Zero	Slow Left	Slow Left	Fast Left
	F	Zero	Zero	Slow Left	Slow Left	Fast Left
	M	Zero	Zero	Slow Left	Slow Left	Fast Left
	N	Zero	Zero	Slow Left	Slow Left	Fast Left
	VN	Zero	Zero	Slow Left	Slow Left	Fast Left

**Cuadro 5.7** Tabla de reglas de la velocidad angular cuando laserLeftZone = F

Laser Left Zone = F		Laser Front Zone				
		VF	F	M	N	VN
Laser Right Zone	VF	Zero	Zero	Slow Left	Slow Left	Fast Left
	F	Zero	Zero	Slow Left	Slow Left	Fast Left
	M	Zero	Zero	Slow Left	Slow Left	Fast Left
	N	Zero	Zero	Slow Left	Slow Left	Fast Left
	VN	Zero	Zero	Slow Left	Slow Left	Fast Left

**Cuadro 5.8** Tabla de reglas de la velocidad angular cuando laserLeftZone = M

Laser Left Zone = M		Laser Front Zone				
		VF	F	M	N	VN
Laser Right Zone	VF	Slow Right	Slow Right	Zero	Slow Left	Fast Left
	F	Slow Right	Slow Right	Zero	Slow Left	Fast Left
	M	Slow Right	Slow Right	Zero	Slow Left	Fast Left
	N	Slow Right	Slow Right	Zero	Slow Left	Fast Left
	VN	Slow Right	Slow Right	Zero	Slow Left	Fast Left

**Cuadro 5.9** Tabla de reglas de la velocidad angular cuando laserLeftZone = N

Laser Left Zone = N		Laser Front Zone				
		VF	F	M	N	VN
Laser Right Zone	VF	Slow Right	Slow Right	Slow Right	Zero	Slow Left
	F	Slow Right	Slow Right	Slow Right	Zero	Slow Left
	M	Slow Right	Slow Right	Slow Right	Zero	Slow Left
	N	Slow Right	Slow Right	Slow Right	Zero	Slow Left
	VN	Slow Right	Slow Right	Slow Right	Zero	Slow Left

**Cuadro 5.10** Tabla de reglas de la velocidad angular cuando laserLeftZone = VN

Laser Left Zone = VN		Laser Front Zone				
		VF	F	M	N	VN
Laser Right Zone	VF	Fast Right	Fast Right	Fast Right	Zero	Zero
	F	Fast Right	Fast Right	Fast Right	Zero	Zero
	M	Fast Right	Fast Right	Fast Right	Zero	Zero
	N	Fast Right	Fast Right	Fast Right	Zero	Zero
	VN	Fast Right	Fast Right	Fast Right	Zero	Zero



---

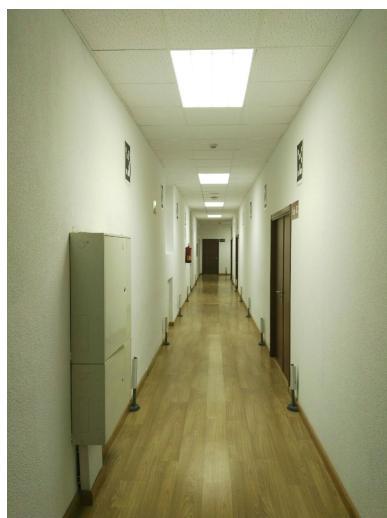
# Pruebas físicas y resultados

---

Una vez que se concluye el desarrollo de la clase `RNHallwayController` y se adaptan las clases encargadas del movimiento de Doris para utilizar el controlador en los casos en los que sea preciso, es necesario comprobar la viabilidad y la fiabilidad del trabajo hecho hasta el momento. En este proyecto se cuenta ya con el robot Doris construido y con todas las funcionalidades necesarias añadidas previamente al desarrollo del trabajo, por tanto, se considera más ventajoso trabajar directamente en el espacio y no realizar simulaciones, ya que supondría un gasto temporal innecesario el diseñar el entorno de pruebas y realizarlas.

## 6.1. Entorno de pruebas

El pasillo de la figura 6.1 tiene una medida de anchura de 175 centímetros y una medida longitudinal superior a 10 metros, lo que permite desarrollar pruebas largas en las que realizar varias comprobaciones cada vez que se lanza el programa. El pasillo seleccionado tiene otras ventajas como la posibilidad de paso de una o dos personas a los lados del robot si se encuentra centrado, la presencia de obstáculos como son las puertas, o radiadores de calefacción. En ciertas pruebas se plantea la posibilidad de añadir obstáculos al pasillo para ver la reacción.



**Figura 6.1** Pasillo en el que se realizan las distintas pruebas.

## 6.2. Ajustes previos

En primera instancia, se realizarán ajustes sobre el láser del robot y las funciones de pertenencia de entrada, para lo cual es posible realizar pruebas en estático.

### 6.2.1. Ajuste del láser

Las primeras pruebas realizadas consisten en realizar comprobaciones sobre la precisión de las medidas láser. Para aumentar la seguridad sobre las mediciones se idean 3 pruebas distintas. Estas pruebas se realizan manteniendo a Doris estática en un punto cualquiera del pasillo, con el láser activado. Las pruebas realizadas son:

- La primera comprobación obviamente es obtener los datos de las 361 mediciones por pantalla. Una vez que se mida con un metro a qué distancia está perpendicularmente Doris de las paredes, se hace un seguimiento de los valores esperando no encontrar un valor altamente discrepante con el resto. También se puede medir la precisión exacta de los ángulos  $0^\circ$  y  $180^\circ$ .
- El segundo test a realizar es comprobar el alcance de los ángulos centrales, ya que estos pueden no alcanzar una posible pared frontal. Esto permitirá acometer el futuro ajuste del controlador con mayor comodidad. Sin embargo, la potencia del láser, que alcanza 32m de distancia es más que suficiente para el estudio del proyecto.
- La última comprobación consiste en lanzar una medición láser, y comprobar si efectivamente, se está calculando de forma correcta el valor definitivo de la distancia en cada sector. Por tanto habrá que comprobar el número de puntos que caen dentro de un sector radial, y si se están desechando los valores fuera del sector radial más cercano, además de si el cálculo de la media es correcto.



**Figura 6.2** Doris en el pasillo realizando las primeras mediciones.

### 6.2.2. Ajustes de las funciones de pertenencia de entrada

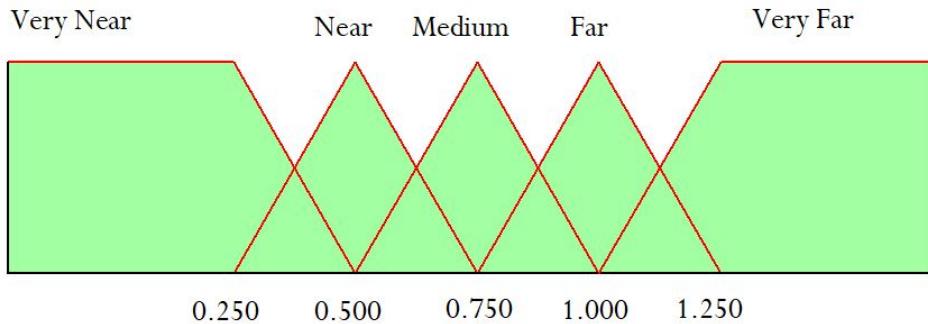
Una vez que se ha demostrado que las mediciones de Doris son fiables, es momento de realizar un primer ajuste de las funciones de pertenencia de las variables de entrada. Obviamente los valores de las 3 funciones de pertenencia deben ser iguales, para que no exista ningún desequilibrio. Los valores de las funciones de pertenencia deben ser suficientemente grandes como para que pequeñas variaciones en las distancias no provoquen grandes alteraciones en las velocidades de salida, lo que significaría la presencia de tirones y movimientos bruscos, así como probablemente cabeceo en el avance del robot. Por otro lado, los valores tienen que estar ajustados para no ser excesivamente grandes y provocar que se esté siempre en valores mínimos de la función de pertenencia, y la respuesta del sistema sea demasiado lenta.

Al ser la anchura del pasillo de 175 centímetros, se decide que el centro de las funciones de pertenencia serán 75 centímetros, de tal forma que en el caso de Doris está centrada, estará moviéndose en el centro de la función de pertenencia. Por otra parte el centro del triángulo **Very Far**, está a 125 centímetros, con lo que incluso en el caso de que Doris estuviera pegada a una pared, la medida del lado alejado, estaría a una distancia razonable para estar dentro de la función de pertenencia. Obviamente, en caso de operar en distintos pasillos, el operario debería ajustar estos datos de manera más laxa para ganar en flexibilidad y poder operar en varias configuraciones.

Para evitar problemas de cálculo al realizar cálculos al borrosificar y desborrosificar, se utilizan triángulos equiláteros como términos de la función de pertenencia. A continuación se muestra en la tabla 6.1 las medidas exactas de dichos triángulos, y en la figura 6.3 se muestra un ejemplo de las 3 funciones de pertenencia **LaserLeftZone**, **LaserFrontZone** y **LaserRightZone**.

	Vértice Izquierdo	Vértice Superior	Vértice Derecho
Very Near	$-\infty$	0.250	0.500
Near	0.250	0.500	0.750
Medium	0.500	0.750	1.000
Far	0.750	1.000	1.250
Very Far	1.000	1.250	$\infty$

**Cuadro 6.1** Medidas de los términos triangulares de las funciones de pertenencia de las variables de entrada (m)



**Figura 6.3** Función de pertenencia de las 3 variables de entrada.

## 6.3. Pruebas dinámicas

### 6.3.1. Ajuste de las funciones de pertenencia de salida

Las primeras pruebas en movimiento, sirven tanto para el ajuste de las velocidades lineales y angulares, como para la comprobación de la navegación del pasillo en su sentido más simple, por lo que el filtro de obstáculos temporales se desactivará.

Se indicará a Doris que debe avanzar una distancia de unos 5 metros en los que la configuración del pasillo se mantiene constante. Por un lado se comprobará que la velocidad no sea excesivamente lenta ni demasiado rápida como para perder el control, y que los giros den una respuesta rápida pero no brusca. Además se comprobará que el robot responde al tránsito de personas por cualquiera de sus dos lados.

Considerando el eje  $y$  como el eje perpendicular a las paredes y la distancia media como el origen, y el eje  $x$  como el eje paralelo considerando el origen como el inicio del pasillo, en la primera prueba se envía a Doris desde el punto (10.300, 0.000) al punto (16.000, 0.000), comprobando que si bien la velocidad angular es válida, se obtienen velocidades lineales cercanas a los  $30mm/s$ , considerada una velocidad baja, por lo que se cambia la función de pertenencia hasta llegar a la de la figura 6.4, de tal forma que la velocidad ronda los  $55m/s$ .

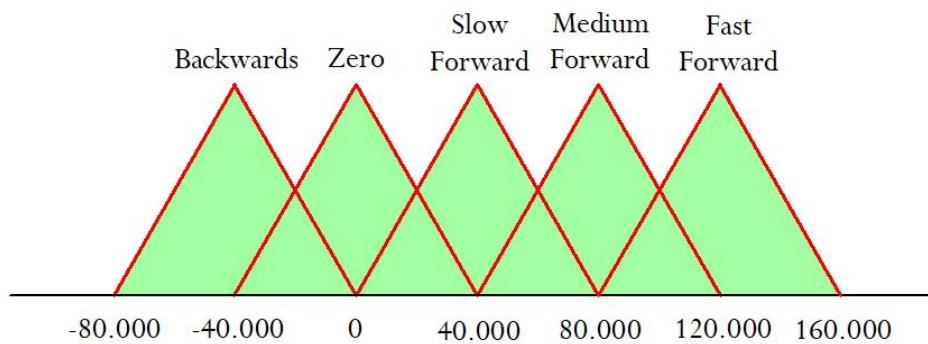
Las siguientes pruebas se realizan con Doris partiendo desde puntos descentrados del eje  $y$  para comprobar mejor cuál es su respuesta, y comprobar que la estabilización desde posiciones desfavorables es correcta.

Por otro lado, se comprueba que efectivamente, ante el tránsito de personas, Doris es capaz de estar recibiendo cambios bruscos en sus entradas al sistema fuzzy, por lo que las reglas elegidas parecen válidas.

A continuación se muestran las funciones de pertenencia de las variables de salida:

	Vértice Izquierdo	Vértice Superior	Vértice Derecho
Backwards	-80.000	-40.000	0
Zero	-40.000	0	40.000
Slow Forward	0	40.000	80.000
Medium Forward	40.000	80.000	120.000
Fast Forward	80.000	120.000	160.000

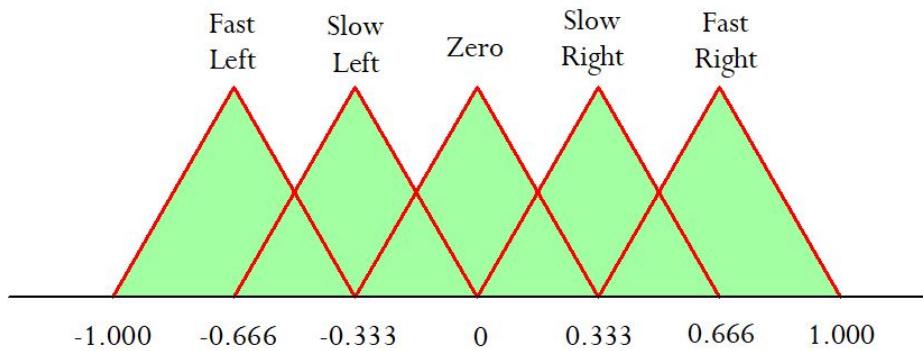
**Cuadro 6.2** Medidas de los términos triangulares de la función de pertenencia de la velocidad lineal (mm/s)



**Figura 6.4** Función de pertenencia de la velocidad lineal.

	Vértice Izquierdo	Vértice Superior	Vértice Derecho
Fast Left	-1.000	-0.666	-0.333
Slow Left	-0.666	-0.333	40.000
Zero	-0.333	0	0.333
Slow Right	40.000	0.333	0.666
Fast Right	0.333	0.666	1.000

**Cuadro 6.3** Medidas de los términos triangulares de la función de pertenencia de la velocidad angular (rad/s)



**Figura 6.5** Función de pertenencia de la velocidad angular.

### 6.3.2. Comprobación del funcionamiento del filtrado de datos

Con las funciones de pertenencias ya ajustadas, el último paso del proyecto es comprobar las magnitudes que marcan el número de iteraciones necesarias para considerar que existe cambio en el pasillo, o las que indican que el robot está estabilizado, y la distancia que provoca una alteración en la estabilización.

Para obtener información sobre dichos datos, se realizarán experimentos sucesivos en los que Doris atraviesa una puerta abierta, un hueco de 6cm. o una persona cruza a su lado a distinta velocidad. Ambos datos se ajustarán simultáneamente.

Si el número de iteraciones es demasiado bajo, el filtro se desactivará demasiado rápido, por lo que habrá una desestabilización, que se descentrará y variará su rumbo para buscar la zona con más espacio libre. Por otra parte, si el valor es demasiado alto, durante todo ese tiempo el robot no esquivará obstáculos que vengan de forma lateral. Además, si existe un pequeño desvío en el último valor aceptado y se mantiene ese rumbo durante demasiado tiempo, el robot podría estar descentrándose a pesar del filtro.

Sobre la distancia necesaria para la activación del filtro, valores muy grandes provocarían que el robot se desplazara del centro. Si por ejemplo la diferencia medida entre los sectores izquierdo y derecho fuera de cinco centímetros, pero el umbral para activar el filtro fuera de diez, el robot se desplazaría 2,5 centímetros hacia el lado más lejano, independientemente de si esa variación se mantiene o no. Pero si el valor es muy bajo, se corre el peligro de estar filtrando durante demasiada parte del trayecto para desviaciones de la línea media que no serían significativas.

#### Valor de estabilización

Al no producirse cabeceo, ni existir oscilaciones en la trayectoria, se considera que un valor del orden de las decenas es suficiente para que el robot se suponga como estable. Se decide que si la diferencia entre los últimos valores aceptados de los sectores laterales está por debajo del umbral de filtrado durante las últimas 20 iteraciones del controlador, el sistema es estable y está por tanto en una posición muy centrada.

Cabe decir que el uso de valores excesivamente altos, provoca con más facilidad un fallo en la navegación, ya que es más probable que se pueda colar una medida falsa. También es notable que el cambio en el umbral de distancia no parece afectar a este valor de estabilización.

### Valor umbral de diferencia de distancia

Ya que en el espacio en el que se encuentra alojado el radiador, su cara externa se encuentra a  $6\text{cm}$ . más de profundidad que la pared, se decide un primer valor de  $5\text{cm}$ . centímetros para considerar la diferencia como significativa. Sin embargo, este valor se encuentra al límite para el caso de estudio, y produce en ocasiones una diferencia no significativa, cortando el funcionamiento del filtro. Si Doris trabajara en entornos con diferencias más grandes, es posible que este fuera un valor perfectamente válido.

Por tanto se disminuye el valor hasta  $3\text{cm}.$ , consiguiendo evitar esos falsos positivos dentro del filtro. No se trata de disminuir más el valor ya que ante los casos de estudio anteriores, es probable que el filtro estuviera actuando demasiadas veces, algo que obviamente no interesa.

### Valor de iteraciones del filtro

En el entorno de pruebas, se mide un ancho de puerta de  $75\text{cm}$ . y un ancho en el hueco del radiador de  $94\text{cm}.$ . Se decide realizar pruebas con tope máximo, es decir, se utilizan valores bajos que se van aumentando hasta conseguir el propósito del filtro. Estos valores no serán exactos porque dependen de en qué punto empieza a leer medidas discrepantes el robot, por lo que se tomará un valor aproximado, buscando también la adaptabilidad a otros espacios.

El primer valor utilizado es un valor de 100, este valor es suficiente para que una persona pueda cruzar junto a Doris sin provocar un cambio de rumbo, ya que el paso de una persona supone unos 20 valores descartados. Sin embargo este valor de 100 es demasiado bajo para poder filtrar completamente los valores discrepantes, ya que el filtro deja de actuar aproximadamente cuando se ha avanzado un tercio de la distancia.

Por esta razón se realizan pruebas sucesivas aumentando esta distancia hasta concluir que 400 repeticiones son suficientes para mantener el rumbo constante sin verse alterado. Si se superan esas repeticiones y la diferencia se mantiene, se considerará que el entorno ha cambiado, y el robot volverá a centrarse.



# Conclusiones y líneas de investigación futuras

En este capítulo se procede a realizar algunas conclusiones acerca de la fiabilidad y la precisión del controlador propuesto, así como líneas de investigación futura.

## 7.1. Conclusiones

El proyecto desarrollado demuestra ser exitoso en muchos de los aspectos buscados. El diseño de un controlador fuzzy con múltiples entradas y múltiples salidas permite un control robusto y rápido del robot.

El movimiento del sistema es quizá algo lento, pero a cambio no existe cabeceo durante el movimiento, esto es posible gracias a la buena precisión del láser, los ajustes realizados en las funciones de pertenencia tanto de las variables de entrada como de las de salida, así como el uso de 5 etiquetas lingüísticas en cada una de las funciones de pertenencia, lo que permite gestionar una alta cantidad de posibilidades, de hecho el sistema consta de 125 reglas distintas.

Sin embargo, sí que existen varias áreas a mejorar. El valor de la diferencia de distancias entre los sectores izquierdo y derecho que provoca cambios en el filtro, obviamente no consigue actuar en cambios en el pasillo por debajo de ese umbral, por lo que en cambios de un par de centímetros, sí que se aprecia un desplazamiento de igual magnitud en el robot.

El otro valor que puede ser conflictivo, es el del número de iteraciones del controlador necesarias para considerar que el pasillo ha cambiado, ya que una vez ajustado, ante dos obstáculos sucesivos, las iteraciones no serían suficientes y se consideraría que el pasillo ha cambiado, provocando, esta vez sí, un desplazamiento del centro más pronunciado. Sin embargo, aumentar en exceso este controlador, provocaría que durante un lapso de tiempo demasiado grande, no se tuviera respuesta del controlador.

Se concluye que si existe un operario con conocimientos suficientes, tras el ajuste de todos los valores, el controlador es perfectamente funcional, y muy exportable.

## 7.2. Líneas de investigación futura

Este proyecto supone una primera aproximación a la navegación con lógica fuzzy dentro del proyecto general de Doris. Se cree que el proyecto es perfectamente adaptable a otras funciones de navegación. Aplicando un diseño muy similar al descrito en el proyecto, es posible crear un controlador dedicado a la navegación en espacios abiertos con obstáculos dinámicos o no mapeados previamente.

Se crean dos líneas de investigación pues, una basada en mantener una basada de manera más estricta en el controlador `RNHallwayController`. Realizando algunos pequeños cambios, y cambiando los bloques de reglas, podría decidirse crear un controlador de navegación en el que en caso de encontrar un obstáculo que no formara parte del mapa, el robot se debería mantener paralelo a una distancia dada del obstáculo en todo momento, intentando bordearlo para poder seguir con la ruta original.

Por otra parte, podría crearse un controlador en el que se trabajara con la creación y destrucción de puntos “fantasma, de tal forma que ante un obstáculo, el robot crearía una submeta en un punto seguro, con la idea de ir a ese punto, y luego volver a la ruta original a partir de otra serie de puntos no creados anteriormente. Obviamente, aunque este navegador también estaría basado en la lógica difusa, se necesitaría el desarrollo de un algoritmo para la creación y destrucción de puntos.

Por último, para decidir qué controlador se está utilizando en cada momento, se necesita la creación de una clase supervisora, encargada de decidir qué condiciones se están produciendo en cada momento para saber qué controlador tiene qué actuar, dónde, y en qué condiciones. Aplicando al proyecto actual, debería decidir a partir de qué punto o zona, se considera que ha entrado en el pasillo, y si es necesaria una posición angular específica o no.

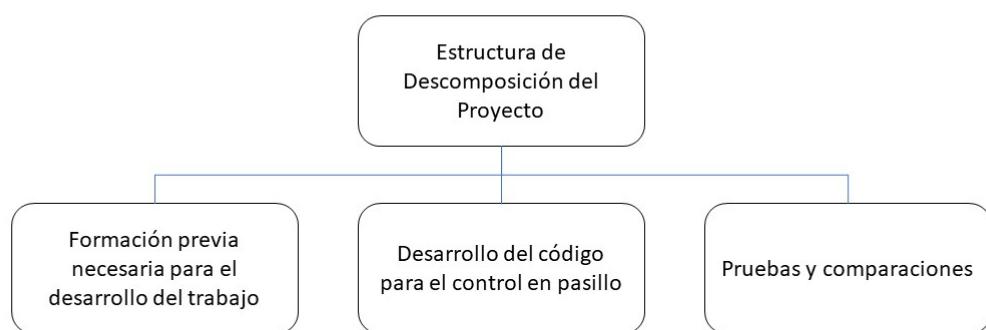
## capítulo 8

---

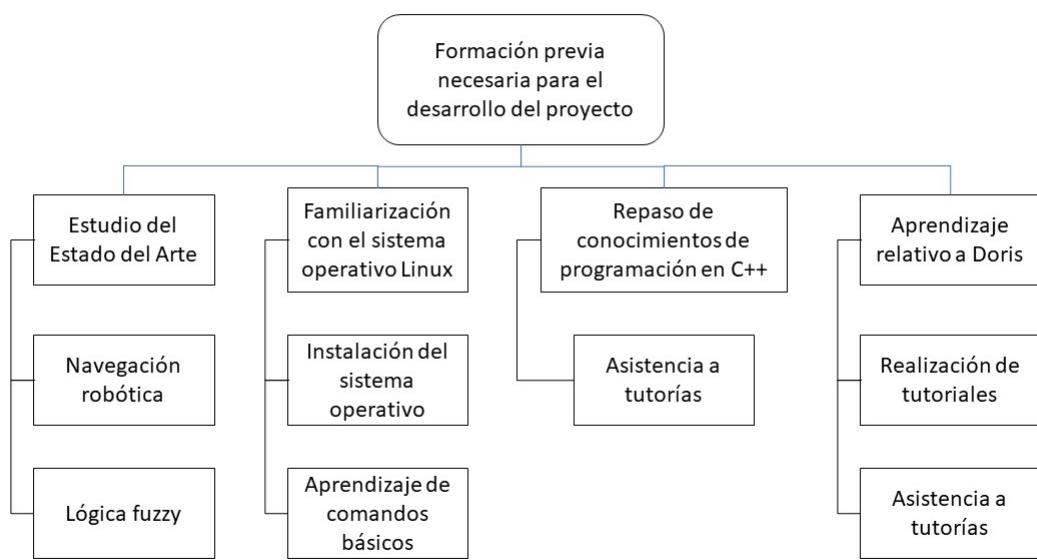
# Anexos

---

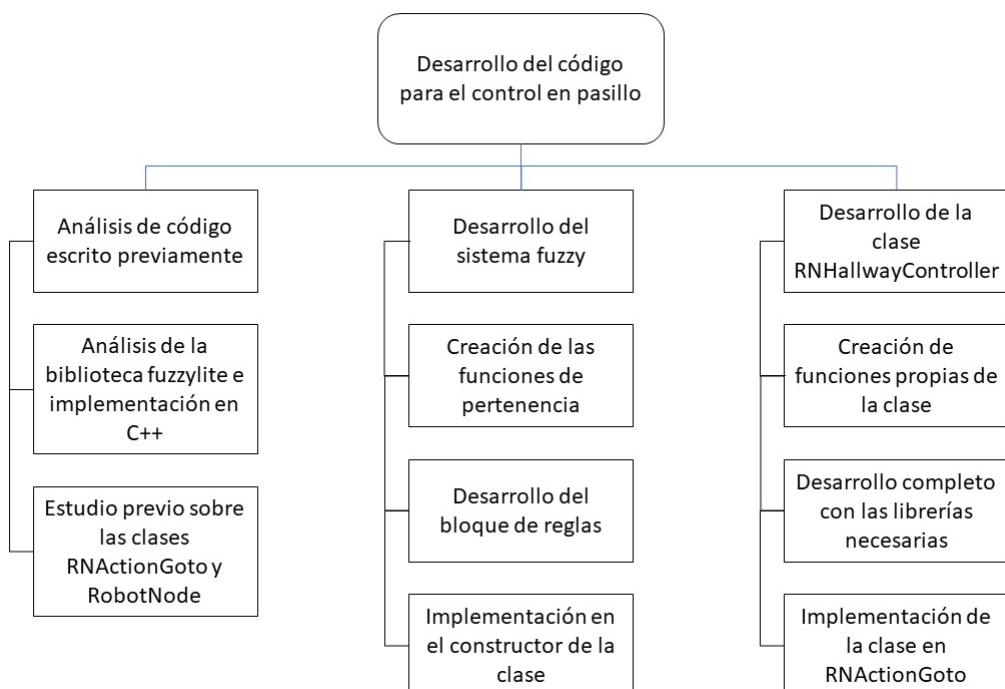
### 8.1. Anexo I: Estructura de Descomposición del Proyecto



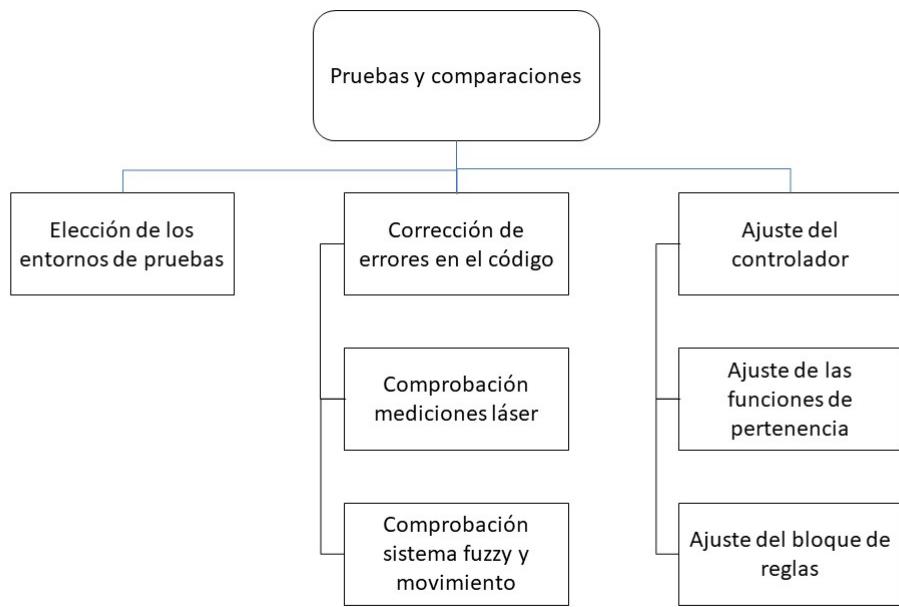
**Figura 8.1** Estructuración de descomposición del proyecto



**Figura 8.2** Formación previa necesaria para el desarrollo del proyecto

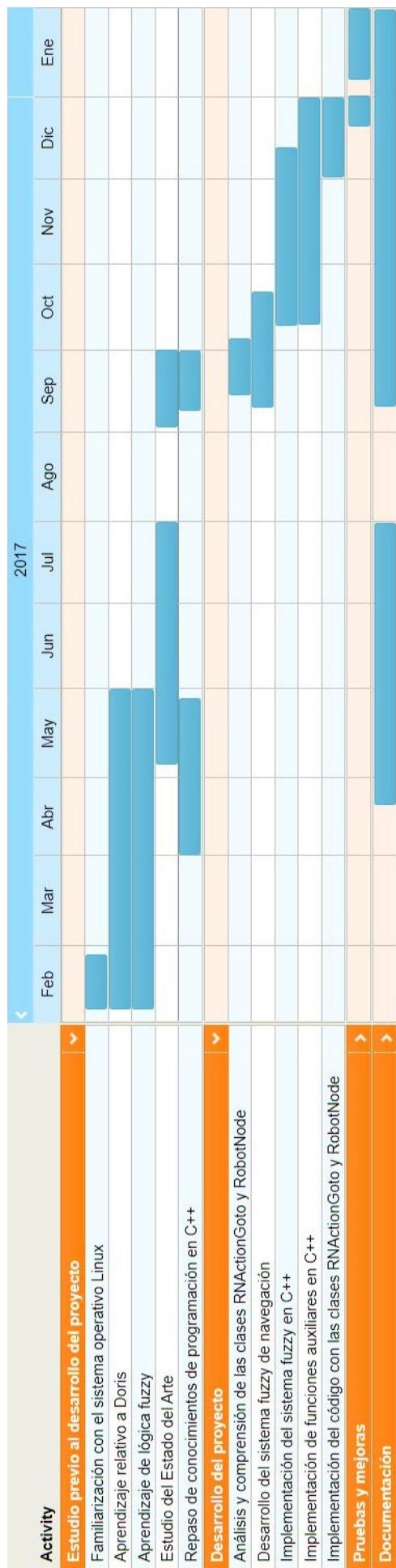


**Figura 8.3** Desarrollo del código para el control en pasillo

**Figura 8.4** Pruebas y comparaciones

## 8.2. Anexo II: Diagrama de Gantt

En la figura 8.5 se expone el diagrama de Gantt de este proyecto. Es notable el hecho de que el periodo de formación previa se extiende a lo largo de más tiempo que el desarrollo de cada algoritmo, aunque el tiempo dedicado a la formación sea menor. Entre los meses de Febrero a Julio se han realizado tareas relacionadas con el trabajo una o dos veces por semana. En cambio, en el periodo de Septiembre a Febrero la dedicación aumenta notablemente a varias jornadas durante la semana para el desarrollo del código y las pruebas, y trabajo en fines de semana relacionado con la documentación y la escritura del trabajo.

**Figura 8.5** Diagrama de Gantt del proyecto

### 8.3. Anexo III: Presupuesto del proyecto

En este anexo se procede a realizar el desglose del presupuesto del proyecto.

Al utilizar durante el desarrollo del trabajo programas software con licencia gratuita, o con licencia para estudiantes, no se incluye el precio de ninguno de estos programas.

En cuanto a los componentes hardware, se ha incluido el precio unitario, y se ha estimado un porcentaje a cada uno en función del tiempo que ha sido dedicado al proyecto.

Por último, los gastos de personal vienen dados por las horas de trabajo estimadas y el salario por hora aproximado.

Concepto	Coste unitario(€)	Uds.	% Dedicado al proyecto	Total(€)
PC portátil principal	800	1	15	120
PC sobremesa principal	2.000	1	10	200
Robot Doris	40.000	1	5	6.000

**Cuadro 8.1** Gastos de material

Concepto	€/h	Horas dedicadas	Total (€)
Autor	7	430	2.940
Tutor	35	70	2.450
Director	50	5	250

**Cuadro 8.2** Gastos de personal

Gastos en personal(€)	5.640
Gastos de material(€)	6.320
<b>TOTAL (€)</b>	<b>11.960</b>

**Cuadro 8.3** Gastos totales



---

# Índice de cuadros

---

4.1. Ejemplo de matriz de reglas . . . . .	35
5.1. Tabla de reglas de la velocidad lineal cuando laserLeftZone = VF . . . . .	63
5.2. Tabla de reglas de la velocidad lineal cuando laserLeftZone = F . . . . .	63
5.3. Tabla de reglas de la velocidad lineal cuando laserLeftZone = M . . . . .	63
5.4. Tabla de reglas de la velocidad lineal cuando laserLeftZone = N . . . . .	64
5.5. Tabla de reglas de la velocidad lineal cuando laserLeftZone = VN . . . . .	64
5.6. Tabla de reglas de la velocidad angular cuando laserLeftZone = VF . . . . .	64
5.7. Tabla de reglas de la velocidad angular cuando laserLeftZone = F . . . . .	64
5.8. Tabla de reglas de la velocidad angular cuando laserLeftZone = M . . . . .	65
5.9. Tabla de reglas de la velocidad angular cuando laserLeftZone = N . . . . .	65
5.10. Tabla de reglas de la velocidad angular cuando laserLeftZone = VN . . . . .	65
6.1. Medidas de los términos triangulares de las funciones de pertenencia de las variables de entrada (m) . . . . .	69
6.2. Medidas de los términos triangulares de la función de pertenencia de la velocidad lineal (mm/s) . . . . .	71
6.3. Medidas de los términos triangulares de la función de pertenencia de la velocidad angular (rad/s) . . . . .	71
8.1. Gastos de material . . . . .	81
8.2. Gastos de personal . . . . .	81
8.3. Gastos totales . . . . .	81



---

# Índice de figuras

---

2.1. Esquema de control básico . . . . .	6
2.2. Esquema de control completo . . . . .	6
2.3. Arquitectura de control . . . . .	7
2.4. RoboX en la Switzerland Expo 2002 . . . . .	11
2.5. Robot Minerva . . . . .	11
2.6. La serie de robots Mobot . . . . .	12
2.7. Rhino realizando demostraciones. Detalle de Rhino. . . . .	13
2.8. Minerva - Cara de Minerva - Minerva realizando demostraciones . . . . .	13
2.9. RoboX realizando desmotraciones - Detalle de RoboX . . . . .	14
2.10. Robot Jinny . . . . .	14
2.11. Alpha en funcionamiento . . . . .	15
2.12. Robotinho . . . . .	15
2.13. Robovie . . . . .	15
2.14. Asimo - Nao - Pepper - REEM - Maggie . . . . .	16
2.15. Blacky . . . . .	16
2.16. URBANO . . . . .	18
3.1. Hardware de Doris . . . . .	20
3.2. Arquitectura de software de Doris . . . . .	20
3.3. Cámara omnidireccional C25 de Mobotix . . . . .	22
3.4. Punto clave utilizado . . . . .	22
3.5. Imágenes obtenidas en el proceso de tratamiento . . . . .	23
3.6. Marcadores reflectantes diseñados por el grupo . . . . .	23
3.7. Representación de las medidas obtenidas por el láser. . . . .	24
4.1. Ejemplo de conjuntos en lógica difusa . . . . .	28
4.2. Conjuntos complementarios y disjuntos en lógica difusa . . . . .	29
4.3. Tipos de lógica . . . . .	31
4.4. Controlador fuzzy . . . . .	34
4.5. Cálculo del grado de pertenencia de una variable . . . . .	35
4.6. Cálculo del centro de gravedad en la desborrosificación . . . . .	37
4.7. Modelo cinemático del robot de Li . . . . .	40
4.8. Esquema de control de un controlador fuzzy de navegación.[42] . . . . .	41
5.1. Diagrama de casos de uso del controlador RNHallwayController . . . . .	45
5.2. Esquema del controlador RNHallwayController . . . . .	46

5.3.	Esquema de la sectorización de las mediciones realizadas con el láser . . . . .	46
5.4.	Esquema de navegación de pasillo de Doris . . . . .	49
5.5.	Función de pertenencia tipo de las variables de entrada . . . . .	50
5.6.	Función de pertenencia tipo de la velocidad lineal . . . . .	50
5.7.	Función de pertenencia tipo de la velocidad angular . . . . .	50
5.8.	Diagrama de actividades . . . . .	51
5.9.	Diagrama de clases del controlador RNHallwayController . . . . .	56
5.10.	Diagrama de estados del controlador RNHallwayController . . . . .	62
6.1.	Pasillo en el que se realizan las distintas pruebas. . . . .	67
6.2.	Doris en el pasillo realizando las primeras mediciones. . . . .	68
6.3.	Función de pertenencia de las 3 variables de entrada. . . . .	70
6.4.	Función de pertenencia de la velocidad lineal. . . . .	71
6.5.	Función de pertenencia de la velocidad angular. . . . .	72
8.1.	Estructuración de descomposición del proyecto . . . . .	77
8.2.	Formación previa necesaria para el desarrollo del proyecto . . . . .	78
8.3.	Desarrollo del código para el control en pasillo . . . . .	78
8.4.	Pruebas y comparaciones . . . . .	79
8.5.	Diagrama de Gantt del proyecto . . . . .	80

---

## Bibliografía

---

- [1] M. Bennewitz, F. Faber, D. Joho, M. Schreiber, S. Behnke, (2005, December), Towards a humanoid museum guide robot that interacts with multiple persons, *In Humanoid Robots, 2005 5th IEEE-RAS International Conference on* (pp. 418-423). IEEE.
- [2] C. Breazeal, (2003), Toward sociable robots. *Robotics and autonomous systems*, 42(3), 167-175.
- [3] C. L. Breazeal, (2004), Designing sociable robots, MIT press
- [4] W. Burgard, et al., (1999), The museum tour-guide robot RHINO, *Autonome Mobile Systeme 1998*, 245-254.
- [5] W. Burgard, D. Fox, G. Lakemeyer, D. Haehnel, D. Schulz, W. Steiner, A. Cremers, (1998), Real robots for the real world—the rhino museum tour-guide project, *In Proc. of the AAAI 1998 Spring Symposium on Integrating Robotics Research, Taking the Next Leap, Stanford, CA.*
- [6] G. Capi, S. Kaneko, B. Hua (2015), Neural Network based Guide Robot Navigation: An Evolutionary Approach, *Procedia Computer Science*, 76, 74-79.
- [7] K. Dautenhahn, A. Billard, (1999, April), Bringing up robots or—the psychology of socially intelligent robots: From theory to implementation, *In Proceedings of the third annual conference on Autonomous Agents* (pp. 366-367), ACM.
- [8] K. Dautenhahn, B. Ogden, T. Quick, (2002), From embodied to socially embedded agents—implications for interaction-aware robots, *Cognitive Systems Research*, 3(3), 397-428.
- [9] K. Dautenhahn, (1995), Getting to know each other—artificial social intelligence for autonomous robots, *Robotics and autonomous systems*, 16(2), 333-356.
- [10] K. Dautenhahn, (1998), The art of designing socially intelligent agents: Science, fiction, and the human in the loop, *Applied artificial intelligence*, 12(7-8), 573-617.
- [11] K. Dautenhahn, (1997), I could be you: The phenomenological dimension of social understanding, *Cybernetics Systems*, 28(5), 417-453.

- [12] K. Dautenhahn and C. Nehaniv, Living with socially intelligent agents: a cognitive technology view, in: K. Dautenhahn, ed., *Human Cognition and Social Agent Technology*, John Benjamins Publishing Company, 2000.
- [13] K. Dautenhahn, (2003). Roles and functions of robots in human society: implications from research in autism therapy, *Robotica*, 21(04), 443-452.
- [14] K. Dautenhahn, (2002), Design spaces and niche spaces of believable social robots, *In Robot and Human Interactive Communication, 2002. Proceedings. 11th IEEE International Workshop on* (pp. 192-197), IEEE.
- [15] D. Dubois , H. Prade, (1980), *Fuzzy Sets and Systems: Theory and Applications*, Academic Press.
- [16] D. Dubois, H. Prade, (1985), Evidence Measures Based on Fuzzy Information, *Automática* 21.
- [17] B. R. Duffy, (2003), Anthropomorphism and the social robot. *Robotics and autonomous systems*, 42(3), 177-190.
- [18] F. Faber, M. Bennewitz, C. Eppner, A. Görög, C. Gonsior, D. Joho, S. Behnke, (2009, September), The humanoid museum tour guide Robotinho, *In Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on* (pp. 891-896). IEEE.
- [19] M. Faisal, R. Hedjar, M. Al Sulaiman, K. Al-Mutib, (2013), Fuzzy Logic Navigation and Obstacle Avoidance by a Mobile Robot in an Unknown Dynamic Environment.
- [20] T. Fong, I. Nourbakhsh, K. Dautenhahn,(2003), A survey of socially interactive robots, *Robotics and autonomous systems*, 42(3), 143-166.
- [21] P. Gaudiano, C. Chang, (1997), Adaptive obstacle avoidance with a neural network for operant conditioning: experiments with real robots, *In the Proceedings of CIRA'97*
- [22] M. A. Goodrich, A. C. Schultz, (2007), Human-robot interaction: a survey, *Foundations and trends in human-computer interaction*, 1(3), 203-275.
- [23] F. Hegel, M. Lohse, A. Swadzba, S. Wachsmuth, K. Rohlfing, B. Wrede, (2007, August), Classes of applications for social robots: A user study, *In Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on* (pp. 938-943). IEEE.
- [24] F. Iida, M. Tabata, F. Hara, (1998), Generating personality character in a Face Robot through interaction with human, *In 7th IEEE International Workshop on Robot and Human Communication* (pp. 481-486).
- [25] F. Iida, H. Ayai, F. Hara, (1999), Behavior learning of a face robot using human natural instruction. *In Robot and Human Interaction, 1999. RO-MAN'99. 8th IEEE International Workshop on* (pp. 171-176). IEEE.
- [26] H. Ishiguro, T. Ono, M. Imai, T. Maeda, T. Kanda, R. Nakatsu, (2001), Robovie: an interactive humanoid robot, *Industrial robot: An international journal*, 28(6), 498-504.

- [27] A. Jiménez, F. Matía, (1994), Control Fuzzy: Estado Actual y Aplicaciones, *Cuadernos Profesionales AADECA*, 2(6).
- [28] R. Kalman, (1960), A new approach to linear filtering and prediction problems, *Transactions of the ASME-Journal of Basic Engineering*, 82, 35-45.
- [29] A. Kandel, (1986), Fuzzy Mathematical Techniques with Applications, Addison-Wesley Publishing Company.
- [30] G. Kim, W. Chung, K. R. Kim, M. Kim, S. Han, R. H. Shinn, (2004, September), The autonomous tour-guide robot Jinny, In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on* (Vol. 4, pp. 3450-3455). IEEE.
- [31] C. J. Klir, T. A. Folger, (1988), Fuzzy Sets, Uncertainty and Information, Prentice-Hall International Editions.
- [32] H. I. Krebs, B. T. Volpe, M. L. Aisen, N. Hogan, (2000), Increasing productivity and quality of care: robot-aided neuro-rehabilitation, *Journal of rehabilitation research and development*, 37(6), 639-652.
- [33] S. Kumar, D. Ramakrushna, A. Kumar, (2009 January), Fuzzy logic techniques for navigation of several mobile robots, *Applied Soft Computing*, 9(1), 290-304.
- [34] X. Li, B. Choi, (2013, May), Design of Obstacle Avoidance System for Mobile Robot using Fuzzy Logic Systems, *International Journal of Smart Home*, 7(3)
- [35] E. H. Mamdani, (1974, December), Application of fuzzy algorithms for control of simple dynamic plant, *Proceedings of the Institution of Electrical Engineers*, 121(12), 1585-1588
- [36] F. Matía, (1994), Análisis y diseño de sistema de control de procesos basados en lógica borrosa, Ph. D. thesis, Universidad Politécnica de Madrid.
- [37] A. Medina-Santiago, J. L. Camas-Anzueto, J. A. Vazquez-Feijoo, H. R. Hernandez de Leon, R. Mota-Grajales, (2014, February), *Journal of Applied Research and Technology*, 12(1), 104-110.
- [38] C. V. Negoita, (1978), Fuzzy Systems, Abacus Press, London.
- [39] N. J. Nilsson, (1984), Shakey the robot, SRI INTERNATIONAL MENLO PARK CA.
- [40] I. R. Nourbakhsh, J. Bobenage, S. Grange, R. Lutz, R. Meyer, A. Soto, (1999), An affective mobile robot educator with a full-time job, *Artificial Intelligence*, 114(1), 95-124.
- [41] A. Ollero, (2001), Robótica: Manipuladores y robots móviles, 17-18.
- [42] H. Omrane, M. Slim, M. Masmoudi, (2016), Fuzzy Logic Based Control for Autonomous Mobile Robot Navigation, *Comput Intell Neurosci*.

- [43] P. Persson, J. Laaksolahti, P. Lönnqvist, (2001), Understanding socially intelligent agents-a multilayered phenomenon, *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 31(5), 349-360.
- [44] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, S. Thrun, (2003), Towards robotic assistants in nursing homes: Challenges and results, *Robotics and Autonomous Systems*, 42(3), 271-281.
- [45] E. Prassler, J. Scholz, P. Fiorini, (2001), A robotics wheelchair for crowded public environment, *Robotics Automation Magazine, IEEE*, 8(1), 38-45.
- [46] J. Rada-Vilela, “<https://www.fuzzylite.com/>”.
- [47] L. Ran, Y. Zhang, Q. Zhang, T. Yang, (2016), Autonomous Wheeled Robot Navigation with Uncalibrated Spherical Images. In *Chinese Conference on Intelligent Visual Surveillance*; Springer: Singapore, pp. 47–55.
- [48] P. Reignier, (1994 April), Fuzzy logic techniques for mobile robot obstacle avoidance, *Robotics and Autonomous Systems*, 12(3-4), 143-153.
- [49] D. Rodriguez-Losada, F. Matia, R. Galan, A. Jimenez, (2002), Blacky, an interactive mobile robot at a trade fair, In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on* (Vol. 4, pp. 3930-3935), IEEE.
- [50] D. Rodriguez-Losada, F. Matia, J. M. Lucas, J.M. Montero, M. Hernando, R. Galan, (2008), Urbano, an interactive mobile tour-guide robot, INTECH Open Access Publisher.
- [51] B.M. Scassellati, (2001). Foundations for a Theory of Mind for a Humanoid Robot.
- [52] J. Schulte, C. Rosenberg, S. Thrun, (1999), Spontaneous, short-term interaction with mobile robots, In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on* (Vol. 1, pp. 658-663). IEEE.
- [53] M. Shiomi, T. Kanda, H. Ishiguro, N. Hagita, (2006, March), Interactive humanoid robots for a science museum, In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction* (pp. 305-312). ACM.
- [54] Siegwart, et. al., (2003), Robox at Expo. 02: A large-scale installation of personal robots, *Robotics and Autonomous Systems*, 42(3), 203-222.
- [55] T. Smithers, (1992), Taking eliminative materialism seriously: A methodology for autonomous systems research, *Proc. of the 1st Euro. Conf. on Artificial Life*.
- [56] T. Srdjan, M. Zeljko, (2010), Fuzzy-Based Controller for Differential Drive Mobile Robot Obstacle Avoidance, *IFAC Proceedings Volumes*, 43(16), 67-72.
- [57] T. Takagi and M. Sugeno, (1985, January), Fuzzy identification of systems and its applications to modeling and control, *IEEE transactions on systems, man, and cybernetics, SMC-15*(1), 116-132.
- [58] K. Tanaka, M. Sano, (1995), Trajectory stabilization of a model car via fuzzy control, *Fuzzy Sets and Systems*, 70(2-3), 155-170.

- [59] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Schulz, (1999), MINERVA: A tour-guide robot that learns, *KI-99: Advances in Artificial Intelligence*, 14-26.
- [60] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Schulz, (1999), MINERVA: A second-generation museum tour-guide robot, *In Robotics and automation, 1999. Proceedings. 1999 IEEE international conference on* (Vol. 3). IEEE.
- [61] E. Trillas, (1980), Conjuntos Borrosos, Vicens Universidad.
- [62] T. Willeke, C. Kunz, I. R. Nourbakhsh, (2001, May), The History of the Mobot Museum Robot Series: An Evolutionary Study, *In FLAIRS Conference* (pp. 514-518).
- [63] L.A.Zadeh, Fuzzy Sets, *Information and Control* 8, New York, Academic Press (1965) 338-353.
- [64] H. J. Zimmermann, (1991), *Fuzzy Set Theory and its Applications*, Kluwer Academic Publisher.