



**Universidad Autónoma del Estado
de Hidalgo**

Instituto de Ciencias Básicas e Ingeniería

**Área Académica de Computación y
Electrónica**

Navegación autónoma de un robot diferencial basado en Inteligencia Computacional

Tesis

Que como parte de los requisitos para obtener el Grado de

**Maestría en Ciencias en Computación Avanzada y
Electrónica**

Presenta

Ing. Salvador Santos Romero

Director

Dr. Virgilio López Morales

Codirector

Dr. Julio Cesar Ramos Fernández

Dra. Elsa Rubio Espino
Presidente

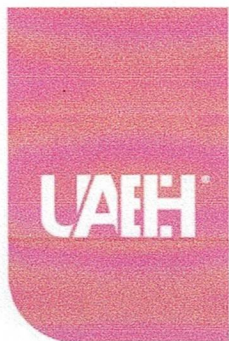
Dr. Manuel Alejandro Ojeda Misses
Secretario

Dr. Virgilio López Morales
Primer Vocal

Dr. Julio Cesar Ramos Fernández
Segundo Vocal

Dra. Anilú Franco Árcega
Suplente

Pachuca de Soto, Hgo. México
26 de agosto de 2026



Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería

School of Engineering and Basic Sciences

Área Académica de Computación y Electrónica

Department of Electronics and Computer Science

Mineral de la Reforma, Hidalgo, a 19 de agosto de 2026

Número de control: ICBI-AACyE/1434/2026

Asunto: Autorización de impresión de tesis.

MTRA. OJUKY DEL ROCIO ISLAS MALDONADO
DIRECTORA DE ADMINISTRACIÓN ESCOLAR DE LA UAEH
PRESENTE

El comité Tutorial de TESIS del alumno Salvador Santos Romero, con **No. de cuenta:** 513025 adscrito al Programa de Posgrado de la Maestría en Ciencias en Computación Avanzada y Electrónica informa que después de leer y revisar el documento de tesis titulado "Navegación autónoma de un robot diferencial basado en Inteligencia Computacional" y con base a lo estipulado en el Artículo 110 del reglamento de Estudios de Posgrado, tiene a bien extender la presente:

AUTORIZACIÓN DE IMPRESIÓN

Por lo que el sustentante deberá cumplir con los requisitos del Reglamento de Estudios de Posgrado y con lo establecido en el proceso de grado vigente para continuar con el proceso de titulación.

Atentamente

"Amor, Orden y Progreso"

Dr. Virgilio López Morales

Director de Tesis

Comité tutorial

Dra. Elsa Rubio Espino

Dr. Manuel Alejandro Ojeda Misses

Dr. Virgilio López Morales

Dr. Julio César Ramos Fernández

Dra. Anilú Franco Árcega

Presidente IPN

Secretario UA EH

Primer vocal UA EH

Segundo vocal UPP

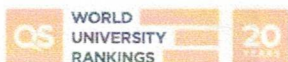
Suplente UA EH



ÁREA ACADÉMICA DE
COMPUTACIÓN Y ELECTRÓNICA

[Handwritten signatures of the committee members]

Ciudad del Conocimiento, Carretera Pachuca-Tulancingo Km. 4.5 Colonia Carboneras, Mineral de la Reforma, Hidalgo, México. C.P. 42184
Teléfono: 52 (771) 71 720 00 Ext. 40052, 40053
aacye_icbi@uaeh.edu.mx, jesus_ordaz@uaeh.edu.mx



Agradecimientos

Quisiera expresar mi más sincero agradecimiento a todas las personas que me guiaron y acompañaron durante esta etapa de formación tan importante para mí y que hicieron posible la realización de este trabajo de tesis:

A **CONAHCYT** por la beca otorgada durante el periodo septiembre 2024 - agosto 2026.

Al Laboratorio Nacional en Vehículos Autónomos y Exoesqueletos de la Universidad Politécnica de Pachuca, por proporcionarme el espacio y equipo necesario para el desarrollo de mi tema de investigación.

A la Universidad Autónoma del Estado de Hidalgo por darme la oportunidad de desarrollar mi tema de investigación dentro del programa educativo de Maestría en Ciencias en Computación Avanzada y Electrónica.

A mis asesores, el Dr. Virgilio López Morales, Dr. Julio Cesar Ramos Fernández y al Dr. Manuel Alejandro Ojeda Misses, que me guiaron y aconsejaron durante todo este proceso de formación y desarrollo de tema de investigación, además de inspirarme y motivarme por este apasionante campo de la investigación con sus experiencias y trabajo.

A mi esposa, por animarme, acompañarme y darme valor en cada uno de mis pasos en este proyecto; a mis padres y hermano, por su apoyo constante e incondicional y por estar allí para escuchar y aconsejarme; y a mis amigos, por su aliento y tiempo.

Finalmente, a todos los profesores que tuve la oportunidad de conocer y que fueron parte indispensable en mi formación y que de igual forma me transmitieron su pasión por este campo científico. Al igual que a mis compañeros de maestría que de alguna forma aportaron al proyecto con su tiempo y recomendaciones.

Nomenclatura y abreviaciones

Sigla	Descripción
ANFIS	Adaptive Neuro-Fuzzy Inference System
APF	Campos Potenciales Artificiales (Artificial Potential Fields)
CPU	Unidad Central de Procesamiento (Central Processing Unit)
DFL	Linealización por Retroalimentación Dinámica (Dynamic Feedback Linearization)
DWA	Dynamic Window Approach
ESP32	Microcontrolador de desarrollo
FIS	Sistema de Inferencia Difusa (Fuzzy Inference System)
FLC	Controlador Lógico Difuso (Fuzzy Logic Controller)
IMU	Unidad de Medición Inercial (Inertial Measurement Unit)
IC	Inteligencia Computacional (Computational Intelligence)
LiDAR	Light Detection and Ranging
LiPo	Batería de Polímero de Litio
MPC	Control Predictivo del Modelo (Model Predictive Control)
MF	Función de Membresía (Membership Function)
MPG	Modus Ponens Generalizado
PD	Controlador Proporcional-Derivativo
PID	Controlador Proporcional-Integral-Derivativo
PSO	Optimización por Enjambre de Partículas (Particle Swarm Optimization)
PWM	Modulación por Ancho de Pulso (Pulse Width Modulation)
RAM	Memoria de Acceso Aleatorio (Random Access Memory)
RMSE	Raíz del Error Cuadrático Medio (Root Mean Square Error)
VPH	Histograma Polar Vectorial (Vector Polar Histogram)
VFF	Campo de Fuerza Virtual (Virtual Force Field)
VFH	Histograma de Campo de Vector (Vector Field Histogram)
Variables Lingüísticas (Antecedentes)	
Dist_P	Error de posición
$error_{\alpha}$	Error de orientación
<i>zero</i>	Cero (distancia) o Error cero (orientación)
<i>ne</i>	Cerca (posición)
<i>med</i>	Media (posición)
<i>far</i>	Lejos (posición)
<i>mf</i>	Muy lejos (posición)
<i>muyneg</i>	Muy negativo (orientación)
<i>neg</i>	Negativo (orientación)
<i>pos</i>	Positivo (orientación)
<i>muypos</i>	Muy positivo (orientación)
Variables Lingüísticas (Consecuentes/Salidas)	
<i>vsl</i>	Muy lento
<i>sl</i>	Lento
<i>me</i>	Medio

Sigla	Descripción
<i>fa</i>	Rápido
<i>vf</i>	Muy rápido
<i>nf</i>	Negativo rápido
<i>nsf</i>	Negativo lento
<i>f</i>	Rápido positivo

Resumen

Este proyecto de investigación tiene como objetivo mejorar la capacidad de navegación autónoma de un robot móvil diferencial mediante la implementación de técnicas avanzadas de control y planificación local de rutas (*Local Path Planning*). Para ello, se integran sistemas de control basados en Inteligencia Computacional (IC), específicamente algoritmos de lógica difusa, así como tecnología de percepción espacial a través de un sensor LiDAR. La evaluación del desempeño se lleva a cabo mediante el diseño y construcción de una plataforma experimental física, complementada con simulaciones numéricas en diversos escenarios de navegación con múltiples obstáculos.

El principal desafío tecnológico abordado es la implementación de estos sistemas de percepción y control en plataformas de hardware con recursos limitados (Edge Computing en inglés), utilizando una arquitectura embarcada basada en Raspberry Pi y Python. La flexibilidad inherente del control difuso permite un diseño modular y adaptable, resultando idóneo para sistemas cuya cinemática y modelado matemático presentan alta complejidad computacional.

Además, dentro de este trabajo se verificó la estabilidad práctica (UUB) para el controlador inteligente, mediante el análisis numérico de la función de Lyapunov en el espacio de fases, garantizando la disipación de energía y la convergencia de las trayectorias hacia el radio de tolerancia establecido.

Los resultados de este trabajo contribuyen al campo de la robótica móvil al optimizar las rutinas de evasión de obstáculos y seguimiento de trayectorias, demostrando la viabilidad técnica y eficiencia de ejecutar algoritmos modernos de IC en sistemas embarcados de bajo costo.

Palabras clave: Navegación autónoma, Control difuso, Inteligencia computacional, Sistemas embarcados, Robot diferencial.

Abstract

This research project aims to enhance the autonomous navigation capabilities of a differential mobile robot through the implementation of advanced control techniques and local path planning. To this end, control systems based on Computational Intelligence (CI)—specifically fuzzy logic algorithms—are integrated alongside spatial perception technology using a LiDAR sensor. Performance evaluation is conducted through the design and construction of a physical experimental platform, complemented by numerical simulations across various navigation scenarios involving multiple obstacles.

The primary technological challenge addressed is the implementation of these perception and control systems on resource-constrained hardware platforms, utilizing an embedded architecture based on Raspberry Pi and Python. The inherent flexibility of fuzzy control enables a modular and adaptable design, making it highly suitable for systems whose kinematics and mathematical modeling present high computational complexity.

Furthermore, this study verified the practical stability (UUB) of the intelligent controller through numerical analysis of the Lyapunov function in phase space, ensuring energy dissipation and the convergence of trajectories toward the established tolerance radius.

The results of this study contribute to the field of mobile robotics by optimizing obstacle avoidance and trajectory-tracking routines, demonstrating the technical viability and efficiency of deploying modern CI algorithms on low-cost embedded systems.

Keywords: Autonomous navigation, Fuzzy control, Computational intelligence, Embedded systems

Índice

1	Introducción	1
1.1	Planteamiento del problema	2
1.2	Justificación	2
1.3	Hipótesis	2
1.4	Objetivo general	3
1.5	Objetivos específicos	3
1.6	Resultados	3
1.7	Estructura del trabajo de tesis	4
2	Estado del arte	5
2.1	Métodos empleados en planificación local de rutas	5
2.2	Métodos empleados en navegación autónoma de robots móviles	7
3	Preliminares matemáticos	10
3.1	Modelo cinemático de un robot diferencial	10
3.2	Odometría	12
3.3	Definición del problema de navegación	14
3.4	Controlador clásico PD	15
3.5	Sistemas difusos	16
3.5.1	Conjuntos difusos	17
3.5.2	Funciones de pertenencia	18
3.5.3	Variable lingüística	20
3.5.4	Regla difusa Si-Entonces	21
3.5.5	FIS Mamdani	23
3.5.6	FIS Sugeno Tipo 1	25
3.5.7	ANFIS: Adaptive Neuro-Fuzzy Inference System	27
3.6	Algoritmos de planificación de rutas locales	29
3.6.1	Vector Polar Histogram (VPH)	29
3.6.2	Enhanced Vector Polar Histogram (VPH+)	31
4	Metodología	33
4.1	Integración y construcción de la plataforma experimental	33
4.1.1	Integración de la plataforma mecánica	33
4.1.2	Arquitectura electrónica y sistema de potencia	36
4.2	Algoritmo de generación dinámica de puntos de evasión	39
4.2.1	Pre-procesamiento de datos	39
4.2.2	Detección	41
4.2.3	Agrupamiento	41
4.2.4	Validación de segmentos	42
4.2.5	Cálculo de puntos de evasión globales	42
4.2.6	Comprobación de visibilidad del objetivo	43
4.2.7	Selección del punto objetivo mediante función de costo	44
4.3	Control PD para la navegación de un robot diferencial	46
4.3.1	Simulación del controlador PD	47

4.4	Control inteligente FIS Mamdani	50
4.4.1	FIS Mamdani para navegación	50
4.4.2	FIS Mamdani para reorientación	54
4.5	Control inteligente FIS Sugeno	58
4.6	Análisis de estabilidad de controlador difuso tipo Sugeno	62
5	Pruebas experimentales	65
5.1	Pruebas experimentales para algoritmos de control de navegación	65
5.1.1	Resultados para el controlador PD	65
5.1.2	Resultados para el controlador Mamdani	70
5.1.3	Resultados para el controlador Sugeno Tipo 1	73
5.1.4	Comparación de controladores	77
5.1.5	Análisis de costo computacional por escenario	82
5.2	Pruebas experimentales para el algoritmo Local Path Planning .	84
5.2.1	Navegación a un punto objetivo con múltiples obstáculos	84
5.2.2	Comparación frente al método VPH	85
6	Conclusiones y perspectivas	89
6.1	Sobre el desempeño de los controladores difusos	89
6.2	Sobre el algoritmo de Local Path Planning	89
6.3	Perspectivas	90

Índice de figuras

1	Configuración de un robot diferencial.	11
2	Representación esquemática de los errores de navegación: distancia al objetivo P y error de orientación α	15
3	Diagrama de bloques para un sistema de control.	16
4	Representación de una función de pertenencia triangular con parámetros a, b, c en 2, 5, 8.	19
5	Representación de una función de pertenencia trapezoidal con parámetros a, b, c, d en 2, 4, 6, 8.	20
6	Representación del razonamiento difuso para una sola regla y una sola premisa. La línea punteada muestra la fuerza de activación w	22
7	Diagrama de bloques de un sistema de inferencia difuso.	24
8	Diagrama de bloques de un sistema de inferencia difuso tipo Mamdani.	24
9	Diagrama de bloques de un sistema de inferencia difuso tipo Sugeno.	26
10	Arquitectura ANFIS para un FIS Sugeno de primer orden con dos reglas Si-Entonces.	27
11	Configuración, renderizado isométrico de Robot Móvil Diferencial en SolidWorks.	34
12	Configuración, renderizado lateral de Robot Móvil Diferencial en SolidWorks.	35
13	Configuración, renderizado frontal de Robot Móvil Diferencial en SolidWorks.	35
14	Arreglo dimensional del Robot Móvil Diferencial.	36
15	Diagrama del circuito de potencia del Robot Móvil diferencial.	37
16	Diagrama de bloques del sistema electrónico y sus componentes principales.	38
17	Configuración inicial del sensor LiDAR, 270° al frente del robot.	40
18	Ajuste, nueva configuración del sensor LiDAR, 0 rad al frente del robot, normalizado a un intervalo $[-\pi, \pi)$	41
19	Localización de índices laterales y detección de segmentos entre obstáculos.	43
20	Cálculo de puntos intermedios para todos los índices laterales en segmentos válidos.	44
21	Validación si un punto objetivo está angularmente dentro de uno de los pasillos, a partir de una rotación de ángulos desde un nuevo eje 0	45
22	Diagrama general de simulación en Simulink.	48
23	Bloque de simulación del control de posición.	48
24	Bloque de simulación para el control de orientación.	48
25	Bloque de simulación de la cinemática del robot diferencial.	49
26	Interfaz de resultados de simulación en <i>App Designer</i>	49
27	Funciones de membresía, variable lingüística error de posición.	51
28	Funciones de membresía, variable lingüística error de orientación.	52

29	Funciones de membresía, variable lingüística velocidad de control de la rueda derecha.	52
30	Funciones de membresía, variable lingüística velocidad de control de la rueda izquierda.	53
31	Superficie de control para la rueda derecha.	54
32	Superficie de control para rueda izquierda.	54
33	Funciones de membresía para la variable error de orientación en reorientación.	55
34	Funciones de membresía para la velocidad de control de la rueda derecha (w_r).	56
35	Funciones de membresía para la velocidad de control de la rueda izquierda (w_l).	56
36	Superficie de control del FIS de reorientación para la rueda derecha (w_r).	57
37	Superficie de control del FIS de reorientación para la rueda izquierda (w_l).	57
38	Espacio de fases de la derivada de la función de Lyapunov para las distintas combinaciones de e_P y e_θ dentro del universo de discurso del FIS Sugeno.	64
39	Trayectoria del robot en condición sin obstáculos, controlador PD, punto de inicio (0,0), punto objetivo (400,200).	66
40	Trayectoria y puntos de evasión del robot en condición obstáculo a la derecha, controlador PD, punto de inicio (0,0), punto objetivo (400,0).	67
41	Trayectoria y puntos de evasión del robot en condición obstáculo a la izquierda, controlador PD, punto de inicio (0,-307), orientación inicial $\pi/4$ rad, punto objetivo (400,200).	68
42	Trayectoria y puntos de evasión del robot en condición dos obstáculos, controlador PD, punto de inicio (0,-307), orientación inicial $\pi/4$ rad, punto objetivo (400,200).	69
43	Trayectoria del robot en condición sin obstáculos, controlador Mamdani, punto de inicio (0,0), punto objetivo (400,200).	70
44	Trayectoria y puntos de evasión del robot en condición obstáculo a la derecha, controlador Mamdani, punto de inicio (0,0), punto objetivo (400,0).	71
45	Trayectoria y puntos de evasión del robot en condición obstáculo a la izquierda, controlador Mamdani, punto de inicio (0,-307), punto objetivo (400,200).	72
46	Trayectoria y puntos de evasión del robot en condición dos obstáculos, controlador Mamdani, punto de inicio (0,-307), orientación $\pi/4$ rad, punto objetivo (400,200).	73
47	Trayectoria del robot en condición sin obstáculos, controlador Sugeno, punto de inicio (0,0), orientación de inicio 0 rad, punto objetivo (400,200).	74

48	Trayectoria y puntos de evasión del robot en condición obstáculo a la derecha, controlador Sugeno, punto de inicio (0,0), orientación de inicio 0 rad, punto objetivo (400,0).	75
49	Trayectoria y puntos de evasión del robot en condición obstáculo a la izquierda, controlador Sugeno, punto de inicio (0,-307), orientación de inicio $\pi/4$ rad, punto objetivo (400,200).	76
50	Trayectoria y puntos de evasión del robot en condición dos obstáculos, controlador Sugeno, punto de inicio (0,-307), orientación de inicio $\pi/4$ rad, punto objetivo (400,200).	77
51	Comparación de los tres controladores por evolución del error de orientación, condición sin obstáculos, posición inicial (0,0), orientación inicial 0 rad, punto objetivo (400,200).	78
52	Comparación de los tres controladores por evolución del error de orientación, condición con un obstáculo a la derecha, posición inicial (0,0), orientación inicial 0 rad, punto objetivo (400,0).	78
53	Comparación de los tres controladores por evolución del error de orientación, condición con un obstáculo a la izquierda, posición inicial (0,-307), orientación inicial $\pi/4$ rad, punto objetivo (400,200).	79
54	Comparación de los tres controladores por evolución del error de orientación, condición con dos obstáculos, posición inicial (0,-307), orientación inicial $\pi/4$ rad, punto objetivo (400,200).	80
55	Comparación de los tres controladores por evolución del error de posición, condición sin obstáculos, posición inicial (0,0), orientación inicial 0 rad, punto objetivo (400,200).	80
56	Comparación de los tres controladores por evolución del error de posición, condición con un obstáculo a la derecha, posición inicial (0,0), orientación inicial 0 rad, punto objetivo (400,0).	81
57	Comparación de los tres controladores por evolución del error de posición, condición con un obstáculo a la izquierda, posición inicial (0,-307), orientación inicial $\pi/4$ rad, punto objetivo (400,200).	81
58	Comparación de los tres controladores por evolución del error de posición, condición con dos obstáculos, posición inicial (0,-307), orientación inicial $\pi/4$ rad, punto objetivo (400,200).	82
59	La trayectoria del robot (línea roja) coincide con los puntos de evasión (puntos verdes y rosas/morados).	85
60	Trayectoria del robot (línea roja) y puntos objetivo seleccionados por la función de costo (puntos con color según el tiempo transcurrido).	86
61	Trayectoria del robot utilizando el algoritmo propuesto. Se muestra la ruta seguida (línea roja) y los puntos objetivo seleccionados por la función de costo.	88
62	Trayectoria del robot utilizando el algoritmo VPH (con modificación de recorte). El color de la trayectoria varía según el tiempo transcurrido.	88

Índice de tablas

2	Base de conocimiento para FIS de navegación, cada celda muestra (w_r, w_l)	53
3	Base de conocimiento para FIS de reorientación. Cada celda muestra (w_r, w_l)	57
4	Parámetros consecuentes calculados para la velocidad de la rueda derecha (w_r) mediante ANFIS.	59
5	Parámetros consecuentes calculados para la velocidad de la rueda izquierda (w_l) mediante ANFIS.	60
6	Parámetros consecuentes de cada regla para Sugeno de reorientación (w_r) obtenidos mediante ANFIS.	60
7	Parámetros consecuentes de cada regla para Sugeno de reorientación (w_l) obtenidos mediante ANFIS.	61
8	Comparación de desempeño computacional: Condición sin obstáculos	83
9	Comparación de desempeño computacional: Condición un obstáculo a la derecha	83
10	Comparación de desempeño computacional: Condición un obstáculo a la izquierda	84
11	Comparación de desempeño computacional: Condición dos obstáculos	84
12	Comparación del costo computacional entre algoritmos de evasión	87

1 Introducción

La robótica móvil autónoma ha experimentado avances significativos en las últimas dos décadas, con aplicaciones en diversas áreas como, por ejemplo, la industria, la exploración y la automatización de distintos procesos, entre otros. Muchos investigadores han previsto que los robots móviles se hagan cargo de diversas labores en plantas de fabricación, almacenes y sitios de construcción. Recientemente, el uso de robots móviles en aplicaciones de manipulación de materiales ha aumentado considerablemente. Sin embargo, los retos que enfrenta la navegación autónoma en tiempo real continúan siendo un área activa de investigación, especialmente cuando se consideran las limitaciones de hardware y la necesidad de implementar algoritmos de inteligencia computacional (IC) en sistemas embarcados de bajo costo.

El presente proyecto se enmarca en el desarrollo de un robot móvil de tipo diferencial, con el objetivo de mejorar su capacidad de navegación mediante la integración de sistemas de control avanzados y tecnologías de percepción como el sensor LiDAR. Se plantea el diseño de un modelo experimental, así como la simulación numérica del robot para evaluar su desempeño en tareas de navegación autónoma. Adicionalmente, se busca implementar dichos sistemas en una plataforma de hardware limitado, como una Raspberry Pi, utilizando Python como lenguaje de programación.

Por otro lado, es bien sabido que las técnicas basadas en IC, como el control difuso, nos permiten implementar en un sistema computacional un control con razonamiento similar al de un ser humano. Esto resulta muy útil cuando el sistema a controlar es complejo y carece de un modelo matemático, o cuando la implementación de dicho modelo consume demasiados recursos, ya que el control difuso opera de forma independiente de estos. En particular, este trabajo hace uso de Sistemas de Inferencia Difusa (FIS) tipo Mamdani, ampliamente utilizados por su facilidad de interpretación y sintonización, en combinación con un FIS tipo Sugeno, preferido en sistemas embarcados por su rapidez y bajo costo computacional, utilizando herramientas como ANFIS para su sintonización. Adicionalmente, la integración de algoritmos de planificación de rutas locales (*Local Path Planning*) permite la navegación de robots móviles desde un punto de inicio a un punto objetivo en un entorno desconocido. En estos escenarios, el robot puede encontrarse con obstáculos estáticos o dinámicos, presentando el reto de calcular rutas de evasión óptimas y seguras haciendo uso de sensores de percepción. El método a aplicar dentro de este trabajo consistirá en la generación dinámica de múltiples rutas de evasión, para posteriormente seleccionar la más eficiente de acuerdo con una función de costo.

El trabajo propuesto tiene el potencial de aportar significativamente al campo de la robótica móvil autónoma, especialmente en la optimización de sistemas de navegación en robots diferenciales mediante la aplicación de técnicas modernas de IC y la implementación de un algoritmo basado en la generación de puntos de evasión en plataformas computacionales de bajo costo.

1.1 Planteamiento del problema

En los últimos años, se han presentado avances considerables en la navegación autónoma de robots móviles y en los algoritmos de planificación de rutas en tiempo real. Gracias a estos desarrollos, las plataformas autónomas se han incorporado de manera más segura y eficiente en sectores clave como la industria, la logística y la medicina, entre otros. Sin embargo, aún persisten desafíos significativos; al enfrentarse a entornos densos, no estructurados y altamente dinámicos, los sistemas de navegación clásicos pueden fallar, llevando al vehículo a colisiones o comportamientos erráticos.

Por otro lado, gran parte de los métodos actuales de planificación de rutas son susceptibles a converger en mínimos locales cuando operan en configuraciones espaciales complejas, tales como esquinas en forma de “U” o corredores estrechos. Asimismo, a altas velocidades de desplazamiento, los algoritmos de evasión tradicionales pueden carecer de la eficiencia computacional necesaria para calcular una trayectoria segura a tiempo. Finalmente, existen enfoques que dependen de funciones de costo propensas a generar trayectorias poco suaves, en los cuales una parametrización incorrecta reduce drásticamente el desempeño general de la tarea de navegación.

1.2 Justificación

Actualmente, es necesario el desarrollo de sistemas de navegación que garanticen la seguridad y eficiencia de los robots móviles en entornos no estructurados y dinámicos. Frente a las limitaciones de los métodos clásicos que presentan dificultades para resolver configuraciones espaciales complejas, como los mínimos locales, o exigen un alto costo computacional para operar a altas velocidades, este trabajo de tesis propone una alternativa basada en inteligencia computacional. Específicamente, se plantea la implementación de Sistemas de Inferencia Difusa para obtener un desempeño rápido y adaptable a las condiciones cambiantes del entorno. La integración de este enfoque con un algoritmo dinámico de evasión de obstáculos permite mitigar los problemas actuales de la planificación de rutas en tiempo real, garantizando una alta tasa de éxito y un bajo consumo de recursos computacionales.

1.3 Hipótesis

La integración de un sistema modular de navegación basado en lógica difusa (Mamdani/Sugeno) con un algoritmo de planificación de rutas por generación dinámica de puntos de evasión, permite una navegación autónoma más robusta en entornos dinámicos que los métodos clásicos, incrementando la tasa de éxito en la evasión de obstáculos complejos (como mínimos locales), logrando trayectorias cinemáticamente más suaves y seguras en entornos no estructurados.

1.4 Objetivo general

Desarrollar un Robot Móvil Diferencial que integra algoritmos de planificación de rutas locales (*Local Path Planning*) y técnicas de control basadas en inteligencia computacional, funcionando completamente con sistemas embarcados, con la finalidad de optimizar la navegación autónoma y evasión de obstáculos.

1.5 Objetivos específicos

1. Construir una plataforma robótica móvil y modular, dimensionada para albergar el hardware de procesamiento, baterías y percepción (sensor LiDAR, encoders, IMU y cámara), con el fin de validar experimentalmente los algoritmos desarrollados.
2. Implementar una arquitectura electrónica de borde, que centralice el procesamiento de alto y bajo nivel, abarcando el control de velocidad de las ruedas y la adquisición de datos sensoriales para la odometría y evasión de obstáculos.
3. Dimensionar el sistema eléctrico de potencia que alimente los distintos módulos del robot, que incluyen la unidad de procesamiento, la alimentación y control de los motores, la temperatura y la plataforma *pan-tilt*.
4. Diseñar un controlador de navegación fundamentado en un Sistema de Inferencia Difuso (FIS) tipo Mamdani, evaluando su desempeño cinemático en contraste con un controlador clásico Proporcional-Derivativo (PD).
5. Desarrollar un Sistema de Inferencia Difuso tipo Sugeno de primer orden, haciendo uso de ANFIS a partir de datos obtenidos del FIS Mamdani sintonizado, para comparar su robustez y desempeño frente al controlador clásico PD.
6. Demostrar mediante análisis numérico de la función de Lyapunov, estabilidad práctica en el espacio de fases, estableciendo el Acotamiento Último Uniforme (UUB) del controlador Sugeno de primer orden.
7. Formular un algoritmo de *Local Path Planning* basado en la generación dinámica de puntos de evasión, validando su eficiencia computacional y tasa de éxito frente al método clásico de Histograma Polar Vectorial (VPH).

1.6 Resultados

El desarrollo e implementación de la arquitectura de navegación autónoma para el robot diferencial basada en Inteligencia Computacional (IC) permitió alcanzar los objetivos planteados, generando los siguientes resultados:

Plataforma Experimental

Se logró la integración mecatrónica de un robot móvil diferencial, el cual es la base de las pruebas experimentales alcanzadas. El sistema incluye la adquisición

y procesamiento de la odometría, además de una arquitectura de hardware y software capaz de soportar la ejecución de modelos de IC en tiempo real.

Control y Evasión de Obstáculos

Desarrollo de una arquitectura de software modular que integra los controladores basados en lógica difusa y los algoritmos reactivos de evasión, gestionando de manera síncrona la ejecución de los procesos computacionales y la de los recursos de hardware.

Producción Científica

Publicación del artículo "*Dynamic Evasion Point System for Mobile Robots*" en la revista *International Journal of Robotics and Control Systems* (IJRCS, Vol. 6, Núm. 1, 2026). La contribución de un planificador local guiado por funciones de costo normalizadas, integrando procesamiento LiDAR 2D y un controlador PD para el seguimiento de pose.

1.7 Estructura del trabajo de tesis

El presente documento se encuentra organizado de la siguiente manera: en el Capítulo 1 se expone el planteamiento del problema, la justificación, la hipótesis, el objetivo general y los objetivos específicos de este trabajo de investigación y desarrollo. Posteriormente, en el Capítulo 2 se aborda el estado del arte, donde se muestran técnicas y métodos empleados además de una comparativa con este trabajo. Después, en el Capítulo 3 se describe el marco teórico. Posteriormente, en el Capítulo 4 se presenta la metodología empleada para el desarrollo e implementación de los algoritmos de control y de evasión de obstáculos. A continuación, el Capítulo 5 muestra los resultados y discusión para los algoritmos implementados en la plataforma experimental y, finalmente, en la Sección de 6 se presentan las conclusiones y perspectivas del presente trabajo de tesis.

2 Estado del arte

Diversos estudios han abordado estrategias para la navegación autónoma y la evasión de obstáculos en robots móviles, utilizando sensores LiDAR y algoritmos de planificación de rutas. Dependiendo de la complejidad del entorno y los requisitos del sistema, la literatura reporta la implementación tanto de controladores clásicos como de técnicas basadas en Inteligencia Computacional. A continuación, se analizan los trabajos más relevantes y sus contribuciones.

2.1 Métodos empleados en planificación local de rutas

La planificación de rutas locales, también conocida como planificación en línea o control reactivo, consiste en generar trayectorias seguras en tiempo real. Este enfoque se basa en las lecturas sensoriales instantáneas del robot para lograr la evasión de obstáculos estáticos y dinámicos. Entre los métodos más destacados en este campo se encuentran los siguientes:

Campos Potenciales Artificiales (APF, por sus siglas en inglés), el cual es un método muy popular clásico que modela el entorno asignando una fuerza de atracción hacia el objetivo y fuerzas de repulsión alrededor de los obstáculos. Aunque es un método simple y computacionalmente eficiente, presenta el defecto crítico de quedar atrapado en mínimos locales, además de generar movimientos oscilatorios al navegar en pasillos estrechos [1].

Para mitigar estas deficiencias, investigaciones recientes han propuesto esquemas híbridos. Por ejemplo, en [2], los autores presentan un marco de planificación para robots móviles que integra la Optimización por Enjambre de Partículas (PSO) con los APF. En dicho trabajo, el método APF aporta adaptabilidad ante diversas situaciones del entorno, mientras que el PSO compensa sus limitaciones inherentes. Además, se han desarrollado variantes del algoritmo, como el método de Múltiples Subobjetivos (MTAPF), diseñadas específicamente para contrarrestar el problema de los mínimos locales.

El algoritmo de Campo de Fuerza Virtual *Virtual Force Field en inglés* (VFF) dirigido a robots equipados con sensores tipo sonar, el cual combina cuadrículas de certeza para el procesamiento de sensores con campos potenciales. La principal debilidad de este algoritmo es una excesiva reducción de detalles del entorno, además de presentar problemas en pasillos con espacio limitado entre obstáculos, rechazando al robot fuera de la trayectoria planeada [3].

Algoritmo de Histograma de Campo Vectorial *Vector Field Histogram en Inglés* (VFH) el cual fue creado para superar las limitaciones del método VFF. El VFH usa un histograma polar unidimensional que representa la densidad de obstáculos a su alrededor para elegir el sector libre más directo al objetivo. Su evolución, el VFH+, mejoró la fiabilidad considerando explícitamente el ancho del robot y aproximando su trayectoria cinemática mediante arcos circulares, además de introducir una histéresis para reducir las oscilaciones [4]. Posteriormente, el VFH* incorporó una verificación de anticipación usando el algoritmo de búsqueda A* para prever las consecuencias de la ruta y evitar que el robot entre en callejones sin salida [5].

Algoritmos de Histograma Polar Vectorial *Vector Polar Histogram en Inglés* (VPH) [6] y VPH+ [7] que se adaptaron del método VFH para la incorporación de sensores tipo LiDAR, los cuales eliminan las cuadrículas de certeza y toman directamente los datos polares del sensor, lo que reduce significativamente el costo computacional. Estos algoritmos procesan los datos del LiDAR para crear un histograma polar que identifica los "valles" (pasajes libres). Posteriormente, seleccionan la dirección óptima dentro del valle basándose en una función de costo que compara la dirección hacia el objetivo con la orientación actual del robot [6].

La versatilidad de los enfoques basados en histogramas y vectores se ha visto potenciada por la alta resolución espacial de los sensores LiDAR. Por ejemplo, Liang *et al.* [8] demostraron la vigencia del algoritmo VFH original al fusionarlo con datos LiDAR para la evasión de obstáculos en vehículos aéreos no tripulados (UAV) en entornos complejos. En el ámbito terrestre, Padgett y Browne [9] implementaron un sistema de evasión puramente vectorial utilizando LiDAR sobre una plataforma de tracción Mecanum, validando cómo la medición directa del tiempo de vuelo del láser simplifica el cálculo de las trayectorias de escape sin necesidad de mapas previos. Sin embargo, las funciones de costo de VPH y VPH+ no suelen estar normalizadas, por lo que una incorrecta sintonización de sus parámetros puede otorgar demasiada libertad al robot (enfocándose en evadir obstáculos sin alcanzar la meta) o, por el contrario, si se asigna un peso mayor a la atracción hacia el objetivo, el robot se vuelve más propenso a sufrir colisiones o a realizar maniobras de evasión bruscas. La principal debilidad de estos métodos radica en que la selección del mejor ángulo instantáneo no garantiza la mejor ruta global. La función de costo original del VPH, por ejemplo, es altamente sensible a los pesos y no toma en consideración la distancia euclidiana total de la trayectoria. En consecuencia, el robot puede elegir una ruta que, aunque sea angularmente óptima en un instante dado, derive en un recorrido total mucho más largo.

Algoritmos de Aproximación de Ventana Dinámica *Dynamic Window Approach en Inglés* (DWA) [10], [11] los cuales, en lugar de construir mapas repulsivos, muestrean combinaciones de velocidades lineales y angulares dentro de una "ventana dinámica", la cual está restringida por los límites de aceleración y la cinemática del robot. El método simula trayectorias a corto plazo y selecciona la más eficiente evaluando el progreso hacia la meta, el margen de seguridad frente a obstáculos y la velocidad del vehículo. Entre las ventajas del DWA destaca la consideración explícita de la dinámica del robot y su bajo costo computacional, habiéndose reportado tiempos de ciclo de control de 0.250 segundos. No obstante, este método puede provocar que el robot quede atrapado en mínimos locales. Además, de manera similar a los algoritmos VPH y VPH+, su desempeño depende en gran medida de funciones objetivo que deben parametrizarse, por lo que una sintonización inadecuada puede impedir que el sistema logre el comportamiento de navegación esperado.

Como alternativas modernas al DWA para superar estos retos, diversos autores han propuesto la integración de escaneos LiDAR en planificadores que consideran restricciones futuras. Ou *et al.* [12] desarrollaron un planificador local basado

en la Banda Elástica Temporal (TEB, por sus siglas en inglés) que utiliza un LiDAR 2D y filtros de Kalman para asociar y predecir las velocidades de obstáculos dinámicos, integrando estas restricciones en un grafo de optimización temporal. Por otro lado, enfoques cinemáticos más ligeros como el de Persecución Pura (*Pure Pursuit*) han sido mejorados mediante LiDAR. Nguyen *et al.* [13] propusieron una adaptación de este método para garantizar la desaceleración segura en curvas y mitigar problemas de pérdida de localización en entornos con características LiDAR dispersas (*sparse LiDAR*).

Finalmente, para superar las limitaciones de parametrización estricta y los mínimos locales inherentes a los métodos clásicos como VPH y DWA, la literatura del estado del arte muestra una transición hacia el uso de Inteligencia Computacional acoplada a sensores LiDAR. Enfoques basados en aprendizaje, como el propuesto por de Heuvel *et al.* [14], emplean mecanismos de atención espaciotemporal mediante Aprendizaje por Refuerzo Profundo (*Deep Reinforcement Learning*) para procesar directamente lecturas de LiDAR 2D, logrando una navegación previsoras en entornos altamente dinámicos. En el ámbito del control lógico, Hanana *et al.* [15] validaron que la Lógica Difusa es una herramienta altamente eficaz al utilizarse en conjunto con un LiDAR, desarrollando un controlador que permite ajustes de trayectoria en tiempo real basados puramente en reglas de inferencia a partir de los datos láser. Estos trabajos demuestran que el procesamiento no lineal de los datos sensoriales es clave para el desarrollo de planificadores locales más reactivos y robustos.

2.2 Métodos empleados en navegación autónoma de robots móviles

A continuación, se presentan las diferentes metodologías y técnicas empleadas en la navegación autónoma de robots móviles, las cuales se dividen en dos enfoques principales: técnicas basadas en control clásico y técnicas basadas en inteligencia computacional. Asimismo, se destaca cómo en los trabajos recientes se ha optado por la hibridación, combinando sistemas clásicos (que garantizan la estabilidad del sistema basándose en modelos físicos) con sistemas orientados a datos y toma de decisiones inspirados en el razonamiento humano. Esta tendencia responde a que ningún método, por sí solo, resulta óptimo para todos los escenarios dinámicos a los que puede enfrentarse un robot móvil.

En el trabajo de [16], los autores introducen un control PID acoplado con un algoritmo de optimización de rutas, permitiendo que el robot se mueva de manera más eficiente mientras evita obstáculos. Esto se logra mediante el procesamiento en tiempo real de los datos provenientes de un sensor LiDAR 2D, facilitando el cálculo de una trayectoria que minimiza los cambios bruscos en la dirección del movimiento.

Existen diferentes tipos de algoritmos para la planificación de rutas. Por ejemplo, en [17] se comparan distintos métodos, destacando el algoritmo de Optimización por Enjambre de Partículas (PSO) debido a su eficiencia computacional y alta tasa de éxito. En dicho estudio, se generan varias trayectorias en el mapa del entorno y, posteriormente, un controlador PID se encarga de seguirlas. Aunque

el controlador PID es el estándar en la industria para el control de velocidad y seguimiento de trayectorias debido a su simplicidad, su rendimiento decae ante dinámicas variables del entorno o del propio robot, ya que sus ganancias son tradicionalmente fijas.

Por otro lado, el Control Predictivo del Modelo (MPC) es valorado porque integra restricciones físicas, cinemáticas (no holonómicas) y de los actuadores dentro de un problema de optimización continua, asegurando que el robot no intente maniobras físicamente imposibles. Su mayor limitación radica en la gran carga computacional que exige para operar en tiempo real en entornos densos [18]. Un ejemplo de su aplicación se observa en [19], donde el MPC se utiliza en combinación con VPH+ para optimizar trayectorias, garantizando una navegación segura alrededor de los obstáculos mientras el vehículo se dirige a su meta.

Para superar las limitaciones de los métodos clásicos, la Inteligencia Computacional provee a los robots de la capacidad de aprender de la experiencia, manejar la incertidumbre y emular el razonamiento humano. Dentro de este campo, gran parte de las aportaciones para navegación se basan en la Lógica Difusa.

En [20], se desarrolla un controlador de evasión de obstáculos para un robot móvil utilizando tecnología LiDAR y lógica difusa. Este enfoque permite al robot detectar y evadir colisiones en entornos no estructurados. Gracias a la lógica difusa, es posible modelar la incertidumbre y tomar decisiones más flexibles en situaciones dinámicas, mejorando significativamente la capacidad de navegación. Trabajos similares, como los de [21], [22] y [23], demuestran la eficacia de los controladores difusos para minimizar los errores de posición y orientación. Estos sistemas traducen los datos sensoriales a variables y valores lingüísticos dentro de un sistema de inferencia, guiando al robot hacia el objetivo mientras evade obstáculos.

Además, el avance hacia arquitecturas más complejas ha dado lugar a la combinación de Sistemas de Inferencia Difusa con Redes Neuronales, como el Sistema de Inferencia Neuro-Difuso Adaptativo (ANFIS). Investigaciones como la de [24] desarrollan sistemas de navegación en entornos desordenados basados en ANFIS, FIS Sugeno y fusión sensorial. Este marco procesa los datos del entorno para generar salidas de control precisas, como los ángulos de dirección. Al ser entrenado con conjuntos de datos empíricos, el modelo ANFIS se adapta eficazmente a diversas situaciones de navegación.

En la misma línea, se han desarrollado exitosamente hibridaciones entre el control clásico y la Inteligencia Computacional. Ejemplos de esto son los trabajos de [25] y [26], donde la lógica difusa se emplea para ajustar dinámicamente, y en tiempo real, los parámetros (ganancias) de un controlador PID clásico, permitiendo una adaptación robusta a condiciones cambiantes.

Por último, en [27] se explora el uso de la Linealización por Retroalimentación Dinámica (DFL, por sus siglas en inglés) para el seguimiento de trayectorias predefinidas. Este método se combina con un Controlador Lógico Difuso (FLC) encargado de la evasión de obstáculos estáticos. Sin embargo, enfoques como el DFL presentan desventajas, como su alta complejidad computacional y la estricta dependencia de un modelado matemático preciso del sistema, lo cual

resulta poco práctico en escenarios dinámicos altamente no lineales.

En conclusión, la revisión de la literatura expone las diferentes técnicas y metodologías empleadas para la navegación autónoma de robots móviles. Por un lado, las técnicas basadas en el control clásico, como el PID, resultan ser alternativas fáciles de implementar; sin embargo, su estructura de parametrización fija puede afectar el desempeño del controlador ante entornos dinámicos. Además, métodos como la Linealización por Retroalimentación Dinámica (DFL) o el Control Predictivo del Modelo (MPC) son altamente convenientes cuando se requiere considerar las restricciones de la planta y garantizar un sistema muy estable; no obstante, su dependencia de un modelo matemático exacto los hace difíciles de modelar en la práctica.

Frente a estas limitaciones, enfoques como el Controlador Lógico Difuso (FLC) representan una solución óptima para los problemas de navegación y evasión de obstáculos. Al no requerir un modelo exacto de la planta, logran modelar la incertidumbre del entorno y se adaptan de manera robusta a las dinámicas cambiantes del sistema. Finalmente, la integración de arquitecturas más complejas, como ANFIS junto con FIS Sugeno, permite la sintonización de modelos a partir de datos empíricos, volviéndose una alternativa altamente eficiente para su implementación en sistemas con recursos computacionales limitados.

3 Preliminares matemáticos

3.1 Modelo cinemático de un robot diferencial

El robot móvil contiene dos ruedas motrices individuales montadas en la parte central del chasis en el mismo eje y dos ruedas (rueda giratoria) en la parte trasera y frontal del chasis para equilibrar el robot móvil durante el movimiento. Estas dos ruedas motrices son impulsadas independientemente por dos actuadores (motores) para lograr el desplazamiento lineal y angular, la Figura 1 muestra la descripción geométrica para el robot diferencial.

El robot diferencial se caracteriza por tres tipos de movimientos. Un desplazamiento lineal, cuando las velocidades de las ruedas son iguales y en la misma dirección. Un desplazamiento angular sobre el CCI cuando las ruedas giran en la misma dirección pero una rueda es más rápida que la otra. Desplazamiento angular sobre su propio eje, cuando las velocidades de las ruedas son iguales pero giran en sentido opuesto.

La configuración de un automóvil que se mueve sobre un plano se describe mediante sus coordenadas generalizadas (cf. (1)), donde el espacio de configuración C describe todas las posibles posiciones y orientaciones que el sistema puede adoptar y $\mathbb{R}^2$ representa el plano donde el robot o móvil se mueve, definido por las coordenadas (x, y) . S representa el espacio de las orientaciones, indicando que θ es un ángulo en el rango de $[0, 2\pi)$ [28, Cap. 4].

$$q = (x, y, \theta) \in C : C \subset \mathbb{R}^2 \times S. \quad (1)$$

Un robot diferencial es un sistema no holonómico porque su movimiento está restringido de tal manera que no puede moverse lateralmente de manera instantánea. Esta restricción se expresa matemáticamente como en la Ec. (2).

$$\dot{y} \cos(\theta) - \dot{x} \sin(\theta) = 0. \quad (2)$$

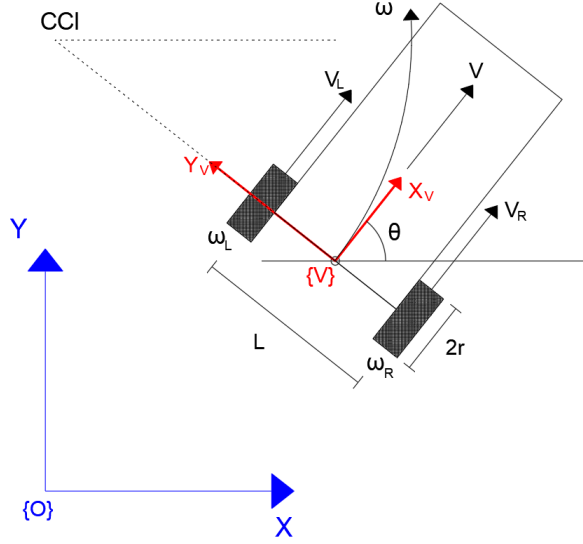


Figura 1: Configuración de un robot diferencial.

donde:

- L : Es la distancia entre las dos ruedas (m)
- r : Es el radio de la rueda (m)
- ω : Es la velocidad angular (rad/s)
- ω_R : Es la velocidad angular en la rueda derecha (rad/s)
- ω_L : Es la velocidad angular en la rueda izquierda (rad/s)
- V : Es la velocidad lineal (m/s)
- V_R : Es la velocidad lineal en la rueda derecha (m/s)
- V_L : Es la velocidad lineal en la rueda izquierda (m/s)
- θ : Es la orientación del robot (rad)
- CCI : Centro de curvatura instantáneo

La velocidad angular ω se puede definir como se indica en las Ecs. (3) y (4). El elemento ω se forma desde el CCI y eje horizontal, cuando V_R y V_L son iguales la velocidad angular $\omega = 0$, es decir el robot tiene un desplazamiento en línea recta.

$$\omega = \frac{V_R - V_L}{L}, \quad (3)$$

$$\omega = \frac{r(\omega_R - \omega_L)}{L}. \quad (4)$$

La velocidad lineal del robot depende de la velocidad angular de cada una de las ruedas y del radio de estas como se muestra en la Ec. (6).

$$V = \frac{r(\omega_R + \omega_L)}{2} \quad (5)$$

$$V = \frac{V_R + V_L}{2} \quad (6)$$

El modelo cinemático de este tipo de robot móvil se describe mediante las ecuaciones no lineales (7), (8) y (9), donde $\dot{x}$ representa la razón de cambio del desplazamiento en x , $\dot{y}$ representa la razón de cambio en y y $\dot{\theta}$ la razón de cambio del desplazamiento angular θ .

$$\dot{x} = \frac{r(\omega_R + \omega_L)}{2} \cos(\theta), \quad (7)$$

$$\dot{y} = \frac{r(\omega_R + \omega_L)}{2} \sin(\theta), \quad (8)$$

$$\dot{\theta} = \frac{r(\omega_R - \omega_L)}{L}, \quad (9)$$

$$\dot{x} = V \cos(\theta), \quad (10)$$

$$\dot{y} = V \sin(\theta), \quad (11)$$

$$\dot{\theta} = \omega. \quad (12)$$

3.2 Odometría

El robot calcula su pose en cada instante mediante odometría utilizando el método de estimación *Dead Reckoning*, el cual requiere una posición y orientación inicial conocida. Para ello, se discretiza el modelo cinemático continuo del robot diferencial mediante la aproximación de Euler con un periodo de muestreo $\Delta T = t_{k+1} - t_k$.

La actualización de la pose del robot en el instante $k + 1$ está dada por:

$$\begin{aligned} x_{k+1} &= x_k + v_k \Delta T \cos(\theta_k), \\ y_{k+1} &= y_k + v_k \Delta T \sin(\theta_k), \\ \theta_{k+1} &= \theta_k + \omega_k \Delta T, \end{aligned} \quad (13)$$

donde v_k (m/s) y ω_k (rad/s) representan las velocidades lineal y angular del centro del robot en el intervalo k , respectivamente.

En la plataforma experimental (microcontrolador ESP32), el desplazamiento físico se calcula a partir de los incrementos en los conteos de pulsos de los encoders (ΔN_R y ΔN_L). La distancia lineal recorrida por las ruedas derecha e izquierda ($\Delta s_R, \Delta s_L$) en metros durante el intervalo ΔT se determina por:

$$\Delta s_R = \pi D \left(\frac{\Delta N_R}{N_{PPR}} \right), \quad \Delta s_L = \pi D \left(\frac{\Delta N_L}{N_{PPR}} \right) \quad (14)$$

donde D es el diámetro de las ruedas (m) y N_{PPR} es la resolución del encoder (pulsos/rev) (para este trabajo, $N_{PPR} = 1200$). A partir de las distancias individuales, el desplazamiento lineal del centro del robot (Δs_k) y la variación angular ($\Delta \theta_k$) se expresan de la siguiente manera:

$$\Delta s_k = \frac{\Delta s_R + \Delta s_L}{2} = \frac{\pi D}{2N_{PPR}} (\Delta N_R + \Delta N_L) \quad (15)$$

$$\Delta \theta_k = \frac{\Delta s_R - \Delta s_L}{L} = \frac{\pi D}{LN_{PPR}} (\Delta N_R - \Delta N_L) \quad (16)$$

de modo que las velocidades promedio del sistema corresponden a $v_k = \Delta s_k / \Delta T$ y $\omega_k = \Delta \theta_k / \Delta T$.

Sustituyendo los desplazamientos (15) y (16) en el modelo de Euler (13), el término del periodo de muestreo ΔT se simplifica, obteniendo las ecuaciones finales de actualización de pose implementadas en el microcontrolador:

$$\begin{aligned} x_{k+1} &= x_k + \frac{\pi D}{2} \left(\frac{\Delta N_R}{N_{PPR}} + \frac{\Delta N_L}{N_{PPR}} \right) \cos(\theta_k), \\ y_{k+1} &= y_k + \frac{\pi D}{2} \left(\frac{\Delta N_R}{N_{PPR}} + \frac{\Delta N_L}{N_{PPR}} \right) \sin(\theta_k), \\ \theta_{k+1} &= \theta_k + \frac{\pi D}{L} \left(\frac{\Delta N_R}{N_{PPR}} - \frac{\Delta N_L}{N_{PPR}} \right), \end{aligned} \quad (17)$$

donde:

- x_k, y_k representan la posición cartesiana del robot en el instante k (m).
- θ_k representa la orientación del robot en el instante k (rad).
- D es el diámetro nominal de las ruedas (m).
- L es la distancia entre los centros de las ruedas (m).
- N_{PPR} es la resolución de los encoders (1200 pulsos por revolución).
- $\Delta N_R, \Delta N_L$ son las diferencias discretas de pulsos leídos en el instante de muestreo ($t_{k+1} - t_k$).

Este método puede introducir errores como los causados por el deslizamiento de las ruedas, los cuales se acumulan a medida que aumenta la distancia recorrida. Sin embargo, la estimación mantiene una eficacia aceptable a bajas velocidades debido a que se minimizan los efectos dinámicos de inercia y fricción. Asimismo, en trayectos o intervalos cortos, la acumulación integral de errores sistemáticos (tales como imperfecciones geométricas en el diámetro de las ruedas D o el eje L) y no sistemáticos (deslizamientos locales y rugosidad del terreno) se mantiene acotada.

3.3 Definición del problema de navegación

La navegación autónoma de un robot móvil se define como la capacidad del sistema para desplazarse desde una configuración inicial hasta una configuración objetivo dentro del plano cartesiano, de manera estable.

Para el diseño de las estrategias de control, se establece un sistema de coordenadas globales donde la postura actual del robot está dada por el vector $q_c = [x_c, y_c, \theta_c]^T$ y la posición del punto objetivo o intermedios por $q_g = [x_g, y_g]^T$. El objetivo de control consiste en minimizar el error de posición y el error de orientación.

El error de posición (o distancia Euclidiana), denotado como e_P , representa la magnitud del desplazamiento entre el robot y la meta (cf. [28, Cap. 4]). Este se define matemáticamente en la Ec. (18):

$$e_P = \sqrt{(x_g - x_c)^2 + (y_g - y_c)^2}, \quad (18)$$

donde (x_c, y_c) son las coordenadas actuales del robot y (x_g, y_g) las coordenadas del objetivo.

La segunda variable de control es el error de orientación, denotado como e_o . Este error mide la desviación angular entre la orientación actual del robot θ_c y el ángulo deseado hacia el objetivo β . Se calcula como se muestra en la Ec. (19):

$$e_o = \beta - \theta_c, \quad (19)$$

donde el ángulo deseado $\beta \in (-\pi, \pi]$ hacia la meta se determina mediante la función arcotangente de cuatro cuadrantes. A diferencia de la función trigonométrica estándar $\arctan(\cdot)$, que presenta discontinuidades y ambigüedad de cuadrante al omitir los signos individuales de las coordenadas, la función matemática $\text{atan2}(\Delta y, \Delta x)$ evalúa el incremento espacial en ambos ejes ($\Delta y = y_g - y_c$ y $\Delta x = x_g - x_c$) para mapear correctamente la orientación en todo el plano cartesiano. Su comportamiento se define matemáticamente mediante la siguiente función a trozos:

$$\beta = \text{atan2}(\Delta y, \Delta x) = \begin{cases} \arctan\left(\frac{\Delta y}{\Delta x}\right) & \text{si } \Delta x > 0 \\ \arctan\left(\frac{\Delta y}{\Delta x}\right) + \pi & \text{si } \Delta x < 0 \text{ y } \Delta y \geq 0 \\ \arctan\left(\frac{\Delta y}{\Delta x}\right) - \pi & \text{si } \Delta x < 0 \text{ y } \Delta y < 0 \\ +\frac{\pi}{2} & \text{si } \Delta x = 0 \text{ y } \Delta y > 0 \\ -\frac{\pi}{2} & \text{si } \Delta x = 0 \text{ y } \Delta y < 0 \\ \text{indefinido} & \text{si } \Delta x = 0 \text{ y } \Delta y = 0 \end{cases} \quad (20)$$

La Figura 2 ilustra geométricamente estas variables.

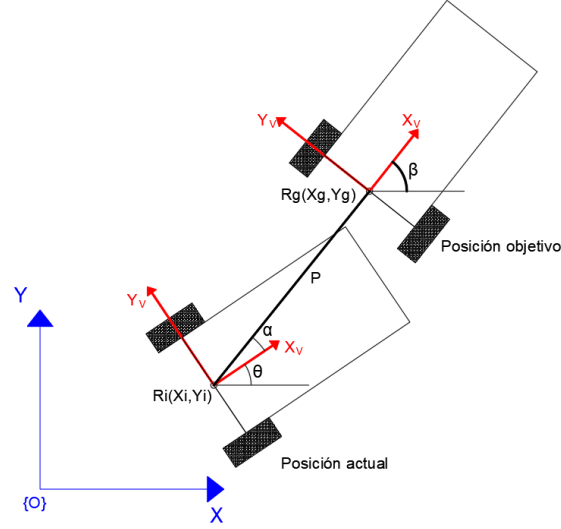


Figura 2: Representación esquemática de los errores de navegación: distancia al objetivo P y error de orientación α .

El sistema de control actúa sobre la planta generando las velocidades de referencia para las ruedas izquierda y derecha, ω_L y ω_R respectivamente. Por lo tanto, la tarea de navegación se considera completada cuando el controlador logra que las variables de error converjan a cero. En este trabajo, se abordará la implementación y comparación de técnicas de controlador clásico Proporcional Derivativo (PD) y Controladores Difusos.

3.4 Controlador clásico PD

El objetivo de un sistema de control automático es reducir la señal de desviación o error a cero o a un valor muy pequeño. Esto lo logra generando una señal de control. La manera en la cual el controlador automático produce la señal de control se denomina acción de control. Un sistema de control automático opera mediante lazo cerrado, compuesto por cuatro elementos esenciales, un controlador automático, un actuador, una planta (sistema a controlar) y un sensor o elemento de medición. La Figura 3 muestra un diagrama de bloques para un sistema de control.

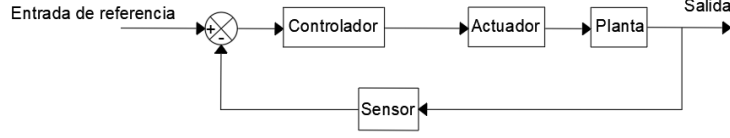


Figura 3: Diagrama de bloques para un sistema de control.

En la Figura 3, el sensor cuantifica la variable de salida, convirtiéndola en una magnitud comparable con la referencia. Basándose en esta comparación, el controlador calcula la desviación o error, $e(t)$, y genera una señal de control, $u(t)$. Esta señal es procesada por el actuador (por ejemplo, los motores del robot), el cual suministra la potencia necesaria a la planta para corregir su dinámica y aproximar la salida al valor deseado (cf. [29, Cap. 5]).

Para el propósito de este trabajo de tesis, se usará como referencia un controlador clásico Proporcional-Derivativo.

La acción proporcional actúa como un amplificador de ganancia ajustable donde la relación entre la salida del controlador $u(t)$ y la señal de error $e(t)$ es:

$$u(t) = K_p e(t), \quad (21)$$

donde K_p se considera la ganancia proporcional. Además, para mejorar la respuesta transitoria y la estabilidad se incorpora la acción derivativa. Esto permite obtener un controlador con alta sensibilidad. Una ventaja de usar una acción de control derivativa es que responde a la velocidad del cambio del error y produce una corrección significativa antes de que la magnitud del error se vuelva demasiado grande.

Así, **la acción derivativa** prevé el error, inicia una acción correctiva oportuna y tiende a aumentar la estabilidad del sistema.

La acción de control de un controlador proporcional-derivativa (PD) se define mediante la Ec. (22).

$$u(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt}, \quad (22)$$

donde T_d es el tiempo derivativo.

3.5 Sistemas difusos

La Lógica Difusa es una rama fundamental del campo del *Soft Computing*, el cual a su vez es un subcampo de la Inteligencia Computacional. Este campo de estudio se ha desarrollado paulatinamente desde su propuesta inicial en la década de 1960. Históricamente, cobró una gran relevancia en Japón, país donde se desarrollaron las principales aportaciones a la teoría y las primeras aplicaciones prácticas en controladores difusos. Actualmente, su uso se ha extendido a diversas áreas, ofreciendo soluciones a problemas de control, toma de decisiones, estrategias de mantenimiento predictivo y predicción de series de tiempo, entre otros.

Como cualquier rama de la Inteligencia Computacional, la Lógica Difusa busca imitar características de la inteligencia humana; específicamente, intenta simular el razonamiento aproximado del ser humano. Su objetivo principal es dotar a las máquinas de la capacidad para manejar la ambigüedad y la incertidumbre de manera similar a como lo procesa el cerebro humano.

3.5.1 Conjuntos difusos

Los sistemas difusos se basan en la teoría de Conjuntos Difusos. Estos se pueden definir a partir de la extensión del concepto de conjunto clásico (o conjunto nítido). Un conjunto clásico se define como una colección de elementos bien definidos dentro de un universo U . Cada uno de los elementos del universo puede pertenecer o no a un determinado conjunto clásico C . Si definimos una función de pertenencia $\mu_C(x)$ para cada uno de los elementos $x \in X$, el resultado es binario: un valor de 0 para los elementos no pertenecientes y 1 para los elementos que pertenecen totalmente al conjunto (cf. [30, Cap. 2]).

En cambio, un conjunto difuso D permite que un elemento pertenezca parcialmente al conjunto. Formalmente, un conjunto difuso se define como un par ordenado:

$$D = \{(x, \mu_D(x)) \mid x \in X\}, \quad (23)$$

donde $\mu_D(x)$ es la función de pertenencia (o función de membresía) del elemento x en el conjunto difuso D . Esta función mapea cada elemento de X a un grado de pertenencia en el intervalo continuo $[0, 1]$. Usualmente, X recibe el nombre de universo de discurso, el cual puede ser discreto (ordenado o no ordenado) o un espacio continuo.

En resumen, un conjunto difuso se distingue de un conjunto clásico por tres características principales:

- **Rango de pertenencia:** La función de pertenencia en un conjunto clásico es binaria ($\{0, 1\}$), mientras que en un conjunto difuso puede tomar cualquier valor real dentro del intervalo $[0, 1]$.
- **Principio del tercero excluido:** El conjunto clásico cumple estrictamente con este principio (un elemento pertenece o no pertenece, sin punto medio). En cambio, en la lógica difusa, un elemento puede pertenecer parcialmente a un conjunto y, simultáneamente, pertenecer parcialmente a su complemento.
- **Fronteras:** Las fronteras de un conjunto clásico son exactas, mientras que las fronteras de un conjunto difuso son graduales y suaves.

Estas características permiten modelar las entradas del sistema de control de una manera más precisa frente a la incertidumbre del mundo real, facilitando además una sintonización más intuitiva del controlador basada en el conocimiento experto.

3.5.2 Funciones de pertenencia

Como se ha mencionado anteriormente, un conjunto difuso está caracterizado completamente por su función de pertenencia (MF, por sus siglas en inglés). Dado que la mayoría de los conjuntos difusos en aplicaciones de control se basan en un universo de discurso continuo $\mathbb{R}$, la manera más concisa de definir una MF es expresándola mediante una fórmula matemática. A continuación, se describen las funciones de membresía empleadas en la metodología de este trabajo.

Función Triangular Una función de pertenencia triangular se define por medio de tres parámetros $\{a, b, c\}$ (con $a < b < c$), los cuales determinan las coordenadas en x de los vértices del triángulo. Los parámetros a y c representan en donde empieza y termina la función (donde la pertenencia es 0) y b representa el pico (donde la pertenencia tiene un valor de 1). Su definición matemática por tramos se muestra en la Ec. (24):

$$\mu_{tri}(x; a, b, c) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a < x \leq b \\ \frac{c-x}{c-b}, & b < x < c \\ 0, & c \leq x \end{cases} \quad (24)$$

Una alternativa compacta para expresar la función triangular, que resulta conveniente para su implementación en código, es utilizando las operaciones de mínimo y máximo como se observa en la Ec. (25):

$$\mu_{tri}(x; a, b, c) = \max \left(\min \left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right). \quad (25)$$

La parametrización directa de estas coordenadas ofrece una ventaja en la etapa de sintonización, permitiendo visualizar fácilmente la localización de las funciones dentro del universo de discurso y cómo se solapan con sus vecinas. La Figura 4 muestra un ejemplo de la forma de la función triangular.

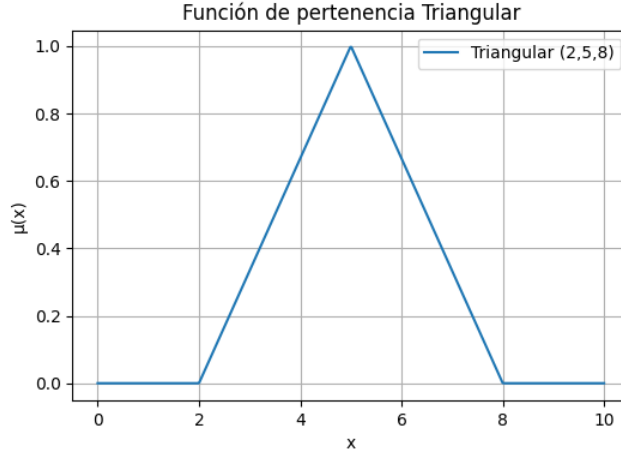


Figura 4: Representación de una función de pertenencia triangular con parámetros a , b , c en 2, 5, 8.

Función Trapezoidal De manera similar, y como una extensión natural de la forma triangular, la función trapezoidal se especifica mediante cuatro parámetros $\{a, b, c, d\}$ (con $a < b < c < d$). En este caso, el intervalo $[b, c]$ representa el núcleo del conjunto, donde el grado de pertenencia tiene un valor de 1. Su definición formal es:

$$\mu_{trap}(x; a, b, c, d) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a < x \leq b \\ 1, & b < x \leq c \\ \frac{d-x}{d-c}, & c < x < d \\ 0, & d \leq x \end{cases} \quad (26)$$

Al igual que en el caso anterior, su representación compacta mediante operadores $\min/\max$ facilita su programación en el código:

$$\mu_{trap}(x; a, b, c, d) = \max \left(\min \left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c} \right), 0 \right). \quad (27)$$

La Figura 5 muestra un ejemplo de la forma de la función trapezoidal.

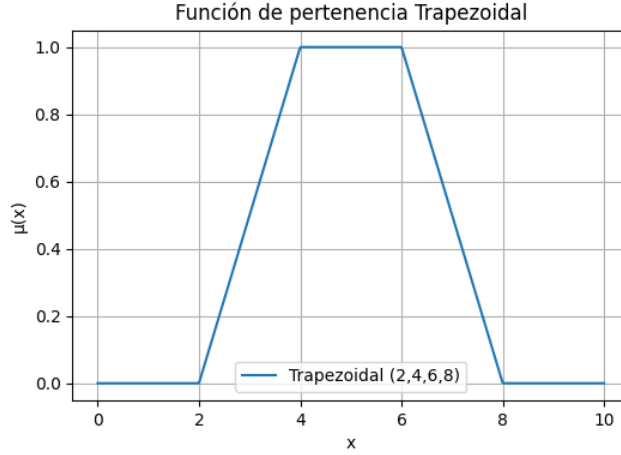


Figura 5: Representación de una función de pertenencia trapezoidal con parámetros a , b , c , d en 2, 4, 6, 8.

Las funciones anteriores son computacionalmente eficientes debido a que su cálculo se basa en operaciones aritméticas simples, evitando el uso de funciones complejas (como exponenciales). Por esta razón, su uso es habitual y recomendado en sistemas embarcados y aplicaciones de control en tiempo real.

3.5.3 Variable lingüística

Los Sistemas de Inferencia Difusa (FIS, por sus siglas en inglés), ampliamente utilizados para el control de sistemas, poseen un componente central denominado base de conocimiento. En esta base, un experto es capaz de describir el comportamiento de la dinámica esperada del sistema mediante el uso de variables lingüísticas y reglas difusas del tipo Si-Entonces (*If-Then*).

El concepto de variable lingüística fue propuesto por Lotfi A. Zadeh como una metodología para modelar de forma más precisa el razonamiento aproximado del ser humano. Esta herramienta permite resumir información y expresarla en términos de conjuntos difusos, haciendo uso de palabras o frases del lenguaje natural en lugar de valores numéricos exactos.

Formalmente, una variable lingüística se caracteriza por la quintupla $(x, T(x), X, G, M)$, donde:

- x : es el nombre o etiqueta conceptual de la variable (por ejemplo, “Distancia al objetivo”).
- $T(x)$: es el conjunto de términos o valores lingüísticos que la variable x puede tomar (por ejemplo, “Cerca”, “Media”, “Lejos”).
- X : es el universo de discurso o el rango físico donde la variable existe.

- G : es una regla sintáctica que genera los términos en $T(x)$ mediante modificadores (por ejemplo, “muy”, “poco”).
- M : es una regla semántica que asocia cada valor lingüístico con su respectiva función de pertenencia dentro del universo de discurso.

3.5.4 Regla difusa Si-Entonces

El comportamiento del sistema se basa en un conjunto de reglas difusas (o implicaciones difusas) (cf. [30, Cap. 3]), cuya estructura tiene la siguiente forma:

$$\text{Si } x \text{ es } A, \text{ entonces } y \text{ es } B \quad (28)$$

Esta estructura se compone de dos partes: la premisa " x es A ", comúnmente llamada antecedente, y la conclusión " y es B ", denominada consecuente. A su vez, A y B son los valores lingüísticos definidos dentro de los universos de discurso X e Y , respectivamente, para las variables x e y .

Un ejemplo práctico de una regla difusa empleada en el controlador de un robot móvil es: "*Si el error de posición es Alto, entonces la velocidad de las ruedas es Alta*".

Matemáticamente, una regla difusa (abreviada como $A \rightarrow B$) describe una relación entre dos variables. Para el diseño de controladores, es importante adoptar la interpretación de conjunción (A *acoplado con* B), donde el grado de activación de la salida está acotado por el grado de verificación de la entrada. Para inferir una conclusión a partir de los hechos observados y la regla difusa Si-Entonces, se emplea una operación matemática denominada Regla Composicional de Inferencia, la cual es el mecanismo más utilizado en los motores de inferencia difusa.

Esta regla de inferencia se expresa matemáticamente mediante la composición de la entrada difusa A' con la relación difusa de la regla R :

$$B' = A' \circ R \quad (29)$$

donde:

- R es la relación difusa que vincula el antecedente A y el consecuente B , modelada por la regla difusa Si-Entonces $A \rightarrow B$.
- A' es el antecedente actual o hecho observado, el cual puede no coincidir exactamente con el antecedente A definido en la regla.
- B' es la conclusión difusa resultante de la composición.

El operador $\circ$ representa la composición, comúnmente basada en la operación *max-min*, la cual se define a continuación en términos de sus funciones de pertenencia:

$$\mu_{B'}(y) = \max_x \min [\mu_{A'}(x), \mu_R(x, y)] \quad (30)$$

Este procedimiento de inferencia, también llamado razonamiento aproximado o Modus Ponens Generalizado (MPG), puede ser ilustrado para un solo antecedente y una sola regla, como se muestra a continuación:

$$\begin{array}{ll} \text{premisa 1 (hecho):} & x \text{ es } A', \\ \text{premisa 2 (regla):} & \text{si } x \text{ es } A \text{ entonces } y \text{ es } B, \\ \hline \text{consecuencia (conclusión):} & y \text{ es } B', \end{array} \quad (31)$$

Para este caso, la Ecuación (30) puede ser simplificada de la siguiente forma:

$$\begin{aligned} \mu_{B'}(y) &= [\vee_x (\mu_{A'}(x) \wedge \mu_A(x))] \wedge \mu_B(y) \\ &= w \wedge \mu_B(y), \end{aligned} \quad (32)$$

donde w corresponde a la fuerza de activación (grado de similitud) entre el hecho de la premisa y el antecedente de la regla difusa. Es esta fuerza de activación la que corta a la función conclusión $\mu_B(y)$; es decir, la función de pertenencia del consecuente queda acotada por w . Lo anterior se ilustra en la Figura 6.

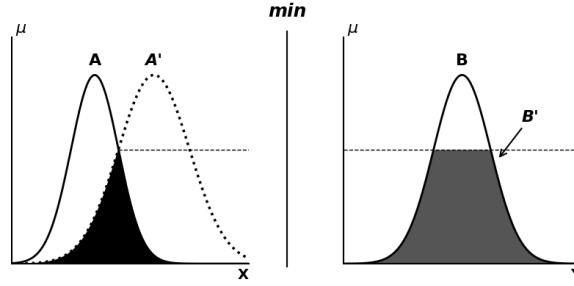


Figura 6: Representación del razonamiento difuso para una sola regla y una sola premisa. La línea punteada muestra la fuerza de activación w .

Por extensión, es posible aplicar este mismo principio a reglas con múltiples premisas, formalizadas como $A \times B \rightarrow C$. Por ejemplo, en este trabajo se aplican reglas difusas bidimensionales del tipo: "*Si la Distancia es Lejos y la Orientación es Positiva, entonces la Velocidad es Rápida*". Una regla difusa con dos premisas se ilustra como:

$$\begin{array}{ll} \text{premisa 1 (hecho):} & x \text{ es } A' , y \text{ es } B', \\ \text{premisa 2 (regla):} & \text{si } x \text{ es } A , y \text{ es } B \text{ entonces } z \text{ es } C \\ \hline \text{consecuencia (conclusión):} & z \text{ es } C' \end{array} \quad (33)$$

Tomando como base la simplificación definida por la Ec. (32) y extendiéndola a dos premisas, se obtiene lo siguiente:

$$\begin{aligned}
\mu_{C'}(z) &= \vee_{x,y} [\mu_{A'}(x) \wedge \mu_{B'}(y)] \wedge [\mu_A(x) \wedge \mu_B(y) \wedge \mu_C(z)] \\
&= \vee_{x,y} \{ [\mu_{A'}(x) \wedge \mu_A(x)] \wedge [\mu_{B'}(y) \wedge \mu_B(y)] \} \wedge \mu_C(z) \\
&= \underbrace{\{ \vee_x [\mu_{A'}(x) \wedge \mu_A(x)] \}}_{w_1} \wedge \underbrace{\{ \vee_y [\mu_{B'}(y) \wedge \mu_B(y)] \}}_{w_2} \wedge \mu_C(z) \\
&= \underbrace{(w_1 \wedge w_2)}_{\text{fuerza de activación}} \wedge \mu_C(z),
\end{aligned} \tag{34}$$

donde w_1 y w_2 son los valores máximos de la intersección (conjunción) de $A \wedge A'$ y $B \wedge B'$, respectivamente. Dado que las premisas están vinculadas por el operador lógico AND (" , "), se utiliza el operador mínimo ($\wedge$) para evaluar la fuerza de activación conjunta de la regla.

Por último, la base de conocimiento de un Sistema de Inferencia Difuso comúnmente se compone de un conjunto de reglas difusas con múltiples antecedentes. El MPG para este escenario se plantea de la siguiente manera:

$$\begin{array}{ll}
\text{premisa 1 (hecho):} & x \text{ es } A' , y \text{ es } B' \\
\text{premisa 2 (regla 1):} & \text{si } x \text{ es } A_1 , y \text{ es } B_1 \text{ entonces } z \text{ es } C_1 \\
\text{premisa 3 (regla 2):} & \text{si } x \text{ es } A_2 , y \text{ es } B_2 \text{ entonces } z \text{ es } C_2 \\
\hline
\text{consecuencia (conclusión):} & z \text{ es } C'
\end{array} \tag{35}$$

Para obtener la salida global del sistema C' , se combinan las inferencias individuales de cada regla activada ($C'_1, C'_2, \dots, C'_n$). En la inferencia tipo Mamdani, esta agregación se realiza calculando la unión geométrica de los conjuntos truncados mediante el operador máximo:

$$C' = C'_1 \cup C'_2 = \max(C'_1, C'_2). \tag{36}$$

3.5.5 FIS Mamdani

El Sistema de Inferencia Difusa (FIS, por sus siglas en inglés) es un marco computacional popular fundamentado en los conceptos de la teoría de conjuntos difusos, reglas difusas *If-Then* (si-entonces) y el razonamiento difuso. Esta metodología ha encontrado aplicaciones exitosas en una amplia variedad de campos, tales como el control automático, la clasificación de datos, el análisis de decisiones, los sistemas expertos, la predicción de series temporales, la robótica y el reconocimiento de patrones (cf [30, Cap. 4]).

La estructura básica de un FIS se ilustra en la Figura 7. En este diagrama se observa una entrada, la cual puede ser nítida (*crisp*) o difusa. Estos valores ingresan a través de variables lingüísticas, las cuales están compuestas por valores lingüísticos definidos mediante funciones de pertenencia (MF) sobre un universo de discurso. De esta forma, cada valor de entrada poseerá un grado de pertenencia entre 0 y 1 asignado por su respectiva función de membresía.

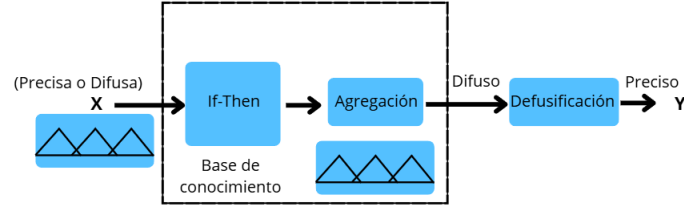


Figura 7: Diagrama de bloques de un sistema de inferencia difuso.

Posteriormente, el grado de pertenencia es evaluado por la base de conocimiento, conformada por un conjunto de reglas del tipo si-entonces. La cantidad total de reglas difusas dependerá directamente del número de variables lingüísticas y de la cantidad de valores lingüísticos por cada variable. Un enfoque típico es la Partición en Cuadrícula (*Grid Partition*), donde, si cada variable de entrada posee m funciones de pertenencia y existen n variables lingüísticas de entrada, el número total de reglas se define como m^n .

El resultado de evaluar cada regla es un valor difuso; cada regla generará una inferencia en la salida, pero aportará un peso distinto. La etapa de agregación permite unificar los resultados ponderados de todas las reglas, entregando, en la mayoría de los casos, un conjunto difuso resultante. Por consiguiente, se requiere un paso final de defusificación para obtener un valor nítido que pueda ser usado.

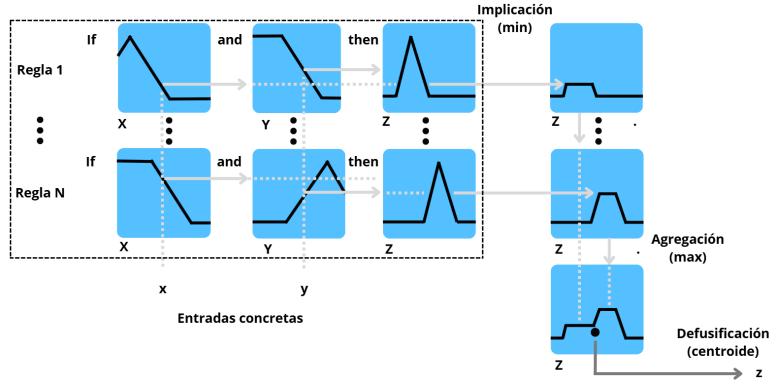


Figura 8: Diagrama de bloques de un sistema de inferencia difuso tipo Mamdani.

La Figura 8 muestra el Sistema de Inferencia Difusa tipo Mamdani, el cual se caracteriza por utilizar reglas de la forma:

$$\text{Regla } i : \text{Si } x \text{ es } A_i, y \text{ es } B_i, \text{ entonces } z \text{ es } C_i. \quad (37)$$

Este FIS hace uso del operador mínimo como T-norma para evaluar el grado de cumplimiento del antecedente compuesto, obteniendo así la fuerza de activación (o disparo) de la regla, w_i como en la Ec. (38):

$$w_i = \min(\mu_{A_i}(x), \mu_{B_i}(y)) \quad (38)$$

Mamdani utiliza la composición max-min, donde la implicación estándar consiste en aplicar el operador mínimo entre la fuerza de activación de la regla y su función de pertenencia consecuente dada por la Ec. (39):

$$\mu_{C'_i}(z) = \min(w_i, \mu_{C_i}(z)) \quad (39)$$

donde C_i es el conjunto difuso original del consecuente y C'_i es el conjunto difuso truncado resultante de la regla i .

Dado que cada regla en la base de conocimiento genera un conjunto difuso parcial, se emplea el operador S-norma (tipo máximo) para lograr la unión de todos los subconjuntos en un único conjunto difuso de salida. Este proceso se conoce como agregación Ec. (40):

$$\mu_{C_{total}}(z) = \max(\mu_{C'_1}(z), \mu_{C'_2}(z), \dots, \mu_{C'_n}(z)) \quad (40)$$

Finalmente, como producto de este FIS se obtiene un conjunto difuso continuo, el cual es indispensable convertir a un valor numérico nítido para que el sistema de actuadores físicos (los motores del robot) pueda ejecutarlo. Aunque existen distintos métodos de defusificación, el más empleado es el método de centro de gravedad o centroide (COA), el cual calcula el centro de área bajo la curva de la función agregada:

$$z^* = \frac{\sum_{j=1}^K \mu_{C_{total}}(z_j) \cdot z_j}{\sum_{j=1}^K \mu_{C_{total}}(z_j)}, \quad (41)$$

donde K es el número discreto de muestras en el que se divide el universo de discurso de salida, y $\mu_{C_{total}}(z_j)$ es el grado de pertenencia evaluado en la muestra z_j .

3.5.6 FIS Sugeno Tipo 1

Un Sistema de Inferencia Difusa Sugeno, también conocido como modelo TSK (propuesto por Takagi, Sugeno y Kang), emplea reglas difusas cuya estructura difiere en el consecuente (cf. [30, Cap. 4]):

$$\text{Regla } i : \text{Si } x \text{ es } A_i \text{ e } y \text{ es } B_i, \text{ entonces } z_i = f(x, y), \quad (42)$$

donde A_i y B_i son conjuntos difusos en el antecedente, mientras que $z_i = f(x, y)$ es una función matemática lineal y nítida como consecuente, dependiente de los valores de entrada. Por lo tanto, este modelo omite el proceso geométrico de agregación y defusificación del área, lo cual representa una reducción significativa en el tiempo de procesamiento por iteración, haciéndolo ideal para la ejecución del control en tiempo real en comparación con un FIS Mamdani.

En la Figura 9 se ilustra el diagrama de bloques de un sistema de inferencia difuso (FIS) tipo Sugeno. Aunque consiste en una estructura similar a la de un

modelo Mamdani, su principal diferencia se basa en la etapa de inferencia, ya que en lugar de generar conjuntos difusos, produce directamente valores nítidos. Dado que se trata de un sistema de primer orden, la salida de cada regla se representa mediante un *singleton* móvil, cuyo valor se determina dinámicamente evaluando una función lineal que depende de las entradas concretas x e y .

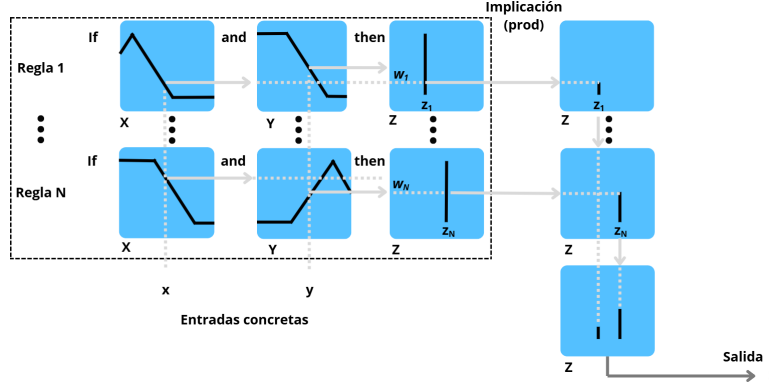


Figura 9: Diagrama de bloques de un sistema de inferencia difuso tipo Sugeno.

Para cada regla evaluada en un FIS Sugeno se generan dos valores fundamentales: w_i , que representa el peso o fuerza de la regla, y z_i , que es su nivel de salida numérico.

En un FIS Sugeno de primer orden, el nivel de salida z_i corresponde a una función lineal. Para un sistema con dos entradas, la ecuación toma la siguiente forma:

$$z_i = a_i x + b_i y + c_i \quad (43)$$

donde x e y son los valores nítidos de la entrada 1 y 2 respectivamente, y a_i, b_i, c_i son coeficientes constantes definidos durante el diseño o sintonización del controlador.

Por otro lado, la fuerza de activación de la regla, w_i , se deriva de la evaluación lógica del antecedente compuesto:

$$w_i = \min(\mu_{A_i}(x), \mu_{B_i}(y)) \quad (44)$$

donde $\mu_{A_i}(x)$ y $\mu_{B_i}(y)$ son los grados de pertenencia evaluados en las funciones de las entradas correspondientes.

La salida individual de cada regla es su nivel ponderado, es decir, el producto de w_i y z_i . El resultado final nítido del sistema de control completo es simplemente el promedio ponderado de las salidas de todas las reglas activadas:

$$\text{Salida} = \frac{\sum_{i=1}^N w_i z_i}{\sum_{i=1}^N w_i} \quad (45)$$

donde N representa el número total de reglas difusas en la base de conocimiento.

3.5.7 ANFIS: Adaptive Neuro-Fuzzy Inference System

El Sistema de Inferencia Neuro-Difuso Adaptativo (ANFIS, por sus siglas en inglés) es una arquitectura de red neuronal artificial basada en el sistema de inferencia difusa Takagi-Sugeno-Kang (TSK). Este modelo combina las técnicas de aprendizaje de las redes neuronales, como el algoritmo de mínimos cuadrados recursivos y la retropropagación (*backpropagation*), con la estructura lógico-lingüística de reglas Si-Entonces de los sistemas de inferencia difusos. Esta integración permite aproximar funciones no lineales y modelar sistemas complejos con precisión, motivo por el cual es una herramienta ampliamente utilizada en áreas como el modelado, la toma de decisiones, el procesamiento de señales y el control automático [30, Cap. 12].

Para ilustrar la arquitectura de un modelo ANFIS, es posible tomar como caso de estudio un Sistema de Inferencia Difuso Sugeno de primer orden (cuyas funciones consecuentes son lineales) conformado por dos reglas. Dicho conjunto de reglas se expresa de la siguiente manera:

$$\begin{aligned} \text{Regla 1: Si } x \text{ es } A_1 \text{ e } y \text{ es } B_1, \text{ entonces } f_1 &= p_1x + q_1y + r_1, \\ \text{Regla 2: Si } x \text{ es } A_2 \text{ e } y \text{ es } B_2, \text{ entonces } f_2 &= p_2x + q_2y + r_2, \end{aligned} \quad (46)$$

La arquitectura ANFIS equivalente para este FIS Sugeno de primer orden con dos reglas se ilustra en la Figura 10. En esta topología, los nodos de la capa 1 describen las funciones de membresía en el antecedente, mientras que los de la capa 4 modelan las ecuaciones lineales del consecuente. Ambos niveles se denotan mediante nodos rectangulares para indicar su característica adaptativa, lo que significa que sus parámetros internos se ajustarán de manera dinámica durante el proceso de entrenamiento.

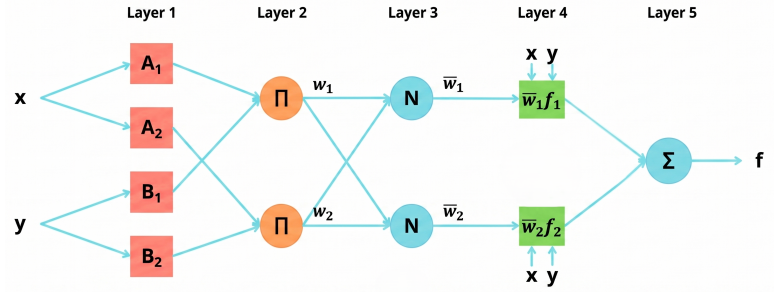


Figura 10: Arquitectura ANFIS para un FIS Sugeno de primer orden con dos reglas Si-Entonces.

La topología de una red ANFIS se compone de una estructura de alimentación hacia adelante (*feedforward*) distribuida en cinco capas. A continuación, se detalla el procesamiento matemático interno que se lleva a cabo en cada una de estas capas:

- Capa 1: Capa de Fusificación. Los nodos de esta capa son adaptativos, lo que significa que poseen parámetros ajustables durante la fase de entrenamiento (también conocidos como parámetros antecedentes). La función de cada nodo consiste en calcular el grado de membresía lingüística de las variables de entrada. Para un sistema con dos entradas (x, y) , el valor de salida de cada nodo se expresa como:

$$O_{1,i} = \mu_{A_i}(x), \quad \text{para } i = 1, 2 \quad (47)$$

$$O_{1,i} = \mu_{B_{i-2}}(y), \quad \text{para } i = 3, 4 \quad (48)$$

Donde $O_{1,i}$ representa la señal de salida del nodo i , y μ_{A_i} o $\mu_{B_{i-2}}$ corresponden a las funciones de membresía implementadas (tales como funciones triangulares o gaussianas). Los parámetros asociados a estas funciones definen la geometría de los conjuntos difusos en el antecedente.

- Capa 2: Capa de Reglas. En esta capa, cada nodo es del tipo fijo y se encarga de modelar el operador lógico de intersección (AND) a través de una T-norma. Comúnmente se emplea el producto aritmético para evaluar la condición de la regla. La salida de cada nodo representa la fuerza de disparo o peso (w_i) de la regla asociada:

$$O_{2,i} = w_i = \mu_{A_i}(x) \cdot \mu_{B_i}(y), \quad i = 1, 2 \quad (49)$$

Cada una de las señales resultantes cuantifica el nivel de activación de la regla difusa correspondiente ante el par de entradas determinado.

- Capa 3: Capa de Normalización. Esta capa está conformada por nodos fijos cuya función principal es normalizar las fuerzas de disparo calculadas en la capa previa. El nodo calcula la relación existente entre la fuerza de disparo de la regla i -ésima y la sumatoria total de las fuerzas de disparo de todas las reglas activas dentro de la base de conocimiento:

$$O_{3,i} = \bar{w}_i = \frac{w_i}{w_1 + w_2}, \quad i = 1, 2 \quad (50)$$

La señal resultante, designada como fuerza de disparo normalizada ($\bar{w}_i$), representa el peso relativo o la contribución proporcional que posee dicha regla sobre el cálculo de la acción de control final.

- Capa 4: Capa del Consecuente. En esta capa, los nodos son adaptativos y están vinculados a los parámetros del consecuente del modelo Takagi-Sugeno. Cada nodo calcula el aporte individual ponderado de su regla respectiva mediante el producto de la fuerza de disparo normalizada y un polinomio lineal de primer orden dependiente de los errores de entrada:

$$O_{4,i} = \bar{w}_i f_i = \bar{w}_i(p_i x + q_i y + r_i), \quad i = 1, 2 \quad (51)$$

Donde $\{p_i, q_i, r_i\}$ corresponde al conjunto de parámetros de diseño lineales modificables del consecuente (también conocidos como parámetros consecuentes), los cuales dan forma a la recta o plano que aproxima localmente la dinámica del sistema en esa región del espacio de estados.

- Capa 5: Capa de Salida. Esta capa final consta de un único nodo fijo que consolida el proceso de defusificación del sistema. Su propósito es calcular la salida neta global mediante la sumatoria algebraica de todas las señales parciales e individuales provenientes de la capa de los consecuentes:

$$O_{5,1} = \text{Salida global} = \sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \quad (52)$$

Esta expresión matemática representa el promedio ponderado final del sistema de inferencia, generando el valor nítido (*crisp*) de velocidad o acción de control que se transmitirá hacia los actuadores del robot móvil.

De la anterior forma se ha creado un modelo de red neuronal equivalente a un sistema de inferencia difuso Sugeno, con la capacidad adicional de ajustar sus parámetros mediante algoritmos de aprendizaje supervisado. Por último, es importante destacar que el proceso de entrenamiento de un modelo ANFIS se basa en la minimización del error cuadrático medio (MSE) entre las salidas predichas por la red y los valores reales de un conjunto de datos de entrenamiento, utilizando técnicas como el método híbrido que combina el algoritmo de mínimos cuadrados recursivos para ajustar los parámetros consecuentes y el algoritmo de retropropagación para ajustar los parámetros antecedentes.

3.6 Algoritmos de planificación de rutas locales

A continuación, se describen los algoritmos de planificación de rutas locales implementados en esta investigación para la navegación autónoma del robot móvil diferencial. Estos algoritmos forman parte de la familia de algoritmos basados en histogramas mencionados en el Estado del Arte (Capítulo 2), que posteriormente evolucionaron para poder ser integrados con sensores tipo LiDAR 2D. Este trabajo se inspira y se compara principalmente con los algoritmos Vector Polar Histogram (VPH) [6] y el algoritmo Vector Polar Histogram Plus (VPH+) [7].

3.6.1 Vector Polar Histogram (VPH)

El algoritmo *Vector Polar Histogram* (VPH) es un método de evasión de obstáculos en tiempo real que procesa directamente la información obtenida por un sensor LiDAR 2D, concluyendo en un valor de dirección junto con una velocidad de control. A continuación, se describen los pasos principales del algoritmo:

1. Creación del Histograma Polar Vectorial: El radar láser detecta la distancia a los obstáculos en cada ciclo de escaneo. Estos datos se transforman en un histograma polar, donde el eje x representa el ángulo de escaneo (cada grado en el plano) y el eje y representa la distancia medida al obstáculo $d(i)$ en cada ángulo.
2. Cálculo de la Función de Umbral Dinámica: VPH calcula una función de umbral $D(i, t)$ que varía en el tiempo. Esta función considera la velocidad en tiempo real del robot (V_t), su desaceleración (a), su radio (R) y una distancia de seguridad (D). La ecuación se define por tramos, dependiendo de si el obstáculo está dentro del campo de visión frontal ($|\theta(t) - i| \leq 90^\circ$) o no:

$$D(i, t) = \begin{cases} \frac{V_t^2 \cdot \cos^2(\theta(t) - i)}{2a} + R + D & \text{si } |\theta(t) - i| \leq 90^\circ \\ R + D & \text{si } |\theta(t) - i| > 90^\circ \end{cases} \quad (53)$$

3. Reducción al Histograma Binario: El histograma polar se compara con la función de umbral para generar un histograma binario $H(i)$. Los valores máximos (1) representan áreas libres y los valores mínimos (0) representan áreas bloqueadas por obstáculos:

$$H(i) = \begin{cases} 1 & \text{si } d(i) \geq D(i, t) \\ 0 & \text{si } d(i) < D(i, t) \end{cases} \quad (54)$$

4. Determinación de Sectores Libres (Candidatos): El algoritmo busca “ventanas” continuas de espacio libre en el histograma binario. Solo se consideran aquellas direcciones i cuya ventana tenga un ancho $W(i)$ mayor o igual a un umbral constante de ancho mínimo W_{max} . Esto genera un conjunto U de direcciones candidatas:

$$U = \{i \mid H(i) = 1 \text{ y } W(i) \geq W_{max}\} \quad (55)$$

5. Selección de la Dirección de Navegación: Para elegir la mejor dirección entre los candidatos, se evalúa una función de costo $C(i)$. Esta función penaliza las direcciones que se alejan de la dirección objetivo (Θ) y de la dirección actual de las ruedas del robot ($\theta(t)$), utilizando las constantes de peso K_1 y K_2 :

$$C(i) = K_1|i - \Theta| + K_2|i - \theta(t)| \quad (56)$$

La dirección elegida Ω será la que minimice esta función de costo:

$$C(\Omega) = \min C(i), \quad i \in U \quad (57)$$

6. Control de Velocidad: Finalmente, el robot ajusta su velocidad V . Intenta mantener su velocidad máxima V_{max} , pero la reduce instantáneamente si detecta un obstáculo cercano en la dirección seleccionada Ω . Esto se

calcula relacionando la distancia medida $d(\Omega)$, el rango máximo del radar láser D_{max} y el rango seguro del robot D_{min} :

$$V = \left(1 - \frac{D_{max} - d(\Omega)}{D_{max} - D_{min}}\right) V_{max} \quad (58)$$

Donde el rango seguro se define como:

$$D_{min} = R + D \quad (59)$$

3.6.2 Enhanced Vector Polar Histogram (VPH+)

El algoritmo *Enhanced Vector Polar Histogram* (VPH+) se caracteriza por agrupar los obstáculos en bloques y optimizar el tiempo de llegada. El algoritmo se divide en cinco pasos principales:

1. Modificación de la información original: Los datos brutos obtenidos por el radar láser se ajustan para tener en cuenta las dimensiones del robot (radio R) y determinar la distancia máxima que el robot puede avanzar en cada dirección. Si s_{ij} es la distancia perpendicular desde un obstáculo j a la línea de visión del obstáculo i , la distancia modificada d'_{ij} se calcula como:

$$s_{ij} = d_j \sin y_{ij} \quad (60)$$

$$d'_{ij} = \begin{cases} d_i & \text{si } s_{ij} > R \\ d_i & \text{si } (s_{ij} \leq R) \text{ y } (d_j \cos y_{ij} > d_i) \\ d_j \cos y_{ij} & \text{en otros casos} \end{cases} \quad (61)$$

La distancia máxima alcanzable D_i en la dirección i es:

$$D_i = \min(d'_{ij}) - R \quad \text{para } j = 0, 1, 2, \dots, 180 \quad (62)$$

2. Agrupación de obstáculos en bloques: Para evitar que el robot intente pasar por pasillos intransitables, los puntos de obstáculos adyacentes se agrupan evaluando la distancia entre ellos $d_{i,i+1}$. Si esta distancia es menor a un umbral d_{thr} , se consideran parte del mismo bloque de obstáculo:

$$d_{i,i+1} = \sqrt{d_i^2 + d_{i+1}^2 - 2d_i d_{i+1} \cos x} \quad (63)$$

3. Construcción de la función simbólica: Se clasifica cada bloque de obstáculos como “cóncavo” o “no cóncavo” comparándolo con sus bloques adyacentes. Las direcciones hacia bloques cóncavos se excluyen como opciones viables mediante la función binaria $B(i)$:

$$B(i) = \begin{cases} 0 & \text{si el punto obstáculo } i \text{ pertenece a un bloque cóncavo} \\ 1 & \text{en otros casos} \end{cases} \quad (64)$$

4. Construcción de la función de umbral: Para asegurar que no haya colisiones mientras el robot se mueve a una velocidad determinada, se establece una distancia segura d_{safe} que toma en cuenta la velocidad actual v , la velocidad máxima v_{max} , el radio de giro mínimo R_{min} , el ciclo de planificación T y un coeficiente de seguridad K .

$$d_{safe} = K \left(\frac{v}{v_{max}} R_{min} + vT \right) \quad (65)$$

La función de umbral $H(i)$ verifica si la distancia disponible supera esta distancia segura:

$$H(i) = \begin{cases} 1 & \text{si } D_i \geq d_{safe} \\ 0 & \text{en otros casos} \end{cases} \quad (66)$$

5. Construcción de la función de costo y control de velocidad: VPH+ utiliza una función de costo $C(i)$ orientada a minimizar el tiempo de llegada. Esta función considera el costo direccional $S(i)$, el cual penaliza la desviación respecto al objetivo (hg) y la desviación respecto al rumbo actual del robot (ho), usando las constantes k_1, k_2, k_3 :

$$S(i) = k_1 \cdot hg + k_2 \cdot ho + k_3 \quad (67)$$

$$C(i) = \frac{B(i)H(i)D_i}{S(i)} \quad (68)$$

La mejor dirección m será aquella que *maximice* esta función de costo:

$$C(m) = \max(C(i)) \quad \text{para } i = 0, 1, 2, \dots, 180 \quad (69)$$

Finalmente, la velocidad para el siguiente ciclo v_{n+1} se ajusta en función de la distancia al obstáculo en la dirección deseada (D_m) y la aceleración máxima a :

$$v_{n+1} = \begin{cases} v_{max} & \text{si } \sqrt{2aD_m} > v_{max} \\ \sqrt{2aD_m} & \text{en otros casos} \end{cases} \quad (70)$$

4 Metodología

El objetivo principal de esta investigación es lograr la navegación autónoma de un robot móvil diferencial mediante la integración de un algoritmo de control inteligente y un algoritmo de planificación de rutas locales (Local Path Planning). Esta combinación tiene el propósito de guiar al robot hacia un punto objetivo de manera segura y eficiente dentro de un entorno con obstáculos.

Para alcanzar este propósito, el desarrollo metodológico se divide en dos partes principales. En primer lugar, se abarca la implementación de dos sistemas de control para la navegación autónoma: un controlador de tipo clásico, que servirá como línea base de referencia, y dos sistemas de control inteligente basados en un Sistema de Inferencia Difuso (FIS, por sus siglas en inglés).

De manera simultánea, se desarrolla un sistema dinámico de evasión de obstáculos que utiliza como entrada las lecturas de un sensor LiDAR 2D. El algoritmo propuesto procesa estos datos en tiempo real para calcular puntos de evasión laterales (por la izquierda y por la derecha) respecto a los obstáculos detectados, permitiendo así seleccionar la ruta más corta y óptima hacia la meta.

Tanto los algoritmos de navegación como los algoritmos de evasión de obstáculos han sido implementados sobre una plataforma experimental (robot móvil diferencial) por lo que parte de este trabajo de tesis incluye la integración y construcción de esta plataforma experimental.

4.1 Integración y construcción de la plataforma experimental

La validación de los algoritmos de navegación y control propuestos en esta investigación requirió de una plataforma física con una capacidad alta de procesamiento en tiempo real. Por tal motivo, se desarrolló una plataforma robótica de tracción diferencial diseñada bajo un enfoque modular. En este apartado, se describe el proceso de integración del prototipo, estructurado en dos campos :

1. **Integración de la plataforma mecánica:** Se detalla la configuración geométrica y cinemática del robot. Se presentan los modelos tridimensionales desarrollados en SolidWorks, especificando la selección de materiales para el chasis, la disposición de los actuadores y las dimensiones generales que definen el modelo matemático del sistema.
2. **Arquitectura electrónica y sistema de potencia:** Se muestra la arquitectura de hardware, abarcando desde el sistema de potencia y regulación de energía, hasta la interconexión del conjunto de sensores (LiDAR, IMU y encoders) con los microcontroladores.

4.1.1 Integración de la plataforma mecánica

La integración mecánica de la plataforma diferencial se inspira en la arquitectura estándar utilizada por la mayoría de los robots industriales móviles. Su configuración alinea el centro del robot con el eje de las ruedas motrices, incorporando dos

ruedas locas (una frontal y una posterior) para garantizar el equilibrio estático y dinámico del chasis. La Fig. 11 presenta una vista isométrica de la arquitectura final. Las Figs 12 y 13 muestran las vistas lateral y frontal, ilustrando la disposición general de los componentes principales.

El sistema de tracción está compuesto por dos motores montados en la sección posterior del cuadro, fijados sobre perfiles deslizables que permiten ajustar su posición. La transmisión de potencia se realiza mediante un mecanismo de cadena y piñón que conecta los motores con dos ejes independientes. Estos ejes se mantienen alineados y centrados mediante chumaceras ancladas al cuadro inferior del robot.

Cada eje motriz fue diseñado a la medida para cumplir con tres propósitos funcionales: acoplarse a los sensores (encoders), sujetar un piñón con posibilidad de desplazamiento longitudinal para facilitar su alineación con el piñón del motor, y proporcionar el soporte necesario para fijar las ruedas neumáticas de tracción en el exterior del chasis.

La distribución del conjunto de perfiles internos se configuró no solo para soportar las ruedas locas, sino también para sostener de manera segura el gabinete de control y el banco de baterías. Finalmente, en la parte frontal superior se ubicó el sensor LiDAR, alineado con el centro de masa del robot, en conjunto con un mecanismo articulado de dos grados de libertad (pan-tilt) dedicado al control de movimiento de la cámara.

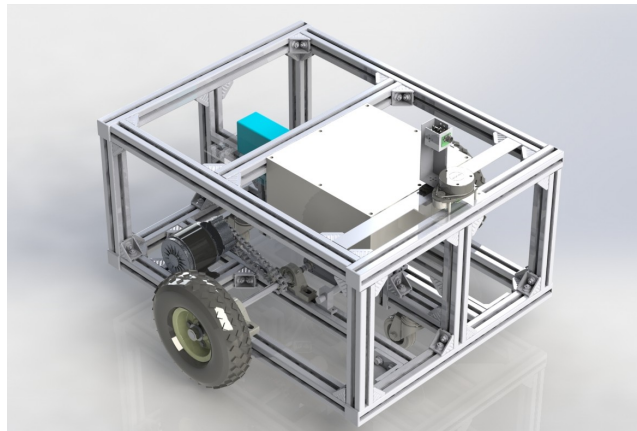


Figura 11: Configuración, renderizado isométrico de Robot Móvil Diferencial en Solid-Works.

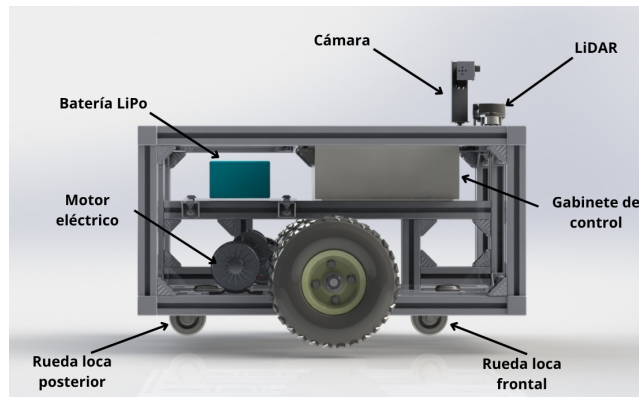


Figura 12: Configuración, renderizado lateral de Robot Móvil Diferencial en SolidWorks.

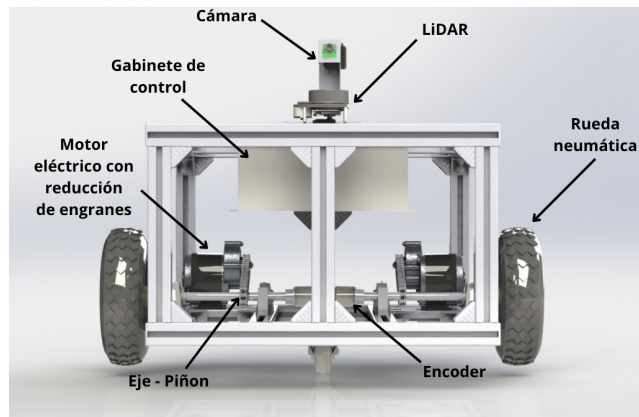


Figura 13: Configuración, renderizado frontal de Robot Móvil Diferencial en SolidWorks.

El Robot móvil diferencial fue diseñado con las siguientes dimensiones generales indicadas en la Fig. 14. Estas dimensiones son consideradas en los algoritmos de evasión de obstáculos (Local Path Planning) y en la estimación de posición del robot (Odometría).

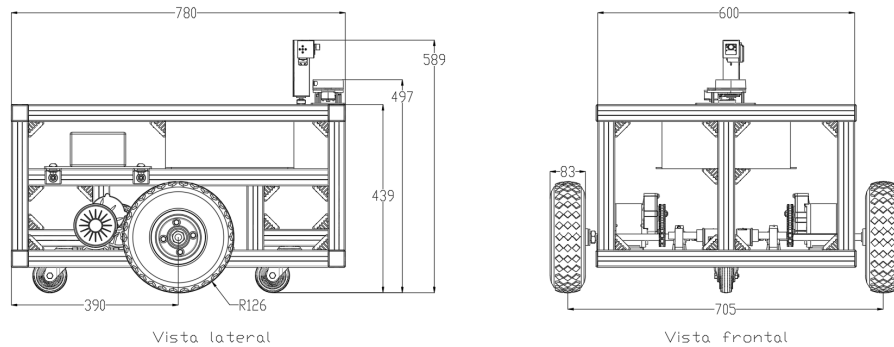


Figura 14: Arreglo dimensional del Robot Móvil Diferencial.

4.1.2 Arquitectura electrónica y sistema de potencia

La integración eléctrica y electrónica parte de un sistema de potencia con la capacidad suficiente para alimentar los distintos subsistemas del robot. Para ello, el robot móvil diferencial emplea una batería de polímero de litio (LiPo). Este tipo de tecnología es ampliamente utilizada en aplicaciones robóticas debido a que la relación peso-potencia es un factor crítico. Además, al utilizar un electrolito en forma de gel en lugar de un estado líquido, se reduce significativamente el riesgo de derrames y explosiones. Finalmente, estas baterías tienen la capacidad de suministrar altas tasas de descarga, importante para manejar los picos de corriente requeridos por los motores de tracción.

El voltaje nominal de la batería a plena carga es de 29.4 VCD, con una capacidad de 30,000 mAh. Para la distribución de energía, se consideró un sistema de reguladores reductores de voltaje (step-down) encargados de suministrar los voltajes nominales requeridos por el circuito de control de temperatura, el conjunto Raspberry Pi 5 - ESP32 y el sistema de servomotores (pan-tilt). El controlador de potencia de los motores se alimenta directamente con el voltaje nominal de la batería. La Fig. 15 ilustra el sistema de potencia propuesto.

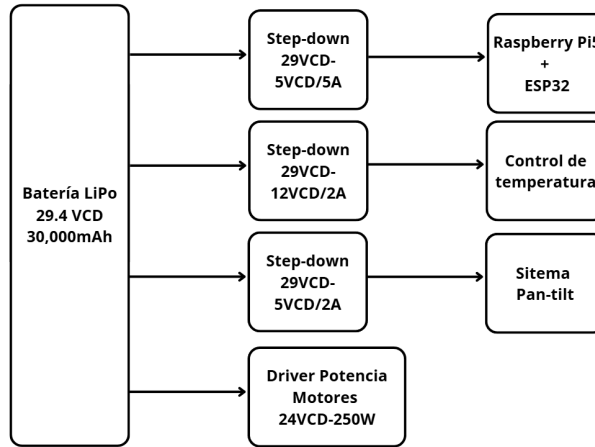


Figura 15: Diagrama del circuito de potencia del Robot Móvil diferencial.

La batería LiPo propuesta provee una energía total de 882 Wh; sin embargo, por seguridad de la misma, para el cálculo de la autonomía del robot se considera únicamente el 80 % de esta capacidad, es decir, 705.6 Wh.

La estimación del consumo energético de la plataforma se realizó bajo las siguientes premisas:

- El consumo teórico de los actuadores de tracción corresponde a la potencia nominal de cada motor de corriente directa (250 W) multiplicada por el número de unidades (2). Tomando en cuenta un desplazamiento a una velocidad constante de 1.5 rad/s, lo cual representa una fracción de la potencia pico, y una masa total que solo incluye la estructura, la batería y los componentes electrónicos, el requerimiento real se estimó en 100 W entre ambos motores.
- La demanda del sistema principal de procesamiento y percepción (Raspberry Pi 5, chip acelerador Hailo 8L, microcontrolador ESP32 y sensor LiDAR) se calculó en 15 W, basándose en el consumo típico de cada dispositivo.
- El gasto energético del mecanismo pan-tilt se fijó en 3 W, contemplando el requerimiento estándar de las dos unidades que lo componen.
- El consumo del sistema de control de temperatura se estimó en 5 W, evaluando la carga del módulo W1209 y el ventilador de 12 VCD asociado.

En conjunto, la demanda total proyectada del robot es de 123 W, lo que se traduce en una autonomía teórica de aproximadamente 5.73 horas bajo condiciones ideales. Cabe destacar que esta aproximación no contempla factores dinámicos como la eficiencia electromecánica de los motores, las pérdidas térmicas en el sistema

de potencia o las fluctuaciones derivadas de variaciones en la velocidad y carga operativa del robot en entornos no estructurados.

La arquitectura electrónica del robot se diseñó con base en los requerimientos funcionales del sistema. Como se observa en la Fig. 16, la unidad de procesamiento principal está conformada por una Raspberry Pi 5 complementada con un chip acelerador de IA Hailo 8L. Este conjunto es el encargado del procesamiento de alto nivel, incluyendo el análisis de datos del sensor RPLIDAR-A1, el procesamiento de la cámara mediante interfaz MIPI CSI-2, y la ejecución de los algoritmos de navegación local (Local Path Planning) y evasión de obstáculos.

Para las tareas de control en tiempo real, la Raspberry Pi 5 se comunica mediante protocolo UART/Serial con un microcontrolador ESP32. Este microcontrolador de bajo nivel es el responsable de leer los sensores encoders (LPD3806 de 1200 p/r) para el cálculo de odometría. Con base en un controlador PI y las referencias de control de velocidad provenientes de la Raspberry Pi 5, el ESP32 genera las señales de control PWM hacia el controlador de motor dual HC-160A S2 para la tracción del robot, así como el control de posición de los servomotores del mecanismo pan-tilt. Adicionalmente, se cuenta con un sistema de control de temperatura (W1209) para gestionar la ventilación del interior del gabinete.

La unidad de procesamiento principal Raspberry Pi 5 tiene las siguientes características: procesador 2.4 GHz Quad-core 64 bits, 8 GB de RAM, y sistema operativo Raspberry Pi OS (basado en Debian Bookworm).

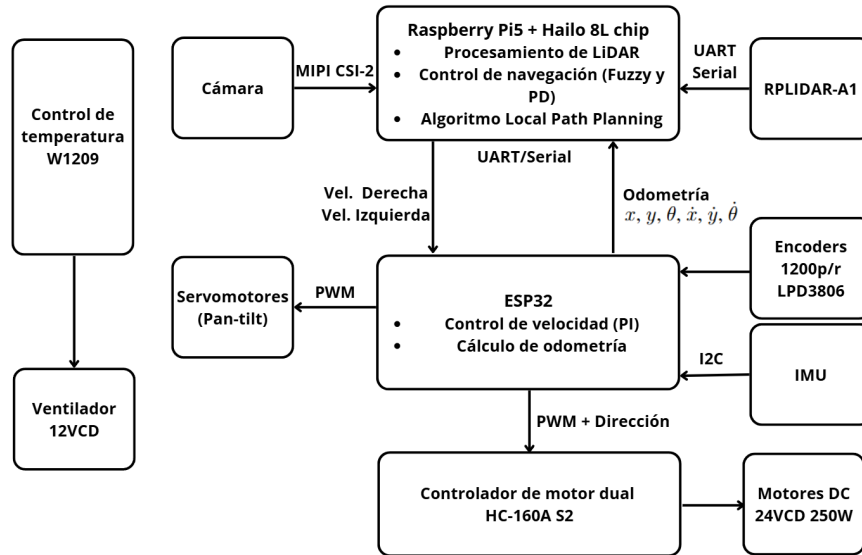


Figura 16: Diagrama de bloques del sistema electrónico y sus componentes principales.

4.2 Algoritmo de generación dinámica de puntos de evasión

En este trabajo, se ha desarrollado un algoritmo de planificación de ruta local (Local Path Planning) basado en la generación de una secuencia continua de puntos entre la posición actual del robot y un punto objetivo final. Dado que el controlador de navegación lleva al robot a una posición o coordenada deseada minimizando la distancia y el error, el algoritmo genera dinámicamente un conjunto de puntos intermedios que permite al robot evitar obstáculos y, posteriormente, alcanzar el punto objetivo final.

El algoritmo sigue una secuencia lógica de pasos para transformar las lecturas del sensor en un punto objetivo inmediato (intermedio o final). El algoritmo se ejecuta cíclicamente y consta de los siguientes pasos.

- **Detección:** Dentro de los datos del LiDAR, se identifican los puntos que indican una posible colisión dentro del radio de detección.
- **Agrupamiento:** Estos puntos de colisión se agrupan para distinguir obstáculos individuales.
- **Validación de segmentos:** Se detectan los segmentos libres (pasillos) entre los obstáculos detectados y se comprueba si son lo suficientemente anchos como para que el robot los atraviese.
- **Generación de puntos de evasión globales:** Se calculan las coordenadas globales de los puntos de evasión dentro de los pasillos seguros.
- **Comprobación de visibilidad del objetivo:** Determina si el punto objetivo final (meta) es directamente accesible sin colisionar.
- **Selección del punto objetivo:** El punto óptimo a seguir (ya sea un punto de evasión o el punto final) se elige mediante una función de coste normalizada que tiene en cuenta la distancia total de la ruta y el cambio de dirección necesario.

A continuación se detalla cada uno de los pasos.

4.2.1 Pre-procesamiento de datos

Como parte del filtrado a las lecturas del sensor LiDAR se ha considerado una corrección a los ángulos, ajustando la forma en que el sensor RPLIDAR-A1 entrega los datos para hacerlos compatibles con el algoritmo. Se busca homogenizar el sistema de coordenadas del sensor LiDAR con el sistema cinemático del robot. El sensor LiDAR entrega un vector de distancias donde el índice $i = 270$ corresponde físicamente al frente del robot, sin embargo el modelo de control requiere que el ángulo 0 rad coincida con el eje X positivo (frente del robot) y que el incremento angular sea antihorario.

Se aplica una transformación lineal y una normalización a los índices fuente del sensor para obtener un ángulo corregido en un intervalo de $[-\pi, \pi)$.

$$\theta_{fuente} = \text{rad}(i), \quad (71)$$

$$\theta_i = \text{norm}(-\theta_{fuente} + \frac{3\pi}{2}), \quad (72)$$

La Fig. 17 muestra el estado inicial de los ángulos, donde el frente del robot corresponde a un ángulo de 270° . La Fig. 18 muestra el ajuste realizado, estableciendo ahora 0 rad en el frente del robot.

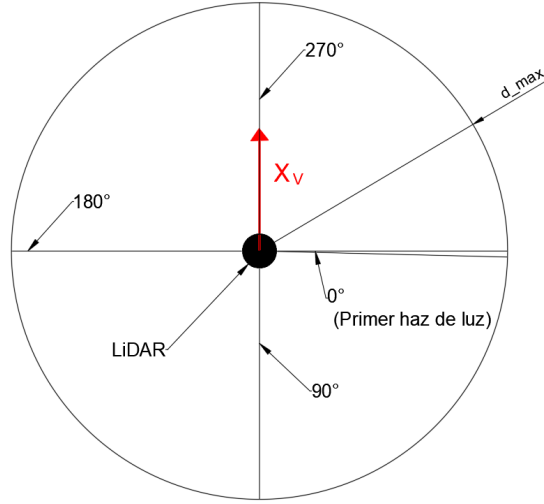


Figura 17: Configuración inicial del sensor LiDAR, 270° al frente del robot.

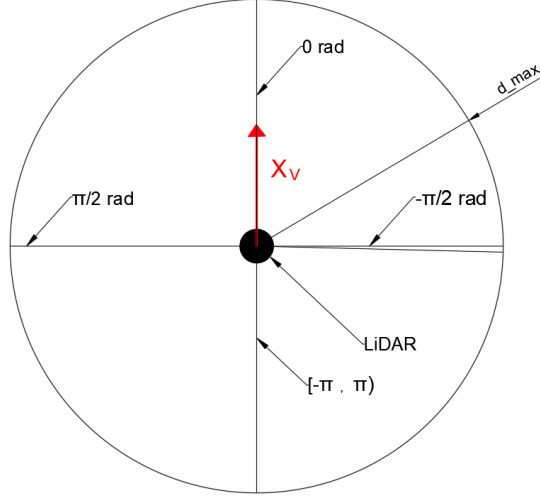


Figura 18: Ajuste, nueva configuración del sensor LiDAR, 0 rad al frente del robot, normalizado a un intervalo $[-\pi, \pi)$.

4.2.2 Detección

El primer paso consiste en procesar el vector de distancias $D = \{d_0, d_1, \dots, d_{359}\}$ obtenido del sensor LiDAR 2D. Dado que el sensor entrega un total de 360 lecturas después del filtro, cada índice i corresponde con su desplazamiento angular en grados dentro del intervalo $[0^\circ - 359^\circ]$.

Se identifican los índices correspondientes a lecturas que representan un obstáculo dentro de un radio de detección ajustable d_{max} . El conjunto de índices de colisión, $V_{colision}$, se define como en (73).

$$V_{colision} = \{i \in \{0, \dots, 359\} \mid 1 < d_i \leq d_{max}\}, \quad (73)$$

Donde d_i es la distancia medida en el índice angular i . Si $V_{colision}$ está vacío, no se detectan obstáculos cercanos y el robot procede directamente hacia el punto objetivo final.

4.2.3 Agrupamiento

Si $V_{colision}$ no está vacío, sus elementos (índices) se ordenan de forma ascendente. Luego, se realiza una agrupación (clustering) basada en la proximidad angular para diferenciar obstáculos individuales. Se define un umbral de diferencia angular máxima, Δ_{max} . El conjunto $V_{colision}$ se particiona en un conjunto de subconjuntos $O = \{O_1, O_2, \dots, O_n\}$, donde cada O_j representa un obstáculo. Un nuevo obstáculo O_{j+1} se crea, si el índice $v_m \in V_{colision}$ con el índice anterior v_{m-1} es mayor que el umbral en (74).

$$v_m - v_{m-1} > \Delta_{max}. \quad (74)$$

El resultado de este paso genera una lista de obstáculos detectados.

4.2.4 Validación de segmentos

Este paso tiene como objetivo obtener los pasillos (segmentos) que sean suficientemente amplios para que el robot pueda pasar de manera segura.

Se identifican los bordes de cada obstáculo O_j . El índice inicial, lado derecho del obstáculo es $i_{R_j} = \min(O_j)$ y el índice final, lado izquierdo del obstáculo es $i_{L_j} = \max(O_j)$. Se calculan los segmentos rectilíneos S_j que conectan el borde final de un obstáculo O_j con el borde inicial del siguiente obstáculo O_{j+1} (incluyendo el segmento de cierre entre O_n y O_1) cf. Fig. 19. La longitud de cada segmento se calcula mediante la Ley de Cosenos como en (75).

$$S_j = \sqrt{d_{i_{L_j}}^2 + d_{i_{R_{j+1}}}^2 - 2d_{i_{L_j}}d_{i_{R_{j+1}}}\cos(\gamma_j)}, \quad (75)$$

Donde d_i es la distancia medida en el índice i , y γ_j es el ángulo absoluto entre los rayos i_{L_j} e $i_{R_{j+1}}$.

Un segmento se considera navegable solo si su longitud S_j es mayor o igual que el ancho mínimo requerido por el robot como en (76).

$$C_{min} = L + 2\epsilon. \quad (76)$$

El resultado es un conjunto de pares de índices como en (77), donde cada par representa los bordes de un pasillo seguro para pasar.

$$S_{valido} = \{(i_{L_j}, i_{R_{j+1}}) \mid S_j \geq C_{min}\}. \quad (77)$$

4.2.5 Cálculo de puntos de evasión globales

Para cada pasillo válido $(i_L, i_R) \in S_{validos}$, se calculan dos puntos de evasión en coordenadas globales (P_L, P_R) . Estos puntos se sitúan a una distancia lateral $d_{IntPoint} = L/2 + \epsilon$ de los bordes del obstáculo. La posición de cada punto (P_x, P_y) se obtiene mediante una transformación de coordenadas que considera la posición actual del robot (x_c, y_c, θ_c) , la distancia del LiDAR al eje del robot d_{Lidar} , la distancia medida d_i y el ángulo α_i medido desde la horizontal. Calculamos la distancia al punto intermedio h_i y el ángulo desde la lateral β_i como en (78), (79), cf. Fig. 20.

$$h_i = \sqrt{d_i^2 + d_{IntPoint}^2}, \quad (78)$$

$$\beta_i = \arctan 2(d_{IntPoint}, d_i). \quad (79)$$

Las coordenadas globales se calculan como en (80).

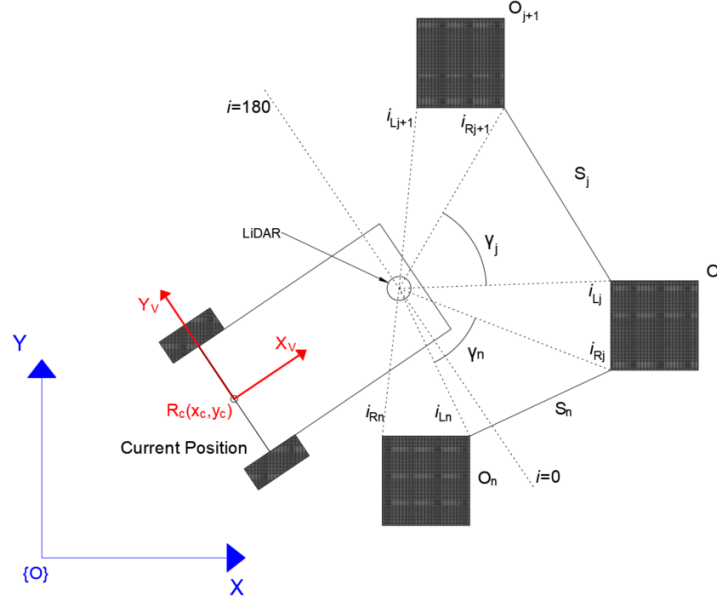


Figura 19: Localización de índices laterales y detección de segmentos entre obstáculos.

$$\begin{aligned}
 P_{L_x} &= x_c + d_{Lidar} \cos(\theta_c) + h_{i_L} \cos(\theta_c + \alpha_{i_L} + \beta_{i_L}), \\
 P_{L_y} &= y_c + d_{Lidar} \sin(\theta_c) + h_{i_L} \sin(\theta_c + \alpha_{i_L} + \beta_{i_L}), \\
 P_{R_x} &= x_c + d_{Lidar} \cos(\theta_c) + h_{i_R} \cos(\theta_c + \alpha_{i_R} - \beta_{i_R}), \\
 P_{R_y} &= y_c + d_{Lidar} \sin(\theta_c) + h_{i_R} \sin(\theta_c + \alpha_{i_R} - \beta_{i_R}).
 \end{aligned} \tag{80}$$

El resultado es un conjunto de pares de puntos de evasión como en (81).

$$P_{evasion} = \{(P_{L_j}, P_{R_j}) \mid \forall (i_{L_j}, i_{R_j}) \in S_{validos}\} \tag{81}$$

4.2.6 Comprobación de visibilidad del objetivo

Antes de seleccionar el punto a seguir, se verifica si el punto objetivo final $P_g = (x_g, y_g)$ es directamente accesible. El punto objetivo final se considera visible si cumple una de estas dos condiciones:

- Está más cerca que cualquier obstáculo detectado:
 $d(P_c, P_g) < \min(d_i), \quad i \in V_{colision}.$
- Angularmente, se encuentra dentro de uno de los pasillos navegables definidos por los puntos de evasión P_L, P_R cf. Fig. 21.

La condición dos se logra haciendo una rotación de ángulos respecto a un nuevo eje 0, trazado desde el punto P_c al P_L . P_g (visible) si $0 < \alpha_g < \alpha_{P_R}$

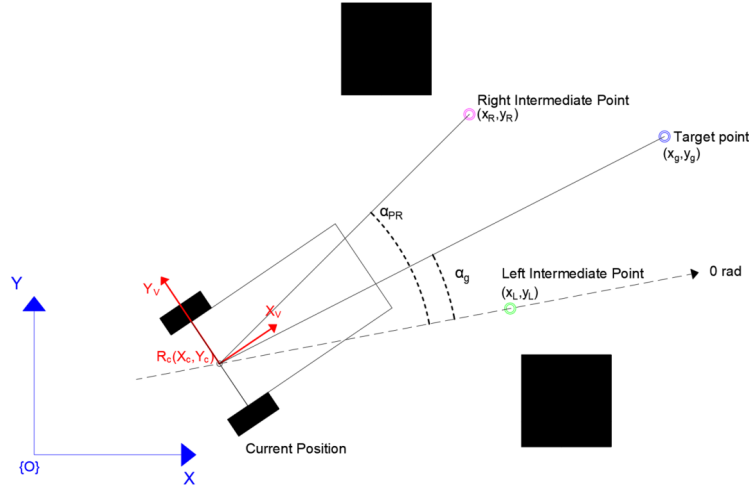


Figura 21: Validación si un punto objetivo está angularmente dentro de uno de los pasillos, a partir de una rotación de ángulos desde un nuevo eje 0.

Las constantes de normalización empleadas en los denominadores de las Ecuaciones (84) y (85) fueron determinadas a partir de los límites físicos y geométricos del sistema, y validadas experimentalmente para mantener los valores de costo en un rango ideal de $[0, 1]$.

Por un lado, el divisor π en $J_{angle_norm}(P)$ representa el límite superior determinado del error de orientación, dado que la máxima desviación angular absoluta que el robot puede tener respecto a un punto candidato es de 180 grados (π radianes).

Por otro lado, el término $2d_{max}$ en $J_{dist_norm}(P)$ actúa como una cota superior heurística para la distancia local. Este valor asume el escenario donde tanto el punto de evasión P como el objetivo P_g se encuentran en el límite del horizonte de percepción del LiDAR.

Cabe destacar que esta estrategia de normalización espacial introduce una restricción al algoritmo: si la distancia total al P_g se encuentra a una distancia significativamente mayor que $2d_{max}$, la función J_{dist_norm} arrojará valores superiores a 1, descompensando el equilibrio de los pesos w_{dist} y w_{angle} . Por consiguiente, $2d_{max}$ corresponde a un parámetro de diseño que se debe de ajustar para garantizar así la estabilidad métrica de la función de costo.

El punto objetivo final P_{target} es aquel que minimiza la función de costo como en (86). Esta función de costo se diferencia de los métodos VPH y VPH+ ya que mientras VPH optimiza casi exclusivamente el costo angular (una comparación entre la dirección al objetivo y la dirección actual), y VPH+ optimiza la distancia de seguridad. Nuestro enfoque optimiza la distancia Euclidiana total de la ruta. Además, el uso de costos normalizados asegura que los pesos w_{dist} y w_{angle} proporcionen un equilibrio entre la búsqueda de la distancia mínima y la penalización por giros bruscos, un comportamiento que es difícil de calibrar en

los métodos VPH anteriores.

$$P_{target} = \arg \min J(P), \quad P \in C. \quad (86)$$

Este punto P_{target} se envía al controlador de navegación como el objetivo inmediato.

4.3 Control PD para la navegación de un robot diferencial

Para el control de posición y orientación del robot diferencial se implementó un controlador PD que servirá de base para comparar el desempeño con los controladores inteligentes.

A partir de las relaciones cinemáticas del robot diferencial presentadas en las Ecs. (3) y (6), se define la relación entre la velocidad lineal del robot (V_{com}) y la velocidad angular de corrección (ω_{com}) generada por el lazo de control. Despejando las velocidades lineales individuales de cada rueda (V_R, V_L), se obtiene:

$$V_R = V_{com} + \frac{L}{2}\omega_{com} \quad (87)$$

$$V_L = V_{com} - \frac{L}{2}\omega_{com} \quad (88)$$

donde L representa la distancia entre ruedas en metros. De igual forma, considerando el radio r de las ruedas, las velocidades de referencia angular (ω_R, ω_L) en rad/s requeridas por los actuadores de cada rueda se obtienen mediante:

$$\omega_R = \frac{V_{com} + \frac{L}{2}\omega_{com}}{r} \quad (89)$$

$$\omega_L = \frac{V_{com} - \frac{L}{2}\omega_{com}}{r} \quad (90)$$

donde ω_{com} es la señal de salida del controlador PD, definida por la ley de control:

$$\omega_{com} = k_P e_o + k_D \dot{e}_o \quad (91)$$

Considerando la definición de error de orientación $e_o = \beta - \theta$, se tiene que:

$$e_o = \beta - \theta_c \quad (92)$$

donde β representa el ángulo de la línea de vista calculado desde la postura actual del robot $R_c = (x_c, y_c)$ hacia la postura objetivo $R_g = (x_g, y_g)$, mediante la función de cuatro cuadrantes:

$$\beta = \text{atan2}(y_g - y_c, x_g - x_c) \quad (93)$$

Al derivar analíticamente el error de orientación con respecto al tiempo ($\dot{e}_o = \dot{\beta} - \dot{\omega}$) y sustituir los términos en la ley de control, la acción de control angular ω_{com} se expresa de la siguiente manera:

$$\omega_{com} = k_P (\text{atan2}(y_g - y_c, x_g - x_c) - \theta_c) + k_D \left(\frac{(y_g - y_c)\dot{x}_c - (x_g - x_c)\dot{y}_c}{(x_g - x_c)^2 + (y_g - y_c)^2} - \omega \right) \quad (94)$$

Para garantizar la estabilidad computacional del algoritmo en el sistema embebido y evitar indeterminaciones matemáticas, el controlador considera lo siguiente. Primero, el uso de la función $\text{atan2}(\cdot)$ previene la división por cero cuando $\Delta x = 0$. Segundo, dado que el denominador del término derivativo representa la distancia al cuadrado hacia la meta, existe un riesgo de división por cero cuando el robot alcanza la coordenada (x_g, y_g) . Para mitigar esto, se implementa un radio de tolerancia: la Ec. (94) deja de evaluarse y la acción de control se anula ($\omega_{com} = 0$) cuando la distancia euclidiana al objetivo es menor a un radio de tolerancia programado ϵ .

Como se observa en la Ecs. (87) y (88), el sistema busca minimizar el error de orientación mientras mantiene un avance constante, lo que permite al robot navegar de forma dirigida hacia su objetivo. Para asegurar la detención del móvil, se implementa una condición de paro donde la velocidad lineal V_{com} se hace nula cuando el error de distancia $e_d \approx 0$.

Una vez que el robot alcanza la posición objetivo R_g , se activa una conmutación hacia el controlador PD de orientación pura. Este esquema busca alinear la estructura del robot con una orientación final deseada β_g . En esta fase, dado que el robot solo debe realizar una rotación sobre su propio eje, la velocidad de avance se define como $V_{com} = 0$, resultando en las siguientes expresiones:

$$V_R = -\omega_{com} \quad (95)$$

$$V_L = \omega_{com} \quad (96)$$

En este estado, el error de orientación se define nuevamente como en (92), donde θ es la orientación actual y β_g es el ángulo objetivo constante. Dado que la derivada de una constante es cero, la señal de control ω_{com} se simplifica a:

$$\omega_{com} = k_P(\beta_g - \theta) - k_D\omega \quad (97)$$

4.3.1 Simulación del controlador PD

Se desarrolló un entorno de simulación en Simulink para validar el controlador propuesto analíticamente antes de su implementación en la plataforma experimental. A continuación, se describen los diferentes bloques de programación y los resultados obtenidos en las pruebas de control.

La Figura 22 ilustra el diagrama general de simulación en Simulink para el controlador PD.

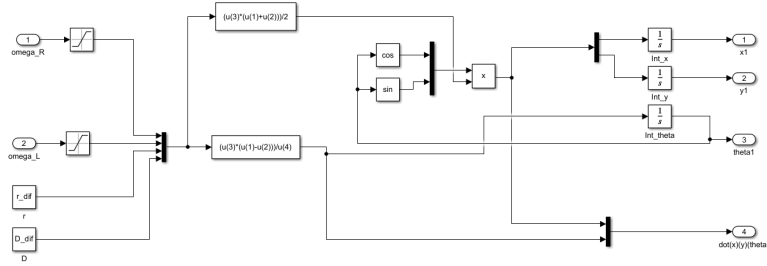


Figura 25: Bloque de simulación de la cinemática del robot diferencial.

Como parte del análisis, se diseñó una interfaz en *App Designer* para visualizar el desplazamiento en el plano XY y la corrección de orientación (*Heading*). La Figura 26 muestra los resultados de una prueba configurada con los siguientes parámetros: punto de inicio $R_i = (1, 1)$ m, punto objetivo $R_g = (5, 5)$ m y un ángulo de orientación inicial y objetivo de π rad. Asimismo, las ganancias del controlador de posición se ajustaron en $K_P = 1,0$ y $K_D = 0,5$, mientras que para el controlador de orientación se definieron en $K_P = 1,0$ y $K_D = 0,5$. El robot simulado cuenta con un radio de rueda de 10 cm y una distancia entre ruedas de 20 cm.

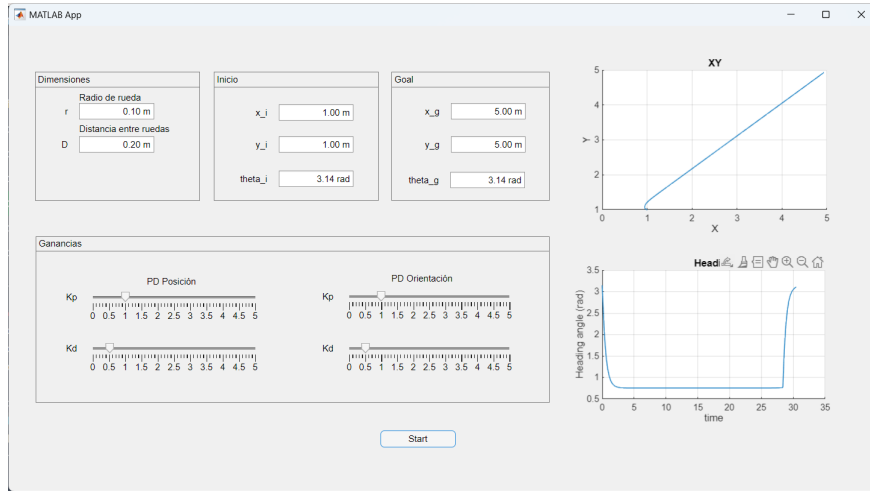


Figura 26: Interfaz de resultados de simulación en *App Designer*.

En la Figura 26 se observa, en la sección derecha, la trayectoria del robot desde el origen hasta el punto final y la evolución de su orientación en el tiempo. Para esta prueba inicial, el desempeño es óptimo: no se presentan oscilaciones y la corrección del error es casi inmediata. Cabe destacar que este comportamiento ideal se debe principalmente a que el controlador se aplica sobre un modelo puramente cinemático, sin considerar perturbaciones externas

o variables dinámicas complejas del robot real.

4.4 Control inteligente FIS Mamdani

A continuación, se presenta el diseño de dos Sistemas de Inferencia Difusa (FIS, por sus siglas en inglés) tipo Mamdani para el control de navegación de un robot diferencial. El primero se encarga de dirigir al robot hacia un punto objetivo, minimizando simultáneamente los errores de posición y orientación. El segundo FIS tiene la función de reorientar al robot una vez que ha alcanzado el punto objetivo, minimizando el error de orientación mediante un giro sobre su propio eje. Es importante destacar que el diseño de ambos controladores se realizó con base en la experiencia empírica obtenida del controlador PD implementado previamente en la plataforma experimental. Esto permitió establecer una base de conocimiento inicial para el diseño de las reglas difusas.

4.4.1 FIS Mamdani para navegación

Para el primer FIS Mamdani, enfocado en la navegación, se definen como variables lingüísticas de entrada el error de posición y el error de orientación, cada una particionada en cinco conjuntos difusos. Para la variable lingüística error de posición, se proponen los valores “cero”, “cerca”, “media”, “lejos” y “muy lejos”, definidos en un universo de discurso de $[0, 700]$ cm. Aun cuando el límite superior del universo de discurso se estableció en 700 cm acorde a las dimensiones del laboratorio, la parametrización de las funciones de pertenencia facilita su extensión. Al extender los parámetros c y d de la función trapezoidal correspondiente al límite superior U_{max} , el sistema conserva la dinámica esperada, manteniendo la validez del control en entornos de mayor tamaño. La Figura 27 ilustra estas funciones de membresía. Como se observa, las funciones triangulares correspondientes a “cero”, “cerca”, “media” y “lejos” se concentran en la primera mitad del universo de discurso, lo que permite una respuesta de control más gradual conforme el robot se aproxima a la meta. Por consiguiente, la función “muy lejos” abarca la segunda mitad, garantizando una respuesta de aproximación rápida ante errores de posición grandes.

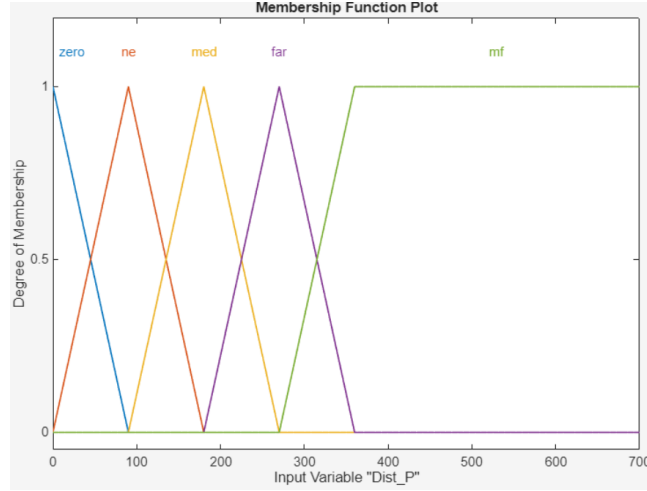


Figura 27: Funciones de membresía, variable lingüística error de posición.

Por otro lado, la variable lingüística error de orientación está compuesta por los valores “muy negativo”, “negativo”, “cero”, “positivo” y “muy positivo”, evaluados en el universo de discurso $[-\pi, \pi)$ rad. La Figura 28 presenta la distribución de estas funciones de membresía. Se puede observar que los conjuntos “muy negativo” y “muy positivo” dominan la mayor parte de los extremos del universo de discurso; esta configuración se diseñó para provocar una respuesta rápida de evasión o corrección ante errores de orientación superiores a 0.8 rad. Además, en la región central, la distribución de las funciones “cero”, “negativo” y “positivo” proporciona una acción de control gradual, mitigando las oscilaciones y generando un movimiento suave durante la alineación final del robot.

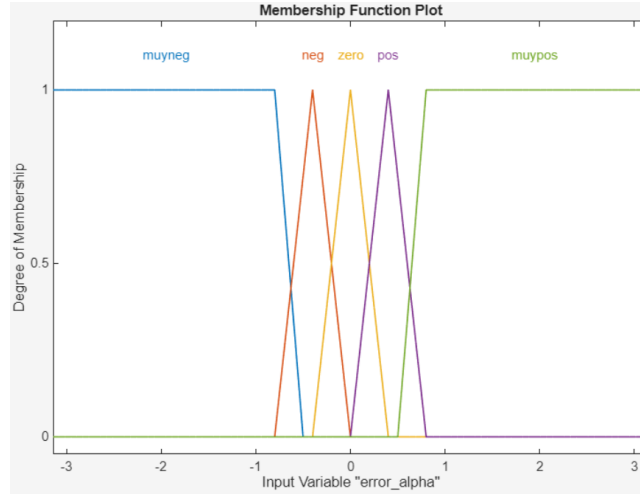


Figura 28: Funciones de membresía, variable lingüística error de orientación.

Las variables de salida del controlador difuso corresponden a las velocidades angulares de control de la rueda derecha (w_r) y de la rueda izquierda (w_l). Para ambas variables de actuación, se establecieron cinco funciones de membresía distribuidas en un universo de discurso de $[0, 4]$ rad/s. Las Figuras 29 y 30 muestran esta parametrización, donde se aprecian los valores lingüísticos “muy lento”, “lento”, “media”, “rápido” y “muy rápido”, representados mediante funciones triangulares simétricas para garantizar una respuesta balanceada en ambos actuadores.

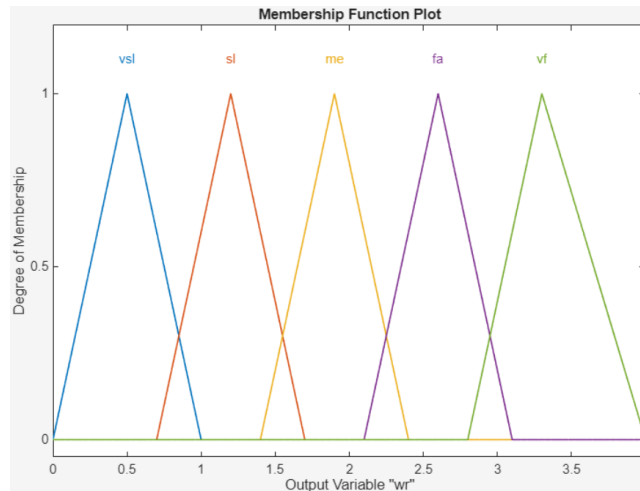


Figura 29: Funciones de membresía, variable lingüística velocidad de control de la rueda derecha.

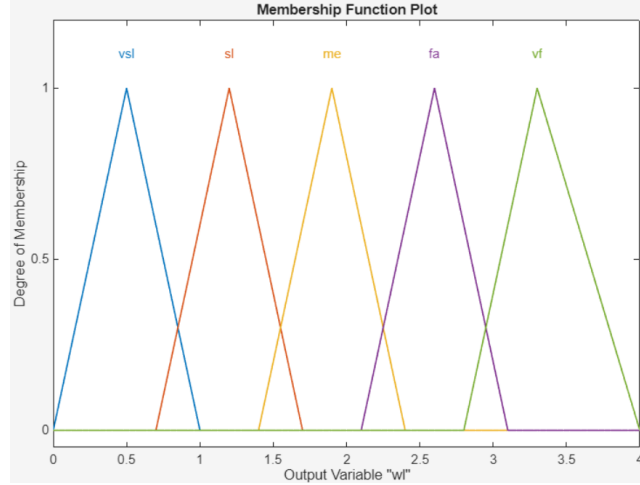


Figura 30: Funciones de membresía, variable lingüística velocidad de control de la rueda izquierda.

Finalmente, la base de reglas se estructuró empleando el método de partición de cuadrícula (*Grid Partition*). Al cruzar los cinco conjuntos de ambas variables de entrada, se genera una matriz completa de 25 reglas difusas, cantidad suficiente para modelar adecuadamente la superficie de control. Estas reglas, detalladas en la Tabla 2, fueron sintonizadas heurísticamente, aprovechando el conocimiento dinámico extraído de la fase de experimentación realizada en esta tesis, como parte fundamental para establecer el control difuso tipo Mamdani.

$e_\theta \setminus e_P$	zero	ne	med	far	mf
muyneg	(vsl, me)	(vsl, me)	(sl, fa)	(me, vf)	(me, vf)
neg	(vsl, sl)	(vsl, sl)	(sl, me)	(me, fa)	(fa, vf)
zero	(vsl, vsl)	(sl, sl)	(me, me)	(fa, fa)	(vf, vf)
pos	(sl, vsl)	(sl, vsl)	(me, sl)	(fa, me)	(vf, fa)
muypos	(me, vsl)	(me, vsl)	(fa, sl)	(vf, me)	(vf, me)

Tabla 2: Base de conocimiento para FIS de navegación, cada celda muestra (w_r, w_l) .

Las Figuras 31 y 32 muestran las superficies de control de la rueda derecha e izquierda respectivamente. En ambas superficies se puede observar que para errores de posición grandes, el sistema asigna velocidades de control altas, mientras que a medida que el error de posición disminuye, las velocidades de control se reducen gradualmente a partir de los 350 cm. Además, para errores de orientación negativos (giro a la derecha), la rueda derecha tiende a recibir velocidades más bajas en comparación con la rueda izquierda, lo que induce un giro hacia la derecha. Por el contrario, para errores de orientación positivos (giro a la izquierda), la rueda izquierda recibe velocidades más bajas, promoviendo un

giro hacia la izquierda. Esta configuración garantiza una respuesta de control coherente con el objetivo de minimizar tanto el error de posición como el error de orientación durante la navegación del robot diferencial.

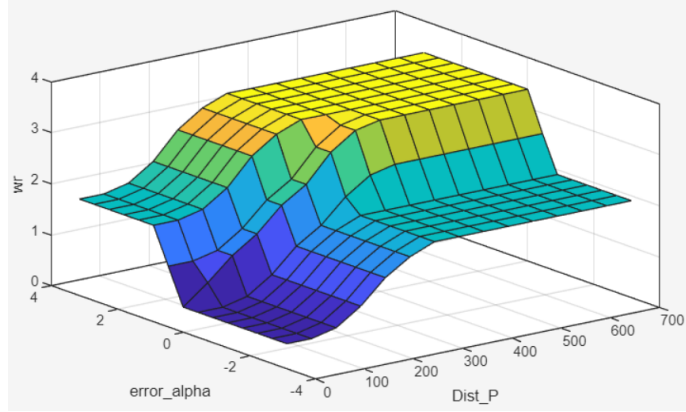


Figura 31: Superficie de control para la rueda derecha.

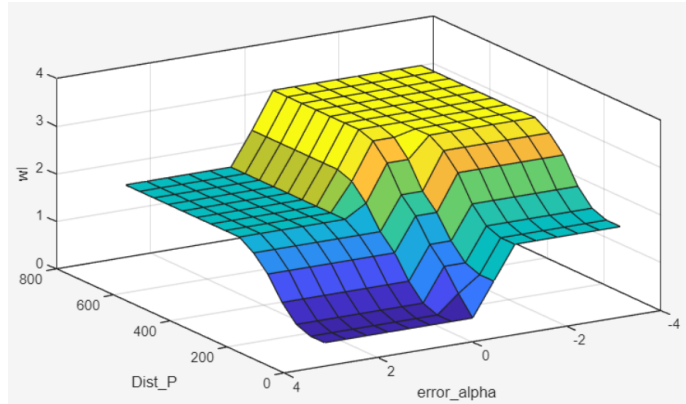


Figura 32: Superficie de control para rueda izquierda.

4.4.2 FIS Mamdani para reorientación

El segundo FIS tipo Mamdani, encargado de reorientar al robot sobre su propio eje, toma como única variable de entrada el error de orientación. La Figura 33 ilustra las funciones de membresía para dicha variable en un universo de discurso $[-\pi, \pi)$. Al igual que en el FIS de navegación, se emplean los valores lingüísticos “muy negativo”, “negativo”, “cero”, “positivo” y “muy positivo”. No obstante, en esta etapa, las funciones de membresía se distribuyen de manera uniforme a lo largo del universo de discurso. Esta configuración se diseñó para garantizar una respuesta de control equilibrada y lineal durante el proceso de reorientación,

permitiendo que el robot realice ajustes precisos en su orientación sin inducir movimientos bruscos u oscilaciones.

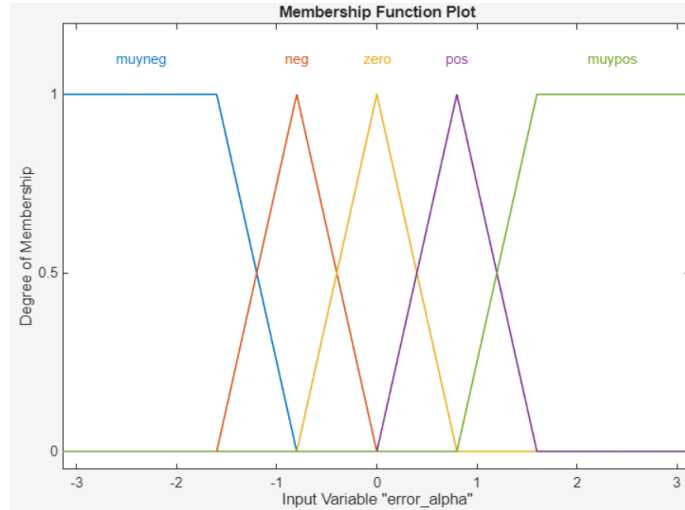


Figura 33: Funciones de membresía para la variable error de orientación en reorientación.

Las variables de salida para este controlador corresponden a las velocidades angulares de control de la rueda derecha (w_r) y de la rueda izquierda (w_l). En este caso, el universo de discurso se establece en $[-2, 2]$ rad/s, dado que el objetivo es permitir que el robot gire sobre su propio eje central. Los valores lingüísticos se definen como “negativo rápido”, “negativo lento”, “cero”, “lento” y “rápido”. Las Figuras 34 y 35 muestran la parametrización de estas funciones para las ruedas derecha e izquierda, respectivamente.

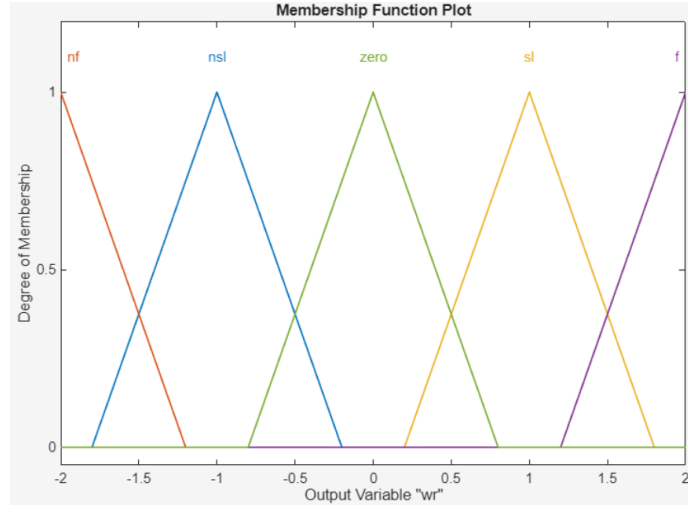


Figura 34: Funciones de membresía para la velocidad de control de la rueda derecha (w_r).

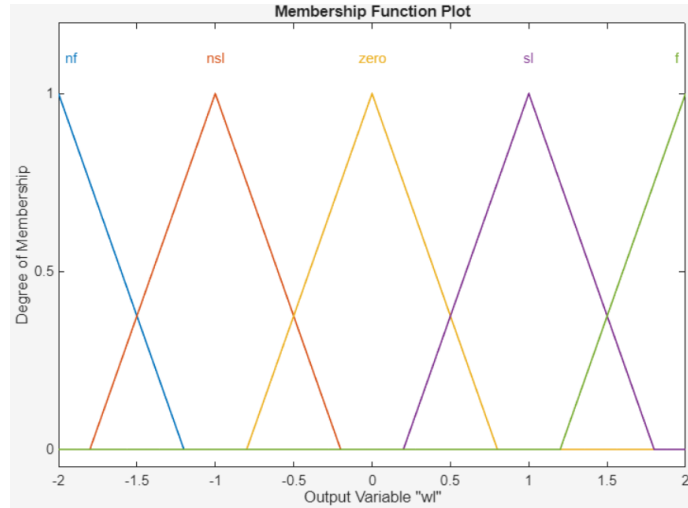


Figura 35: Funciones de membresía para la velocidad de control de la rueda izquierda (w_l).

La Tabla 3 detalla las cinco reglas difusas empleadas en este sistema. El conjunto de reglas permite asignar velocidades de signo contrario a las ruedas ante errores de orientación distintos de cero, induciendo así el giro necesario sobre el eje del robot.

Las Figuras 36 y 37 muestran las superficies de control para las velocidades de las ruedas derecha e izquierda, respectivamente. Aunque estas superficies

e_θ	(w_r, w_l)
muyneg	(nf, f)
neg	(nsl, sl)
zero	(zero, zero)
pos	(sl, nsl)
muypos	(f, nf)

Tabla 3: Base de conocimiento para FIS de reorientación. Cada celda muestra (w_r, w_l) .

son similares a las del FIS de navegación, presentan una diferencia en la lógica de actuación: para errores de orientación negativos, la rueda derecha recibe velocidades negativas mientras que la izquierda recibe velocidades positivas, generando un giro hacia la derecha. Inversamente, ante errores positivos, la rueda derecha recibe velocidades positivas y la izquierda negativas, generando un giro hacia la izquierda.

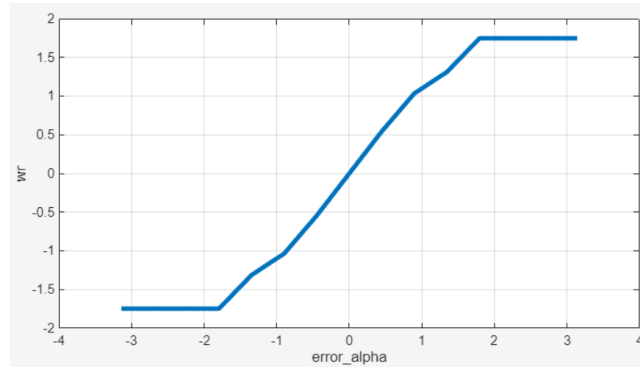


Figura 36: Superficie de control del FIS de reorientación para la rueda derecha (w_r).

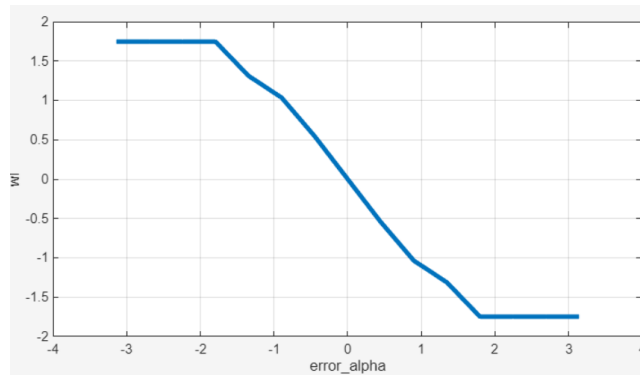


Figura 37: Superficie de control del FIS de reorientación para la rueda izquierda (w_l).

4.5 Control inteligente FIS Sugeno

El controlador difuso tipo Takagi-Sugeno de primer orden presenta ventajas significativas computacionales en comparación con el método de Mamdani, lo cual lo convierte en una alternativa óptima para su implementación en sistemas embarcados con recursos limitados, como la plataforma Raspberry Pi 5. Mediante la integración de esta arquitectura, se busca reducir el tiempo de procesamiento en la etapa de inferencia durante cada iteración del ciclo de control en la plataforma experimental. Esto debido principalmente a que el algoritmo FIS Mamdani requiere de una etapa de agregación y defucificación basada en cálculo de áreas y centroides, mientras que FIS Sugeno corresponde solo a un producto ponderado. Para determinar tanto los parámetros de los conjuntos difusos antecedentes como las funciones lineales de los consecuentes del modelo Sugeno, se optó por utilizar un conjunto de datos generado mediante simulación numérica. Específicamente, se tomó como base el modelo FIS Mamdani previamente sintonizado de manera empírica en la fase de experimentación con la plataforma. Utilizando el entorno de MATLAB, se extrajeron los datos de entrada y salida correspondientes: para cada combinación posible entre el error de posición y el error de orientación, se registraron las velocidades angulares de salida requeridas en las ruedas derecha e izquierda.

Como resultado de este proceso de entrenamiento, se logró replicar el comportamiento dinámico del controlador Mamdani original dentro de una arquitectura Sugeno de primer orden. Las Tablas 4 y 5 muestran los parámetros numéricos resultantes de las funciones lineales consecuentes para cada regla, correspondientes a la velocidad de control de la rueda derecha (w_r) y de la rueda izquierda (w_l), respectivamente. En cada una de las 25 reglas, se especifican los coeficientes a_i , b_i y c_i que definen la superficie lineal del sistema.

Los resultados presentados anteriormente fueron obtenidos mediante la herramienta ANFIS, empleando un método de optimización híbrido que combina la estimación de mínimos cuadrados (*Least Squares Estimation*, LSE) en el paso hacia adelante, con el algoritmo de descenso del gradiente (*Backpropagation*) en el paso hacia atrás. Debido a que la red neuro-difusa se inicializó con los antecedentes sintonizados del modelo Mamdani y fijando los parámetros iniciales de los consecuentes lineales en cero, la tarea de optimización se redujo a un problema de regresión lineal. Como consecuencia de esta simplificación, el sistema logró una convergencia óptima, requiriendo únicamente dos épocas de entrenamiento para estabilizar el error cuadrático medio resultante en 0.020388.

De igual forma se aplicó el mismo proceso de entrenamiento para el FIS de reorientación, obteniendo los parámetros lineales a_i y b_i de cada regla que se muestran en las Tablas 6 y 7 para las ruedas derecha e izquierda, respectivamente.

Regla	$Dist_P$	$error_\alpha$	a_i	b_i	c_i
1	zero	muyneg	-0.0002499	0.0022889	0.51648576
2	zero	neg	0.0028782	0.2829495	0.59474275
3	zero	zero	-0.0018369	-0.7887398	0.55176241
4	zero	pos	-0.0035441	0.5990259	0.98194582
5	zero	muypos	0.0016034	0.0063536	1.89351506
6	med	muyneg	0.0018265	-0.0011445	0.87115226
7	med	neg	0.0010556	0.2518663	1.15495895
8	med	zero	0.0011324	0.1995713	1.67317199
9	med	pos	-0.0003083	0.1933752	1.89748801
10	med	muypos	0.0038493	-0.0002503	1.91398654
11	far	muyneg	0.0010705	0.0042380	1.58903013
12	far	neg	0.0022646	-0.3134823	1.20468337
13	far	zero	0.0016820	-0.2251282	2.12479515
14	far	pos	-0.0006958	-0.3127686	2.96475699
15	far	muypos	0.0009804	0.0049415	3.06001897
16	mf	muyneg	0.0000067	0.0085716	1.91590406
17	mf	neg	-0.0000164	0.4756009	2.85110444
18	mf	zero	-0.0000269	0.1198199	3.34080489
19	mf	pos	0.0000310	-0.0920762	3.39797216
20	mf	muypos	0.0000136	-0.0003543	3.36077077
21	ne	muyneg	0.0009550	0.0003513	0.44615999
22	ne	neg	0.0006370	0.2400335	0.56998887
23	ne	zero	-0.0003040	-0.1893422	1.21094905
24	ne	pos	-0.0003048	-0.0138105	1.25596117
25	ne	muypos	0.0026646	0.0052087	1.67820152

Tabla 4: Parámetros consecuentes calculados para la velocidad de la rueda derecha (w_r) mediante ANFIS.

Regla	$Dist_P$	$error_\alpha$	a_i	b_i	c_i
1	zero	muyneg	0.0014723	-0.0058124	1.8969318
2	zero	neg	-0.0051708	-0.5663535	1.0076959
3	zero	zero	-0.0001522	0.9458293	0.4889281
4	zero	pos	0.0029046	-0.2061864	0.5614589
5	zero	muypos	-0.0003940	-0.0023372	0.5165189
6	med	muyneg	0.0038895	-0.0043545	1.8982554
7	med	neg	0.0005459	-0.2310766	1.7407787
8	med	zero	0.0007471	-0.2133034	1.7380910
9	med	pos	0.0011044	-0.2713497	1.1650490
10	med	muypos	0.0016978	-0.0026770	0.9011719
11	far	muyneg	0.0009735	-0.0032504	3.0655394
12	far	neg	-0.0006317	0.4841832	3.0145689
13	far	zero	0.0013833	0.4344447	2.2017421
14	far	pos	0.0026623	0.3987460	1.0429800
15	far	muypos	0.0010337	-0.0035486	1.5987078
16	mf	muyneg	0.0000093	0.0004252	3.3631136
17	mf	neg	0.0000355	0.0677330	3.3839918
18	mf	zero	-0.0000603	-0.1626280	3.3604311
19	mf	pos	0.0000432	-0.4636061	2.8174957
20	mf	muypos	0.0000052	-0.0091957	1.9182252
21	ne	muyneg	0.0026987	-0.0057354	1.6771853
22	ne	neg	-0.0010772	0.0662981	1.3923367
23	ne	zero	-0.0007051	0.1340043	1.2425687
24	ne	pos	0.0006492	-0.2969128	0.5990574
25	ne	muypos	0.0007791	-0.0010822	0.4621941

Tabla 5: Parámetros consecuentes calculados para la velocidad de la rueda izquierda (w_l) mediante ANFIS.

Regla	error_$_\alpha$	a_i	b_i
1	mneg	-0.0008837	-1.7356746
2	neg	-0.6776203	-1.4822065
3	zero	-0.2816089	-0.0205031
4	pos	-0.7234748	1.5237912
5	mpos	0.0013360	1.7298015

Tabla 6: Parámetros consecuentes de cada regla para Sugeno de reorientación (w_r) obtenidos mediante ANFIS.

Regla	error_ α	a_i	b_i
1	mneg	0.0009296	1.7357888
2	neg	0.6358034	1.4516246
3	zero	0.2458233	0.0224665
4	pos	0.6827539	-1.4911610
5	mpos	-0.0013661	-1.7297220

Tabla 7: Parámetros consecuentes de cada regla para Sugeno de reorientación (w_i) obtenidos mediante ANFIS.

4.6 Análisis de estabilidad de controlador difuso tipo Sugeno

Para este estudio, se ha optado por un enfoque de estabilidad práctica en lugar de un análisis de estabilidad asintótica global, debido a la naturaleza altamente no lineal del sistema y a las restricciones no holonómicas del robot diferencial. El teorema de Brockett [31] establece un conjunto de condiciones matemáticas necesarias para estabilizar sistemas no lineales utilizando retroalimentación continua basada puramente en el estado. Dicho teorema demuestra que muchos sistemas no lineales controlables no pueden ser estabilizados en un punto de equilibrio mediante leyes de control continuas. En particular, los robots de tracción diferencial con restricciones no holonómicas (donde el número de velocidades controlables es menor al número de coordenadas espaciales) no cumplen con la condición de suprayectividad topológica exigida por el teorema de Brockett.

A continuación, se presenta el análisis de estabilidad del controlador difuso tipo Sugeno de primer orden basado en el criterio de estabilidad práctica o Acotamiento Último Uniforme (*Uniform Ultimate Boundedness*, UUB) (cf. [32, Cap. 4]). Este criterio establece que un sistema es UUB si existe una constante $b > 0$ (la cota o el límite) independiente del tiempo inicial, tal que para cualquier condición inicial $\mathbf{x}(t_0)$, existe un tiempo finito $T > 0$ donde la trayectoria del sistema satisface:

$$\|\mathbf{x}(t)\| \leq b, \quad \forall t \geq t_0 + T \quad (98)$$

En otras palabras, sin importar desde qué posición u orientación inicial comience el robot ($\mathbf{x}(t_0)$), después de un tiempo finito T , el sistema entrará invariablemente a una región compacta de radio $b > 0$ alrededor del origen y no volverá a salir. Para demostrar que el sistema controlado por el FIS Sugeno exhibe estabilidad UUB, se utiliza un enfoque basado en una función de Lyapunov. Este método consiste en proponer una función candidata $V(\mathbf{x})$ definida positiva, cuya derivada a lo largo de las trayectorias del sistema sea estrictamente negativa fuera de una bola de radio operativo.

La función de Lyapunov candidata debe ser continua, poseer primeras derivadas parciales continuas y cumplir con la condición de acotamiento:

$$\alpha_1(\|\mathbf{x}\|) \leq V(\mathbf{x}) \leq \alpha_2(\|\mathbf{x}\|) \quad (99)$$

donde α_1 y α_2 son funciones de clase $\mathcal{K}$, es decir, funciones estrictamente crecientes que pasan por cero. La siguiente función de energía cumple con dichas condiciones:

$$V(\mathbf{x}) = \frac{1}{2}e_P^2 + \frac{1}{2}e_\theta^2 \quad (100)$$

donde $\mathbf{x} = [e_P, e_\theta]^T$ representa el vector de estados de error de posición y orientación.

A fin de considerar las restricciones físicas del sistema y evitar la indeterminación matemática de la cinemática en el origen, la dinámica del error se establece como un sistema conmutado que depende del radio de tolerancia de llegada $\epsilon = 0,1$ m:

$$\dot{\mathbf{x}} = \begin{cases} \begin{bmatrix} -\frac{r}{2} \cos(e_\theta)(\omega_R + \omega_L) \\ \left(\frac{r \sin(e_\theta)}{2e_P} - \frac{r}{L}\right) \omega_R + \left(\frac{r \sin(e_\theta)}{2e_P} + \frac{r}{L}\right) \omega_L \end{bmatrix}, & \text{si } e_P > \epsilon \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, & \text{si } e_P \leq \epsilon \end{cases} \quad (101)$$

donde r es el radio de las ruedas, L es la distancia entre ruedas, y ω_R, ω_L son las velocidades de rueda generadas por el FIS Sugeno. Puesto que el sistema opera únicamente cuando $e_P > \epsilon$, el término $2e_P$ se mantiene estrictamente positivo ($2e_P \geq 2\epsilon > 0$), evitando la indeterminación matemática en el origen. Cuando el robot ingresa al radio de tolerancia ($e_P \leq \epsilon$), las acciones de control de navegación se anulan ($\omega_R = \omega_L = 0$), deteniendo la evolución del sistema antes de alcanzar el punto singular $e_P = 0$.

Para garantizar la convergencia, la derivada temporal de la función de Lyapunov debe disipar energía a lo largo de las trayectorias del sistema, satisfaciendo la condición de drenaje fuera de la región de tolerancia μ :

$$\dot{V}(\mathbf{x}) \leq -W_3(\mathbf{x}) < 0 \quad \forall \|\mathbf{x}\| \geq \mu \quad (102)$$

donde $W_3(\mathbf{x})$ es una función definida positiva continua y μ es una constante positiva que representa el radio de tolerancia de llegada a la meta. Aplicando la regla de la cadena, la derivada analítica se define como:

$$\dot{V}(\mathbf{x}) = e_P \cdot \dot{e}_P + e_\theta \cdot \dot{e}_\theta \quad (103)$$

Debido a la complejidad algebraica dada por las 25 reglas del controlador Sugeno, la demostración de la Ecuación (102) se ha efectuado mediante un mapeo numérico en el espacio de fases [33, Cap. 2]. Para cada coordenada (e_P, e_θ) dentro del universo de discurso, el algoritmo evalúa el sistema de inferencia difusa completo para extraer ω_R y ω_L , inyecta estas señales en la cinemática de la planta y evalúa la tasa de cambio de energía $\dot{V}(\mathbf{x})$.

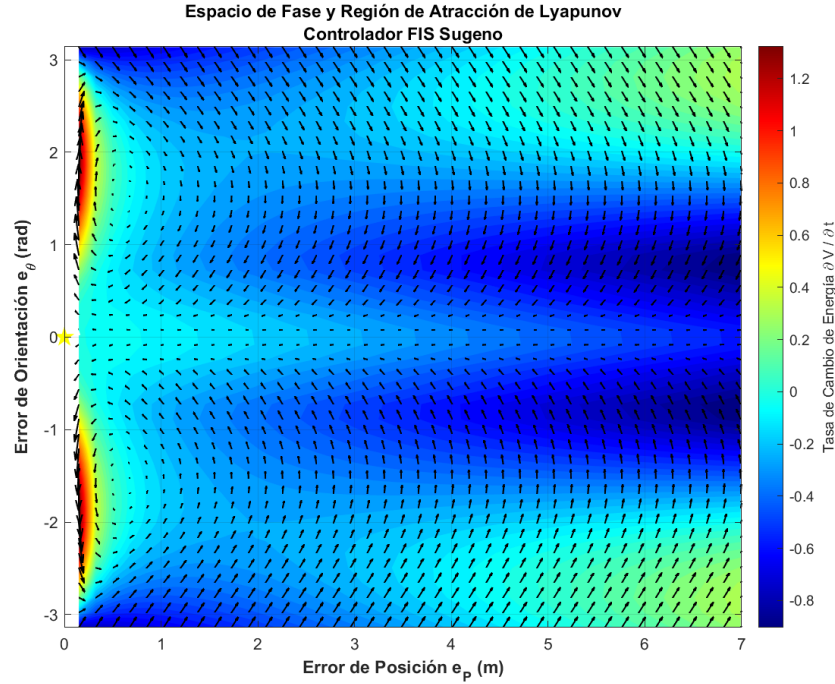


Figura 38: Espacio de fases de la derivada de la función de Lyapunov para las distintas combinaciones de e_P y e_θ dentro del universo de discurso del FIS Sugeno.

El mapeo computacional de la Figura 38 muestra que la derivada de Lyapunov es estrictamente negativa ($\dot{V}(\mathbf{x}) < 0$) en el 85 % del espacio de estados evaluado. El 15 % restante presenta regiones con valores instantáneamente positivos ($\dot{V}(\mathbf{x}) > 0$), las cuales se concentran en cuatro zonas de maniobra transitoria derivadas de las restricciones no holonómicas del robot:

En primer lugar, las regiones rojas donde e_P es muy cercano a cero y $e_\theta \approx \pm\pi/2$. Esta singularidad cinemática ocurre cuando el robot está en la frontera de la meta pero orientado de forma paralela a la misma; la restricción no holonómica obliga al controlador a ejecutar una maniobra rápida de reorientación, alejando el chasis para corregir el ángulo antes de entrar al radio objetivo. En segundo lugar, se observan zonas de menor intensidad positiva cuando e_P es grande con errores cercanos a $\pm\pi$. Esto representa al robot situado de espaldas al objetivo; el impulso de avance del controlador combinado con el giro de alineación provoca que la trayectoria describa un arco, incrementando temporalmente la distancia antes de orientarse hacia el objetivo.

El análisis del espacio de fases en la Figura 38 muestra que no se cumple estrictamente con (102) sin embargo se confirma que estas cuatro regiones no representan puntos de equilibrio inestables, ni zonas de atascamiento. Por el contrario, el campo de vectores demuestra que las trayectorias atraviesan rápidamente estas regiones transitorias de rotación e ingresan de forma contundente a la región dominante donde $\dot{V}(\mathbf{x}) < 0$, garantizando que el estado (e_P, e_θ) converja de

manera estable hacia el radio de tolerancia especificado ($\epsilon = 0,1$ m).

5 Pruebas experimentales

Los resultados experimentales de este trabajo se dividen en dos fases principales. En la primera fase, se evalúa y compara el desempeño cinemático de tres controladores (PD, Difuso Mamdani y Difuso Sugeno Tipo 1) en distintos escenarios de navegación. Esta evaluación se basa en el principio de seguimiento de puntos y analiza los errores de posición y orientación, probando el sistema tanto en espacios libres como en presencia de un solo obstáculo.

Este primer conjunto de resultados fue obtenido mediante la plataforma experimental diseñada y descrita en la Sección 4.1. Asimismo, para estas pruebas, la primera versión del algoritmo de generación dinámica de puntos de evasión fue parametrizada con una distancia máxima de detección (*MaxDist*) de 110 cm y una tolerancia de seguridad entre el obstáculo y el robot (ϵ) de 60 cm.

En la segunda fase, se analiza el comportamiento del algoritmo de planificación de ruta local (*Local Path Planning*) propuesto. Para validar su eficiencia y robustez, se compara su desempeño frente al método clásico VPH (*Vector Polar Histogram*) en un entorno de múltiples obstáculos, empleando la misma plataforma experimental descrita anteriormente.

5.1 Pruebas experimentales para algoritmos de control de navegación

5.1.1 Resultados para el controlador PD

El primer conjunto de pruebas evalúa el desplazamiento del robot móvil diferencial empleando el controlador clásico Proporcional-Derivativo (PD). Para analizar su comportamiento, el sistema fue sometido a cuatro escenarios de navegación: espacio libre (sin obstáculos), evasión de un obstáculo por la derecha, evasión por la izquierda, y navegación entre dos obstáculos.

Las Figuras 39 a 42 ilustran las trayectorias ejecutadas por el robot desde su punto de inicio hasta la coordenada objetivo en cada uno de los escenarios mencionados, mostrando la ruta seguida y los puntos de evasión calculados.

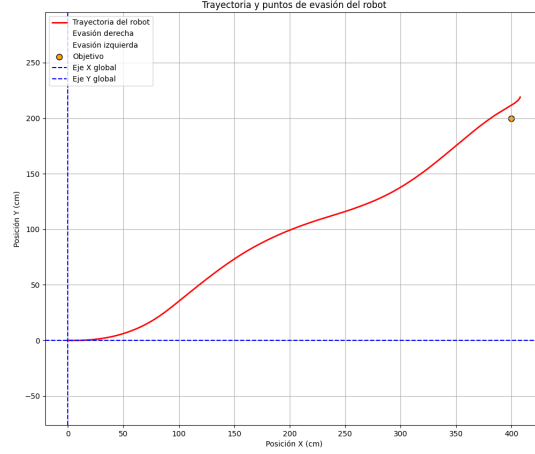


Figura 39: Trayectoria del robot en condición sin obstáculos, controlador PD, punto de inicio (0,0), punto objetivo (400,200).

La Figura 39 muestra el desplazamiento del robot en un entorno libre de obstáculos. El sistema inicia en la coordenada (0, 0) con una orientación de 0 rad, estableciendo su objetivo en la coordenada (400, 200). Para este conjunto de pruebas, el controlador fue parametrizado con las ganancias $k_P = 1,0$ y $k_D = 0,5$. Aunque estos parámetros generan una ligera oscilación durante la navegación en espacios libres, se seleccionaron debido a que, en la práctica, proveen un mejor desempeño para la evasión de obstáculos al proporcionar una respuesta rápida del controlador ante la detección de objetos cercanos.

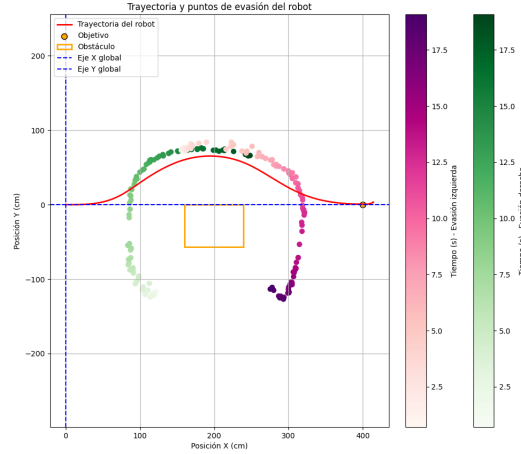


Figura 40: Trayectoria y puntos de evasión del robot en condición obstáculo a la derecha, controlador PD, punto de inicio (0,0), punto objetivo (400,0).

La Figura 40 ilustra al robot móvil diferencial partiendo desde la misma condición inicial (0, 0) con orientación cero hacia un punto objetivo ubicado directamente al frente en la coordenada (400, 0). En este escenario se colocó un obstáculo a la mitad del trayecto, desplazado hacia la derecha del eje de avance, con la finalidad de forzar al robot a realizar una maniobra de evasión por la izquierda. Este recorrido muestra la generación de dos conjuntos de puntos por parte del algoritmo de planificación local: el primer conjunto en color verde corresponde a los puntos de evasión calculados para rodear el obstáculo por la derecha, mientras que el conjunto morado representa la ruta de evasión por la izquierda. En el gráfico, los primeros puntos generados se muestran en un tono claro, oscureciéndose gradualmente conforme avanza el desplazamiento.

Como resultado, se observa que tanto los puntos morados como los verdes terminan encerrando al obstáculo. Para este escenario, el robot persigue la ruta de evasión izquierda; aunque la trayectoria real del chasis no se superpone de manera exacta con los puntos planificados, se aproxima a ellos de forma considerable. Finalmente, en el instante en que el punto objetivo global es visible de manera directa, sin puntos intermedios que interfieran, el robot deja la maniobra de evasión y se dirige directamente hacia su meta.

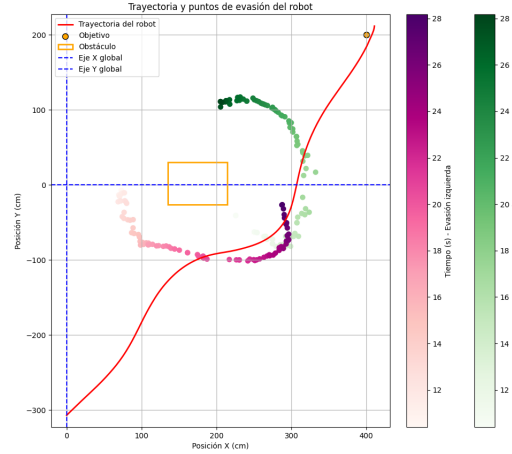


Figura 41: Trayectoria y puntos de evasión del robot en condición obstáculo a la izquierda, controlador PD, punto de inicio (0,-307), orientación inicial $\pi/4$ rad, punto objetivo (400,200).

La Figura 41 muestra la trayectoria del robot en un entorno con un solo obstáculo. El sistema debe recorrer el espacio de prueba en diagonal para alcanzar la coordenada (400,200), partiendo desde la posición (0,-307) con una orientación inicial de $\pi/4$ rad hacia el punto objetivo.

Para esta prueba, el obstáculo se ubicó ligeramente desplazado hacia la izquierda del eje de la trayectoria ideal, obligando al robot a realizar una maniobra de evasión por la derecha. La trayectoria resultante presenta una ligera oscilación al principio del recorrido, comportamiento característico del controlador PD, logrando evadir exitosamente el obstáculo para posteriormente orientarse hacia el objetivo final.

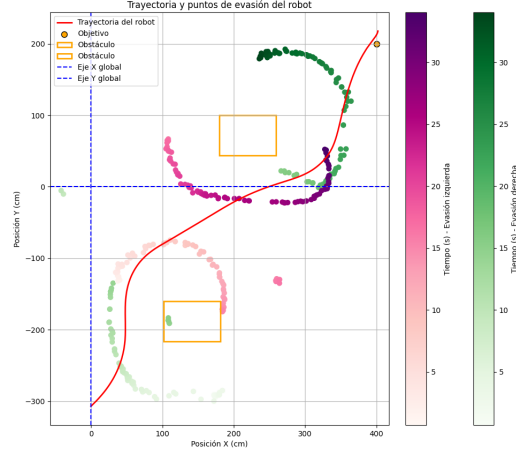


Figura 42: Trayectoria y puntos de evasión del robot en condición dos obstáculos, controlador PD, punto de inicio (0,-307), orientación inicial $\pi/4$ rad, punto objetivo (400,200).

Una prueba de mayor complejidad se presenta en la Figura 42, donde el robot nuevamente debe desplazarse en diagonal sobre el espacio de prueba. Sin embargo, en este escenario se colocaron dos obstáculos estratégicamente para forzar una maniobra de evasión doble (por la izquierda y por la derecha) en un solo recorrido. Los obstáculos mantienen una separación calculada para que, cuando el LiDAR detecte el segundo elemento, el primero ya haya salido de su campo de visión. El robot logra evadir el primer obstáculo por la izquierda; tras superarlo, busca retomar una trayectoria rectilínea hacia el objetivo final. Al detectar el segundo obstáculo, el algoritmo de navegación ejecuta una nueva evasión, esta vez por la derecha.

Es importante observar cómo esta primera versión del algoritmo de evasión (diseñada originalmente para un solo obstáculo) generó puntos intermedios por la izquierda y por la derecha alrededor de las coordenadas (260,-130) y (-40,0), respectivamente. Esto ocurrió debido a que el sensor LiDAR alcanzó a detectar ambos obstáculos simultáneamente durante un par de iteraciones, resolviendo la situación sin afectar significativamente la fluidez del recorrido.

Finalmente, cabe destacar que, durante los recorridos ejecutados bajo el controlador PD, el robot mostró un desplazamiento más allá del punto objetivo. Una vez que el algoritmo de control determinó la detención de los motores, el chasis se desplazó linealmente hasta 15 cm después del punto de frenado. Esto se debe a que la velocidad de avance lineal se mantuvo constante; en consecuencia, a pesar de operar a baja velocidad, la inercia propia del robot provocó este desplazamiento adicional al final de la trayectoria. Una manera de mitigar este comportamiento podría ser incorporar una acción de control sobre el despla-

miento lineal en lugar de mantenerlo constante, logrando un frenado oportuno y suave.

5.1.2 Resultados para el controlador Mamdani

Las Figuras 43 a 46 ilustran el desplazamiento del robot móvil diferencial en los mismos escenarios de navegación: espacio libre, evasión de obstáculo por la derecha, evasión de obstáculo por la izquierda, y navegación entre dos obstáculos. Este conjunto de resultados fue obtenido haciendo uso de la sintonización de funciones de pertenencia antecedentes y consecuentes indicadas para el FIS Mamdani en la Sección 4.4.

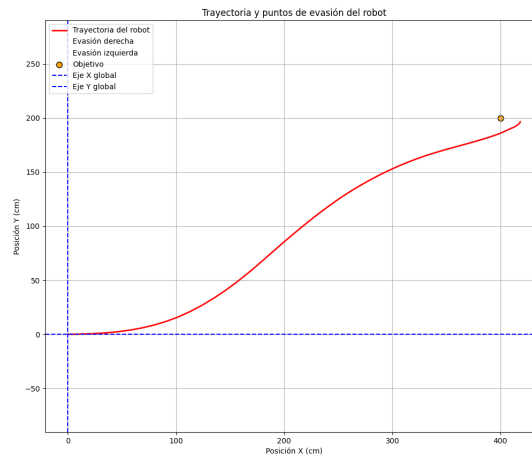


Figura 43: Trayectoria del robot en condición sin obstáculos, controlador Mamdani, punto de inicio (0,0), punto objetivo (400,200).

La Figura 43 muestra el resultado del desplazamiento del robot desde una posición inicial en (0,0) con orientación 0 rad, hacia un punto objetivo en (400,200) en un entorno libre de obstáculos. La trayectoria muestra una oscilación inicial, aunque de menor magnitud en comparación con el controlador PD. Además, se observa que el robot presenta un ligero error al término del recorrido, llegando levemente desplazado a la derecha del punto objetivo exacto.

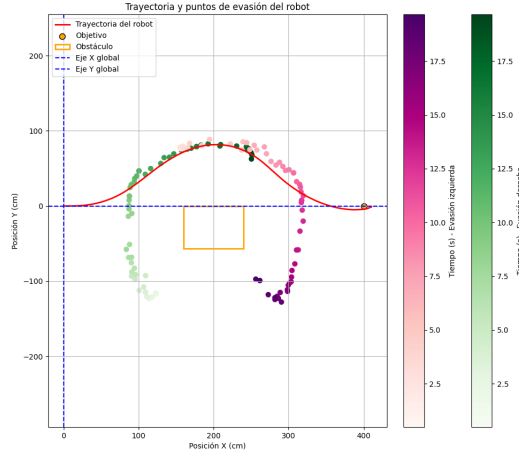


Figura 44: Trayectoria y puntos de evasión del robot en condición obstáculo a la derecha, controlador Mamdani, punto de inicio (0,0), punto objetivo (400,0).

La Figura 44 ilustra el recorrido del robot en un entorno con un obstáculo posicionado a la derecha del eje ideal de trayectoria, forzando una maniobra de evasión por la izquierda. Durante este recorrido, destaca cómo la trayectoria del chasis se superpone con el conjunto de puntos intermedios generados por el algoritmo de evasión, demostrando la alta robustez del controlador Mamdani para el seguimiento de trayectorias locales.

Adicionalmente, se puede observar que la cantidad de puntos intermedios generados en esta prueba es menor en comparación con lo mostrado en la Figura 40 (controlador PD). Este resultado es una consecuencia directa del mayor costo computacional que exige el procesamiento de inferencia y defusificación del FIS Mamdani en cada iteración del sistema embarcado.

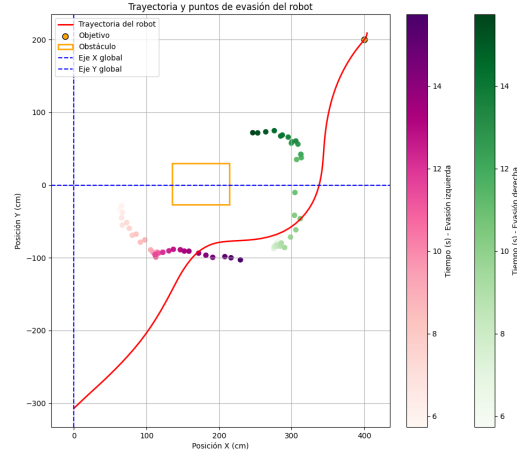


Figura 45: Trayectoria y puntos de evasión del robot en condición obstáculo a la izquierda, controlador Mamdani, punto de inicio (0,-307), punto objetivo (400,200).

En la Figura 45, el robot es posicionado en la coordenada (0,-307) con una orientación inicial de $\pi/4$ rad, con el objetivo en (400,200). El sistema debe recorrer el espacio en diagonal y evadir por la derecha un obstáculo situado a la mitad del trayecto. Nuevamente, se presenta una reducción en la densidad de puntos de evasión debido a la carga computacional del controlador. Sin embargo, a pesar de contar con menos referencias intermedias, el robot logra evadir el obstáculo de manera fluida y alcanzar el objetivo final con un error mínimo.

Es importante resaltar el comportamiento dinámico que entrega el controlador Mamdani en este escenario. El robot se aproxima al obstáculo con mayor rapidez y a una distancia mucho más cercana antes de ejecutar el giro de evasión, en contraste con el recorrido conservador del controlador PD (ver Figura 41). Esto se justifica por la lógica de activación de la base de reglas: al inicio, al no existir puntos intermedios, el algoritmo evalúa el error de posición a más de 700 cm de distancia, dándole un peso mayor a la regla *"Si el error de distancia es Lejos y el error de orientación es Cero, entonces la velocidad es Alta"*. En el instante en que el sensor detecta el obstáculo y se genera un punto de evasión cercano, los pesos de la inferencia difusa se reajustan rápidamente, provocando una desaceleración rápida y un giro por la derecha.

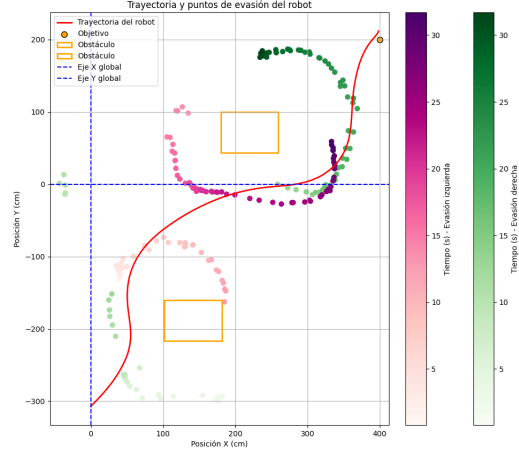


Figura 46: Trayectoria y puntos de evasión del robot en condición dos obstáculos, controlador Mamdani, punto de inicio (0,-307), orientación $\pi/4$ rad, punto objetivo (400,200).

La prueba de mayor complejidad se presenta en la Figura 46, donde los obstáculos están dispuestos estratégicamente para forzar una evasión inicial por la izquierda seguida de una por la derecha. El robot completa la doble evasión con éxito, demostrando que la precisión de seguimiento del FIS Mamdani compensa la menor densidad de puntos de la ruta generada.

Finalmente, destaca la capacidad del controlador difuso para llegar al punto objetivo sin el excedente de desplazamiento que caracterizaba al controlador PD. Dado que el FIS Mamdani evalúa explícitamente la variable lingüística de "Distancia", el sistema reduce progresiva y naturalmente su velocidad lineal a medida que se aproxima al punto objetivo, mitigando casi por completo el desplazamiento excedente provocado por la inercia mecánica.

5.1.3 Resultados para el controlador Sugeno Tipo 1

Las Figuras 47 a 50 ilustran el desplazamiento del robot móvil diferencial en los mismos escenarios de navegación previamente evaluados: espacio libre, evasión de obstáculo por la derecha, evasión por la izquierda, y navegación entre dos obstáculos.

A pesar del excelente comportamiento mostrado por el controlador FIS Mamdani (mostrado en la Sección 5.1.2), su elevado costo computacional por iteración castiga la capacidad de la unidad de procesamiento principal (Raspberry Pi 5) para ejecutar otras tareas de manera óptima, afectando específicamente la frecuencia de actualización del algoritmo de evasión. La consecuencia de no procesar este algoritmo con la velocidad suficiente es el riesgo de generar puntos

ciegos durante la trayectoria, obligando a limitar el sistema a velocidades más bajas para mitigar el riesgo de colisión. La aplicación del controlador FIS Sugeno Tipo 1, sintonizado como se indicó en la Sección 4.5, permitió replicar los resultados de navegación del FIS Mamdani, pero con un consumo de recursos computacionales reducido, que se iguala al de un controlador clásico PD.

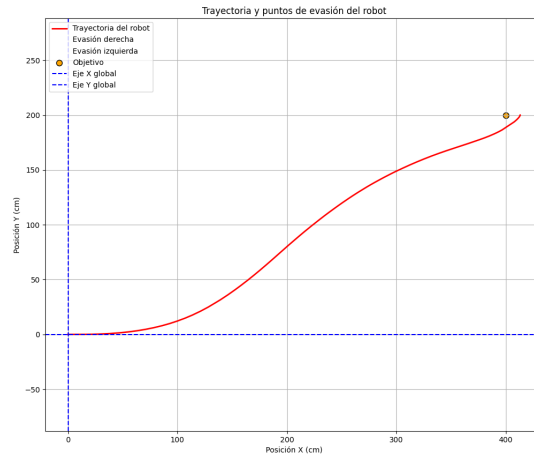


Figura 47: Trayectoria del robot en condición sin obstáculos, controlador Sugeno, punto de inicio (0,0), orientación de inicio 0 rad, punto objetivo (400,200).

La Figura 47 muestra el desplazamiento del sistema en un entorno sin obstáculos, partiendo de la posición (0,0) con orientación 0 rad, hacia un punto objetivo en la coordenada (400,200). De esta prueba destaca la alta similitud de la trayectoria resultante con la observada en la Figura 43 (FIS Mamdani). Este comportamiento es el esperado, dado que el controlador FIS Sugeno fue entrenado tomando como referencia el mapeo de entrada-salida del modelo Mamdani previamente sintonizado.

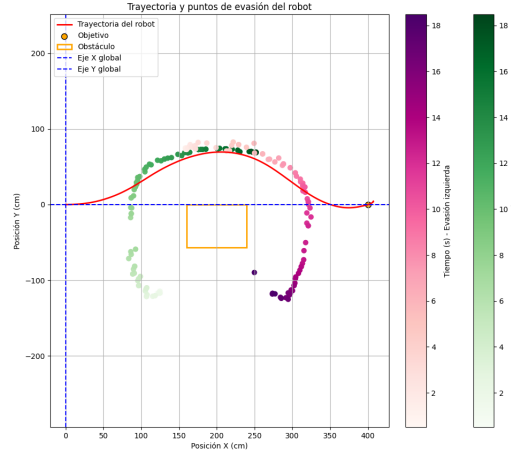


Figura 48: Trayectoria y puntos de evasión del robot en condición obstáculo a la derecha, controlador Sugeno, punto de inicio (0,0), orientación de inicio 0 rad, punto objetivo (400,0).

La Figura 48 ilustra la prueba con un obstáculo posicionado a la mitad del trayecto, desplazado ligeramente a la derecha del eje ideal para forzar una maniobra de evasión por la izquierda. El resultado es geométricamente igualable al observado en la Figura 44, donde el recorrido del chasis se aproxima considerablemente a los puntos intermedios generados, confirmando que Sugeno recibe la robustez de seguimiento de trayectorias locales de Mamdani.

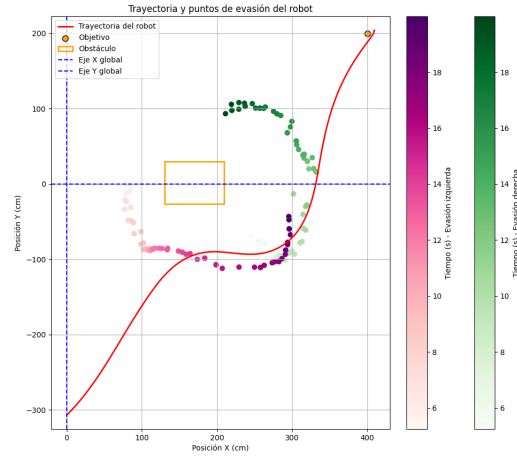


Figura 49: Trayectoria y puntos de evasión del robot en condición obstáculo a la izquierda, controlador Sugeno, punto de inicio (0,-307), orientación de inicio $\pi/4$ rad, punto objetivo (400,200).

En la Figura 49, el robot parte de la posición (0,-307) en diagonal y enfrenta un obstáculo desplazado a la izquierda, el cual evade exitosamente por la derecha. Es importante destacar dos mejoras en comparación con el desempeño de Mamdani (ver Figura 45): primero, el giro de evasión se ejecuta de manera más anticipada; segundo, se observa un incremento considerable en la densidad de los puntos intermedios proyectados. Este aumento de puntos es evidencia directa de que el ciclo de control logra ejecutarse a una mayor frecuencia gracias a la rapidez de Sugeno, permitiendo actualizaciones de puntos más oportunas y reduciendo el riesgo de colisión.

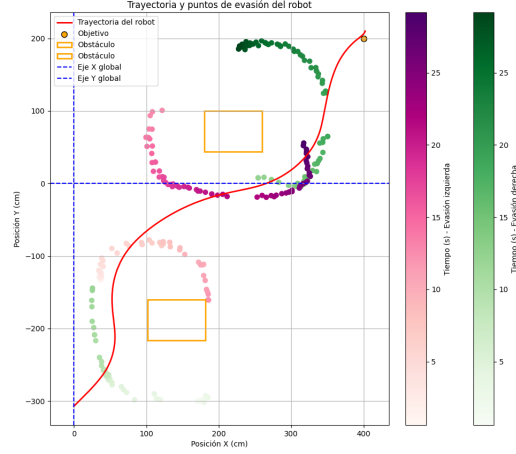


Figura 50: Trayectoria y puntos de evasión del robot en condición dos obstáculos, controlador Sugeno, punto de inicio (0,-307), orientación de inicio $\pi/4$ rad, punto objetivo (400,200).

Finalmente, la Figura 50 presenta el escenario más exigente, forzando al robot a evadir secuencialmente por la izquierda y luego por la derecha. Esta prueba confirma la eficacia del controlador FIS Sugeno Tipo 1, mantiene la precisión y suavidad de navegación de Mamdani, pero gana una alta frecuencia de actualización del algoritmo de Local Path Planning que es indispensable para un sistema de bajo costo computacional, implementado en un hardware de recursos limitados.

5.1.4 Comparación de controladores

Esta sección presenta la comparación del desempeño de los tres controladores implementados, evaluados mediante la evolución del error de orientación y del error de posición a lo largo del tiempo. Los datos mostrados corresponden al mismo conjunto de pruebas presentado en las secciones 5.1.1, 5.1.2 y 5.1.3.

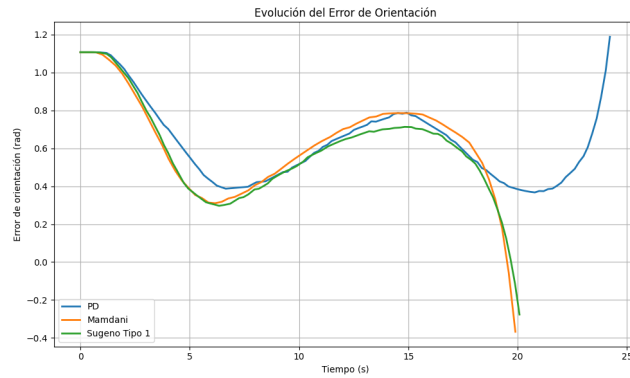


Figura 51: Comparación de los tres controladores por evolución del error de orientación, condición sin obstáculos, posición inicial (0,0), orientación inicial 0 rad, punto objetivo (400,200).

La Figura 51 muestra la comparación de la evolución del error de orientación de los tres controladores para un entorno libre de obstáculos. En primer lugar, se destaca la reducción de oscilación obtenida por los controladores difusos en comparación con el controlador clásico PD. Además, se observa que los modelos difusos logran orientar al robot hacia el punto objetivo en un tiempo de 20 segundos, mientras que el controlador clásico completa la tarea en un tiempo cercano a los 25 segundos. Esto se debe principalmente al avance constante del controlador clásico, mientras que los controladores difusos aplican una velocidad dinámica que depende del error de posición. Por último, se evidencia una alta similitud en la respuesta de los controladores Mamdani y Sugeno Tipo 1, mostrando un solapamiento en sus curvas de evolución.

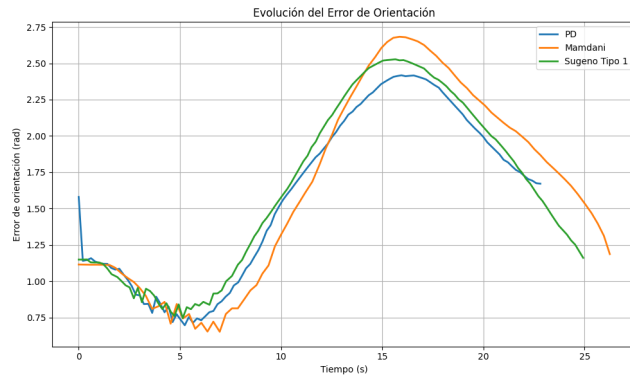


Figura 52: Comparación de los tres controladores por evolución del error de orientación, condición con un obstáculo a la derecha, posición inicial (0,0), orientación inicial 0 rad, punto objetivo (400,0).

La Figura 52 ilustra la evolución del error de orientación en la condición de un obstáculo a la derecha, escenario que obliga al robot a evadir por la izquierda. Se observa que, durante la primera mitad del recorrido, el error de orientación de los tres controladores oscila con pequeños saltos. Esto ocurre porque el punto intermedio de evasión que persigue el robot al inicio del trayecto está cambiando de posición dinámicamente y, por lo tanto, el error calculado respecto a la meta inmediata varía en cada instante.

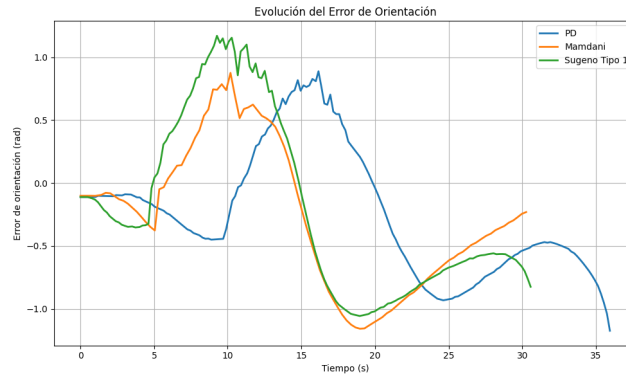


Figura 53: Comparación de los tres controladores por evolución del error de orientación, condición con un obstáculo a la izquierda, posición inicial (0,-307), orientación inicial $\pi/4$ rad, punto objetivo (400,200).

La Figura 53 muestra el comportamiento de los controladores frente a un obstáculo a la izquierda. Se aprecia un cambio abrupto en la tendencia del error de orientación: para los controladores Mamdani y Sugeno, esta inflexión ocurre a los 5 segundos, mientras que para el controlador PD se presenta más tarde, a los 10 segundos. Este comportamiento refleja cómo se calcula el error: durante la primera parte del recorrido, el error corresponde a la diferencia entre la orientación actual y el punto objetivo final; sin embargo, al detectar el obstáculo, el sistema recalcula el error apuntando hacia el nuevo punto intermedio de evasión, lo que genera este salto en la gráfica.

Derivado de la dinámica que proveen los controladores difusos, el robot logra aproximarse y evadir el obstáculo con mayor rapidez en comparación con el controlador PD, completando la tarea 5 segundos antes.

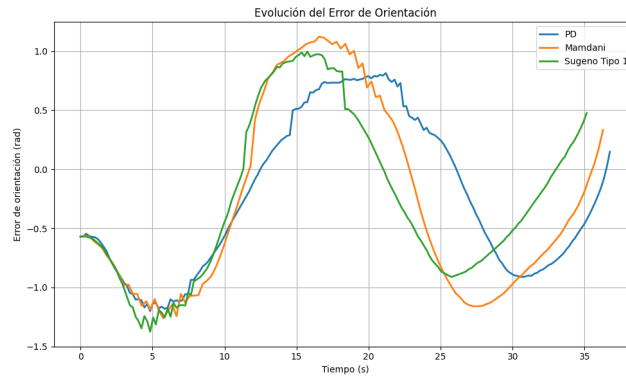


Figura 54: Comparación de los tres controladores por evolución del error de orientación, condición con dos obstáculos, posición inicial (0,-307), orientación inicial $\pi/4$ rad, punto objetivo (400,200).

La Figura 54 presenta la evolución del error en el escenario más complejo con dos obstáculos. A diferencia de la prueba anterior, el primer obstáculo es detectado desde el inicio del recorrido, por lo que el comportamiento de evasión de los tres controladores resulta bastante similar durante los primeros 10 segundos. Posteriormente, los controladores difusos toman ventaja al dirigir el chasis más rápidamente hacia el segundo obstáculo.

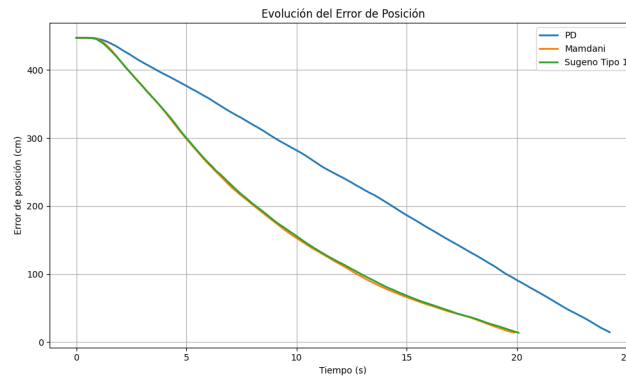


Figura 55: Comparación de los tres controladores por evolución del error de posición, condición sin obstáculos, posición inicial (0,0), orientación inicial 0 rad, punto objetivo (400,200).

La Figura 55 detalla el error de posición en un recorrido libre. Nuevamente se observa un solapamiento en el desempeño de los controladores difusos, los cuales completan la tarea en 20 segundos, mientras que el controlador PD requiere 24 segundos. Por otro lado, la tasa de reducción del error de posición en el controlador PD es constante. En contraste, los controladores difusos muestran

un decremento mucho más pronunciado al inicio del recorrido, atenuando su velocidad conforme el sistema se aproxima a la meta.

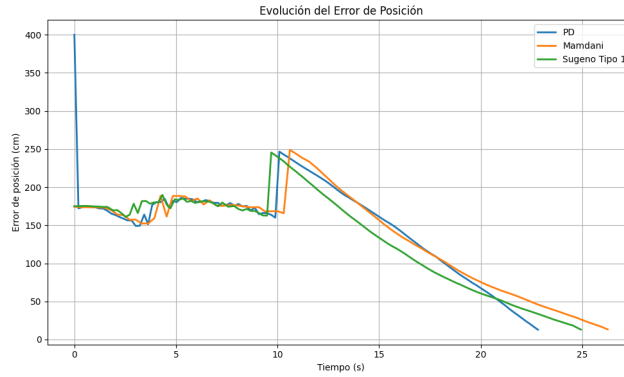


Figura 56: Comparación de los tres controladores por evolución del error de posición, condición con un obstáculo a la derecha, posición inicial (0,0), orientación inicial 0 rad, punto objetivo (400,0).

La Figura 56 ilustra el error de posición en un entorno con evasión por la izquierda. Durante los primeros 10 segundos, se mantiene un error cercano a los 170 cm, el cual corresponde a la distancia hacia los puntos intermedios generados para rodear el obstáculo. Una vez que el robot supera al obstáculo, el controlador PD muestra un decremento de error lineal, mientras que los algoritmos difusos exhiben una caída más rápida, seguida de una desaceleración en los últimos segundos antes de alcanzar el objetivo.

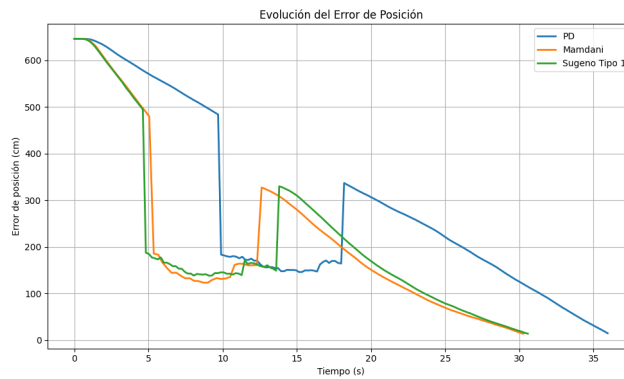


Figura 57: Comparación de los tres controladores por evolución del error de posición, condición con un obstáculo a la izquierda, posición inicial (0,-307), orientación inicial $\pi/4$ rad, punto objetivo (400,200).

La Figura 57 expone el error de posición al evadir por la derecha. Los tres

controladores inician con un error de 645 cm. Sin embargo, los controladores difusos alcanzan la zona del obstáculo a los 5 segundos, mientras que el controlador clásico requiere 10 segundos. Posteriormente, el error cae bruscamente de 490 cm a menos de 200 cm; esto sucede porque, al detectarse el objeto, el punto objetivo inmediato (punto de evasión) se genera a una distancia mucho más corta. Una vez superado el obstáculo, el error de posición se recalcula tomando como referencia la meta final.

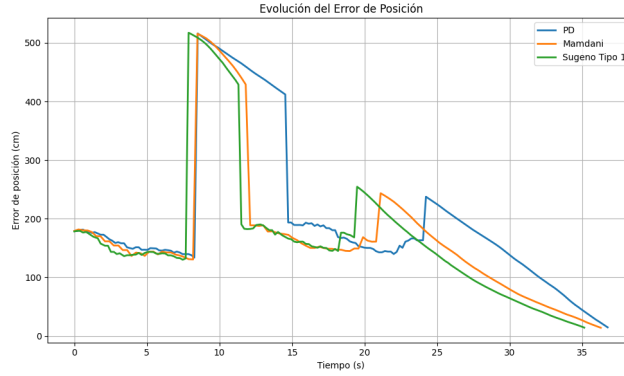


Figura 58: Comparación de los tres controladores por evolución del error de posición, condición con dos obstáculos, posición inicial (0,-307), orientación inicial $\pi/4$ rad, punto objetivo (400,200).

Finalmente, la Figura 58 muestra los resultados en el escenario de mayor complejidad. Durante los primeros 8 segundos, el error de posición oscila alrededor de los 160 cm, ya que el robot arranca detectando el primer obstáculo y ejecutando inmediatamente el seguimiento de los puntos intermedios por la izquierda. Además, se reafirma que los controladores difusos se desplazan con mayor rapidez hacia el segundo obstáculo, completando la segunda maniobra de evasión en menor tiempo.

5.1.5 Análisis de costo computacional por escenario

A continuación, se presenta el desempeño computacional del controlador clásico y de los dos controladores difusos para las cuatro condiciones de prueba analizadas en las secciones anteriores. Los resultados se encuentran organizados según el entorno de la prueba: la Tabla 8 corresponde al escenario libre de obstáculos; la Tabla 9, a la condición con un obstáculo a la derecha; la Tabla 10, con un obstáculo a la izquierda; y la Tabla 11, al entorno más complejo con dos obstáculos.

Las Tablas 8 a 11 muestran la comparativa de la mediana del uso de memoria empleado en la ejecución de cada algoritmo durante los cuatro escenarios. Se destaca que, en todas las pruebas realizadas, el consumo de memoria se mantuvo constante y con variaciones insignificantes entre los tres controladores. Esto

Tabla 8: Comparación de desempeño computacional: Condición sin obstáculos

Métrica	PD	Mamdani	Sugeno Tipo 1
Tiempo de Ejecución			
Tiempo promedio	196.89 ms	301.85 ms	214.14 ms
Uso de Memoria			
Mediana	188.43 MB	188.69 MB	191.10 MB

Tabla 9: Comparación de desempeño computacional: Condición un obstáculo a la derecha

Métrica	PD	Mamdani	Sugeno Tipo 1
Tiempo de Ejecución			
Tiempo promedio	200.93 ms	302.55 ms	205.29 ms
Uso de Memoria			
Mediana	191.10 MB	191.15 MB	192.91 MB

demuestra que, si bien los sistemas de inferencia difusa requieren almacenar un mayor número de parámetros (variables lingüísticas, funciones de pertenencia y reglas difusas), esto no implica un impacto significativo de memoria frente a un controlador más ligero como lo es el PD.

En cuanto al tiempo de ejecución, se observa un incremento significativo en el costo computacional del controlador Mamdani, el cual requiere en promedio 100 ms más por iteración en comparación con los controladores PD y Sugeno Tipo 1, que rondan los 200 ms. Este retardo refleja la carga de procesamiento inherente al método de Mamdani, originada principalmente en los pasos de agregación y en la integración numérica para el cálculo del centroide para la defusificación. Por consiguiente, resalta el hecho de que el algoritmo Sugeno Tipo 1 logró igualar el tiempo de ejecución del PD, permitiendo aprovechar la suavidad de un sistema de inferencia difusa sin comprometer la velocidad de respuesta del sistema embarcado.

Por último, existe una comunicación serial UART a 115200 baudios, también cabe señalar que la base de estos tiempos de ejecución está directamente relacionada con la latencia de sincronización entre la computadora Raspberry Pi 5 y el microcontrolador ESP32. El firmware de la ESP32 opera mediante un bucle programado con un intervalo de 100 ms, durante el cual se calcula el control PI de los motores, se actualiza la odometría y se gestiona la comunicación por el puerto serial. Debido a esta cuantización del tiempo, en un ciclo de trabajo regular, los algoritmos PD y Sugeno leen los datos de odometría en los primeros 100 ms y, tras calcular rápidamente las acciones de control, transmiten la respuesta para que sea ejecutada en la siguiente ventana de 100 ms, sumando una latencia total de 200 ms. En contraste, el controlador Mamdani, al exigir mayor procesamiento para la defusificación, excede la ventana de los 200 ms; en consecuencia, pierde el ciclo inmediato de transmisión y se ve forzado a esperar un tercer ciclo de lectura/escritura, lo que eleva su tiempo de iteración a 300 ms.

Tabla 10: Comparación de desempeño computacional: Condición un obstáculo a la izquierda

Métrica	PD	Mamdani	Sugeno Tipo 1
Tiempo de Ejecución			
Tiempo promedio	203.13 ms	305.20 ms	213.37 ms
Uso de Memoria			
Mediana	189.07 MB	190.75 MB	192.87 MB

Tabla 11: Comparación de desempeño computacional: Condición dos obstáculos

Métrica	PD	Mamdani	Sugeno Tipo 1
Tiempo de Ejecución			
Tiempo promedio	202.03 ms	308.24 ms	211.35 ms
Uso de Memoria			
Mediana	192.91 MB	192.09 MB	193.86 MB

5.2 Pruebas experimentales para el algoritmo Local Path Planning

Se realizaron diversas pruebas experimentales en el robot diferencial descrito en la Sección 4.1. El conjunto de pruebas se dividió en dos grupos:

1. Validación del algoritmo propuesto: Se evaluó el comportamiento del algoritmo en un entorno diseñado para incluir tres escenarios comunes de obstáculos en la planificación de rutas locales: una estructura en forma de U, un pasillo estrecho y una sección con múltiples obstáculos cerca del objetivo final.
2. Comparación con VPH: Se realizó una comparación cuantitativa del rendimiento de nuestro algoritmo frente al método VPH [6] en un escenario específico.

La configuración y los resultados de estas pruebas se detallan a continuación.

5.2.1 Navegación a un punto objetivo con múltiples obstáculos

Para esta prueba, el robot inicia desde la coordenada $(x, y) = (0, 0)$ con una orientación de $\theta = 0$ rad. El objetivo final está ubicado en la posición $(x, y) = (85, 750)$ cm, con una orientación deseada de $\theta = 0$ rad. La pose del robot (posición y orientación) se estima utilizando la odometría calculada por el microcontrolador ESP32, como se describe en la Sección 3.2. En la Raspberry Pi 5, se procesan los datos del LiDAR y se genera un punto objetivo inmediato (ya sea un punto intermedio de evasión o el objetivo final) utilizando el algoritmo de planificación de rutas local detallado en la Sección 4.2. Este conjunto de pruebas emplea un controlador PD descrito en la Sección 4.3.

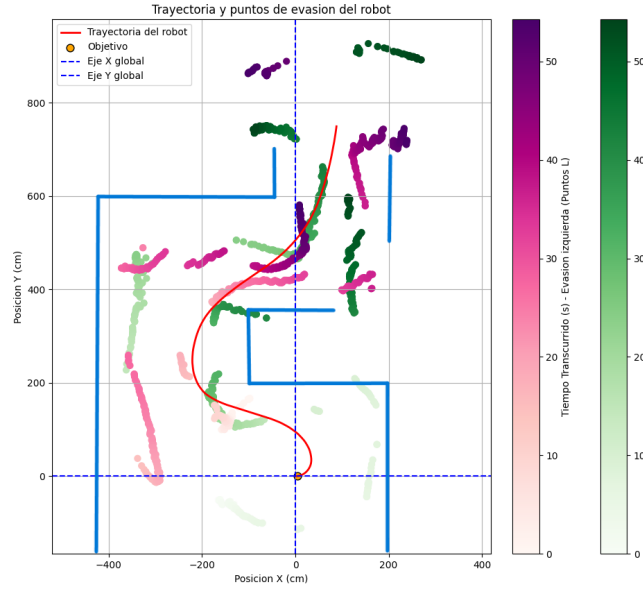


Figura 59: La trayectoria del robot (línea roja) coincide con los puntos de evasión (puntos verdes y rosas/morados).

El entorno experimental para esta prueba es un escenario complejo compuesto por varias secciones: el robot comienza dentro de una estructura en forma de U, navega a través de un pasillo en forma de L y, en el tramo final hacia el objetivo, debe gestionar la detección simultánea de al menos tres obstáculos.

Los parámetros utilizados en esta prueba se establecieron de manera empírica: para el controlador PD, las ganancias fueron $k_p = 1,1$ y $k_d = 0,5$, con una velocidad lineal de comando $V_{com} = 1,5$ rad/s. Para el algoritmo de evasión, se configuró un radio de detección de $d_{max} = 200$ cm y una distancia de seguridad de $\epsilon = 35$ cm. Los pesos para la función de costo normalizada fueron $w_{dist} = 1$ y $w_{angle} = 0,5$. La Fig. 59 ilustra la trayectoria seguida por el robot desde el inicio hasta el objetivo final, superpuesta con el conjunto de todos los puntos de evasión calculados durante el recorrido. Finalmente, la Fig. 60 presenta la misma trayectoria, pero resaltando únicamente los puntos objetivo seleccionados por la función de costo en cada instante.

5.2.2 Comparación frente al método VPH

Para evaluar el rendimiento relativo del algoritmo propuesto, se realizó una comparación con el método VPH en un entorno de navegación específico. Se configuró un entorno consistente en un pasillo, donde el robot debe navegar

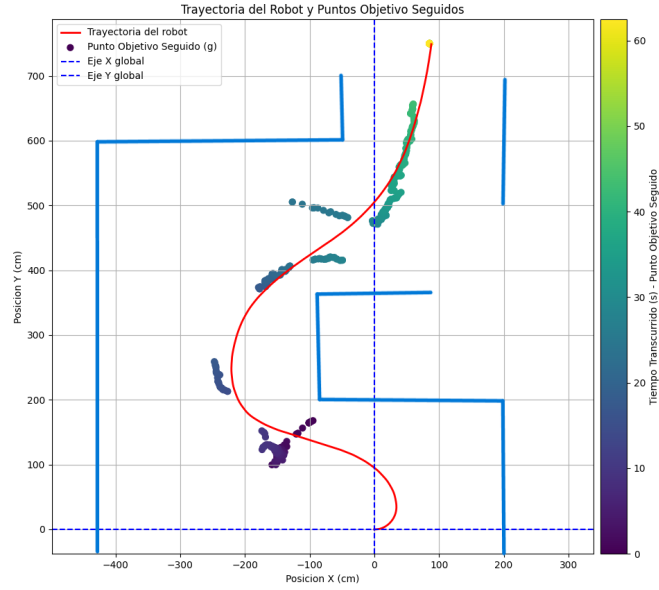


Figura 60: Trayectoria del robot (línea roja) y puntos objetivo seleccionados por la función de costo (puntos con color según el tiempo transcurrido).

desde la coordenada inicial $(x, y) = (0, 0)$ hasta el punto objetivo final $(x, y) = (700, -450)$.

Los parámetros utilizados en nuestro algoritmo corresponden a los descritos en la Sección 5.2.1. Para el algoritmo VPH, fue necesario definir sus parámetros clave, incluyendo la distancia de seguridad (D_f), el ancho mínimo de la ventana (W_{max}) y los pesos de su función de costo (K_1 , que prioriza mantener la dirección actual, y K_2 , que prioriza la dirección hacia el objetivo). Cabe destacar que la función de costo del VPH original es sensible a la proporción entre K_1 y K_2 , donde un valor ligeramente mayor de K_2 en comparación con K_1 tiende a seleccionar ángulos muy cercanos a la dirección del objetivo, incluso si estos se encuentran cerca de los bordes de un obstáculo. Para mitigar este comportamiento en nuestra implementación comparativa del VPH, se aplicó un recorte adicional a los valles candidatos, eliminando las direcciones más cercanas a los bordes detectados del obstáculo.

La Fig. 61 muestra la trayectoria resultante al ejecutar nuestro algoritmo. Se observa un movimiento fluido y una velocidad de avance constante, completando el recorrido en aproximadamente 53 segundos. La figura también muestra los puntos objetivo seleccionados por la función de costo en cada instante. En contraste, la Fig. 62 muestra el comportamiento del robot utilizando el algoritmo

VPH (con la modificación de recorte). Inicialmente se mueve a una mayor velocidad en las secciones rectas del pasillo y tiene dificultades al acercarse a las paredes frontales, exhibiendo una reducción drástica en su velocidad y giros menos suaves. En la curva final, el algoritmo VPH no logró alcanzar el objetivo, tardando más de 800 segundos.

Finalmente, la Tabla 12 muestra el tiempo de ejecución de los algoritmos. El algoritmo propuesto presenta un tiempo de 1.2661 ms por iteración, mayor en comparación con los 0.2016 ms del algoritmo VPH. Este incremento es esperado debido a los procesos adicionales que integra la propuesta, tales como el cálculo geométrico de los puntos de evasión y la validación de segmentos. Sin embargo, el tiempo de ejecución sigue siendo lo suficientemente bajo, por lo que resulta altamente recomendable para su implementación en sistemas en tiempo real.

Respecto al uso de memoria, ambos algoritmos mostraron un comportamiento muy similar. Aunque el algoritmo propuesto requiere una cantidad ligeramente superior (20.30 kB), este consumo sigue siendo perfectamente adecuado para dispositivos con recursos limitados.

Tabla 12: Comparación del costo computacional entre algoritmos de evasión

Métrica	Algoritmo VPH	Algoritmo Propuesto
Tiempo de ejecución/iteración	0.2016 ms	1.2661 ms
Uso de memoria	16.14 kB	20.30 kB

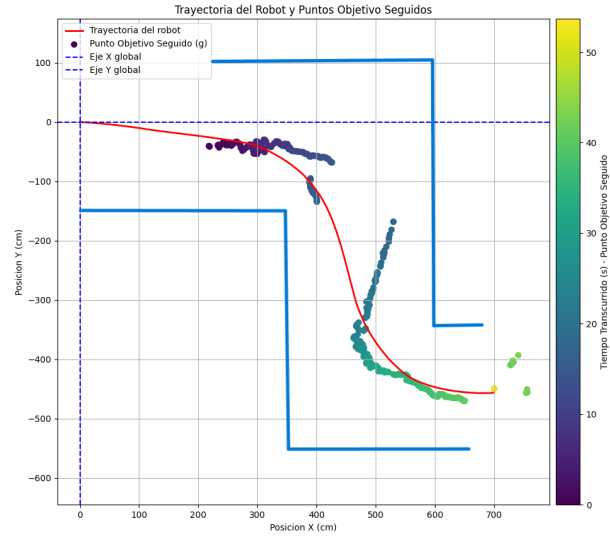


Figura 61: Trayectoria del robot utilizando el algoritmo propuesto. Se muestra la ruta seguida (línea roja) y los puntos objetivo seleccionados por la función de costo.

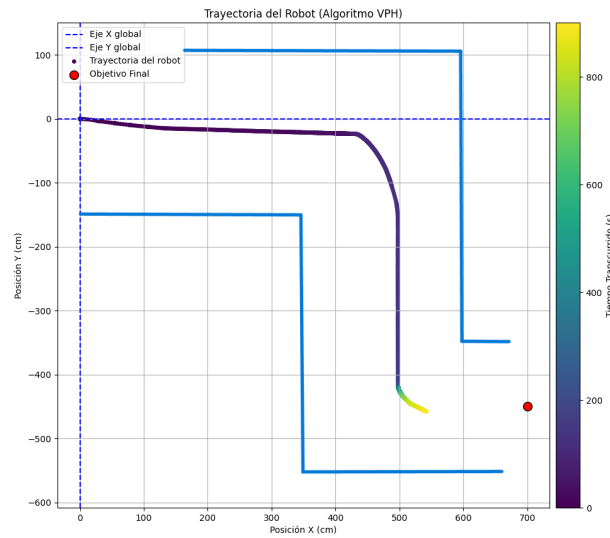


Figura 62: Trayectoria del robot utilizando el algoritmo VPH (con modificación de recorte). El color de la trayectoria varía según el tiempo transcurrido.

6 Conclusiones y perspectivas

Tras el desarrollo, implementación y validación experimental de la plataforma robótica y los algoritmos propuestos, se presentan las siguientes conclusiones generales derivadas de los resultados obtenidos:

6.1 Sobre el desempeño de los controladores difusos

La evaluación comparativa entre el control clásico y el inteligente permitió identificar que, si bien el controlador PD es sencillo de implementar, presenta limitaciones en la navegación autónoma, tales como oscilaciones en la trayectoria y un excedente de desplazamiento (15 cm) provocado por la inercia mecánica al no considerar dinámicamente la velocidad lineal.

Por su parte, el controlador FIS Mamdani demostró una mayor robustez y suavidad cinemática, mitigando las oscilaciones y eliminando el error por inercia gracias a su capacidad de evaluar la variable lingüística de distancia para reducir la velocidad de manera progresiva. Sin embargo, su alto costo computacional y la latencia generada (300 ms) limitan su uso en sistemas de tiempo real con hardware de recursos limitados. Por consiguiente, se concluye que el controlador Sugeno Tipo 1 representa la solución óptima para este trabajo; al ser entrenado mediante datos empíricos del modelo Mamdani, logra replicar el comportamiento fluido pero con una eficiencia computacional igualable a la del controlador PD (200 ms), permitiendo una alta frecuencia de actualización indispensable para el cumplimiento de la tarea de navegación.

Por último, la implementación física quedó respaldada por el análisis de estabilidad desarrollado. Se comprobó que el desempeño del robot se relaciona con un campo vectorial de atracción global. La evaluación de la función de Lyapunov en el espacio de fases demostró que, a pesar de las restricciones no holonómicas inherentes al robot, el controlador Sugeno de primer orden garantiza una Estabilidad Práctica (UUB). El 85 % del espacio operativo muestra una disipación de energía, mientras que el porcentaje restante actúa como transitorio, forzando al sistema hacia la región de tolerancia de 10 cm.

6.2 Sobre el algoritmo de Local Path Planning

El algoritmo de planificación de rutas basado en la generación dinámica de puntos de evasión demostró ser eficaz para resolver escenarios donde los métodos clásicos suelen fallar. Las pruebas experimentales confirmaron que la propuesta es capaz de navegar con éxito en estructuras complejas en forma de “U”, pasillos estrechos y entornos con múltiples obstáculos simultáneos.

Al comparar el desempeño frente al método clásico de Histograma Polar Vectorial (VPH), se concluye que, aunque el algoritmo propuesto exige un tiempo de ejecución ligeramente superior (1.26 ms vs 0.20 ms), ofrece un desempeño significativamente mayor. Mientras que VPH mostró una alta sensibilidad a la parametrización y falló en completar trayectorias con curvas pronunciadas, nuestra propuesta mantuvo un movimiento fluido y una velocidad de avance

constante, demostrando que la validación geométrica de segmentos y la función de costo normalizada garantizan trayectorias más seguras y cinemáticamente suaves.

Finalmente, los resultados permiten validar la hipótesis planteada al inicio de esta investigación. Se ha demostrado que la integración de sistemas de inferencia difusa con un algoritmo de planificación local dinámico mejora considerablemente el desempeño de la navegación autónoma en comparación con los esquemas de control y planificación clásicos.

El trabajo conjunto entre el controlador Sugeno (que aporta suavidad y eficiencia) y el algoritmo de puntos de evasión (que aporta robustez ante mínimos locales) permite que un robot de bajo costo computacional opere en entornos no estructurados con una alta tasa de éxito. Adicionalmente, la arquitectura modular propuesta desacopla la etapa de control del sistema de planificación de rutas. Esta característica permite integrar el algoritmo de puntos dinámicos con diversos enfoques de control, superando la rigidez de los métodos clásicos que calculan directamente las salidas de control de los actuadores. Al evitar este acoplamiento directo, se previene que un controlador inadecuado para las condiciones específicas del entorno comprometa el desempeño global de la tarea de navegación.

6.3 Perspectivas

Como trabajo futuro de esta tesis se proponen los siguientes puntos:

- Implementación de algoritmos de fusión de sensores (Filtro de Kalman) combinando la odometría de los encoders y los datos de la Unidad de Medición Inercial (IMU). Esto con el propósito de mitigar la acumulación de errores y robustecer la estimación de la posición y orientación del robot diferencial en entornos dinámicos.
- Integración de un sistema de visión artificial basado en un acelerador de IA (Hailo 8L) para el procesamiento de redes neuronales convolucionales en tiempo real, para su implementación en casos de estudio específicos, tales como la agricultura de precisión en tareas de detección de enfermedades o daños en cultivos.
- Desarrollo e implementación de algoritmos de Mapeo y Localización Simultánea (SLAM) mediante fusión de datos LiDAR-Cámara, para dar al robot la capacidad de navegar de forma completamente autónoma en entornos desconocidos y no estructurados sin depender de mapas.

Referencias

- [1] Y. Tang, M. A. Zakaria y M. Younas, “Path Planning Trends for Autonomous Mobile Robot Navigation: A Review”, en, *Sensors*, vol. 25, n.º 4, pág. 1206, ene. de 2025, ISSN: 1424-8220. DOI: 10.3390/s25041206 visitado 11 de mayo de 2026. dirección: <https://www.mdpi.com/1424-8220/25/4/1206>
- [2] A. Benmachiche, M. Derdour, M. S. Kahil, M. C. Ghanem y M. Deriche, “Adaptive Hybrid PSO–APF Algorithm for Advanced Path Planning in Next-Generation Autonomous Robots”, en, *Sensors*, vol. 25, n.º 18, pág. 5742, ene. de 2025, ISSN: 1424-8220. DOI: 10.3390/s25185742 visitado 11 de mayo de 2026. dirección: <https://www.mdpi.com/1424-8220/25/18/5742>
- [3] J. Borenstein e Y. Koren, “The vector field histogram-fast obstacle avoidance for mobile robots”, *IEEE Transactions on Robotics and Automation*, vol. 7, n.º 3, págs. 278-288, jun. de 1991, ISSN: 2374-958X. DOI: 10.1109/70.88137 visitado 12 de mayo de 2026. dirección: <https://ieeexplore.ieee.org/document/88137>
- [4] I. Ulrich y J. Borenstein, “VFH+: reliable obstacle avoidance for fast mobile robots”, en *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, ISSN: 1050-4729, vol. 2, mayo de 1998, 1572-1577 vol.2. DOI: 10.1109/ROBOT.1998.677362 visitado 12 de mayo de 2026. dirección: <https://ieeexplore.ieee.org/document/677362>
- [5] I. Ulrich y J. Borenstein, “VFH/sup */: local obstacle avoidance with look-ahead verification”, en *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, ISSN: 1050-4729, vol. 3, abr. de 2000, 2505-2511 vol.3. DOI: 10.1109/ROBOT.2000.846405 visitado 12 de mayo de 2026. dirección: <https://ieeexplore.ieee.org/abstract/document/846405>
- [6] D. An y H. Wang, “VPH: a new laser radar based obstacle avoidance method for intelligent mobile robots”, en *Fifth World Congress on Intelligent Control and Automation*, vol. 5, jun. de 2004, 4681-4685 Vol.5. DOI: 10.1109/WCICA.2004.1342407 visitado 12 de mayo de 2026. dirección: <https://ieeexplore.ieee.org/document/1342407>
- [7] J. Gong, Y. Duan, Y. Man y G. Xiong, “VPH+: An Enhanced Vector Polar Histogram Method for Mobile Robot Obstacle Avoidance”, en *2007 International Conference on Mechatronics and Automation*, ISSN: 2152-744X, ago. de 2007, págs. 2784-2788. DOI: 10.1109/ICMA.2007.4304000 visitado 12 de mayo de 2026. dirección: <https://ieeexplore.ieee.org/document/4304000>

- [8] Q. Liang, Z. Wang, Y. Yin, W. Xiong, J. Zhang y Z. Yang, “Autonomous aerial obstacle avoidance using LiDAR sensor fusion”, en, *PLOS ONE*, vol. 18, n.º 6, e0287177, jun. de 2023, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0287177 visitado 9 de ago. de 2026. dirección: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0287177>
- [9] S. T. Padgett y A. F. Browne, “Vector-based robot obstacle avoidance using LIDAR and mecanum drive”, en *SoutheastCon 2017*, ISSN: 1558-058X, mar. de 2017, págs. 1-5. DOI: 10.1109/SECON.2017.7925312 visitado 9 de ago. de 2026. dirección: <https://ieeexplore.ieee.org/document/7925312>
- [10] D. Fox, W. Burgard y S. Thrun, “The dynamic window approach to collision avoidance”, *IEEE Robotics & Automation Magazine*, vol. 4, n.º 1, págs. 23-33, mar. de 1997, ISSN: 1558-223X. DOI: 10.1109/100.580977 visitado 12 de mayo de 2026. dirección: <https://ieeexplore.ieee.org/document/580977>
- [11] Y. Cheng y G. Y. Wang, “Mobile robot navigation based on lidar”, en *2018 Chinese Control And Decision Conference (CCDC)*, ISSN: 1948-9447, jun. de 2018, págs. 1243-1246. DOI: 10.1109/CCDC.2018.8407319 visitado 12 de mayo de 2026. dirección: <https://ieeexplore.ieee.org/document/8407319>
- [12] X. Ou, Z. You y X. He, “Local Path Planner for Mobile Robot Considering Future Positions of Obstacles”, en, *Processes*, vol. 12, n.º 5, pág. 984, mayo de 2024, ISSN: 2227-9717. DOI: 10.3390/pr12050984 visitado 9 de ago. de 2026. dirección: <https://www.mdpi.com/2227-9717/12/5/984>
- [13] P. T.-T. Nguyen, S.-W. Yan, J.-F. Liao y C.-H. Kuo, “Autonomous Mobile Robot Navigation in Sparse LiDAR Feature Environments”, en, *Applied Sciences*, vol. 11, n.º 13, pág. 5963, ene. de 2021, ISSN: 2076-3417. DOI: 10.3390/app11135963 visitado 9 de ago. de 2026. dirección: <https://www.mdpi.com/2076-3417/11/13/5963>
- [14] J. d. Heuvel, X. Zeng, W. Shi, T. Sethuraman y M. Bennewitz, *Spatio-temporal Attention Enhances Lidar-Based Robot Navigation in Dynamic Environments*, arXiv:2310.19670 [cs.RO], feb. de 2024. DOI: 10.48550/arXiv.2310.19670 visitado 9 de ago. de 2026. dirección: <http://arxiv.org/abs/2310.19670>
- [15] S. Hanana, N. Krichen, M. S. Masmoudi y M. Masmoudi, “Fuzzy Logic Obstacle Avoidance Controller for a Mobile Robot using LiDAR”, en *2024 IEEE 7th International Conference on Advanced Technologies, Signal and Image Processing (ATSIP)*, ISSN: 2687-878X, vol. 1, jul. de 2024, págs. 387-390. DOI: 10.1109/ATSIP62566.2024.10638842 visitado 9 de ago. de 2026. dirección: <https://ieeexplore.ieee.org/document/10638842>

- [16] G. Csaba, L. Somlyai y Z. Vámosy, “Mobil robot navigation using 2D LIDAR”, en *2018 IEEE 16th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, feb. de 2018, págs. 000 143-000 148. DOI: 10.1109/SAMI.2018.8324002 visitado 13 de mayo de 2026. dirección: <https://ieeexplore.ieee.org/document/8324002>
- [17] A. J. Moshayedi, A. Abbasi, L. Liao y S. Li, “Path planning and trajectory tracking of a mobile robot using bio-inspired optimization algorithms and PID control”, en *2019 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, ISSN: 2377-9322, jun. de 2019, págs. 1-6. DOI: 10.1109/CIVEMSA45640.2019.9071596 visitado 13 de mayo de 2026. dirección: <https://ieeexplore.ieee.org/document/9071596>
- [18] T. Xia y H. Chen, “A Survey of Autonomous Vehicle Behaviors: Trajectory Planning Algorithms, Sensed Collision Risks, and User Expectations”, en, *Sensors*, vol. 24, n.º 15, pág. 4808, ene. de 2024, ISSN: 1424-8220. DOI: 10.3390/s24154808 visitado 13 de mayo de 2026. dirección: <https://www.mdpi.com/1424-8220/24/15/4808>
- [19] K. Liu, J. Gong y H. Chen, *VPH+ and MPC Combined Collision Avoidance for Unmanned Ground Vehicle in Unknown Environment*, arXiv:1805.08089 [cs.RO], mayo de 2018. DOI: 10.48550/arXiv.1805.08089 visitado 13 de mayo de 2026. dirección: <http://arxiv.org/abs/1805.08089>
- [20] S. Hanana, N. Krichen, M. S. Masmoudi y M. Masmoudi, “Fuzzy Logic Obstacle Avoidance Controller for a Mobile Robot using LiDAR”, en *2024 IEEE 7th International Conference on Advanced Technologies, Signal and Image Processing (ATSIP)*, ISSN: 2687-878X, vol. 1, jul. de 2024, págs. 387-390. DOI: 10.1109/ATSIP62566.2024.10638842 visitado 13 de mayo de 2026. dirección: <https://ieeexplore.ieee.org/document/10638842>
- [21] P. M. Tuan, N. D. Tai, T. Q. Huy y N. T. Thinh, “Flexible Path Planning of Mobile Robot for Avoiding the Dynamic Obstacles Using Fuzzy Controllers”, *International Journal of Mechanical Engineering and Robotics Research*, vol. 13, n.º 1, págs. 126-132, 2024, ISSN: 22780149. DOI: 10.18178/ijmerr.13.1.126-132 visitado 13 de mayo de 2026. dirección: <https://www.ijmerr.com/show-235-1910-1.html>
- [22] A. Bakdi, A. Hentout, H. Boutami, A. Maoudj, O. Hachour y B. Bouzouia, “Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control”, *Robotics and Autonomous Systems*, vol. 89, págs. 95-109, mar. de 2017, ISSN: 0921-8890. DOI: 10.1016/j.robot.2016.12.008 visitado 13 de mayo de 2026. dirección: <https://www.sciencedirect.com/science/article/pii/S0921889016302512>
- [23] M. Faisal, R. Hedjar, M. Al Sulaiman y K. Al-Mutib, “Fuzzy Logic Navigation and Obstacle Avoidance by a Mobile Robot in an Unknown Dynamic Environment”, EN, *International Journal of Advanced Robotic Systems*,

- vol. 10, n.º 1, pág. 37, ene. de 2013, ISSN: 1729-8806. DOI: 10.5772/54427 visitado 13 de mayo de 2026. dirección: <https://doi.org/10.5772/54427>
- [24] M. H. Haider et al., “Robust mobile robot navigation in cluttered environments based on hybrid adaptive neuro-fuzzy inference and sensor fusion”, *Journal of King Saud University - Computer and Information Sciences*, vol. 34, n.º 10, Part B, págs. 9060-9070, nov. de 2022, ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2022.08.031 visitado 13 de mayo de 2026. dirección: <https://www.sciencedirect.com/science/article/pii/S1319157822003135>
 - [25] Aqeel-Ur-Rehman y C. Cai, “Autonomous Mobile Robot Obstacle Avoidance Using Fuzzy-PID Controller in Robot’s Varying Dynamics”, en *2020 39th Chinese Control Conference (CCC)*, ISSN: 1934-1768, jul. de 2020, págs. 2182-2186. DOI: 10.23919/CCC50068.2020.9188467 visitado 13 de mayo de 2026. dirección: <https://ieeexplore.ieee.org/document/9188467>
 - [26] S. M. Saidi, R. Mellah, A. Fekik y A. T. Azar, “Real-Time Fuzzy-PID for Mobile Robot Control and Vision-Based Obstacle Avoidance”, en, *International Journal of Service Science, Management, Engineering, and Technology (IJSSMET)*, vol. 13, n.º 1, págs. 1-32, 2022, ISSN: 1947-959X. DOI: 10.4018/IJSSMET.304818 visitado 13 de mayo de 2026. dirección: <https://www.igi-global.com/gateway/article/www.igi-global.com/gateway/article/304818>
 - [27] S. Louda, N. Karkar, F. Seghir y O. Boutalbi, “Fuzzy Dynamic Feedback Linearization for Efficient Mobile Robot Trajectory Tracking and Obstacle Avoidance in Autonomous Navigation”, en, *International Journal of Robotics and Control Systems*, vol. 5, n.º 2, págs. 881-901, mar. de 2025, ISSN: 2775-2658. DOI: 10.31763/ijrcs.v5i2.1780 visitado 13 de mayo de 2026. dirección: <https://pubs2.ascee.org/index.php/IJRCS/article/view/1780>
 - [28] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms In MATLAB® Second, Completely Revised, Extended And Updated Edition*, en. Springer, mayo de 2017, Google-Books-ID: d4EkDwAAQBAJ, ISBN: 978-3-319-54413-7.
 - [29] K. Ogata, *Ingeniería de control moderna*, es. Pearson Educación, 2003, Google-Books-ID: QK148EPC_m0C, ISBN: 978-84-205-3678-1.
 - [30] J.-S. R. Jang, C.-T. Sun y E. Mizutani, *Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, en. Prentice Hall, 1997, Google-Books-ID: vN5QAAAAMAAJ, ISBN: 978-0-13-261066-7.
 - [31] R. Brockett, “Asymptotic stability and feedback stabilization”, 1982. visitado 5 de jun. de 2026. dirección: <https://www.semanticscholar.org/paper/Asymptotic-stability-and-feedback-stabilization-Brockett/af3539894339c03c350e522afd584b961d591d4f>

- [32] H. K. Khalil, *Nonlinear Systems*, en. Prentice Hall, 2002, Google-Books-ID: t_d1QgAACAAJ, ISBN: 978-0-13-067389-3.
- [33] J.-J. E. Slotine y W. Li, *Applied Nonlinear Control*, en. Prentice Hall, 1991, Google-Books-ID: cwpRAAAAMAAJ, ISBN: 978-0-13-040890-7.