



Universidad Autónoma de Querétaro

Facultad de Ingeniería

Maestría en Ciencias (Control en Sistemas Embebidos)

Implementación de un sistema sensoriomotor a vehículo terrestre no tripulado

TESIS

Que como parte de los requisitos para obtener el grado de

Maestro en Ciencias

Presenta:

Ing. Arturo Iram de la Fuente Sánchez

Dirigido por:

Dr. Gerardo Israel Pérez Soto

Dr. Jesús Franco Robles

SINODALES

Dr. Gerardo Israel Pérez Soto

Presidente

Dr. Jesús Franco Robles

Secretario

Dr. Juvenal Rodríguez Reséndiz

Vocal

Dr. Jesús Rooney Rivera Guillén

Suplente

Dra. Karla Anhel Camarillo Gómez

Suplente

Centro Universitario

Querétaro, QRO

México.

Agosto 2026

La presente obra está bajo la licencia:
<https://creativecommons.org/licenses/by-nc-nd/4.0/deed.es>



CC BY-NC-ND 4.0 DEED

Atribución-NoComercial-SinDerivadas 4.0 Internacional

Usted es libre de:

Compartir — copiar y redistribuir el material en cualquier medio o formato

La licenciante no puede revocar estas libertades en tanto usted siga los términos de la licencia

Bajo los siguientes términos:



Atribución — Usted debe dar [crédito de manera adecuada](#), brindar un enlace a la licencia, e [indicar si se han realizado cambios](#). Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciante.



NoComercial — Usted no puede hacer uso del material con [propósitos comerciales](#).



SinDerivadas — Si [remezcla, transforma o crea a partir](#) del material, no podrá distribuir el material modificado.

No hay restricciones adicionales — No puede aplicar términos legales ni [medidas tecnológicas](#) que restrinjan legalmente a otras a hacer cualquier uso permitido por la licencia.

Avisos:

No tiene que cumplir con la licencia para elementos del material en el dominio público o cuando su uso esté permitido por una [excepción o limitación](#) aplicable.

No se dan garantías. La licencia podría no darle todos los permisos que necesita para el uso que tenga previsto. Por ejemplo, otros derechos como [publicidad, privacidad, o derechos morales](#) pueden limitar la forma en que utilice el material.

A mi familia, por estar presente en todo momento, por su apoyo incondicional y por darme el aliento necesario para continuar en cada etapa.

A mi padre, por ser mi respaldo en la vida, por su confianza, sus consejos y por estar siempre, incluso en los momentos difíciles.

A mi madre, por su cariño, sus enseñanzas y por formar parte de mi vida y de la persona que soy hoy.

A Gaby, por acompañarme durante estos dos años, por ser mi apoyo incondicional y por brindarme la fortaleza, la paciencia y el cariño que tantas veces necesité. Gracias por estar a mi lado a lo largo de este proceso y por compartir conmigo los momentos de esfuerzo, incertidumbre y satisfacción.

Este logro también es de ustedes.

Agradecimientos

Quiero agradecer a la Universidad Autónoma de Querétaro por brindarme el espacio y las condiciones necesarias para cursar la maestría, así como la oportunidad de continuar con mi formación académica y profesional. Agradezco también el apoyo económico que me fue otorgado durante este proceso. De igual manera, agradezco a la Universidad de Limoges por abrirme las puertas y brindarme la oportunidad de vivir una experiencia académica excepcional en el extranjero, que me permitió aprender, colaborar con un equipo multidisciplinario y desarrollarme en el ámbito de la investigación. A la SECIHTI, por el apoyo brindado a mi manutención durante esta etapa.

Agradezco especialmente a mi director de tesis por todo el apoyo, la orientación y la confianza que me ha brindado desde mis primeros pasos en este camino. A mi codirector, por ser parte fundamental de mi estancia en el extranjero y por compartir conmigo sus conocimientos, su experiencia y su apoyo académico. A todos mis sinodales, por formar parte de este proyecto y por sus valiosos comentarios, consejos y sugerencias, que contribuyeron significativamente al desarrollo de este trabajo. A todos ustedes, gracias por las enseñanzas y por las experiencias compartidas; cada uno, desde su propia perspectiva, ha contribuido a mi crecimiento y ha dejado una huella importante en mi formación y en lo que hoy conforma mi trayectoria profesional.

Quiero agradecer también a mi familia, especialmente a mis abuelos y tíos paternos, a mi madrina y a toda su familia, por estar siempre presentes, por interesarse en lo que hacía y en cómo me encontraba, y por regalarme, en todo momento, palabras de aliento y apoyo. Gracias por acompañarme durante este proceso y por apoyarme incluso en los momentos en que el camino se volvió más complicado, incluido el apoyo económico que me brindaron cuando fue necesario.

Gracias a mi padre, porque gran parte de lo que soy y de lo que he logrado se lo debo a sus enseñanzas, a su guía y a su apoyo incondicional. Gracias por enseñarme a enfrentar los retos, por confiar en mí y por estar presente en cada etapa de mi vida. Sin todo lo que me has enseñado y sin tu respaldo, llegar hasta donde estoy habría sido mucho más complicado y quizá incluso imposible.

Gracias a mi madre por sus enseñanzas, por formar parte de mi vida y por contribuir a mi crecimiento personal. Gracias por ser parte de mi historia y de este camino que culmina con uno de los logros más importantes de mi vida.

Y, finalmente, gracias a mi prometida, Gaby. Gracias por acompañarme durante estos dos años y por estar a mi lado en cada etapa de este proceso. Gracias por ser mi apoyo, mi pilar y mi motivación; por escucharme cuando necesitaba hablar, por tener paciencia cuando las cosas no salían como esperaba y por recordarme, incluso en los momentos en que yo mismo lo olvidaba, que podía lograrlo. Gracias por no soltar mi mano en ningún momento. Por estar ahí en los días buenos, pero sobre todo en aquellos días difíciles en los que el cansancio, la frustración o la incertidumbre hacían que todo pareciera más complicado. Gracias por entender mis ausencias, mis preocupaciones,

mis desvelos y todas esas veces en las que mi mente estaba completamente ocupada. Gracias por celebrar conmigo cada pequeño avance y por hacer tuyos también mis logros. Gracias por creer en mí, por confiar en mí incluso cuando yo mismo dudaba y, sobre todo, por caminar conmigo durante todo este proceso. No sé si habría sido igual sin ti, pero sé que llegar hasta aquí a tu lado lo hizo mucho más especial.

A todos los que, de una u otra manera, estuvieron presentes durante este camino: gracias por confiar en mí, por acompañarme y por formar parte de este logro.

Resumen

El alcance de las investigaciones relacionadas con robots móviles terrestres está creciendo continuamente, por lo que contar con un sistema de control de posición adecuado, capaz de superar las limitaciones asociadas al uso de sensores de posicionamiento global, resulta fundamental. Así, en el presente trabajo se implementa un sistema sensoriomotor para estimar la posición y controlar un robot móvil diferencial a partir de información visual. Se emplea una cámara fija en configuración ojo a mano para obtener las coordenadas en píxeles y establecer, mediante redes neuronales de funciones de base radial, un mapeo directo entre el espacio de la imagen y las coordenadas espaciales del entorno de trabajo. El desempeño de la estimación se evalúa experimentalmente y se compara con el de un método convencional. La configuración propuesta alcanza una precisión promedio de 0.027 m y posteriormente se emplea como retroalimentación para implementar estrategias de control de posición y de seguimiento de trayectoria. El comportamiento del sistema de control se analiza mediante simulaciones y pruebas experimentales. Los resultados muestran que el sistema propuesto permite integrar la percepción visual, la estimación de posición y el control de movimiento en una única arquitectura sensoriomotora.

Palabras clave: sensoriomotor, ojo a mano, red neuronal, función de base radial.

Abstract

The scope of research on terrestrial mobile robots is continually expanding, so an adequate position control system that overcomes the limitations of global positioning sensors is essential. In this work, a sensorimotor system is implemented to estimate position and control a differential mobile robot using visual information. A fixed camera in an eye-to-hand configuration captures pixel coordinates, and radial basis function neural networks establish a direct mapping between image space and the spatial coordinates of the work environment. The estimation performance is evaluated experimentally and compared with a conventional method. The proposed configuration achieves an average precision of 0.027 m and is then used as feedback to implement position control and trajectory tracking strategies. The behavior of the control system is analyzed through simulations and experimental tests. The results show that the proposed system integrates visual perception, position estimation, and motion control within a single sensorimotor architecture.

Keywords: sensorimotor, eye-to-hand, neural network, radial basis function.

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO



FACULTAD DE INGENIERÍA
MAESTRÍA EN CIENCIAS (CONTROL EN SISTEMAS EMBEBIDOS)

Implementación de un sistema sensoriomotor a vehículo terrestre no tripulado

TESIS

Que como parte de los requisitos para obtener el grado de
Maestro en Ciencias

Presenta:

ING. ARTURO IRAM DE LA FUENTE SÁNCHEZ

Dirigido por:

DR. GERARDO ISRAEL PÉREZ SOTO

DR. JESÚS FRANCO ROBLES

Presidente:

DR. GERARDO ISRAEL PÉREZ SOTO

Secretario:

DR. JESÚS FRANCO ROBLES

Vocal:

DR. JUVENAL RODRÍGUEZ RESÉNDIZ

Suplente:

DR. JESÚS ROONEY RIVERA GUILLÉN

Suplente:

DRA. KARLA ANHEL CAMARILLO GÓMEZ

Querétaro, Qro. a agosto de 2026

Índice general

Dedication	I
Agradecimientos	II
Resumen	IV
abstract	V
1. Introducción	1
1.1. Antecedentes	1
1.2. Estado del arte	2
1.3. Justificación	4
1.4. Descripción del problema	5
2. Fundamentación Teórica	6
2.1. Vehículo terrestre no tripulado	6
2.2. Esquema corporal y el sistema sensoriomotor	7
2.2.1. El esquema corporal en la robótica	8
2.3. Inteligencia artificial	8
3. Metodología	12
3.1. Hipótesis	12
3.2. Objetivos	12
3.2.1. Objetivo General	12
3.2.2. Objetivos específicos	12
3.3. Materiales y recursos	13
3.3.1. Buenas prácticas	14
3.4. Metodología	15
3.5. Sistema sensoriomotor y robot terrestre no tripulado	16
4. Diseño, modelado y simulación	18
4.1. Robot Móvil	18
4.1.1. Modelo dinámico del Robot Móvil	20
4.1.2. Modelo dinámico de las ruedas	30
4.1.3. Tarjeta electrónica	33

4.2.	Seguimiento del robot	35
4.2.1.	Método de aprendizaje de la representación espacial.	35
4.2.2.	Método de marcadores ArUco	37
4.3.	Control aplicado al robot móvil	38
4.3.1.	Control de posición para llegar a la meta	39
4.3.2.	Control de seguimiento de trayectoria	43
5.	Resultados y discusión	48
5.1.	Seguimiento del robot	48
5.1.1.	Adquisición sistemática de datos del espacio de trabajo mediante un patrón de referencia.	49
5.1.2.	Método de marcador ArUco	53
5.2.	Simulación del método de aprendizaje de la representación espacial	55
5.3.	Control de posición con seguimiento visual	59
5.3.1.	Control de posición para llegar a la meta basado en representación espacial	59
5.3.2.	Control de posición para llegar a la meta empleando el método de los marcadores ArUco	65
5.4.	Discusión de los resultados	69
5.5.	Control de seguimiento de trayectoria con rastreo visual	72
6.	Conclusiones	78
6.1.	Conclusión	78
6.2.	Impactos	79
6.2.1.	Impacto científico	79
6.2.2.	Impacto tecnológico	79
6.2.3.	Impacto social	79
6.3.	Productos logrados	79
6.4.	Trabajo a futuro	79
	Referencias	84
	A. Código de adquisición de datos del espacio de trabajo	85
	B. Código de simulación	88
	C. Código de microcontrolador Raspberry Pi Pico	92
	D. Código de procesamiento para el método de aprendizaje de la representación espacial	97
	E. Código de procesamiento para el método de marcadores ArUco	103

Índice de figuras

1.1. Red bibliométrica.	2
2.1. Ejemplos populares de las diferentes configuraciones de los robots móviles terrestres.	7
2.2. Perceptrón multicapa.	9
3.1. Diagrama de bloques de la metodología.	15
3.2. Esquema que ilustra la relación entre el sistema sensoriomotor y el control de la posición del robot móvil terrestre.	17
4.1. Bases del robot. Unidades en mm.	18
4.2. Pilares del robot. Unidades en mm.	19
4.3. Vistas del robot móvil.	20
4.4. Marcos de referencia del robot móvil.	22
4.5. Restricción de movimiento por rodamiento puro. Donde R es el radio de la rueda, V_t es la velocidad tangencial y $\dot{\theta}$ es la velocidad angular de la rueda.	23
4.6. Posición y velocidad de las ruedas del robot móvil.	24
4.7. Modelo de un motor de corriente directa que separa la parte eléctrica de la mecánica.	30
4.8. Diagrama de bloques del motor de corriente directa con salida de velocidad.	31
4.9. Diagrama de bloques del motor de corriente directa, incluyendo el servoamplificador.	32
4.10. Lazo de control PI para el motor de corriente directa con servoamplificador.	32
4.11. Diagrama de bloques de las etapas que componen la tarjeta electrónica.	33
4.12. Tarjeta electrónica fabricada.	34
4.13. Diagrama ilustrativo del método de seguimiento propuesto.	35
4.14. Diagrama ilustrativo del método de seguimiento mediante marcadores ArUco.	38
4.15. Esquema general del objetivo del controlador de posición para alcanzar la meta.	40
4.16. Esquema general del objetivo del controlador de seguimiento de trayectoria.	44
5.1. Espacio de trabajo del robot y ubicación de la cámara cenital.	48
5.2. Obtención del conjunto de datos de entrenamiento para el mapeo entre coordenadas en píxeles y coordenadas espaciales.	49
5.3. Curva de RMSE en función del número de neuronas K	50
5.4. Curva de aprendizaje del RMSE en función de los datos de entrenamiento.	51
5.5. Seguimiento de marcador con la predicción neuronal.	52
5.6. Vista de la cámara y presencia de marcadores ArUco.	54
5.7. Distribución del error en función de la posición.	56

5.8. Diagrama de flujo de la simulación del sensor de percepción virtual.	57
5.9. Simulación.	58
5.10. Obtención de la posición angular del robot móvil mediante dos marcadores.	60
5.11. Prueba experimental del controlador de posición de llegada a la meta, con orientación inicial de 0 rad. Método de aprendizaje de la representación espacial.	60
5.12. Gráficas de error de posición en los ejes x y y , y de la norma del error ρ . La prueba experimental es con $\alpha = 0$ rad.	61
5.13. Prueba experimental del controlador de posición de llegada a la meta, con una orientación inicial de $\frac{\pi}{2}$ rad. Método de aprendizaje de la representación espacial.	62
5.14. Gráficas de error de posición en los ejes x y y , y de la norma del error ρ . La prueba experimental es con $\alpha = \frac{\pi}{2}$ rad.	62
5.15. Prueba experimental del controlador de posición de llegada a la meta, con orientación inicial de π rad. Método de aprendizaje de la representación espacial.	63
5.16. Gráficas de error de posición en los ejes x y y , y de la norma del error ρ . La prueba experimental es con $\alpha = \pi$ rad.	64
5.17. Prueba experimental del controlador de posición de llegada a la meta, con orientación inicial de 0 rad. Método de los marcadores ArUco.	65
5.18. Gráficas de error de posición en los ejes x y y y de la norma del error ρ , obtenidas mediante seguimiento visual basado en marcadores ArUco. La prueba experimental se realiza con $\alpha = 0$ rad.	66
5.19. Prueba experimental del controlador de posición de llegada a la meta, con una orientación inicial de $\frac{\pi}{2}$ rad. Método de los marcadores ArUco.	67
5.20. Gráficas de error de posición en los ejes x y y y de la norma del error ρ , obtenidas mediante seguimiento visual basado en marcadores ArUco. La prueba experimental se realiza con $\alpha = \frac{\pi}{2}$ rad.	67
5.21. Prueba experimental del controlador de posición de llegada a la meta, con orientación inicial de π rad. Método de los marcadores ArUco.	68
5.22. Gráficas de error de posición en los ejes x y y y de la norma del error ρ , obtenidas mediante seguimiento visual basado en marcadores ArUco. La prueba experimental se realiza con $\alpha = \pi$ rad.	69
5.23. Gráficas de la prueba experimental del controlador de seguimiento de trayectoria circular, utilizando la estimación de posición basada en aprendizaje.	72
5.24. Gráficas de error de la prueba experimental del controlador de seguimiento de trayectoria circular.	73
5.25. Gráficas de la prueba experimental del controlador de seguimiento de trayectoria lemniscata de Bernoulli.	75
5.26. Gráficas de error de la prueba experimental del controlador de seguimiento de la trayectoria lemniscata de Bernoulli.	76

Índice de tablas

1.1. Estado del arte	3
3.1. Herramientas y componentes del robot móvil.	13
3.2. Especificaciones del equipo de cómputo.	13
3.3. Características de la cámara óptica.	14
3.4. Software empleado.	14
5.1. Métricas de error de seguimiento del marcador móvil (robot) cuando este se encuentra en posición estática. Método de aprendizaje de la representación espacial.	52
5.2. Métricas de error de seguimiento del marcador móvil (robot) cuando este se encuentra en posición estática. Método de marcador ArUco.	54
5.3. Comparación de ambos métodos con los promedios de los errores obtenidos.	70
5.4. Comparación del método propuesto con otras metodologías.	71
5.5. Métricas de error de la prueba experimental del seguimiento de trayectoria circular.	74
5.6. Métricas de error de la prueba experimental del seguimiento de la trayectoria lemniscata de Bernoulli.	76

Introducción

1.1. Antecedentes

En la última década, los vehículos terrestres no tripulados (UGV, por sus siglas en inglés) se han usado ampliamente en investigación y, aunque existen diferentes tipos, una de las configuraciones comúnmente empleadas es la configuración diferencial. El alcance de las investigaciones de estos dispositivos está creciendo continuamente, desde su uso en el sector de la construcción [1] hasta su aplicación en plantas nucleares [2]. Entre las aplicaciones de este tipo de dispositivos se encuentran la reconstrucción digital en entornos cerrados, la evasión de obstáculos en entornos industriales y la teleoperación en escenarios de búsqueda y rescate [3, 4, 5]. Aunque la clasificación de robots móviles varía según diversos factores, como el modo de locomoción, la operación, el ambiente u otras características, en [6] se mencionan cinco tipos: robots biomiméticos, robots con ruedas, robots submarinos, robots aéreos y micro y nanorrobots. Los robots móviles autónomos constan, de manera general, de tres problemas a resolver: la locomoción, la percepción, la localización y la navegación [7]. La locomoción engloba la mecánica, la cinemática, la dinámica y el control del robot. La percepción se relaciona con la tecnología sensorial del robot. Por último, la navegación y la localización constituyen el conjunto de algoritmos y métodos empleados para vincular la información de la percepción y lograr que la locomoción del robot alcance el objetivo deseado, ya sea una posición específica, la evasión de obstáculos o el seguimiento de una trayectoria, entre otros. Actualmente, la mayor parte de los esfuerzos de los investigadores se centra en la búsqueda de metodologías que permitan determinar la localización del robot móvil sin recurrir a sistemas de navegación global satelital (como el GPS), ya que la señal de estos dispositivos puede ser débil o inexistente, lo que impide obtener la localización [8].

Existen diversas razones por las que puede presentarse una falla de la señal del GPS; algunas de ellas se describen en [9]. Con la intención de evitar la pérdida de información sobre la localización del UGV ante un posible fallo del sistema de posicionamiento, se presenta un método de localización basado en la visión.

Dado que el presente trabajo se enfoca exclusivamente en vehículos terrestres no tripulados, en lo sucesivo, el término robot móvil se referirá a este tipo de plataformas y ambos términos se emplearán indistintamente.

1.2. Estado del arte

En el trabajo de Silva Oliveira et al. [10] menciona que, al analizar las técnicas empleadas para la localización de un robot móvil, los métodos basados en visión por computadora son más eficientes que otros en la precisión y tiempo de localización y de ejecución de tareas. Cuando se diseña un controlador visual, existen dos configuraciones básicas: ojo en mano y ojo a mano [11]. En la configuración ojo en mano, la cámara se encuentra en el robot móvil y se desplaza junto con él. Por otro lado, en la configuración ojo a mano, la cámara se coloca por encima del robot o, por lo general, sobre el área de trabajo del robot y se fija al techo. Esta última configuración es la que se emplea.

La Figura 1.1 presenta un mapa bibliométrico generado con VOSviewer que ilustra las relaciones entre las palabras clave asociadas a robots móviles que emplean visión y control de posición. Este contexto general abarca una amplia área de búsqueda; sin embargo, el estado del arte del presente proyecto se centra específicamente en metodologías que emplean sistemas de visión cenital, en configuración ojo a mano y que realizan el control de posición.

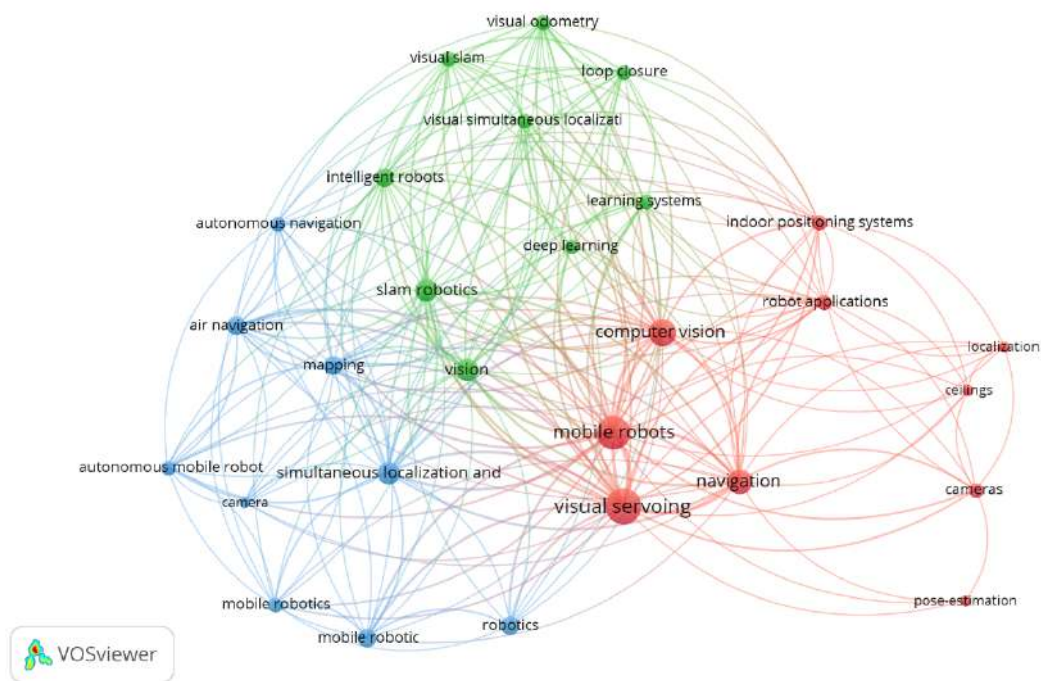


Figura 1.1: Red bibliométrica.

La Tabla 1.1 presenta un listado de trabajos en los que se ajusta la localización de robots móviles terrestres mediante la técnica ojo a mano y de otros que utilizan una propuesta sensoriomotora.

Tabla 1.1: Estado del arte

Referencia	Año	Método de ajuste	Dispositivo	Resultados	Error
[12]	2006	Control no lineal	UGV y cámara	Teórico	-
[13]	2013	Control de 2 grados de libertad para rastreo de trayectoria	UGV y cámara	Experimental	$SAE = 1505$ px
[14]	2015	Red de Gas Neuronal - Aprendizaje no supervisado	Microvehículo aéreo	Simulación	-
[15]	2020	Control basado en ángulo. Control basado en gráficos	UGV y cámara	Experimental	-
[16]	2020	Rastreo dinámico local empleando control diferencial	UGV y cámara	Experimental	≈ 5 cm
[17]	2021	Red de Gas Neuronal - Aprendizaje no supervisado	Dron cuadricóptero	Simulación	-
[18]	2022	Control cinemático diferencial	UGV y dos cámaras	Experimental	≈ 4 mm
[19]	2023	Mapa cortical neuronal de 2 niveles	Dron cuadricóptero	Simulación	-
[20]	2026	Mapa cortical neuronal y LQR	Dron cuadricóptero	Simulación	-

Para delimitar el alcance del estado del arte se distingue entre los trabajos que emplean información visual directamente en la ley de control y aquellos que utilizan un sistema de visión externa como sensor para la localización del robot móvil. En particular, en [13] y [15] se presentan enfoques de servocontrol visual en los que la información visual participa directamente en la estrategia de control. Estos trabajos son antecedentes de la aplicación de sistemas de visión en el control de robots móviles, pero no constituyen una referencia metodológica directa para la propuesta que se desarrolla en el presente trabajo.

Con respecto de la Tabla 1.1, el trabajo [12] presenta un sistema visual que emplea un vehículo aéreo y desglosa la propuesta matemática de control de pose de un robot móvil mediante retroalimentación de cámara. En este trabajo se presenta el análisis de estabilidad, que demuestra la convergencia asintótica; sin embargo, no incluye pruebas experimentales.

Otra propuesta se observa en el trabajo de Zdešar et al. [13], en este caso se rastrea la posición marcando el robot para su reconocimiento y, posteriormente, mediante un subsistema de visión por computadora. Se combinan un lazo de realimentación y otro de avance para implementar un control de dos grados de libertad. El lazo de avance conduce el sistema a la trayectoria de referencia, y el lazo de realimentación elimina los errores debidos a las perturbaciones. Además, se emplean: un filtro de Kalman para estimar los estados del robot móvil y un modelo predictivo de control. Se concluye que la implementación del sistema presenta un controlador robusto frente al ruido y un buen desempeño en el rastreo de la trayectoria. No obstante, el algoritmo de control parte de la suposición de que la homografía de la cámara es conocida con precisión, aunque en realidad se estima durante el proceso de calibración.

En [15] se emplean métodos de control “ir a la meta” basados en visión para robots móviles interiores no holonómicos: el basado en gráficos y el basado en ángulo. La investigación se divide en tres etapas: la primera consiste en calibrar la cámara, colocar en el robot figuras geométricas de colores diferentes a los del suelo y emplear filtros de visión por computadora para detectar bordes y obtener las coordenadas del robot móvil. En la segunda etapa se obtiene la posición del robot con respecto al objetivo mediante una representación gráfica y una representación triangular. La última etapa presenta los resultados obtenidos al aplicar los métodos de control mencionados, además de una comparación entre el sistema propuesto y un controlador PID difuso y un controlador PID clásico. En este trabajo no se presentan públicamente métricas de error para analizar el desempeño del controlador.

En [16] se utiliza la cámara para generar un mapeo global; de esta manera, la identificación y el posicionamiento del robot móvil se obtienen mediante un algoritmo que combina el método de diferencia entre cuadros con un rastreo dinámico local. Para construir el mapa global, los autores convierten las coordenadas físicas en píxeles a partir de 4 puntos característicos del espacio de trabajo del robot; posteriormente, se calculan las coordenadas en píxeles mediante una transformación proyectiva. Posterior a la construcción del mapa global, se emplean métodos de visión por computadora para rastrear y posicionar el robot móvil en todo momento. Se emplea el control diferencial para ajustar dinámicamente el estado del recorrido lineal.

Por último, en [18] presentan una propuesta que emplea dos cámaras: una ubicada perpendicularmente al suelo y otra, en un ángulo de 45 grados, alejada de la primera a 2 metros. También emplean una marca en el robot; en este caso, corresponde a un sistema de posicionamiento de código abierto llamado Apriltag. Se menciona que, para obtener la orientación del robot, se emplean 2 marcas distintas: una en el centro y otra en la parte frontal, inmediatamente después de la anterior, de forma paralela. Además de la localización del robot, se presenta un algoritmo de evasión de obstáculos que emplea un sensor ultrasónico para modificar la trayectoria ante situaciones inesperadas. Se concluye que el sistema propuesto presenta alta flexibilidad, confiabilidad, expansibilidad y precisión móvil, además de ventajas frente a otros sistemas en términos de costo. El sistema se implementa mediante una transformación de perspectiva para observar y procesar el área de trabajo del robot móvil. Este artículo se presenta de color verde en la Tabla 1.1 porque, si bien el trabajo provee la base teórica para que los errores que se presentan puedan convertirse en distancias reales, no brinda los parámetros de calibración ni la homografía exacta para corroborar la información.

En el presente trabajo de tesis se propone un método de localización para un dispositivo UGV, en específico, de un robot móvil con ruedas de configuración diferencial, que utiliza un sistema de visión. La información visual no se incorpora directamente en la ley de control. En su lugar, el sistema de visión es el mecanismo de percepción que se encarga de estimar la posición del robot en su espacio de trabajo.

Como se aprecia en el análisis de los trabajos de la Tabla 1.1, suele requerirse, para la identificación del robot móvil en el espacio, el uso de transformaciones proyectivas, la calibración de la cámara o procedimientos geométricos para establecer la correspondencia entre las coordenadas de la imagen y las coordenadas físicas del espacio de trabajo. Además de la aplicación de filtros que reducen la incertidumbre en la localización del robot. En este sentido, el método propuesto emplea un sistema de visión basado en redes neuronales para aprender directamente la relación entre las coordenadas del plano de imagen y las coordenadas espaciales, evitando la necesidad de una transformación de perspectiva explícita o de una calibración geométrica de la cámara. Así, la principal diferencia con los trabajos analizados es que la información visual no se incorpora en la ley de control, sino que constituye el mecanismo mediante el cual se transforma la información en una estimación de la posición. Se encuentra inspiración en el sistema bioinspirado que presentan los trabajos en color amarillo de la Tabla 1.1 y así se relaciona el sistema sensoriomotor humano con el autoajuste de la localización de un robot móvil con ruedas.

1.3. Justificación

Cuando se habla de tareas que requieren el uso de vehículos terrestres no tripulados, tales como reconocimiento, vigilancia, mapeo geográfico y rescate de emergencia, entre otras, contar con un

sistema de navegación autónomo permite ampliar el alcance de su aplicación. Debido a lo anterior, el control de posición de los dispositivos UGV es un campo ampliamente abordado por la comunidad de investigación. Dado que estos dispositivos se emplean en actividades que abarcan desde el sector militar hasta el sector civil [21], lograr un control capaz de garantizar que un robot móvil esté apto para seguir la posición deseada, considerando las restricciones en las que se encuentre, es una tarea que debe realizarse, ya que la tendencia de estos dispositivos apunta a un cambio significativo de las operaciones terrestres y de la vida humana.

Aún persisten desafíos importantes en la navegación autónoma, la localización, la percepción y el control en tiempo real [22, 23]; por ello, seguirá siendo necesaria la investigación en esta área.

En el presente trabajo de investigación se aplicarán métodos de inteligencia artificial para proponer un método de localización de un robot móvil con ruedas de configuración diferencial, relacionándolo con los conceptos del sistema sensoriomotor humano.

1.4. Descripción del problema

La obtención de la localización de vehículos terrestres no tripulados es una tarea que debe realizarse debido al incremento de las aplicaciones en las que se han visto involucrados recientemente. Para localizar un dispositivo UGV, se deben tener en cuenta diversos factores que pueden o no afectar directamente. Como se observa en la sección 1.2, un factor relevante es el uso de un GPS, que, según las condiciones del entorno, puede perder información sobre la posición del dispositivo. Además, existen intentos de solucionar el problema del GPS mediante métodos que ofrecen un arreglo parcial y solo por un breve tiempo; sin embargo, lo ideal es que la solución funcione independientemente de la duración de la falla del dispositivo.

Por otro lado, es conveniente que el UGV sea capaz de identificar su posición, ya sea estática o dinámica, en cualquier instante, incluso en condiciones no deseadas o extraordinarias. Sobre todo en espacios interiores.

En este trabajo se propone un método para resolver el problema descrito anteriormente mediante una cámara cenital que proporcione la posición de un vehículo terrestre no tripulado en el plano. Esta implementación se inspira en el esquema sensoriomotor humano y lo relaciona con un robot móvil.

Fundamentación Teórica

En esta sección, se presentan los fundamentos teóricos necesarios para el presente trabajo de tesis.

2.1. Vehículo terrestre no tripulado

Los vehículos terrestres no tripulados, o UGVs, son máquinas móviles que incorporan sensores y sistemas de control y realizan actividades de transporte, detección de variables físicas y trazado de rutas, entre otras. [24]. En la literatura también se les conoce como robots móviles terrestres, vehículos autónomos no tripulados y vehículos guiados automatizados.

Existe un gran número de configuraciones de ruedas en los vehículos terrestres, según el número de ruedas. Cada una presenta ventajas y desventajas distintas. Algunas de ellas son [7]:

- Configuración de dos ruedas, con una rueda de dirección al frente y otra de tracción trasera.
- Configuración de dos ruedas centradas de transmisión diferencial con un tercer punto de contacto.
- Configuración de tres ruedas motorizadas en triángulo con movimiento omnidireccional.
- Configuración de cuatro ruedas, con dos motorizadas en la parte trasera y dos de dirección en la parte delantera.

En la Figura 2.1 se muestran ejemplos de los tipos de configuración de robots móviles mencionados anteriormente.

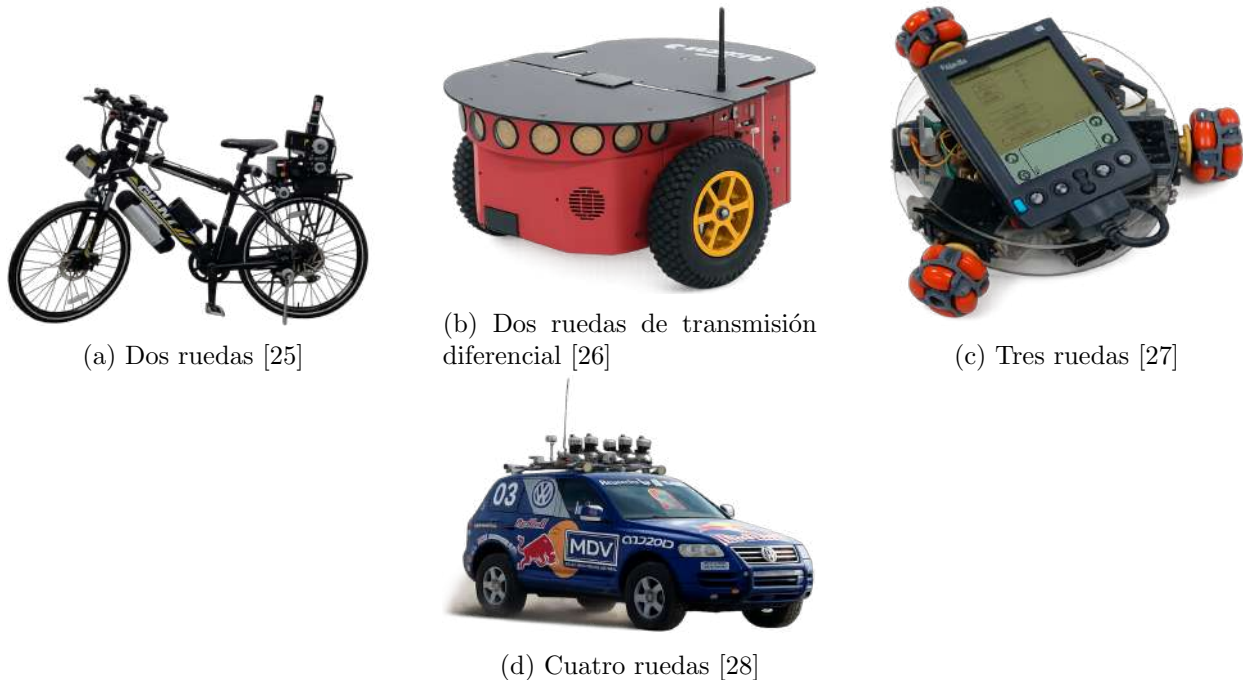


Figura 2.1: Ejemplos populares de las diferentes configuraciones de los robots móviles terrestres.

2.2. Esquema corporal y el sistema sensoriomotor

En referencia a los seres humanos, el esquema corporal (BoS) es el término que designa la conciencia del propio cuerpo, que incluye la ubicación y orientación de sus extremidades y su movimiento relativo en el espacio y el tiempo, así como su integridad funcional [29]. En otras palabras, son procesos inconscientes que nos permiten saber dónde se encuentran las partes del cuerpo en el espacio, lo que nos permite utilizar nuestras extremidades sin tener que prestarles atención [30].

El esquema corporal contiene información sobre cómo se combinan las informaciones cinestésicas, propioceptivas, táctiles y visuales para que tengamos un modelo actualizado de nuestro cuerpo en todo momento [31]. Dicho modelo encapsula tanto la postura como la superficie corporal para recrear el sistema sensoriomotor [17]. El esquema corporal biológico relaciona múltiples señales de control sensoriomotoras entre sí [32]. Las diferentes modalidades de sensores se pueden dividir en:

- Exterocepción: percepción de sensaciones que pueden provenir de la superficie corporal o del entorno y son perceptibles a través de los sentidos.
- Propiocepción: percepción del propio estado corporal.

Por lo tanto, el sistema sensoriomotor se refiere a la relación entre los sistemas sensorial y motor, que comprende tanto el ciclo de percepción como el de acción. Incluye la obtención de estímulos del entorno a través de los sentidos (vista, tacto, gusto, olfato, oído); posteriormente, esa información llega al cerebro para la planificación y la ejecución de movimientos específicos como respuesta [33].

2.2.1. El esquema corporal en la robótica

En la ingeniería de control y la robótica, por lo general se dispone de un modelo explícito del robot para su control. El modelo de un robot es una representación que se utiliza para guiar las acciones y puede asemejarse a la conceptualización del esquema corporal. Sin embargo, como se revisó previamente, las características de este tipo de modelos no constituyen del todo un esquema corporal biológico. Es deseable que un robot pueda desarrollar, calibrar y adaptar sus modelos de forma autónoma. Para lograrlo, además del modelo explícito, se recurre a un modelo implícito, tradicionalmente entendido como la integración de un sistema de aprendizaje. Dentro de los sistemas basados en redes neuronales artificiales, los modelos bayesianos son populares en este campo [34].

Una forma de recrear el esquema corporal en robots es mediante el aprendizaje automático (*machine learning*), que consiste principalmente en construir mapeos entre las señales exteroceptivas y propioceptivas, conocidos como mapas sensoriomotores. La creación de estos mapas generalmente se basa en una regresión funcional supervisada y continua que aprende a partir de observaciones en forma de valores de entrada propioceptivos y de salida exteroceptivos. Un ejemplo de relación sensoriomotora es la cinemática directa, uno de los aspectos fundamentales del esquema corporal [32]. El aprendizaje del modelo de un robot es un problema de regresión en el que las muestras de entrenamiento se obtienen de la percepción y el estado del sistema a lo largo del tiempo [35].

2.3. Inteligencia artificial

La inteligencia artificial (IA) es la capacidad de una máquina para emplear algoritmos, aprender a partir de los datos que recibe y aplicar ese aprendizaje en diversas tareas, de manera similar a como lo haría un ser humano [36]. Una de sus ramas es el aprendizaje automático y, dentro de esta, se encuentran las redes neuronales artificiales.

Redes Neuronales Artificiales

Las redes neuronales artificiales (RNA) son modelos no lineales inspirados en el funcionamiento del cerebro humano. Tienen una estructura basada en principios observados en los sistemas nerviosos de humanos y animales, compuesta por unidades de procesamiento, o neuronas, que se conectan entre sí. Dichas conexiones presentan una cierta intensidad que es determinada por los pesos sinápticos. La Figura 2.2 muestra un esquema de una red típica completamente conectada, llamada perceptrón multicapa.

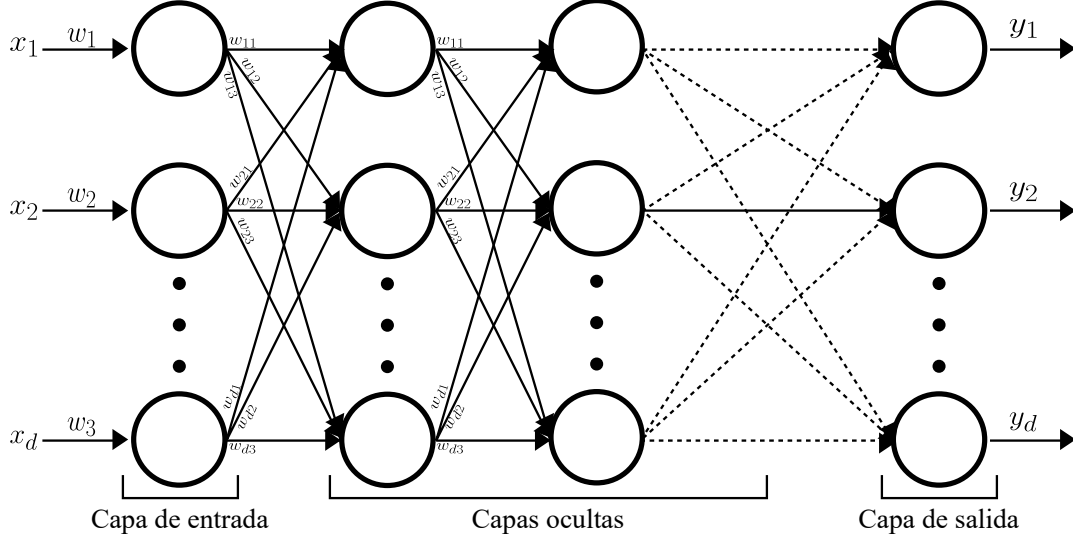


Figura 2.2: Perceptrón multicapa.

Para obtener la salida y de una neurona, primero se debe realizar la suma ponderada a de las entradas, que se conoce como la activación de la neurona.

$$a = \sum_{i=1}^d w_i x_i \quad (2.1)$$

donde:

w_i : peso de la conexión

x_i : señales de entrada

A partir del valor a , se obtiene la salida y mediante una función de activación, como se muestra en (2.2).

$$y = g(a) = g\left(\sum_{i=1}^d w_i x_i\right) \quad (2.2)$$

La función de activación actúa como un filtro o limitador mediante un umbral y modifica el valor de activación antes de que este se transmita a otra neurona. En muchos casos, la función de activación aporta la capacidad de modelar relaciones no lineales de los datos. Algunos ejemplos de funciones de activación son:

1. Función escalón:

$$g(a) = \begin{cases} 0 & \text{si } a < 0 \\ 1 & \text{si } a \geq 0 \end{cases} \quad (2.3)$$

2. Función lineal:

$$g(a) = a \quad (2.4)$$

3. Función sigmoidea:

$$g(a) = \frac{1}{1 + e^{-a}} \quad (2.5)$$

4. Función tangente hiperbólica:

$$g(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \quad (2.6)$$

5. Función Gaussiana:

$$g(a) = \exp\left(-\frac{(a - \mu)^2}{2\sigma^2}\right) \quad (2.7)$$

Cada neurona recibe como entrada la salida de otras neuronas o datos externos, para posteriormente procesar la información y emitir una salida que se distribuye a otras neuronas.

Observando nuevamente la Figura 2.2, las entradas externas se introducen en la primera capa, la capa de entrada, y las salidas se obtienen en la última capa, la capa de salida. A las capas internas se les denomina capas ocultas.

Entrenamiento de Redes Neuronales

Las redes neuronales reciben datos de entrada y los transforman para producir una salida deseada. Esta transformación consiste en ajustar los parámetros de la red neuronal. Dichos parámetros suelen ser los pesos sinápticos, que constituyen el medio mediante el cual la red neuronal artificial almacena conocimiento sobre el problema a resolver.

El conocimiento de la red neuronal se almacena mediante el aprendizaje o entrenamiento y, al igual que el aprendizaje humano, este se basa idealmente en el uso de ejemplos que representan el problema. A estos ejemplos se les conoce como conjunto de entrenamiento.

Debido a la forma en que se constituye el conjunto de entrenamiento, se distinguen dos tipos de aprendizaje: supervisado y no supervisado.

- Supervisado: consiste en introducir entradas etiquetadas que corresponden a las salidas deseadas.
- No supervisado: consiste en encontrar características entre grupos (clusters) de patrones de entrada; en este caso, no se tiene conocimiento directo de las salidas deseadas.

Redes Neuronales de Función de Base Radial

Las funciones de base radial (RBF) presentan un comportamiento característico en el que su respuesta decrece o aumenta de manera monótonica con la distancia a un punto central. Estas pueden emplearse en modelos lineales y no lineales, así como en redes de una capa o multicapa; sin embargo, las redes de función de base radial tradicionalmente se asocian a redes de una sola capa oculta, cuyas neuronas emplean funciones de base radial como funciones de activación [37].

Las funciones de base radial construyen un espacio lineal de funciones que depende de las posiciones de los puntos de datos conocidos, acordes con una distancia arbitraria. Es decir:

$$s(x) = \sum_{j=1}^m \lambda_j \phi(\|x - \varsigma_j\|), \quad x \in \mathbb{R}^n \quad (2.8)$$

donde $s(x)$ son las funciones interpolantes, x es el vector de datos de entrada, m es el número de funciones de base, λ_j es el peso asociado a la j -ésima función, $\phi(\cdot)$ es la función de base radial, $\|\cdot\|$ es la norma euclidiana y ς_j corresponde al centro de la j -ésima función de base [38].

La función de base radial representa la función de activación y la más común es la función gaussiana (2.7). Para el presente caso de las redes RBF, se puede representar esta función como:

$$\phi_j(x) = \exp\left(-\frac{\|x - \varsigma_j\|^2}{2\sigma^2}\right) \quad (2.9)$$

donde ς es el centroide y σ es el ancho de la función gaussiana, también llamado factor de suavidad [39].

La etapa de aprendizaje de la red neuronal consiste en ajustar sus parámetros a partir de los datos de entrenamiento. En la red RBF, se involucra la determinación de los centros de las funciones radiales, sus parámetros de dispersión y los pesos de conexión hacia la capa de salida. Además, es necesario establecer determinados hiperparámetros, como el número de funciones de base empleadas. Para lograr lo anterior, una de las técnicas es el *clustering*.

El *clustering* se emplea para encontrar el conjunto de centros en función de la distribución de los datos de entrada. Una estrategia ampliamente utilizada es el algoritmo de *K-means* [40]. Esta metodología presenta dos desventajas: la primera es que se pueden obtener largos periodos de convergencia, y la segunda es que puede convertirse en un proceso de prueba y error para establecer el tamaño de la capa oculta, dado que el número de centros no puede determinarse automáticamente [41]. Sin embargo, la selección del número de centros K puede requerir procedimientos adicionales para determinar un valor adecuado; por ejemplo, la estrategia del método del codo, que analiza la variación del error de agrupamiento en función del número de grupos.

Metodología

En esta sección se desglosa la metodología empleada en el presente trabajo de tesis.

3.1. Hipótesis

Con la implementación de un controlador de posición para un vehículo terrestre no tripulado, basado en una metodología sensoriomotora e inteligencia artificial, se logra el autoajuste de la localización con un error de traslación absoluto máximo inferior a 5 cm.

3.2. Objetivos

En esta sección se presentan el objetivo general y los objetivos específicos de este trabajo de tesis.

3.2.1. Objetivo General

Implementar un sistema sensoriomotor que emplee inteligencia artificial para lograr el autoajuste de la localización de un UGV.

3.2.2. Objetivos específicos

- Obtener la localización del dispositivo mediante redes neuronales para determinar la posición en un espacio dado.
- Comparar la información de localización obtenida mediante inteligencia artificial con los datos proporcionados por otros métodos para reducir el error de localización del dispositivo.
- Realizar una simulación del sistema con un software especializado en robótica para evaluar el desempeño del entrenamiento neuronal.
- Instrumentar un banco de pruebas y adaptar el espacio existente para implementar el controlador de posición en un UGV real.
- Comparar los resultados reales con los de simulación para observar las diferencias en el comportamiento del sistema.

3.3. Materiales y recursos

En esta sección se describen, de manera general, los recursos que se utilizan en el presente trabajo de tesis.

La Tabla 3.1 presenta los materiales más importantes empleados en la construcción del robot móvil.

Tabla 3.1: Herramientas y componentes del robot móvil.

Nombre	Descripción
Motorreductor con encoder Velocidad: 100 rpm Resolución: 638 ppr Alimentación: 12 V	Actuador de cada una de las ruedas del robot móvil
Raspberry Pi Pico	Se utiliza para la recepción y envío de datos. Ejecuta los comandos de acción para operar los motorreductores. Fija el periodo de muestreo
Bluetooth HC-05	Encargado de la comunicación inalámbrica entre la computadora y el robot móvil
Batería tipo LiPo	Encargada de suministrar energía eléctrica al sistema
Impresora 3D Creality Ender-3	Utilizada para crear la estructura del robot móvil
Filamento PLA de 1.75 mm	Material de fabricación de la estructura del robot

La Tabla 3.2 presenta las características del equipo de cómputo que se emplea en el presente trabajo.

Tabla 3.2: Especificaciones del equipo de cómputo.

Modelo	HP Pavilion Laptop 15-cw1012la
Procesador	AMD Ryzen 3 3300U
Unidad de procesamiento gráfico	AMD Radeon Vega 6 Graphics (2.10 GHz)
RAM instalada	12 Gb
Arquitectura del procesador	64 bits
Sistema operativo	Windows 11 <i>Home single</i>

La Tabla 3.3 presenta las especificaciones de la cámara óptica.

Tabla 3.3: Características de la cámara óptica.

Modelo	Intel RealSense D435i
Sensor de imagen	OmniVision OV2740
Resolución del sensor	2 Mpx
Resolución máxima de video	1920 × 1080 px
Velocidad de captura	30 fps
Campo de visión (FOV)	Horizontal 69.4° Vertical 42.5°

La descripción de la Tabla 3.3 corresponde al sensor RGB de la cámara. La adquisición de imagen se realiza mediante la interfaz estándar de video USB (UVC). La cámara es capaz de proporcionar información de estimación de profundidad, sensores infrarrojos y funciones avanzadas de procesamiento; no obstante, no se hace uso de ninguna de estas características.

La Tabla 3.4 muestra el software más relevante empleado para la adquisición y el procesamiento de video, la programación del microcontrolador y la ejecución e implementación de los controladores.

Tabla 3.4: Software empleado.

Nombre	Descripción
Python 3.13.5	Lenguaje de programación para la escritura de código referente al procesamiento de video, entrenamiento neuronal y ejecución de la ley de control
C++	Lenguaje de programación para la escritura de código embebido del microcontrolador
Visual Studio Code 1.106.2	Editor de código para la programación y ejecución de todos los programas
Librería OpenCV 4.13.0	Para la adquisición y procesamiento de imágenes y video
Librería Threading	Uso de hilos para la división de tareas
Librería SKLearn 1.8.0	Para el entrenamiento neuronal.
Librería Scipy 1.16.3	Para la implementación de métodos numéricos de integración

3.3.1. Buenas prácticas

Durante el desarrollo de este proyecto, se acatan las normas y lineamientos de seguridad establecidos en los laboratorios y espacios de la Facultad de Ingeniería de la Universidad Autónoma de Querétaro, para salvaguardar la integridad física de los involucrados en el proyecto, así como de las terceras personas que puedan estar haciendo uso del mismo espacio. De igual manera, todo el equipo

y los materiales se manejan conforme a las especificaciones de los fabricantes de los dispositivos.

Todos los residuos, incluidos los componentes electrónicos y/o las baterías, se depositan en los lugares correspondientes.

3.4. Metodología

La Figura 3.1 muestra un diagrama de bloques de los pasos generales de la realización del presente trabajo de tesis.

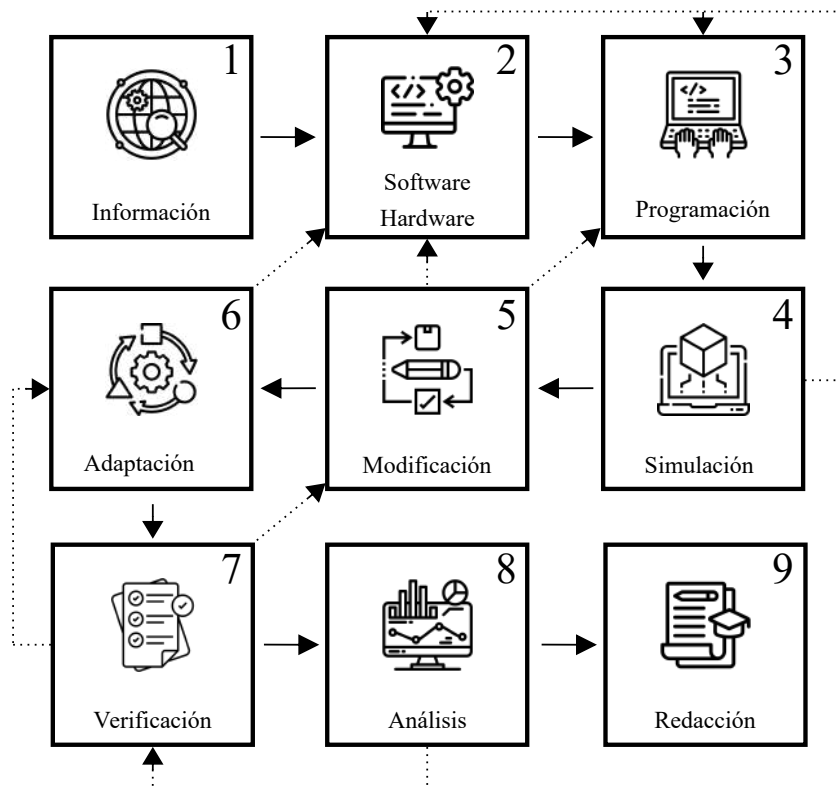


Figura 3.1: Diagrama de bloques de la metodología.

1. Búsqueda de información: Conocer la estructura y el funcionamiento de las redes neuronales artificiales. Entender el sistema sensoriomotor humano y comprender la relación con la robótica.
2. Recopilación de materiales y herramientas: Identificar los materiales, herramientas, software y hardware necesarios para alcanzar los objetivos.
3. Aprendizaje de programación e implementación de las RNA: Conocer las librerías de programación que permiten emplear las RNA y adaptarlas adecuadamente.
4. Simulación del sistema: Realizar una simulación con software para observar la respuesta del dispositivo UGV derivada del entrenamiento neuronal.

5. Modificación del dispositivo: Variar los componentes y materiales necesarios del dispositivo UGV para lograr la comunicación efectiva con los datos de entrenamiento.
6. Adaptación del espacio de trabajo: Adecuar el entorno de trabajo del dispositivo para permitir el uso adecuado del sistema propuesto.
7. Verificación de funcionamiento: Analizar la respuesta obtenida durante la implementación del sistema propuesto y comparar los resultados con los de la simulación.
8. Análisis de resultados: Observar los resultados obtenidos y ajustar los parámetros de entrenamiento para obtener los resultados deseados.
9. Redacción de tesis.

En el contexto del presente trabajo, el término “autoajuste de la localización” se refiere a la capacidad del sistema para obtener y actualizar automáticamente la estimación de la posición del robot móvil a partir de información visual. El concepto no implica (ni intenta) una adaptación en línea de los parámetros de la red neuronal ni de las ganancias del controlador.

3.5. Sistema sensoriomotor y robot terrestre no tripulado

La inspiración biológica del sistema propuesto se basa en el funcionamiento del sistema sensoriomotor humano, como se ilustra en la Figura 3.2. En el ser humano, la información proveniente de la percepción visual es procesada por el sistema nervioso para construir una representación interna del espacio que permite conocer la ubicación del propio cuerpo con respecto al entorno y al objetivo a alcanzar. Esta representación, asociada al concepto de conciencia espacial y, en general, al esquema corporal descrito en la Sección 2.2, constituye un elemento fundamental para la planificación y la ejecución del movimiento.

Análogamente, en el robot móvil terrestre, la percepción visual se implementa mediante una cámara cenital que captura continuamente el entorno de trabajo. Sin embargo, la imagen adquirida no proporciona directamente la representación espacial del robot. Más aún, dicha información se procesa mediante una red neuronal de función de base radial previamente entrenada para aprender el espacio de trabajo del sistema, lo que genera una representación interna que permite estimar la posición del robot respecto al objetivo. Esta representación cumple un papel equivalente al de la conciencia espacial del sistema biológico y sirve de referencia para el controlador de posición encargado de ejecutar el movimiento del robot. Esta analogía funcional no pretende la equivalencia biológica literal, sino únicamente servir para estructurar el diseño del sistema. La función de la red neuronal no se limita a estimar la posición del robot, sino que también abarca el aprendizaje de la representación continua del espacio de trabajo a partir de la información visual. En este sentido, la red neuronal actúa como un modelo interno del entorno, lo que permite transformar la percepción visual en información espacial aprovechable para el controlador.

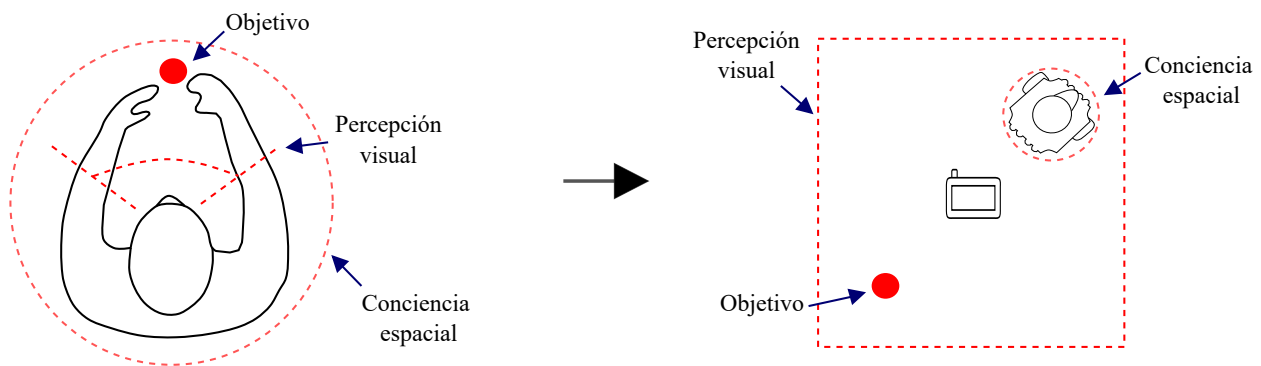


Figura 3.2: Esquema que ilustra la relación entre el sistema sensoriomotor y el control de la posición del robot móvil terrestre.

Diseño, modelado y simulación

4.1. Robot Móvil

El robot móvil empleado en el presente proyecto encuentra inspiración en el TurtleBot3 de la marca Robotis [42]. Para la estructura del robot se fabrican placas y pilares mediante impresión 3D; las dimensiones de ambos se muestran en las Figuras 4.1 y 4.2.

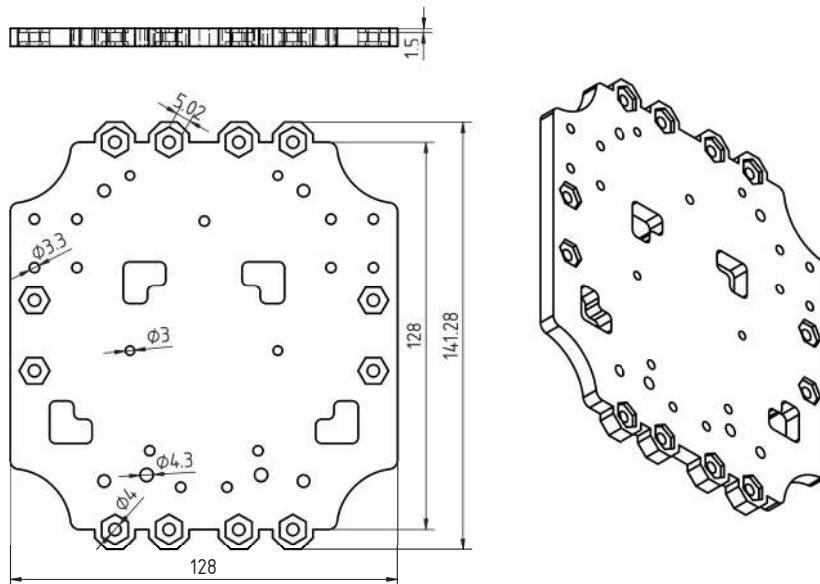


Figura 4.1: Bases del robot. Unidades en mm.

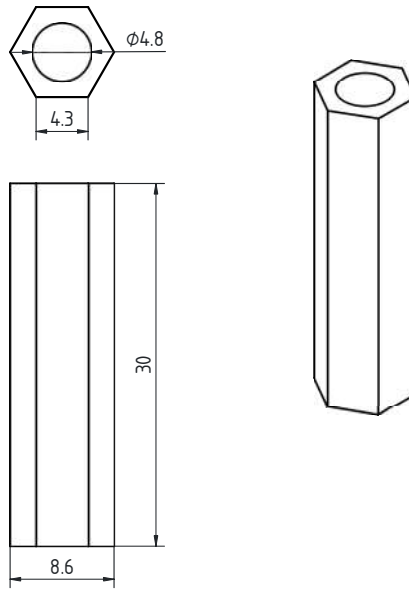


Figura 4.2: Pilares del robot. Unidades en mm.

Al realizar el ensamble de los componentes mostrados en la Tabla 3.1, junto con las bases y los pilares mostrados en las Figuras 4.1 y 4.2. El robot móvil armado se muestra en la Figura 4.3.

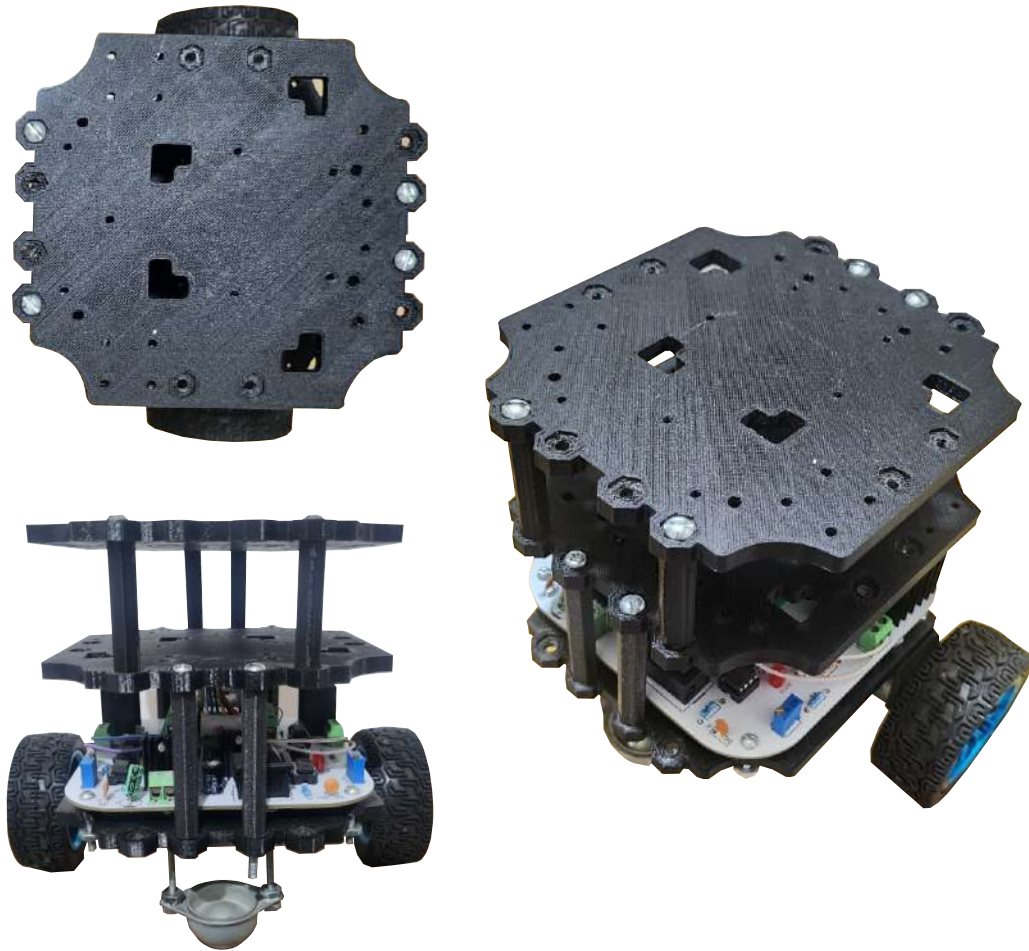


Figura 4.3: Vistas del robot móvil.

Como se observa en la Figura 4.3, el robot está formado por tres niveles. En el primero se ubican los componentes electrónicos encargados del control y del accionamiento de los motorreductores. El nivel intermedio se reserva como un espacio destinado a facilitar la incorporación de sensores, dispositivos de procesamiento o sistemas de adquisición de datos requeridos para futuras etapas de desarrollo, sin necesidad de modificar la estructura mecánica del robot. El nivel superior se destina a la colocación de los marcadores (diferenciadores) detectados por el sistema de visión, a fin de asegurar una adecuada visibilidad desde la cámara y minimizar posibles oclusiones durante la navegación.

4.1.1. Modelo dinámico del Robot Móvil

El modelo dinámico que se desarrolla en esta sección no forma parte de las leyes de control, en las que únicamente se emplea el modelo cinemático. Sin embargo, la finalidad es contar con una representación del robot para la etapa de simulación, en la que las acciones de control se aplican al modelo dinámico. De esta manera, la respuesta del sistema, al considerar sus características físicas, complementa los resultados experimentales.

Marcos de referencia

Para modelar matemáticamente el robot, es necesario definir un marco de referencia inercial y otro móvil. El marco de referencia inercial, denotado como $\{x_i, y_i\}$, es el marco de referencia global y fijo ubicado en el plano en el que el robot tendrá movimiento. El marco de referencia móvil, denotado como $\{x_r, y_r\}$, es el marco de referencia local, ligado al robot y que se desplaza en conjunto con el dispositivo.

Los dos marcos de referencia se muestran en la Figura 4.4. El origen del marco de referencia móvil es el punto medio del eje de ambas ruedas dado por o_r en las coordenadas globales (x_{or}, y_{or}) .

La posición del robot con respecto al marco inercial es:

$$P^i = \begin{bmatrix} x_{or} \\ y_{or} \\ \alpha_i \end{bmatrix} \quad (4.1)$$

Para relacionar los marcos de referencia, dado que el marco de referencia móvil puede cambiar su orientación, se emplea la matriz de rotación:

$$R(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.2)$$

por lo que, la posición y orientación del robot en cualquier punto, es:

$$P^i = R(\alpha)P^r$$

$$\begin{bmatrix} x^i \\ y^i \\ \alpha^i \end{bmatrix} = R(\alpha) \begin{bmatrix} x^r \\ y^r \\ \alpha^r \end{bmatrix} \quad (4.3)$$

y la velocidad es:

$$\dot{P}^i = R(\alpha)\dot{P}^r$$

$$\begin{bmatrix} \dot{x}^i \\ \dot{y}^i \\ \dot{\alpha}^i \end{bmatrix} = R(\alpha) \begin{bmatrix} \dot{x}^r \\ \dot{y}^r \\ \dot{\alpha}^r \end{bmatrix} \quad (4.4)$$

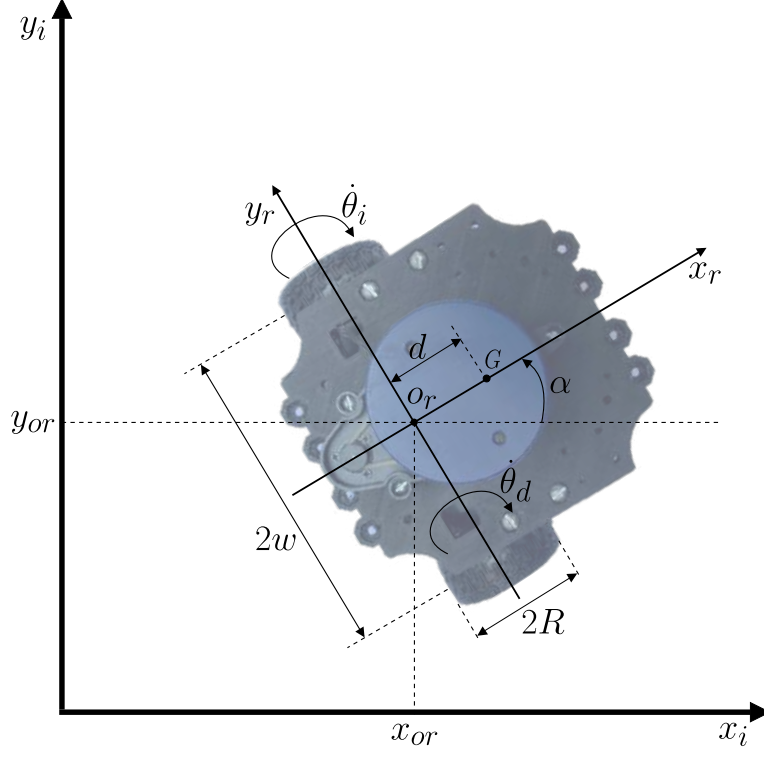


Figura 4.4: Marcos de referencia del robot móvil.

Con respecto de la Figura 4.4, $\dot{\theta}_i$ y $\dot{\theta}_d$ son las velocidades angulares de las ruedas izquierda y derecha, w es la distancia medida desde el eje x_r hasta el centro de las ruedas, y d es la distancia desde el centro de ambas ruedas hasta el centro de masa del robot móvil G .

Restricciones no holonómicas

El movimiento del robot tiene dos restricciones no holonómicas, las cuales se basan en las siguientes suposiciones:

- No hay movimiento lateral en el eje y_r : Esta restricción establece que el robot no puede desplazarse únicamente a lo largo de dicho y_r . La posición del robot con respecto al marco de referencia móvil es:

$$\begin{bmatrix} x^r \\ y^r \\ \alpha^r \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{or} \\ y_{or} \\ \alpha_i \end{bmatrix} = \begin{bmatrix} x_{or} \cos(\alpha) + y_{or} \sin(\alpha) \\ -x_{or} \sin(\alpha) + y_{or} \cos(\alpha) \\ \alpha_i \end{bmatrix} \quad (4.5)$$

La velocidad del robot es:

$$\frac{d}{dt} \begin{bmatrix} x^r \\ y^r \\ \alpha^r \end{bmatrix} = \begin{bmatrix} \dot{x}_{or} \cos(\alpha) - x_{or} \sin(\alpha) \dot{\alpha} + \dot{y}_{or} \sin(\alpha) + y_{or} \cos(\alpha) \dot{\alpha} \\ -\dot{x}_{or} \sin(\alpha) - x_{or} \cos(\alpha) \dot{\alpha} + \dot{y}_{or} \cos(\alpha) - y_{or} \sin(\alpha) \dot{\alpha} \\ \dot{\alpha}_i \end{bmatrix} \quad (4.6)$$

Dado que el movimiento únicamente a lo largo del eje y_r implica tener un ángulo α constante, esta restricción holonómica queda como:

$$-x_{or} \sin(\alpha) + y_{or} \cos(\alpha) = 0 \quad (4.7)$$

- Rodamiento puro: No hay deslizamiento entre las llantas y el suelo; además, solo hay un punto de contacto. Lo anterior se ilustra en la Figura 4.5.

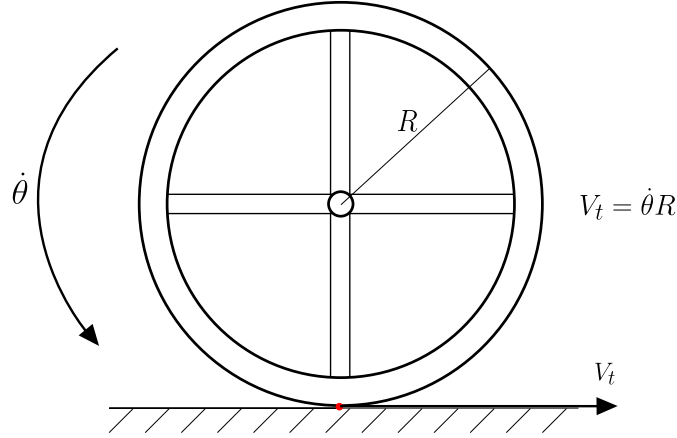


Figura 4.5: Restricción de movimiento por rodamiento puro. Donde R es el radio de la rueda, V_t es la velocidad tangencial y $\dot{\theta}$ es la velocidad angular de la rueda.

De la Figura 4.5, la velocidad traslacional de la rueda derecha (V_{wd}) y de la izquierda (V_{wi}) quedan definidas por (4.8) y (4.9) respectivamente.

$$V_{wd} = \dot{\theta}_d R \quad (4.8)$$

$$V_{wi} = \dot{\theta}_i R \quad (4.9)$$

La posición del centro de masa de la rueda derecha (G_{wd}), con respecto al marco inercial, es:

$$\begin{bmatrix} x_{wd}^i \\ y_{wd}^i \\ \alpha^i \end{bmatrix} = \begin{bmatrix} x_{or} + w \sin(\alpha) \\ y_{or} - w \cos(\alpha) \\ \alpha^i \end{bmatrix} \quad (4.10)$$

y de la rueda izquierda:

$$\begin{bmatrix} x_{wi}^i \\ y_{wi}^i \\ \alpha^i \end{bmatrix} = \begin{bmatrix} x_{or} - w \sin(\alpha) \\ y_{or} + w \cos(\alpha) \\ \alpha^i \end{bmatrix} \quad (4.11)$$

por lo que las velocidades quedan definidas por (4.12) y (4.13).

$$\begin{bmatrix} \dot{x}_{wd}^i \\ \dot{y}_{wd}^i \\ \dot{\alpha}^i \end{bmatrix} = \begin{bmatrix} \dot{x}_{or} + w \cos(\alpha) \dot{\alpha} \\ \dot{y}_{or} + w \sin(\alpha) \dot{\alpha} \\ \dot{\alpha}^i \end{bmatrix} \quad (4.12)$$

$$\begin{bmatrix} \dot{x}_{wi}^i \\ \dot{y}_{wi}^i \\ \dot{\alpha}^i \end{bmatrix} = \begin{bmatrix} \dot{x}_{or} - w \cos(\alpha) \dot{\alpha} \\ \dot{y}_{or} - w \sin(\alpha) \dot{\alpha} \\ \dot{\alpha}^i \end{bmatrix} \quad (4.13)$$

Empleando la matriz de rotación $R(\alpha)^{-1}$ para tener las ecuaciones con respecto al marco de referencia del robot, se obtiene:

$$\begin{bmatrix} \dot{x}_{wd}^r \\ \dot{y}_{wd}^r \\ \dot{\alpha}^r \end{bmatrix} = \begin{bmatrix} \dot{x}_{wd}^i \cos(\alpha) + \dot{y}_{wd}^i \sin(\alpha) \\ -\dot{x}_{wd}^i \sin(\alpha) + \dot{y}_{wd}^i \cos(\alpha) \\ \dot{\alpha}^i \end{bmatrix} \quad (4.14)$$

$$\begin{bmatrix} \dot{x}_{wi}^r \\ \dot{y}_{wi}^r \\ \dot{\alpha}^r \end{bmatrix} = \begin{bmatrix} \dot{x}_{wi}^i \cos(\alpha) + \dot{y}_{wi}^i \sin(\alpha) \\ -\dot{x}_{wi}^i \sin(\alpha) + \dot{y}_{wi}^i \cos(\alpha) \\ \dot{\alpha}^i \end{bmatrix} \quad (4.15)$$

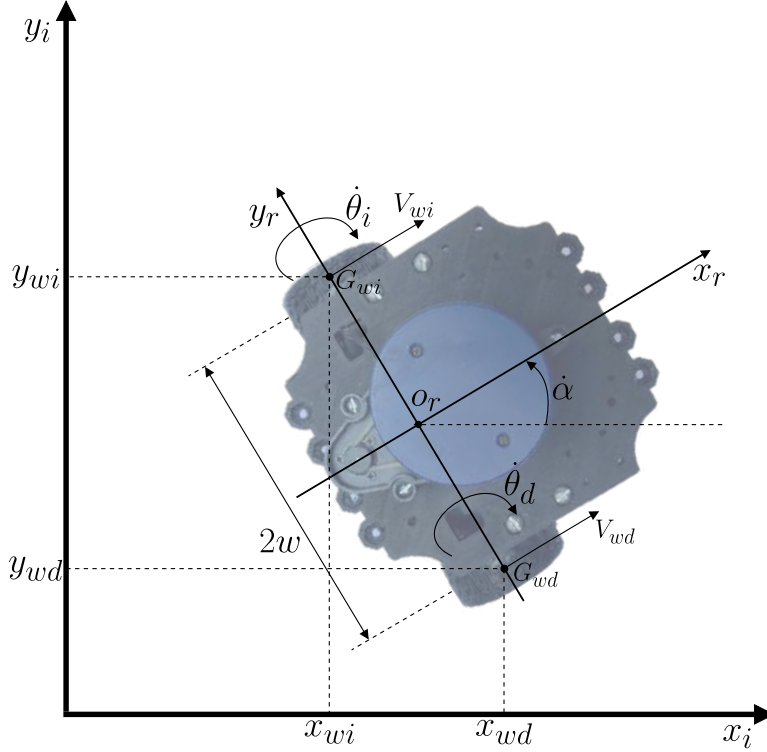


Figura 4.6: Posición y velocidad de las ruedas del robot móvil.

Observando la Figura 4.6, las velocidades $\dot{y}_{wd}^r$ y $\dot{y}_{wi}^r$ son nulas (se comprueba en la restricción no holonómica anterior); por lo tanto, las ecuaciones del rodamiento puro quedan definidas como se muestra en (4.16) y (4.17).

$$\begin{aligned} V_{wd} &= \dot{x}_{wd}^r \\ \dot{\theta}_d R &= \dot{x}_{or} \cos(\alpha) + \dot{y}_{or} \sin(\alpha) + w \dot{\alpha} \end{aligned} \quad (4.16)$$

$$\begin{aligned} V_{wi} &= \dot{x}_{wi}^r \\ \dot{\theta}_i R &= \dot{x}_{or} \cos(\alpha) + \dot{y}_{or} \sin(\alpha) - w\dot{\alpha} \end{aligned} \quad (4.17)$$

Las restricciones no holonómicas anteriores quedan definidas por las ecuaciones:

$$\begin{aligned} -\dot{x}_{or} \sin(\alpha) + \dot{y}_{or} \cos(\alpha) &= 0 \\ \dot{x}_{or} \cos(\alpha) + \dot{y}_{or} \sin(\alpha) + w\dot{\alpha} - \dot{\theta}_d R &= 0 \\ \dot{x}_{or} \cos(\alpha) + \dot{y}_{or} \sin(\alpha) - w\dot{\alpha} - \dot{\theta}_i R &= 0 \end{aligned} \quad (4.18)$$

de forma matricial:

$$\begin{aligned} \Psi(q)\dot{q} &= 0 \\ \Psi(q) &= \begin{bmatrix} -\sin(\alpha) & \cos(\alpha) & 0 & 0 & 0 \\ \cos(\alpha) & \sin(\alpha) & w & -R & 0 \\ \cos(\alpha) & \sin(\alpha) & -w & 0 & -R \end{bmatrix} \\ \dot{q} &= [\dot{x}_{or} \quad \dot{y}_{or} \quad \dot{\alpha} \quad \dot{\theta}_d \quad \dot{\theta}_i]^T \end{aligned} \quad (4.19)$$

Modelo cinemático del robot móvil

La cinemática del robot móvil consiste en representar la velocidad del robot como función del movimiento de sus ruedas, teniendo en cuenta la geometría del robot. De la Figura 4.6, se puede notar que la velocidad traslacional del robot es debida al promedio de las velocidades tangenciales de ambas ruedas, por lo que, empleando las ecuaciones (4.8) y (4.9), se obtiene:

$$\dot{x}^r = R \left(\frac{\dot{\theta}_d + \dot{\theta}_i}{2} \right) \quad (4.20)$$

Por otro lado, suponiendo que ambas ruedas giran a la misma velocidad pero en sentido contrario, se puede deducir que el robot mantiene su posición traslacional, rotando alrededor del eje ortogonal en o_r , y manteniendo velocidades traslacionales nulas, es decir, $\dot{x}^r = 0$, $\dot{y}^r = 0$.

A continuación, la velocidad angular $\dot{\alpha}^r$ se obtiene considerando que solo gira una rueda mientras la otra se mantiene estática, de esta manera, la rueda estacionaria actúa como eje de rotación y así, empleando (4.8) y considerando que la rueda en movimiento es la rueda derecha, se tiene:

$$\dot{\alpha}^r = \frac{V_{wd}}{2w} \quad (4.21)$$

Por consiguiente, considerando el caso contrario en el que la rueda izquierda gira y la rueda derecha permanece estática, $\dot{\alpha}^r$ queda completamente definida, como se muestra en (4.22).

$$\dot{\alpha}^r = R \left(\frac{\dot{\theta}_d - \dot{\theta}_i}{2w} \right) \quad (4.22)$$

Finalmente, la cinemática del robot móvil se expresa completamente en (4.23).

$$\begin{bmatrix} \dot{x}^r \\ \dot{y}^r \\ \dot{\alpha}^r \end{bmatrix} = \begin{bmatrix} R \left(\frac{\dot{\theta}_d + \dot{\theta}_i}{2} \right) \\ 0 \\ R \left(\frac{\dot{\theta}_d - \dot{\theta}_i}{2w} \right) \end{bmatrix} = \begin{bmatrix} \frac{R}{2} & \frac{R}{2} \\ 0 & 0 \\ \frac{R}{2w} & -\frac{R}{2w} \end{bmatrix} \begin{bmatrix} \dot{\theta}_d \\ \dot{\theta}_i \end{bmatrix} \quad (4.23)$$

Con respecto al marco de referencia inercial, se tiene:

$$\begin{bmatrix} \dot{x}^i \\ \dot{y}^i \\ \dot{\alpha}^i \end{bmatrix} = \begin{bmatrix} \frac{R \cos(\alpha)(\dot{\theta}_d + \dot{\theta}_i)}{2} \\ \frac{R \sin(\alpha)(\dot{\theta}_d + \dot{\theta}_i)}{2} \\ \frac{R(\dot{\theta}_d - \dot{\theta}_i)}{2w} \end{bmatrix} = \begin{bmatrix} \frac{R \cos(\alpha)}{2} & \frac{R \cos(\alpha)}{2} \\ \frac{R \sin(\alpha)}{2} & \frac{R \sin(\alpha)}{2} \\ \frac{R}{2w} & -\frac{R}{2w} \end{bmatrix} \begin{bmatrix} \dot{\theta}_d \\ \dot{\theta}_i \end{bmatrix} \quad (4.24)$$

Modelo dinámico del robot móvil

Para el modelo dinámico se hace uso de la formulación Euler-Lagrange para sistemas mecánicos con restricciones no holonómicas y fuerzas externas, cuya ecuación es [43]:

$$\frac{d}{dt} \left[\frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}_i} \right] - \frac{\partial \mathcal{L}(q, \dot{q})}{\partial q_i} = \Psi(q)^T \lambda + B(q) \tau \quad (4.25)$$

donde:

$$\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{U}(q) \quad (4.26)$$

o en su forma compacta:

$$M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + g(q) = \Psi(q)^T \lambda + B(q) \tau \quad (4.27)$$

De la ecuación (4.25), $\mathcal{L}(q, \dot{q})$ es el lagrangiano, $B(q)$ es la matriz de entrada, τ es el vector de fuerzas y torques externos (para el robot móvil representa los torques de los motores de las ruedas), $\Psi(q)$ es la matriz de restricciones no holonómicas y λ son los multiplicadores de Lagrange. Para la ecuación (4.26), $\mathcal{K}(q, \dot{q})$ es la energía cinética; $\mathcal{U}(q)$ es la energía potencial. Por último, de la ecuación (4.27), $M(q)$ es la matriz de inercias, $C(q, \dot{q})$ es la matriz de fuerzas centrífugas y coriolis y $g(q)$ es el vector de gravedad.

El primer paso para construir el modelo dinámico del robot móvil consiste en calcular la energía cinética. Observando la Figura 4.4, la posición del centro de masa G del robot con respecto al marco de referencia inercial es:

$$G = \begin{bmatrix} x_{or} + d \cos(\alpha) \\ y_{or} + d \sin(\alpha) \\ \alpha \end{bmatrix} \quad (4.28)$$

por lo que la velocidad traslacional es:

$$\dot{G} = \begin{bmatrix} \dot{x}_{or} - d \sin(\alpha) \dot{\alpha} \\ \dot{y}_{or} + d \cos(\alpha) \dot{\alpha} \\ \dot{\alpha} \end{bmatrix} \quad (4.29)$$

Se sabe que la energía cinética está definida por:

$$\mathcal{K}(q, \dot{q}) = \frac{1}{2} m V^2 + \frac{1}{2} I \omega^2 \quad (4.30)$$

donde m es la masa, V es la velocidad traslacional, I es el momento de inercia de masa y ω es la velocidad angular.

Por lo tanto, la energía cinética con respecto al centro de masa del robot se define como:

$$\begin{aligned}\mathcal{K}_G(q, \dot{q}) &= \frac{1}{2}m_G\dot{G}^T\dot{G} + \frac{1}{2}I_G\dot{\alpha}^2 \\ &= \frac{1}{2}m_G\{\dot{x}_{or}^2 + \dot{y}_{or}^2 + 2d\dot{\alpha}[-\dot{x}_{or}\sin(\alpha) + \dot{y}_{or}\cos(\alpha)] + d^2\dot{\alpha}^2\} + \frac{1}{2}I_G\dot{\alpha}^2\end{aligned}\quad (4.31)$$

La posición de ambas ruedas del robot se define en (4.10) y (4.11); de igual modo, las velocidades se definen en (4.12) y (4.13). Por consiguiente, la energía cinética con respecto al centro de masa de cada una de las ruedas del robot queda definida como se muestra en (4.32) y (4.33).

$$\mathcal{K}_{wd}(q, \dot{q}) = \frac{1}{2}m_{wd}\{\dot{x}_{or}^2 + \dot{y}_{or}^2 + 2w\dot{\alpha}[\dot{x}_{or}\cos(\alpha) + \dot{y}_{or}\sin(\alpha)] + w^2\dot{\alpha}^2\} + \frac{1}{2}I_{wd}\dot{\theta}_d^2 + \frac{1}{2}I_{Gwd}\dot{\alpha}^2 \quad (4.32)$$

$$\mathcal{K}_{wi}(q, \dot{q}) = \frac{1}{2}m_{wi}\{\dot{x}_{or}^2 + \dot{y}_{or}^2 - 2w\dot{\alpha}[\dot{x}_{or}\cos(\alpha) + \dot{y}_{or}\sin(\alpha)] + w^2\dot{\alpha}^2\} + \frac{1}{2}I_{wi}\dot{\theta}_i^2 + \frac{1}{2}I_{Gwi}\dot{\alpha}^2 \quad (4.33)$$

donde I_{wd} y I_{wi} son los momentos de inercia de las ruedas derecha e izquierda, medidos con respecto al eje del motor. I_{Gwd} y I_{Gwi} son los momentos de inercia de las ruedas derecha e izquierda, medidos respecto de un eje de rotación que pasa por el diámetro de las ruedas (a través de su centro de masa).

Entonces la energía cinética total del robot, es:

$$\begin{aligned}\mathcal{K}(q, \dot{q}) &= \mathcal{K}_G(q, \dot{q}) + \mathcal{K}_{wd}(q, \dot{q}) + \mathcal{K}_{wi}(q, \dot{q}) \\ &= \frac{1}{2}\left\{(m_G + m_{wd} + m_{wi})(\dot{x}_{or}^2 + \dot{y}_{or}^2) + [m_{wd} - m_{wi}]2w\dot{\alpha}[\dot{x}_{or}\cos(\alpha) + \dot{y}_{or}\sin(\alpha)] \right. \\ &\quad \left. + (m_{wd} + m_{wi})w^2\dot{\alpha}^2 + m_G[2d\dot{\alpha}[-\dot{x}_{or}\sin(\alpha) + \dot{y}_{or}\cos(\alpha)] + d^2\dot{\alpha}^2] \right. \\ &\quad \left. + \dot{\alpha}^2(I_G + I_{Gwd} + I_{Gwi}) + I_{wd}\dot{\theta}_d^2 + I_{wi}\dot{\theta}_i^2\right\}\end{aligned}\quad (4.34)$$

Con la intención de simplificar la ecuación (4.34), se realizan las siguientes suposiciones:

$$\begin{aligned}m_{wd} &\approx m_{wi} \\ I_{wd} &\approx I_{wi} \\ I_{Gwd} &\approx I_{Gwi} \\ m &= m_G + m_{wd} + m_{wi} = m_G + 2m_w \\ I &= I_G + I_{Gwd} + I_{Gwi}\end{aligned}$$

las cuales son debidas a que los motores y las ruedas empleadas en el robot, son del mismo fabricante y por lo tanto, de las mismas características.

Simplificando la energía cinética total se puede escribir como:

$$\begin{aligned}
\mathcal{K}(q, \dot{q}) &= \frac{1}{2} \left\{ m(\dot{x}_{or}^2 + \dot{y}_{or}^2) + 2m_w w^2 \dot{\alpha}^2 + m_G [2d\dot{\alpha}[-\dot{x}_{or} \sin(\alpha) + \dot{y}_{or} \cos(\alpha)] + d^2 \dot{\alpha}^2] \right. \\
&\quad \left. + I \dot{\alpha}^2 + I_w(\dot{\theta}_d^2 + \dot{\theta}_i^2) \right\} \\
&= \frac{m}{2}(\dot{x}_{or}^2 + \dot{y}_{or}^2) + m_G d \dot{\alpha}[-\dot{x}_{or} \sin(\alpha) + \dot{y}_{or} \cos(\alpha)] + \frac{1}{2} I_a \dot{\alpha}^2 + \frac{1}{2} I_w(\dot{\theta}_d^2 + \dot{\theta}_i^2)
\end{aligned} \tag{4.35}$$

donde $I_a = 2m_w w^2 + m_G d^2 + I$

La energía potencial gravitacional del robot es nula dado por el hecho que el robot se encuentra en el plano horizontal a nivel del suelo, por lo tanto, el efecto de la gravedad no se presenta en el análisis energético. Debido a lo anterior, el Lagrangiano es $\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q})$.

Siendo el vector q :

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} = \begin{bmatrix} x_{or} \\ y_{or} \\ \alpha \\ \theta_d \\ \theta_i \end{bmatrix} \tag{4.36}$$

el modelo dinámico del robot, empleando la formulación Euler-Lagrange vista en (4.25), se representa como se observa en (4.37).

$$\begin{aligned}
\frac{d}{dt} \left[\frac{\partial \mathcal{L}(q, \dot{q})}{\partial \dot{q}_i} \right] - \frac{\partial \mathcal{L}(q, \dot{q})}{\partial q_i} &= \begin{bmatrix} m\ddot{x}_{or} - m_G d[\ddot{\alpha} \sin(\alpha) + \dot{\alpha}^2 \cos(\alpha)] \\ m\ddot{y}_{or} + m_G d[\ddot{\alpha} \cos(\alpha) - \dot{\alpha}^2 \sin(\alpha)] \\ m_G d[\ddot{y}_{or} \cos(\alpha) - \ddot{x}_{or} \sin(\alpha)] + \ddot{\alpha} I_a \\ I_w \ddot{\theta}_d \\ I_w \ddot{\theta}_i \end{bmatrix} \\
B(q) &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \\
\tau &= [\tau_1 \quad \tau_2 \quad \tau_3 \quad \tau_4 \quad \tau_5]^T \\
\Psi(q)^T &= \begin{bmatrix} -\sin(\alpha) & \cos(\alpha) & \cos(\alpha) \\ \cos(\alpha) & \sin(\alpha) & \sin(\alpha) \\ 0 & w & -w \\ 0 & -R & 0 \\ 0 & 0 & -R \end{bmatrix} \\
\lambda &= [\lambda_1 \quad \lambda_2 \quad \lambda_3]^T
\end{aligned} \tag{4.37}$$

En forma compacta:

$$\begin{aligned}
M(q) &= \begin{bmatrix} m & 0 & -m_G d \sin(\alpha) & 0 & 0 \\ 0 & m & m_G d \cos(\alpha) & 0 & 0 \\ -m_G d \sin(\alpha) & m_G d \cos(\alpha) & I_a & 0 & 0 \\ 0 & 0 & 0 & I_w & 0 \\ 0 & 0 & 0 & 0 & I_w \end{bmatrix} \\
C(q, \dot{q}) &= \begin{bmatrix} 0 & 0 & -m_G d \cos(\alpha) \dot{\alpha} & 0 & 0 \\ 0 & 0 & -m_G d \sin(\alpha) \dot{\alpha} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
g(q) &= 0
\end{aligned} \tag{4.38}$$

El modelo obtenido puede ajustarse eliminando la matriz de restricciones y los multiplicadores de Lagrange, ya que estos constituyen las incógnitas del sistema. Para ajustar el modelo, partiendo de la cinemática del sistema (4.24), considerando todos los elementos de q , se tiene:

$$\begin{bmatrix} \dot{x}_{or} \\ \dot{y}_{or} \\ \dot{\alpha} \\ \dot{\theta}_d \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \frac{R \cos(\alpha)}{2} & \frac{R \cos(\alpha)}{2} \\ \frac{R \sin(\alpha)}{2} & \frac{R \sin(\alpha)}{2} \\ \frac{R}{2w} & -\frac{R}{2w} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_d \\ \dot{\theta}_i \end{bmatrix} \tag{4.39}$$

lo que se puede escribir como:

$$\dot{q} = \kappa(q) \omega \tag{4.40}$$

De las restricciones no holonómicas (4.19), siendo $\Psi(q)\dot{q} = 0$, sustituyendo (4.40), se tiene:

$$\Psi(q) \kappa(q) \omega = 0 \tag{4.41}$$

donde es posible demostrar que $\Psi(q) \kappa(q) = 0$.

Derivando (4.40) con respecto del tiempo :

$$\ddot{q} = \dot{\kappa}(q) \omega + \kappa(q) \dot{\omega} \tag{4.42}$$

Sustituyendo (4.40) y (4.42) en (4.27), se tiene:

$$\begin{aligned}
M(q) [\dot{\kappa}(q) \omega + \kappa(q) \dot{\omega}] + C(q, \dot{q}) [\kappa(q) \omega] &= \Psi(q)^T \lambda + B(q) \tau \\
M(q) \kappa(q) \dot{\omega} + [M(q) \dot{\kappa}(q) + C(q, \dot{q}) \kappa(q)] \omega &= \Psi(q)^T \lambda + B(q) \tau
\end{aligned} \tag{4.43}$$

Aplicando la propiedad de la transpuesta de dos matrices $(AB)^T = B^T A^T$, se puede escribir:

$$\Psi(q) \kappa(q) = 0 \rightarrow (\Psi(q) \kappa(q))^T = 0 = \kappa(q)^T \Psi(q)^T$$

Premultiplicando (4.43) por $\kappa(q)^T$, se tiene:

$$\kappa(q)^T M(q) \kappa(q) \dot{\omega} + \kappa(q)^T [M(q) \dot{\kappa}(q) + C(q, \dot{q}) \kappa(q)] \omega = \kappa(q)^T B(q) \tau \quad (4.44)$$

Por consiguiente, el modelo dinámico del robot se expresa ahora como:

$$\begin{aligned} \widehat{M(q)} &= \kappa(q)^T M(q) \kappa(q) \\ \widehat{C(q, \dot{q})} &= \kappa(q)^T [M(q) \dot{\kappa}(q) + C(q, \dot{q}) \kappa(q)] \\ \widehat{B(q)} &= \kappa(q)^T B(q) \\ \widehat{M(q)} \dot{\omega} + \widehat{C(q, \dot{q})} \omega &= \widehat{B(q)} \tau \end{aligned} \quad (4.45)$$

En forma extendida:

$$\begin{bmatrix} \frac{mR^2}{4} + \frac{I_a R^2}{4w^2} + I_w & \frac{mR^2}{4} - \frac{I_a R^2}{4w^2} \\ \frac{mR^2}{4} - \frac{I_a R^2}{4w^2} & \frac{mR^2}{4} + \frac{I_a R^2}{4w^2} + I_w \end{bmatrix} \begin{bmatrix} \ddot{\theta}_d \\ \ddot{\theta}_i \end{bmatrix} + \begin{bmatrix} 0 & \frac{m_G d R^2 \dot{\alpha}}{2w} \\ -\frac{m_G d R^2 \dot{\alpha}}{2w} & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_d \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \tau_4 \\ \tau_5 \end{bmatrix} \quad (4.46)$$

De la ecuación (4.46) se aprecia que el modelo dinámico del robot se ajusta considerando únicamente los pares externos de ambos motores y, por lo tanto, incluye las restricciones no holonómicas.

Para obtener un modelo dinámico en función de las coordenadas del marco de referencia del robot, se utiliza la ecuación (4.23). Reescribiendo el modelo dinámico del robot, se tiene:

$$\begin{bmatrix} \frac{mR}{2} + \frac{I_w}{R} & \frac{I_a R}{2w} + \frac{w I_w}{R} \\ \frac{mR}{2} + \frac{I_w}{R} & -\frac{I_a R}{2w} - \frac{w I_w}{R} \end{bmatrix} \begin{bmatrix} \ddot{x}^r \\ \ddot{\alpha}^r \end{bmatrix} + \begin{bmatrix} \frac{m_G d R \dot{\alpha}}{2w} & -\frac{m_G d R \dot{\alpha}}{2} \\ -\frac{m_G d R \dot{\alpha}}{2w} & -\frac{m_G d R \dot{\alpha}}{2} \end{bmatrix} \begin{bmatrix} \dot{x}^r \\ \dot{\alpha}^r \end{bmatrix} = \begin{bmatrix} \tau_4 \\ \tau_5 \end{bmatrix} \quad (4.47)$$

4.1.2. Modelo dinámico de las ruedas

El movimiento del robot se debe a los motorreductores ubicados en la parte inferior del robot. Por lo tanto, el esquema general de un motor de corriente directa se muestra en la Figura 4.7.

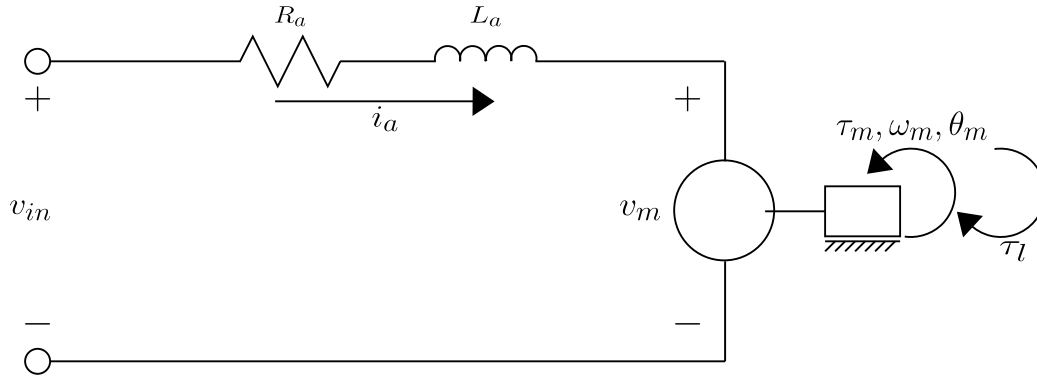


Figura 4.7: Modelo de un motor de corriente directa que separa la parte eléctrica de la mecánica.

Como se observa en la Figura 4.7, un motor de corriente directa se puede separar en la dinámica eléctrica y la mecánica; de esta manera, se pueden emplear leyes físicas conocidas y así encontrar un modelo matemático. Para la dinámica eléctrica se aplican las leyes de Kirchhoff y, para la mecánica, las leyes de Newton. Las ecuaciones que modelan el motor de corriente directa son [44]:

$$v_{in}(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + v_m(t) \quad (4.48)$$

$$\tau_m(t) = \tau_l(t) + J \frac{d^2\theta(t)}{dt^2} + B \frac{d\theta(t)}{dt} \quad (4.49)$$

$$\tau_m(t) = K_\tau i_a(t) \quad (4.50)$$

$$v_m(t) = K_b \frac{d\theta(t)}{dt} = K_b \omega(t) \quad (4.51)$$

donde $v_{in}(t)$ es el voltaje aplicado en la entrada, R_a es la resistencia de armadura, $i_a(t)$ es la corriente, L_a es la inductancia de armadura, $v_m(t)$ es la fuerza contraelectromotriz, $\tau_m(t)$ es el par del motor, $\tau_l(t)$ es el par de carga, J es el momento de inercia del motor, $\theta(t)$ es la posición angular, B es el coeficiente de fricción viscosa, K_τ es la constante de par que relaciona el flujo magnético del motor con la corriente, K_b es la constante de la fuerza contraelectromotriz y $\omega(t)$ es la velocidad angular. La ecuación (4.50) muestra que la corriente eléctrica del motor es proporcional al torque aplicado.

Sustituyendo (4.51) en (4.48) y (4.50) en (4.49), se tiene:

$$v_{in}(t) = R_a i_a(t) + L_a \frac{di_a(t)}{dt} + K_b \frac{d\theta(t)}{dt} \quad (4.52)$$

$$K_\tau i_a(t) = \tau_l(t) + J \frac{d^2\theta(t)}{dt^2} + B \frac{d\theta(t)}{dt} \quad (4.53)$$

Las ecuaciones (4.52) y (4.53) rigen el comportamiento dinámico del motor de corriente directa. Aplicando la transformada de Laplace a estas ecuaciones, suponiendo $\tau_l(t) = 0$ y condiciones iniciales nulas, se obtiene la función de transferencia mostrada en (4.54), donde la salida es la posición angular.

$$\frac{\Theta(s)}{V_{in}(s)} = \frac{K_\tau}{JL_a s^3 + s^2(JR_a + BL_a) + s(K_\tau K_b + BR_a)} \quad (4.54)$$

Si la velocidad angular es la salida, se tiene:

$$\begin{aligned} \frac{\Omega(s)}{V_{in}(s)} &= \frac{K_\tau}{JL_a s^2 + s(JR_a + BL_a) + (K_\tau K_b + BR_a)} \\ &= \frac{K_\tau}{(Js + B)(L_a s + R_a) + K_\tau K_b} \end{aligned} \quad (4.55)$$

El diagrama de bloques correspondiente a la ecuación (4.55) se presenta en la Figura 4.8

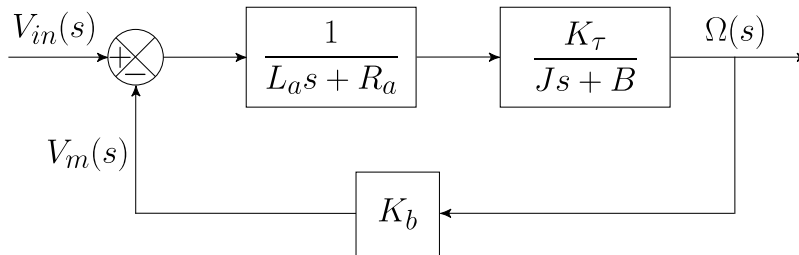


Figura 4.8: Diagrama de bloques del motor de corriente directa con salida de velocidad.

Para ajustar la velocidad de las ruedas del robot móvil, se implementa un servoamplificador con un lazo de corriente interno para controlar el par. En este caso, siendo el efecto del servoamplificador $K_a u(t)$, donde K_a representa la ganancia del servoamplificador y $u(t)$ la señal de entrada, se modela la corriente como:

$$i_a(t) \approx K_a u(t) \quad (4.56)$$

así, se obliga a que la corriente del motor siga la señal de control.

Observando la Figura 4.8, la ecuación (4.56) sustituye a:

$$I_a(s) = \frac{V_{in}(s) - V_m(s)}{L_a s + R_a}; \quad V_m(s) = K_b \Omega(s) \quad (4.57)$$

Por lo tanto, el diagrama de bloques se simplifica al que se muestra en la Figura 4.9. Aparece un bloque $G_c(s)$, que representa la acción de un controlador de velocidad. La entrada es la velocidad deseada $\Omega_d(s)$.

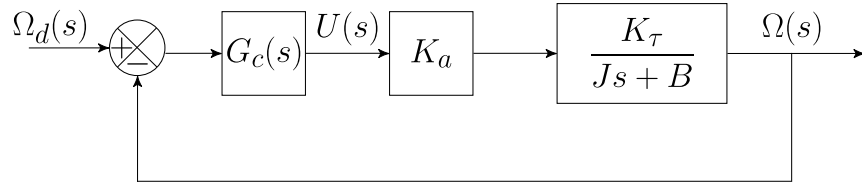


Figura 4.9: Diagrama de bloques del motor de corriente directa, incluyendo el servoamplificador.

Control de velocidad PI

La propuesta de control de la velocidad de las ruedas del robot móvil es un controlador proporcional-integral (PI). La Figura 4.10 muestra el diagrama de bloques del motor de corriente directa con servoamplificador y controlador de velocidad PI.

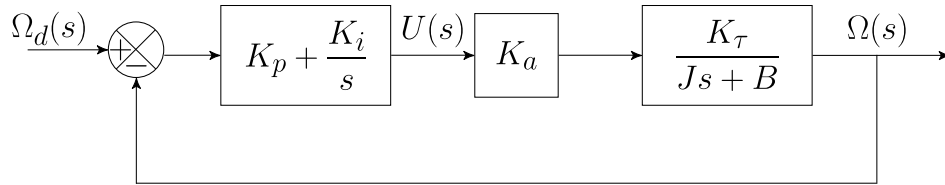


Figura 4.10: Lazo de control PI para el motor de corriente directa con servoamplificador.

La ecuación de la trayectoria directa es:

$$\frac{(K_p s + K_i) K_a K_\tau}{s(Js + B)} \quad (4.58)$$

entonces los polos se encuentran en:

$$\begin{aligned} s_1 &= 0 \\ s_2 &= -\frac{B}{J} \end{aligned} \quad (4.59)$$

y el cero en:

$$s = -\frac{K_i}{K_p} \quad (4.60)$$

Suponiendo el caso en el que se tienen polos dominantes, entonces $-\frac{K_i}{K_p} < -\frac{B}{J}$; por lo tanto, aplicando la condición de magnitud y la condición de ángulo, es posible obtener las siguientes ecuaciones:

$$K_p = \frac{K_i}{\gamma} \quad (4.61)$$

$$K_i = \sqrt{\frac{[(\zeta\omega_n - \frac{B}{J})^2 + \omega_d^2] [\zeta^2\omega_n^2 + \omega_d^2]}{P^2 \left(\frac{\omega_d^2 + \zeta^2\omega_n^2}{\gamma^2} + 1 - \frac{2\zeta\omega_n}{\gamma} \right)}} \quad (4.62)$$

$$\gamma = \frac{\omega_d + \tan(\phi)\zeta\omega_n}{\tan(\phi)} \quad (4.63)$$

donde K_p es la ganancia proporcional del controlador PI, K_i es la ganancia integral del controlador PI, ω_n es la frecuencia natural no amortiguada, ζ es el factor de amortiguamiento relativo, ω_d es la frecuencia natural amortiguada y ϕ es el ángulo perteneciente al cero.

4.1.3. Tarjeta electrónica

La Figura 4.11 muestra el diagrama general de funcionamiento de la tarjeta electrónica del servoamplificador.

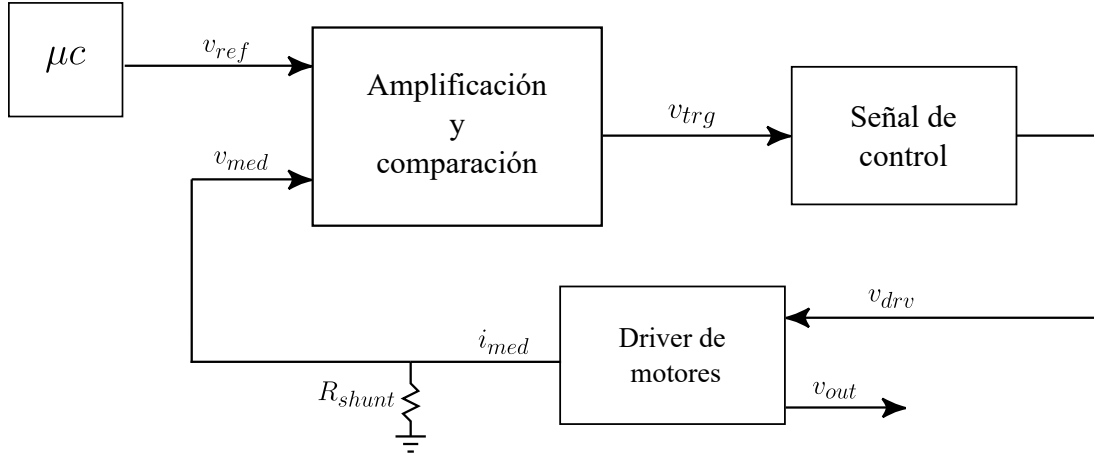


Figura 4.11: Diagrama de bloques de las etapas que componen la tarjeta electrónica.

Con respecto de la Figura 4.11, el funcionamiento de cada uno de los bloques es:

- **Amplificación y comparación:** Este bloque recibe como entradas el voltaje de referencia (v_{ref}), en modo PWM, proveniente del microcontrolador, y el voltaje medido (v_{med}), que representa la corriente de consumo de los motores. Esta conversión entre voltaje y corriente se realiza mediante el método de la resistencia *shunt*, en la que, haciendo uso de la ley de Ohm y

conociendo el valor de la resistencia, se puede determinar el voltaje, que depende proporcionalmente del consumo de corriente de los motores. El bloque cuenta con una etapa de filtrado y amplificación de la señal v_{med} , además de una etapa de comparación entre v_{med} y v_{ref} , lo que da como salida una señal lógica de tipo *trigger* (v_{trg}).

- Señal de control: Este bloque recibe como entrada la señal v_{trg} , que determina el encendido y apagado de la señal de conducción del driver de los motores (v_{drv}).
- Driver de motores: En este bloque, la entrada es la señal v_{drv} , que habilita el arranque de los motores. Por lo tanto, dependiendo de la señal v_{ref} y del consumo de corriente, es posible variar la velocidad y, en consecuencia, el torque. Así, el sistema es capaz de controlar el torque, dado que se tiene un control continuo de la corriente que consume el motor.

La tarjeta electrónica cuenta, por lo tanto, con dos etapas servoamplificadoras, una para cada motor del robot; estas se muestran en la Figura 4.12.

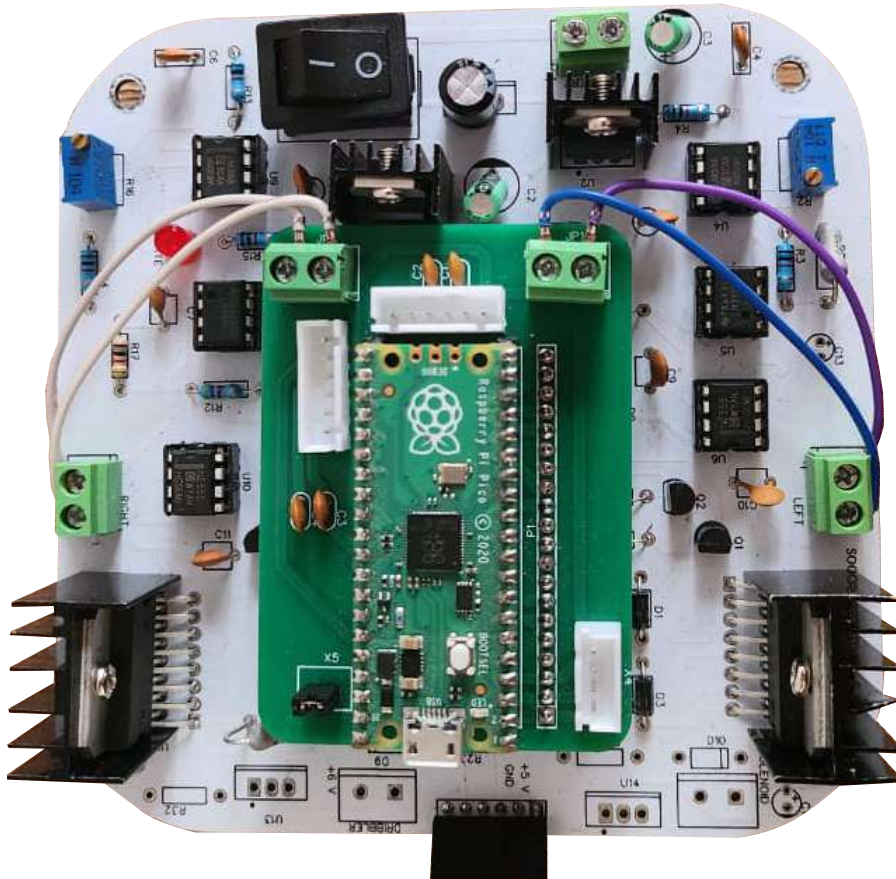


Figura 4.12: Tarjeta electrónica fabricada.

Los servoamplificadores implementados son de baja potencia y cuentan con mitigadores de ruido propios de la aplicación. Particularmente, se incorporan filtros pasa-bajos pasivos, además de capacitores de desacoplo en las fuentes de alimentación para reducir las variaciones de tensión.

Con respecto al funcionamiento y a las condiciones de operación del sistema, estas medidas resultan suficientes ya que los motores empleados son de bajo consumo de corriente (alrededor de 1 A). Así, esta solución permite una calidad adecuada de las señales de control sin requerir una etapa de acondicionamiento o de aislamiento de mayor complejidad.

Las características de diseño deben considerarse únicamente dentro del alcance específico de la presente investigación. No se plantea como una arquitectura general para aplicaciones de mayor potencia. Por lo tanto, la mitigación de ruido empleada se considera apropiada, pero su aplicación a sistemas de mayor escala requiere el rediseño de la electrónica y la incorporación de instrumentación adicional de protección, filtrado y aislamiento entre las etapas de control y de potencia.

4.2. Seguimiento del robot

En esta sección se describen los métodos de rastreo del robot móvil que emplean una cámara cenital y que se aplican en el presente proyecto.

4.2.1. Método de aprendizaje de la representación espacial.

El método de seguimiento propuesto requiere un patrón de referencia similar a un tablero de ajedrez, en el que se conoce previamente la distancia entre las esquinas de los recuadros interiores (l). Posteriormente, mediante una red neuronal de función de base radial, se realiza un mapeo de coordenadas espaciales en metros para cada esquina. Es decir, la red neuronal recibe como entrada las coordenadas en píxeles (x_{px}, y_{px}) de cada una de las esquinas del patrón y, a la salida, se obtiene la coordenada espacial en metros (x, y). Es necesario un punto de referencia arbitrario. La Figura 4.13 muestra un diagrama que ilustra el método propuesto.

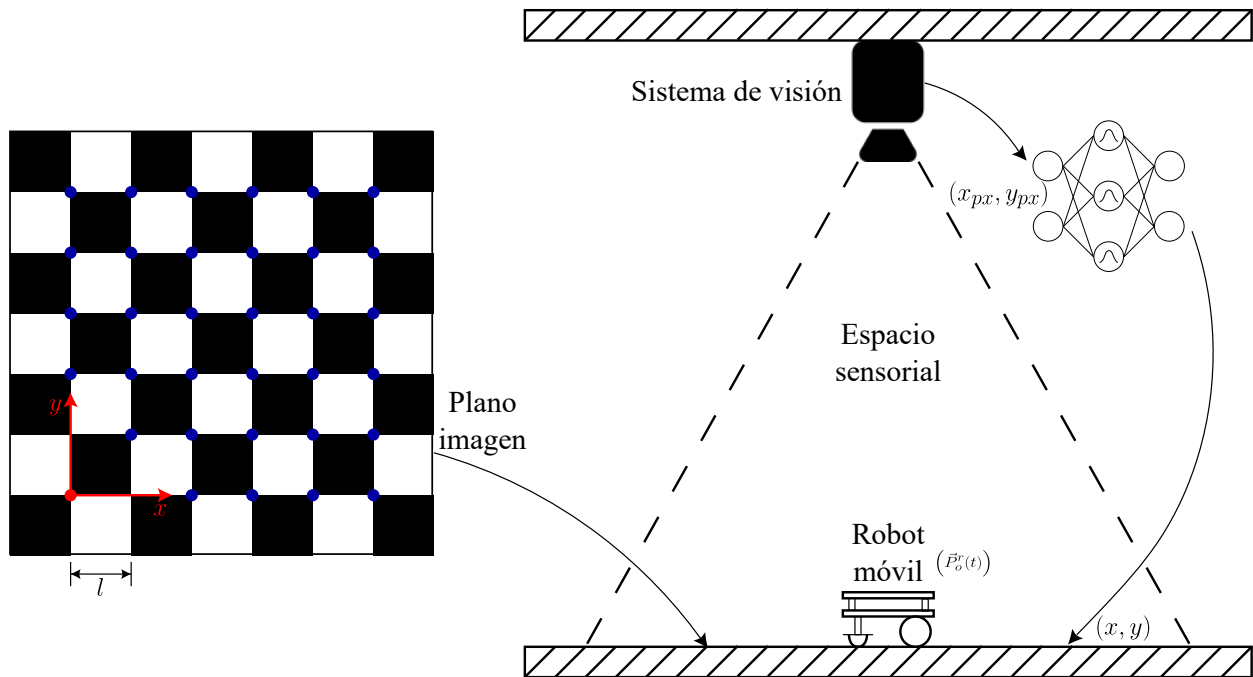


Figura 4.13: Diagrama ilustrativo del método de seguimiento propuesto.

Con respecto de la Figura 4.13, se pretende obtener un mapeo directo entre el plano de imagen observado por la cámara y las coordenadas espaciales del área de trabajo del robot móvil: $(x_{px}, y_{px}) \rightarrow (x, y)$ m. Así, es posible rastrear continuamente al robot y obtener su posición en tiempo real ($\bar{P}_o^r(t)$). Para este método no es necesaria la calibración de la cámara ni el conocimiento de los parámetros intrínsecos ni de la distorsión, ya que únicamente se trabaja con los píxeles de las esquinas observadas.

Conjunto de datos de entrenamiento

La red neuronal que se emplea es de tipo supervisado; en consecuencia, debe obtenerse el conjunto de datos de entrenamiento y proporcionarle las etiquetas correspondientes. Cada esquina interna corresponde a una coordenada en píxeles, por lo que es posible elegir el origen deseado $\mathbf{O}_{px}$.

La ecuación (4.64) calcula el factor de escala que relaciona la imagen con el plano.

$$s_x = \frac{(n_x - 1)l}{\frac{1}{n_y} \sum_{r=1}^{n_y} \|\mathbf{p}_{r,n_x} - \mathbf{p}_{r,1}\|}, \quad s_y = \frac{(n_y - 1)l}{\frac{1}{n_x} \sum_{c=1}^{n_x} \|\mathbf{p}_{n_y,c} - \mathbf{p}_{1,c}\|} \quad (4.64)$$

donde s_x y s_y son los factores de escala en $\frac{\text{m}}{\text{px}}$, l es la distancia entre cada esquina interna medida en metros, n_x y n_y son el número de esquinas internas en el eje x y el eje y respectivamente, $\mathbf{p}_{r,c}$ es la coordenada en píxeles de cada esquina en la fila r y columna c y $\|\cdot\|$ es la norma euclidiana en píxeles.

El resultado de (4.64) pertenece al vector de factores de escala ($\mathbf{s}$) y es necesario para calcular la ecuación (4.65).

$$\Delta_{px} = \mathbf{p}_{(0,0)} - \mathbf{O}_{px}, \quad \delta = \Delta_{px} \mathbf{s} \quad (4.65)$$

donde Δ_{px} es la distancia entre la primera esquina detectada por software $\mathbf{p}_{(0,0)}$ (considerada como la esquina de origen) y el origen deseado. δ es la compensación de la distancia en metros.

El cálculo de las ecuaciones (4.64) y (4.65) ajusta la traslación entre el sistema de píxeles y el sistema de referencia deseado. Las coordenadas en el plano físico, en metros, de cada esquina ($\mathbf{m}_{r,c}$) se obtienen directamente a partir de la geometría conocida del patrón de referencia, como se muestra en la ecuación (4.66).

$$\mathbf{m}_{r,c} = (cl, rl) \quad (4.66)$$

A cada coordenada se le resta la compensación de distancia para obtener el mapeo directo entre las coordenadas en píxeles de cada esquina y las coordenadas espaciales en metros, respetando el origen deseado. Dichas coordenadas se almacenan y se etiquetan para construir el conjunto de datos. La cantidad de datos depende del espacio de trabajo del robot.

Entrenamiento neuronal

El entrenamiento neuronal se lleva a cabo mediante el método descrito en [45], en el que se emplea el algoritmo *K-means* para determinar la localización de cada centro (ς). Los centros representan las regiones del espacio de entrada con mayor concentración de datos y se consideran la ubicación de las neuronas de la capa oculta. La función de activación es la gaussiana de la ecuación (4.67).

$$\phi_j(\mathbf{p}_i) = \exp\left(-\frac{\|\mathbf{p}_i - \varsigma_j\|^2}{2\sigma^2}\right) \quad (4.67)$$

donde $\mathbf{p}_i$ es la coordenada en píxel i -ésima, ς_j es el centro j -ésimo y σ es el parámetro de dispersión que controla el ancho de la función gaussiana.

El entrenamiento neuronal proporciona los pesos W , por lo que, para cualquier otra entrada $\mathbf{p}$, se predice la posición espacial ($\hat{m}$) mediante la ecuación (4.68).

$$\hat{m}(\mathbf{p}) = \Phi(\mathbf{p})W \quad (4.68)$$

donde Φ representa la matriz de activación que incluye la respuesta ante cualquier entrada.

4.2.2. Método de marcadores ArUco

Con la intención de comparar el método propuesto, se propone además un método tradicional utilizado en robótica: el método de los marcadores ArUco, cuya documentación completa se encuentra en [46]. En este método es necesaria la calibración de la cámara, ya que la posición y la orientación de los marcadores son relativas a la de la cámara.

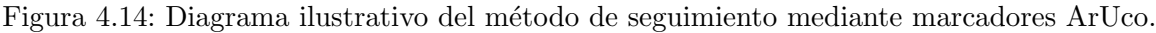
Para este método se considera el modelo *pinhole* de la cámara, el cual es representado por la ecuación (4.69) [47].

$$\mathbf{x} = \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X} \quad (4.69)$$

donde:

- $\mathbf{x} = [x \ y \ z]^T$ es un punto de la imagen.
- $\mathbf{X} = [X \ Y \ Z \ 1]^T$ es un punto en el espacio.
- $\mathbf{K} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}$ es la matriz de parámetros intrínsecos de la cámara.
- $[\mathbf{R}|\mathbf{t}] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$ es la matriz de parámetros extrínsecos de la cámara.

Cuando se detecta un marcador ArUco, se obtienen cuatro puntos que corresponden a las esquinas; estos se miden a partir del centro del marcador y, dado que se conoce la longitud l , se les asignan coordenadas específicas tanto en metros como en píxeles; así, se dispone de cuatro correspondencias $\mathbf{x}_i \leftrightarrow \mathbf{X}_i$ donde $i = 1, 2, 3, 4$. Entonces, el problema consiste en resolver el conjunto de ecuaciones (4.69) para $[\mathbf{R}|\mathbf{t}]$. A este problema se le conoce como problema de la perspectiva, dado n puntos o PnP por sus siglas en inglés [48]. La Figura 4.14 presenta un diagrama que ilustra cómo se realiza el seguimiento del robot móvil mediante marcadores ArUco.


$$\begin{aligned} \{\mathbf{x}_{f,i}, \mathbf{X}_{f,i}\} &\xrightarrow{\text{PnP}} cT_f \\ \{\mathbf{x}_{r,i}, \mathbf{X}_{r,i}\} &\xrightarrow{\text{PnP}} cT_r \end{aligned} \quad (4.70)$$

Al obtener (4.70), se puede considerar el marco de referencia fijo como el marco de referencia global y conocer la posición y orientación del robot con respecto a este (fT_r); matemáticamente, esto es:

$${}^fT_r = ({}^cT_f)^{-1} {}^cT_r \quad (4.71)$$

Quando se habla de controlar la posición de robots móviles y en específico robots terrestres con ruedas, existe una amplia variedad de propuestas reportadas en la literatura, algunas involucran únicamente el modelo cinemático y otras emplean el modelo dinámico, una de las diferencias principales entre ambos métodos es que el control que emplea el modelo dinámico suele tener mayor precisión dado que se involucran variables físicas con las que cuenta el robot, como la masa, el momento de inercia y la fricción. Sin embargo, en robots lentos y pequeños se considera que su

movimiento se realiza únicamente en el suelo (sin despegarse) y que las fuerzas dinámicas pueden despreciarse [49, 50].

Debido a lo anterior, los métodos de control propuestos se limitan a la cinemática del robot. El modelo dinámico se incorpora en este trabajo para representar, en la etapa de simulación, el comportamiento físico del sistema ante las acciones de control y así obtener una representación más completa.

Es importante aclarar que los controles cinemáticos se suponen ideales, ya que se asume que las variables de las ruedas, como la velocidad, el torque, entre otras, son ideales y que las ruedas alcanzan la velocidad deseada. Dado que lo anterior no es físicamente correcto, se utiliza un controlador de la velocidad de las ruedas, como se muestra en la sección 4.1.2.

En esta sección se describen dos modelos de control cinemático: el de posición para llegar a la meta y el de seguimiento de trayectoria. El motivo por el que se emplean dos leyes de control diferentes es validar y comparar los métodos de rastreo de la sección anterior, mostrar sus diferencias y justificar el alcance del método propuesto, es decir, el de aprendizaje de la representación espacial.

4.3.1. Control de posición para llegar a la meta

Considerando la Figura 4.15, el robot comienza en una posición y orientación arbitrarias $[x, y, \alpha]^T$ y debe llegar a la posición deseada $[x_d, y_d, \alpha]^T$ sin importar la trayectoria. Por lo tanto, el error medido con respecto al marco de referencia inercial $\mathbf{e}(t) = [e_x(t), e_y(t), e_\alpha(t)]^T$ debe aproximarse a cero, es decir:

$$\lim_{t \rightarrow \infty} \mathbf{e}(t) = 0$$

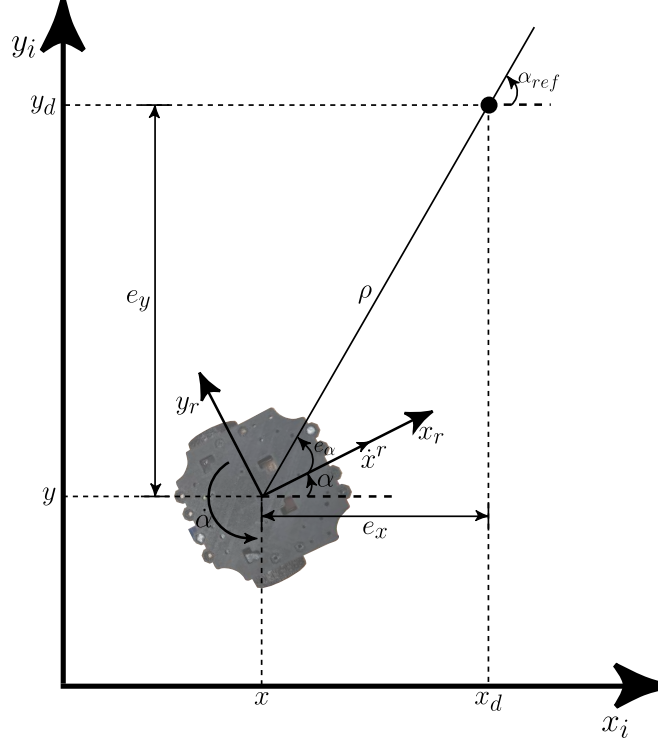


Figura 4.15: Esquema general del objetivo del controlador de posición para alcanzar la meta.

De la Figura 4.15, se obtiene:

$$\begin{aligned}
 e_x(t) &= x_d - x(t) \\
 e_y(t) &= y_d - y(t) \\
 e_\alpha(t) &= \alpha_{ref}(t) - \alpha(t) \\
 \rho(t) &= \sqrt{e_x^2 + e_y^2} \\
 \alpha_{ref}(t) &= \arctan 2(e_y, e_x)
 \end{aligned} \tag{4.72}$$

donde $\alpha_{ref} \in (-\pi, \pi]$. Hay que notar que ρ y e_α representan el error en coordenadas polares.

Siendo el modelo cinemático del robot representado por la ecuación (4.24), se puede reescribir en términos de la velocidad traslacional del robot $\dot{x}^r$ y de la velocidad angular $\dot{\alpha}^r$ que es:

$$\begin{bmatrix} \dot{x}^i \\ \dot{y}^i \\ \dot{\alpha}^i \end{bmatrix} = \begin{bmatrix} \cos(\alpha) & 0 \\ \sin(\alpha) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}^r \\ \dot{\alpha}^r \end{bmatrix} \tag{4.73}$$

Para reescribir el modelo cinemático (4.73) en términos del error y en coordenadas polares, se derivan ρ y α_{ref} . El análisis de $\dot{\rho}$ es:

$$\begin{aligned}\frac{d}{dt}(\rho(t)) &= \frac{d}{dt} \left(\sqrt{e_x^2 + e_y^2} \right) \\ \dot{\rho} &= \frac{1}{\sqrt{e_x^2 + e_y^2}} (e_x \dot{e}_x + e_y \dot{e}_y)\end{aligned}\tag{4.74}$$

donde:

$$\begin{aligned}\dot{e}_x &= -\dot{x} = -\cos(\alpha)\dot{x}^r \\ \dot{e}_y &= -\dot{y} = -\sin(\alpha)\dot{x}^r\end{aligned}\tag{4.75}$$

debido al modelo cinemático (4.73), por lo tanto:

$$\begin{aligned}\dot{\rho} &= \frac{-e_x \cos(\alpha)\dot{x}^r - e_y \sin(\alpha)\dot{x}^r}{\rho} \\ &= -\frac{\dot{x}^r (e_x \cos(\alpha) + e_y \sin(\alpha))}{\rho}\end{aligned}\tag{4.76}$$

Observando nuevamente la Figura 4.15 por identidad geométrica, se obtiene:

$$\begin{aligned}e_x &= \rho \cos(\alpha + e_\alpha) \\ e_y &= \rho \sin(\alpha + e_\alpha)\end{aligned}\tag{4.77}$$

y por identidades trigonométricas:

$$\begin{aligned}e_x &= \rho(\cos(\alpha) \cos(e_\alpha) - \sin(\alpha) \sin(e_\alpha)) \\ e_y &= \rho(\sin(\alpha) \cos(e_\alpha) + \sin(e_\alpha) \cos(\alpha))\end{aligned}\tag{4.78}$$

Sustituyendo (4.78) en (4.76):

$$\dot{\rho} = -\dot{x}^r \cos(e_\alpha)\tag{4.79}$$

El análisis de $\dot{\alpha}_{ref}$ es:

$$\begin{aligned}\frac{d}{dt}(e_\alpha(t)) &= \frac{d}{dt}(\alpha_{ref} - \alpha) = \frac{d}{dt}(\arctan 2(e_y, e_x) - \alpha) \\ \dot{e}_\alpha &= \frac{e_x \dot{e}_y - \dot{e}_x e_y}{e_x^2 + e_y^2} - \dot{\alpha}\end{aligned}\tag{4.80}$$

Sustituyendo (4.75) y (4.78) en (4.80), la expresión se reduce a:

$$\dot{e}_\alpha = \frac{\dot{x}^r \sin(e_\alpha)}{\rho} - \dot{\alpha}^r\tag{4.81}$$

Las ecuaciones (4.79) y (4.81) se reescriben en forma matricial como:

$$\begin{bmatrix} \dot{\rho} \\ \dot{e}_\alpha \end{bmatrix} = \begin{bmatrix} -\cos(e_\alpha) & 0 \\ \frac{\sin(e_\alpha)}{\rho} & -1 \end{bmatrix} \begin{bmatrix} \dot{x}^r \\ \dot{\alpha}^r \end{bmatrix} \quad (4.82)$$

La ecuación (4.82) representa el espacio de estados del robot móvil y describe la posición relativa al objetivo. El control cinemático para llegar a la meta está basado en el que se muestra en [7]; sin embargo, en el presente trabajo se realiza una modificación donde el interés es el control únicamente de posición, omitiendo el control de la orientación final, así las ecuaciones de control son:

$$\begin{aligned} \dot{x}^r &= k_\rho \rho \\ \dot{\alpha}^r &= k_\alpha e_\alpha \end{aligned} \quad (4.83)$$

Por lo tanto, la ecuación de lazo cerrado es:

$$\begin{bmatrix} \dot{\rho} \\ \dot{e}_\alpha \end{bmatrix} = \begin{bmatrix} -\cos(e_\alpha) & 0 \\ \frac{\sin(e_\alpha)}{\rho} & -1 \end{bmatrix} \begin{bmatrix} k_\rho \rho \\ k_\alpha e_\alpha \end{bmatrix} = \begin{bmatrix} -k_\rho \rho \cos(e_\alpha) \\ k_\rho \sin(e_\alpha) - k_\alpha e_\alpha \end{bmatrix} \quad (4.84)$$

donde el punto de equilibrio es $(\rho, e_\alpha) = (0, 0)$.

Análisis de estabilidad

Considerando el controlador modificado, se propone la función candidata de Lyapunov $V = \frac{1}{2} q^T P q$, $q^T = [\rho \ e_\alpha]$, $P > 0$, se tiene:

$$V = \frac{1}{2} [\rho \ e_\alpha] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \rho \\ e_\alpha \end{bmatrix} = \frac{1}{2} \rho^2 + \frac{1}{2} e_\alpha^2 \quad (4.85)$$

Derivando (4.85):

$$\dot{V} = [\rho \ e_\alpha] \begin{bmatrix} -k_\rho \rho \cos(e_\alpha) \\ k_\rho \sin(e_\alpha) - k_\alpha e_\alpha \end{bmatrix} = -k_\rho \rho^2 \cos(e_\alpha) + k_\rho e_\alpha \sin(e_\alpha) - k_\alpha e_\alpha^2 \quad (4.86)$$

Con respecto de (4.86), si se asume que $|e_\alpha| < \frac{\pi}{2}$ y, en específico, $e_\alpha \approx 0$, entonces $\cos(e_\alpha) \approx 1$ y $\sin(e_\alpha) \approx e_\alpha$. Por lo tanto, la ecuación (4.86) se reduce a la aproximación local:

$$\dot{V} \approx -k_\rho \rho^2 + k_\rho e_\alpha^2 - k_\alpha e_\alpha^2 = -k_\rho \rho^2 - e_\alpha^2 (k_\alpha - k_\rho) \quad (4.87)$$

Para que (4.87) sea definida negativa localmente, se debe cumplir:

$$\begin{aligned} (k_\alpha - k_\rho) &> 0 \\ k_\alpha &> k_\rho \end{aligned} \quad (4.88)$$

Por lo tanto, se comprueba que el control de posición para alcanzar la meta es localmente asintóticamente estable.

4.3.2. Control de seguimiento de trayectoria

En la Figura 4.16 se presenta el esquema del robot, los marcos de referencia y la trayectoria deseada que, para fines ilustrativos, es una circunferencia. Para este sistema de control se emplean tres configuraciones; la configuración actual del robot $\mathbf{p}_a(t) = [x_a(t) \ y_a(t) \ \alpha_a(t)]^T$, la configuración de la referencia $\mathbf{p}_{ref}(t) = [x_{ref}(t) \ y_{ref}(t) \ \alpha_{ref}(t)]^T$ y la configuración del error $\mathbf{p}_e(t) = [x_e(t) \ y_e(t) \ \alpha_e(t)]^T$.

La trayectoria de referencia depende del tiempo; esta se debe definir de forma análoga a como el modelo cinemático del robot, lo que en este trabajo se menciona como robot virtual (rv), por lo tanto, el modelo cinemático de la trayectoria es:

$$\begin{bmatrix} \dot{x}_{ref}^i \\ \dot{y}_{ref}^i \\ \dot{\alpha}_{ref}^i \end{bmatrix} = \begin{bmatrix} \cos(\alpha_{ref}) & 0 \\ \sin(\alpha_{ref}) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x}_{ref}^{rv} \\ \dot{y}_{ref}^{rv} \\ \dot{\alpha}_{ref}^{rv} \end{bmatrix} = \begin{bmatrix} \dot{x}_{ref}^{rv} \cos(\alpha_{ref}) \\ \dot{x}_{ref}^{rv} \sin(\alpha_{ref}) \\ \dot{\alpha}_{ref}^{rv} \end{bmatrix} \quad (4.89)$$

De la ecuación (4.89) se aprecia que:

$$\frac{\dot{y}_{ref}^i}{\dot{x}_{ref}^i} = \frac{\sin(\alpha_{ref})}{\cos(\alpha_{ref})} = \tan(\alpha_{ref}) \quad (4.90)$$

entonces, con la intención de que $\alpha_{ref} \in (-\pi, \pi]$:

$$\alpha_{ref} = \arctan 2(\dot{y}_{ref}^i, \dot{x}_{ref}^i) \quad (4.91)$$

representa el ángulo del vector tangente a la trayectoria con respecto al marco de referencia inercial. Además se observa que:

$$\begin{aligned} \dot{x}_{ref}^i &= \dot{x}_{ref}^{rv} \cos(\alpha_{ref}) \Rightarrow \dot{x}_{ref}^{rv} = \frac{\dot{x}_{ref}^i}{\cos(\alpha_{ref})} \\ \dot{y}_{ref}^i &= \dot{x}_{ref}^{rv} \sin(\alpha_{ref}) \Rightarrow \dot{x}_{ref}^{rv} = \frac{\dot{y}_{ref}^i}{\sin(\alpha_{ref})} \\ \frac{\dot{x}_{ref}^i}{\cos(\alpha_{ref})} &= \frac{\dot{y}_{ref}^i}{\sin(\alpha_{ref})} \\ \dot{x}_{ref}^i \sin(\alpha_{ref}) &= \dot{y}_{ref}^i \cos(\alpha_{ref}) \end{aligned} \quad (4.92)$$

El resultado final de (4.92) se debe a la restricción no holonómica del robot virtual. Este resultado se presenta también en su análogo del robot real presentado en la ecuación (4.18).

El error se define como:

$$\begin{aligned} \mathbf{p}_e &= \begin{bmatrix} x_e^r \\ y_e^r \\ \alpha_e^r \end{bmatrix} = \begin{bmatrix} \cos(\alpha_a) & \sin(\alpha_a) & 0 \\ -\sin(\alpha_a) & \cos(\alpha_a) & 0 \\ 0 & 0 & 1 \end{bmatrix} (\mathbf{p}_{ref} - \mathbf{p}_a) = R^{-1}(\alpha_a)(\mathbf{p}_{ref} - \mathbf{p}_a) \\ &= R^{-1}(\alpha_a) \begin{bmatrix} x_{ref} - x_a \\ y_{ref} - y_a \\ \alpha_{ref} - \alpha_a \end{bmatrix} = \begin{bmatrix} (x_{ref} - x_a) \cos(\alpha_a) + (y_{ref} - y_a) \sin(\alpha_a) \\ -(x_{ref} - x_a) \sin(\alpha_a) + (y_{ref} - y_a) \cos(\alpha_a) \\ \alpha_{ref} - \alpha_a \end{bmatrix} \end{aligned} \quad (4.93)$$

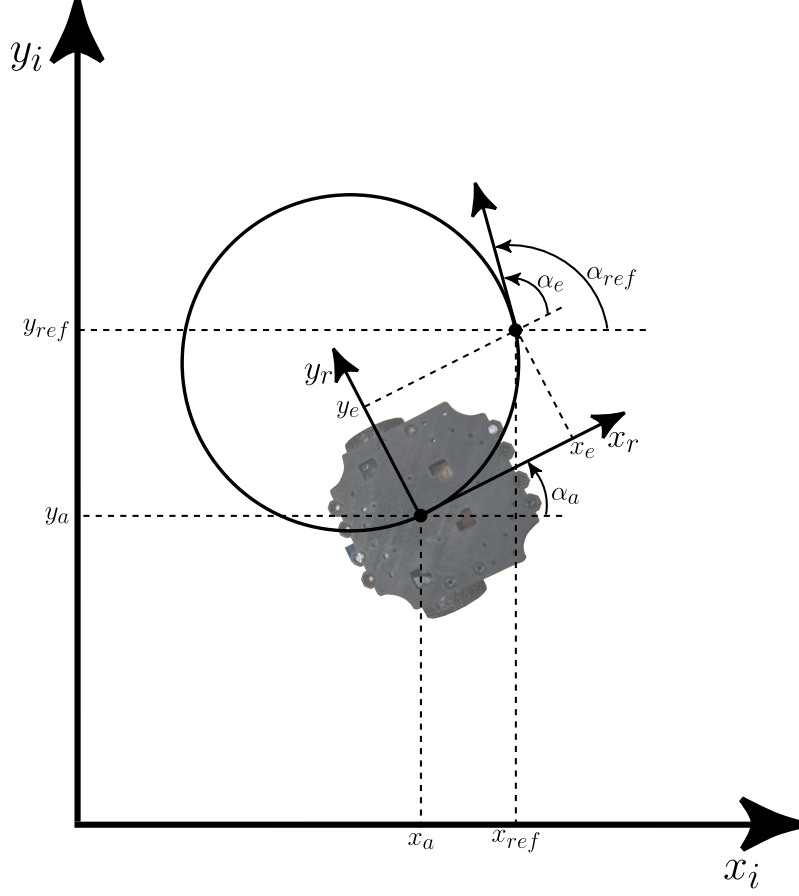


Figura 4.16: Esquema general del objetivo del controlador de seguimiento de trayectoria.

De manera similar a lo que se hace en el controlador anterior, se debe obtener el espacio de estados del robot en función del error. Derivando (4.93), el análisis de $\dot{x}_e$ es:

$$\begin{aligned}
 \frac{d}{dt}(x_e(t)) &= \frac{d}{dt}[(x_{ref} - x_a) \cos(\alpha_a) + (y_{ref} - y_a) \sin(\alpha_a)] \\
 \dot{x}_e &= (\dot{x}_{ref} - \dot{x}_a) \cos(\alpha_a) + (\dot{y}_{ref} - \dot{y}_a) \sin(\alpha_a) + \\
 &\quad + \dot{\alpha}_a(-(x_{ref} - x_a) \sin(\alpha_a) + (y_{ref} - y_a) \cos(\alpha_a)) \\
 &= (\dot{x}_{ref} - \dot{x}_a) \cos(\alpha_a) + (\dot{y}_{ref} - \dot{y}_a) \sin(\alpha_a) + y_e \dot{\alpha}_a \\
 &= \dot{x}_{ref} \cos(\alpha_a) + \dot{y}_{ref} \sin(\alpha_a) - (\dot{x}_a \cos(\alpha_a) + \dot{y}_a \sin(\alpha_a)) + y_e \dot{\alpha}_a
 \end{aligned} \tag{4.94}$$

Sustituyendo el modelo cinemático de la configuración actual del robot, que es igual a la cinemática del robot de la ecuación (4.73), en (4.94), se tiene:

$$\begin{aligned}
 \dot{x}_e &= \dot{x}_{ref} \cos(\alpha_a) + \dot{y}_{ref} \sin(\alpha_a) - (\dot{x}_a^r \cos^2(\alpha_a) + \dot{x}_a^r \sin^2(\alpha_a)) + y_e \dot{\alpha}_a \\
 &= \dot{x}_{ref} \cos(\alpha_a) + \dot{y}_{ref} \sin(\alpha_a) - \dot{x}_a^r + y_e \dot{\alpha}_a
 \end{aligned} \tag{4.95}$$

De (4.93) se sabe que $\alpha_e^r = \alpha_{ref} - \alpha_a$, entonces $\alpha_a = \alpha_{ref} - \alpha_e^r$, por lo que, sustituyendo en (4.95):

$$\dot{x}_e = \dot{x}_{ref} \cos(\alpha_{ref} - \alpha_e^r) + \dot{y}_{ref} \sin(\alpha_{ref} - \alpha_e^r) - \dot{x}_a^r + y_e \dot{\alpha}_a \quad (4.96)$$

Empleando identidades trigonométricas de adición [51] a (4.96):

$$\begin{aligned} \dot{x}_e &= \dot{x}_{ref} [\cos(\alpha_{ref}) \cos(\alpha_e^r) + \sin(\alpha_{ref}) \sin(\alpha_e^r)] + \\ &\quad + \dot{y}_{ref} [\sin(\alpha_{ref}) \cos(\alpha_e^r) - \sin(\alpha_e^r) \cos(\alpha_{ref})] - \dot{x}_a^r + y_e \dot{\alpha}_a \\ &= \cos(\alpha_e^r) [\dot{x}_{ref} \cos(\alpha_{ref}) + \dot{y}_{ref} \sin(\alpha_{ref})] + \\ &\quad + \sin(\alpha_e^r) [\dot{x}_{ref} \sin(\alpha_{ref}) - \dot{y}_{ref} \cos(\alpha_{ref})] - \dot{x}_a^r + y_e \dot{\alpha}_a \end{aligned} \quad (4.97)$$

Debido a la restricción no holonómica, tal como se observa en (4.92), la ecuación (4.97) se reduce a:

$$\dot{x}_e = \cos(\alpha_e^r) [\dot{x}_{ref} \cos(\alpha_{ref}) + \dot{y}_{ref} \sin(\alpha_{ref})] - \dot{x}_a^r + y_e \dot{\alpha}_a \quad (4.98)$$

Sustituyendo el modelo cinemático de la trayectoria (4.89), $\dot{x}_e$ es:

$$\begin{aligned} \dot{x}_e &= \cos(\alpha_e^r) [\dot{x}_{ref}^{rv} \cos^2(\alpha_{ref}) + \dot{x}_{ref}^{rv} \sin^2(\alpha_{ref})] - \dot{x}_a^r + y_e \dot{\alpha}_a \\ &= \cos(\alpha_e^r) \dot{x}_{ref}^{rv} - \dot{x}_a^r + y_e \dot{\alpha}_a \end{aligned} \quad (4.99)$$

El análisis de $\dot{y}_e$ es:

$$\begin{aligned} \frac{d}{dt}(y_e(t)) &= \frac{d}{dt} [-(x_{ref} - x_a) \sin(\alpha_a) + (y_{ref} - y_a) \cos(\alpha_a)] \\ \dot{y}_e &= -(\dot{x}_{ref} - \dot{x}_a) \sin(\alpha_a) + (\dot{y}_{ref} - \dot{y}_a) \cos(\alpha_a) - \\ &\quad - \dot{\alpha}_a ((x_{ref} - x_a) \cos(\alpha_a) + (y_{ref} - y_a) \sin(\alpha_a)) \\ &= -(\dot{x}_{ref} - \dot{x}_a) \sin(\alpha_a) + (\dot{y}_{ref} - \dot{y}_a) \cos(\alpha_a) - x_e \dot{\alpha}_a \\ &= -\dot{x}_{ref} \sin(\alpha_a) + \dot{y}_{ref} \cos(\alpha_a) + \dot{x}_a \sin(\alpha_a) - \dot{y}_a \cos(\alpha_a) - x_e \dot{\alpha}_a \end{aligned} \quad (4.100)$$

Al sustituir la cinemática de la configuración actual (4.73), en (4.100), se tiene:

$$\begin{aligned} \dot{y}_e &= -\dot{x}_{ref} \sin(\alpha_a) + \dot{y}_{ref} \cos(\alpha_a) + \dot{x}_a^r \cos(\alpha_a) \sin(\alpha_a) - \dot{x}_a^r \sin(\alpha_a) \cos(\alpha_a) - x_e \dot{\alpha}_a \\ &= -\dot{x}_{ref} \sin(\alpha_a) + \dot{y}_{ref} \cos(\alpha_a) - x_e \dot{\alpha}_a \end{aligned} \quad (4.101)$$

Se sustituye $\alpha_a = \alpha_{ref} - \alpha_e^r$ en (4.101) y se aplican identidades trigonométricas de adición:

$$\begin{aligned} \dot{y}_e &= -\dot{x}_{ref} [\sin(\alpha_{ref}) \cos(\alpha_e^r) - \cos(\alpha_{ref}) \sin(\alpha_e^r)] + \\ &\quad + \dot{y}_{ref} [\cos(\alpha_{ref}) \cos(\alpha_e^r) + \sin(\alpha_e^r) \sin(\alpha_{ref})] - x_e \dot{\alpha}_a \\ &= \cos(\alpha_e^r) [-\dot{x}_{ref} \sin(\alpha_{ref}) + \dot{y}_{ref} \cos(\alpha_{ref})] + \\ &\quad + \sin(\alpha_e^r) [\dot{x}_{ref} \cos(\alpha_{ref}) + \dot{y}_{ref} \sin(\alpha_{ref})] - x_e \dot{\alpha}_a \end{aligned} \quad (4.102)$$

Por la restricción no holonómica, la ecuación (4.102) se reduce a:

$$\dot{y}_e = \sin(\alpha_e^r)[\dot{x}_{ref} \cos(\alpha_{ref}) + \dot{y}_{ref} \sin(\alpha_{ref})] - x_e \dot{\alpha}_a \quad (4.103)$$

Finalmente, al sustituir el modelo cinemático de la trayectoria (4.89) en (4.103), $\dot{y}_e$ es:

$$\begin{aligned} \dot{y}_e &= \sin(\alpha_e^r)[\dot{x}_{ref}^{rv} \cos^2(\alpha_{ref}) + \dot{x}_{ref}^{rv} \sin^2(\alpha_{ref})] - x_e \dot{\alpha}_a \\ &= \sin(\alpha_e^r) \dot{x}_{ref}^{rv} - x_e \dot{\alpha}_a \end{aligned} \quad (4.104)$$

El análisis de $\dot{\alpha}_e$ es:

$$\begin{aligned} \frac{d}{dt}(\alpha_e) &= \frac{d}{dt}[\alpha_{ref} - \alpha_a] \\ \dot{\alpha}_e &= \dot{\alpha}_{ref} - \dot{\alpha}_a \end{aligned} \quad (4.105)$$

La representación en el espacio de estados del robot móvil en función del error queda definida como se muestra en (4.106).

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\alpha}_e \end{bmatrix} = \begin{bmatrix} \cos(\alpha_e^r) \dot{x}_{ref}^{rv} - \dot{x}_a^r + y_e \dot{\alpha}_a \\ \sin(\alpha_e^r) \dot{x}_{ref}^{rv} - x_e \dot{\alpha}_a \\ \dot{\alpha}_{ref} - \dot{\alpha}_a \end{bmatrix} \quad (4.106)$$

El control de seguimiento de trayectoria se basa en el método propuesto por Kanayama et al. [52], sin embargo, en el presente trabajo se emplea una modificación en la que las ecuaciones de control son:

$$\begin{aligned} \dot{x}^r &= \dot{x}_{ref}^{rv} \cos(\alpha_e^r) + K_x x_e \\ \dot{\alpha}^r &= \dot{\alpha}_{ref} + K_y y_e + K_\alpha \sin(\alpha_e^r) \end{aligned} \quad (4.107)$$

De este modo, la ecuación de lazo cerrado es:

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\alpha}_e \end{bmatrix} = \begin{bmatrix} -K_x x_e + y_e [\dot{\alpha}_{ref} + K_y y_e + K_\alpha \sin(\alpha_e^r)] \\ \sin(\alpha_e^r) \dot{x}_{ref}^{rv} - x_e [\dot{\alpha}_{ref} + K_y y_e + K_\alpha \sin(\alpha_e^r)] \\ -K_y y_e - K_\alpha \sin(\alpha_e^r) \end{bmatrix} \quad (4.108)$$

y el punto de equilibrio es $(x_e, y_e, \alpha_e) = (0, 0, 0)$.

Análisis de estabilidad

La modificación propuesta en la ley de control elimina el factor de velocidad lineal de referencia en la acción angular; por lo tanto, la demostración clásica de estabilidad propuesta por Kanayama et al. no es aplicable. Por este motivo, se realiza el análisis local mediante la linealización de la ecuación (4.108) a través de la matriz jacobiana en el punto de equilibrio $x^* = (0, 0, 0)$. La linealización, es:

$$\begin{aligned} \mathbf{J}_f(x)|_{x^*} &= \begin{bmatrix} -K_x & \dot{\alpha}_{ref} + 2K_y y_e & y_e K_\alpha \cos(\alpha_e^r) \\ -\dot{\alpha}_{ref} - K_y y_e - K_\alpha \sin(\alpha_e^r) & -x_e K_y & \dot{x}_{ref}^{rv} \cos(\alpha_e^r) - x_e K_\alpha \cos(\alpha_e^r) \\ 0 & -K_y & -K_\alpha \cos(\alpha_e^r) \end{bmatrix} \Big|_{x^*} \\ &= \begin{bmatrix} -K_x & \dot{\alpha}_{ref} & 0 \\ -\dot{\alpha}_{ref} & 0 & \dot{x}_{ref}^{rv} \\ 0 & -K_y & -K_\alpha \end{bmatrix} \end{aligned} \quad (4.109)$$

El polinomio característico de la matriz linealizada es:

$$s^3 + s^2(K_\alpha + K_x) + s(\dot{x}_{ref}^{rv}K_y + K_\alpha K_x + \dot{\alpha}_{ref}^2) + \dot{x}_{ref}^{rv}K_x K_y + \dot{\alpha}_{ref}^2 K_\alpha = 0 \quad (4.110)$$

Se sabe que, en un sistema de tercer orden como el mostrado en (4.111), el criterio de Routh-Hurwitz asegura la estabilidad asintótica cuando $a_1 a_2 > a_0 a_3$.

$$a_0 s^3 + a_1 s^2 + a_2 s + a_3 = 0 \quad (4.111)$$

Entonces, considerando el polinomio característico (4.110), se obtiene:

$$(K_\alpha + K_x)(\dot{x}_{ref}^{rv}K_y + K_\alpha K_x + \dot{\alpha}_{ref}^2) > \dot{x}_{ref}^{rv}K_x K_y + \dot{\alpha}_{ref}^2 K_\alpha \quad (4.112)$$

De (4.112) se desprende que la condición de estabilidad de Routh-Hurwitz se cumple para $K_x > 0$, $K_y > 0$ y $K_\alpha > 0$, lo que garantiza que el origen es localmente asintóticamente estable. Este resultado confirma que, mientras el robot se mantenga muy cerca de la trayectoria deseada, el error es cercano a cero.

Resultados y discusión

5.1. Seguimiento del robot

Como se observa en la sección 4.2, se proponen dos métodos de seguimiento del robot, que se comparan en esta sección para evaluar sus características y su desempeño.

La Figura 5.1 muestra el espacio de trabajo en el que se realiza la experimentación. La cámara se coloca utilizando dos trípodes a los costados y un tubo de aluminio de 3 m de longitud y 0.05 m de diámetro. La altura desde el marcador del robot móvil hasta la cámara es de 1.80 m y la orientación es paralela al piso.

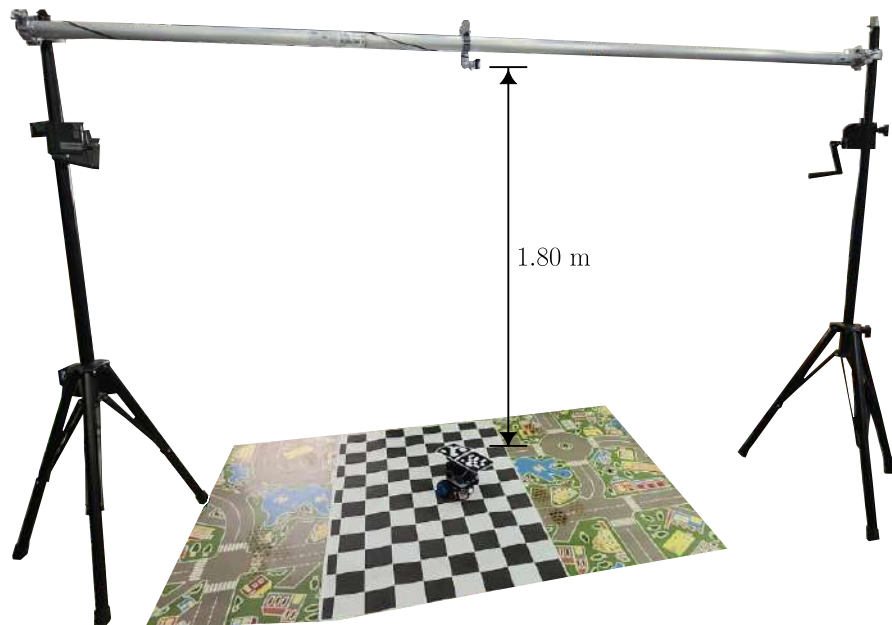


Figura 5.1: Espacio de trabajo del robot y ubicación de la cámara cenital.

5.1.1. Adquisición sistemática de datos del espacio de trabajo mediante un patrón de referencia.

La propuesta de seguimiento visual inicia en la etapa de recolección de datos de entrenamiento; para ello, se adaptan el espacio visual y el patrón de referencia para generar un conjunto de datos de alta densidad. La Figura 5.2 presenta cuatro vistas en las que el tablero patrón se desplaza para cubrir todo el espacio visual de la cámara y se obtiene el mapeo discreto de coordenadas en píxeles correspondiente a las coordenadas espaciales ya medidas y conocidas. En cada vista se adquieren aproximadamente 5,000 coordenadas, lo que da un total de 20,000 datos. La distancia l es de 0.1 m; así, el área del plano visual es de 1.50 m $\times$ 1.20 m.

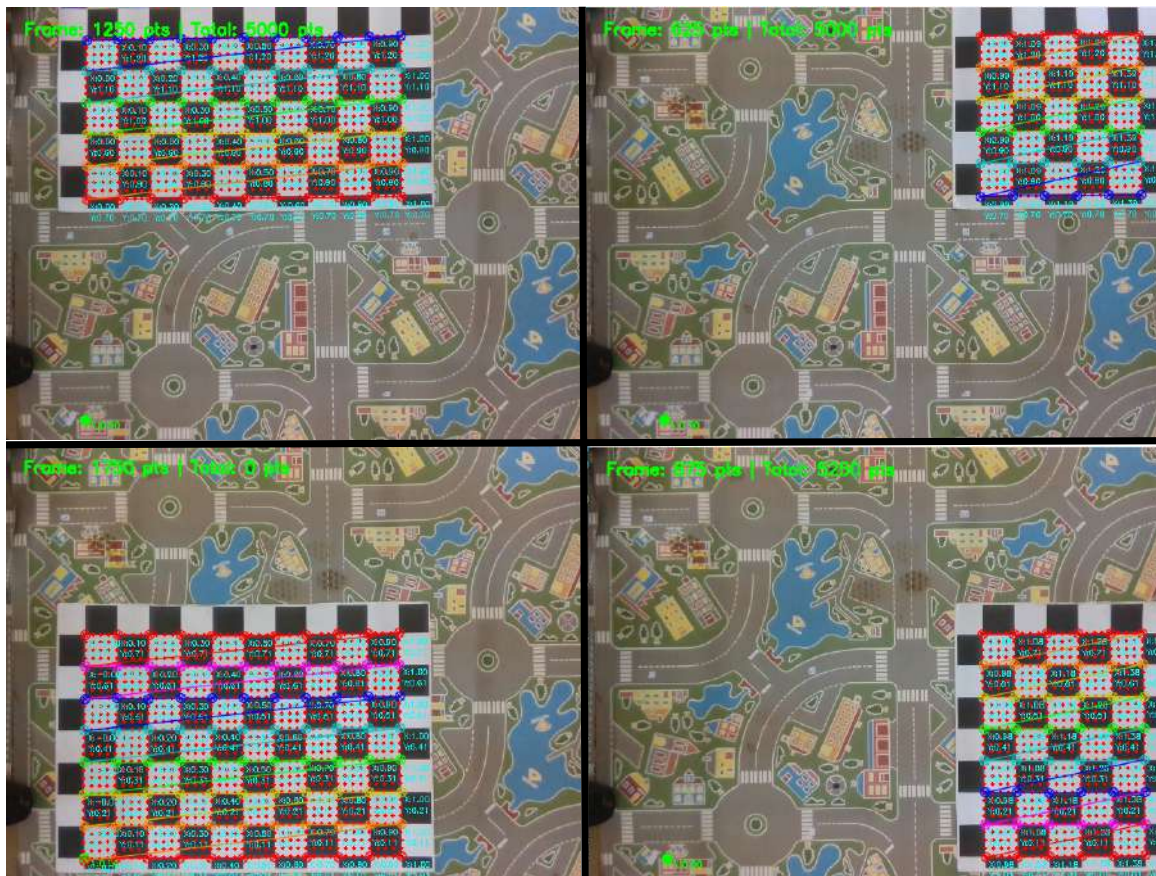


Figura 5.2: Obtención del conjunto de datos de entrenamiento para el mapeo entre coordenadas en píxeles y coordenadas espaciales.

El código de programación que se emplea en la presente sección y cuya compilación y ejecución permiten obtener la Figura 5.2 se presenta en el anexo A.

Entrenamiento neuronal

Uno de los datos relevantes en el entrenamiento neuronal es el número de neuronas; obtener este dato adecuadamente permite lograr un equilibrio entre la precisión de la predicción y el costo computacional. La Figura 5.3 compara la raíz del error cuadrático medio (RMSE) con el número

de neuronas (K). Se puede percatar de que, a partir de $K = 200$, la mejora en el desempeño, en términos de precisión de la predicción, es escasa.

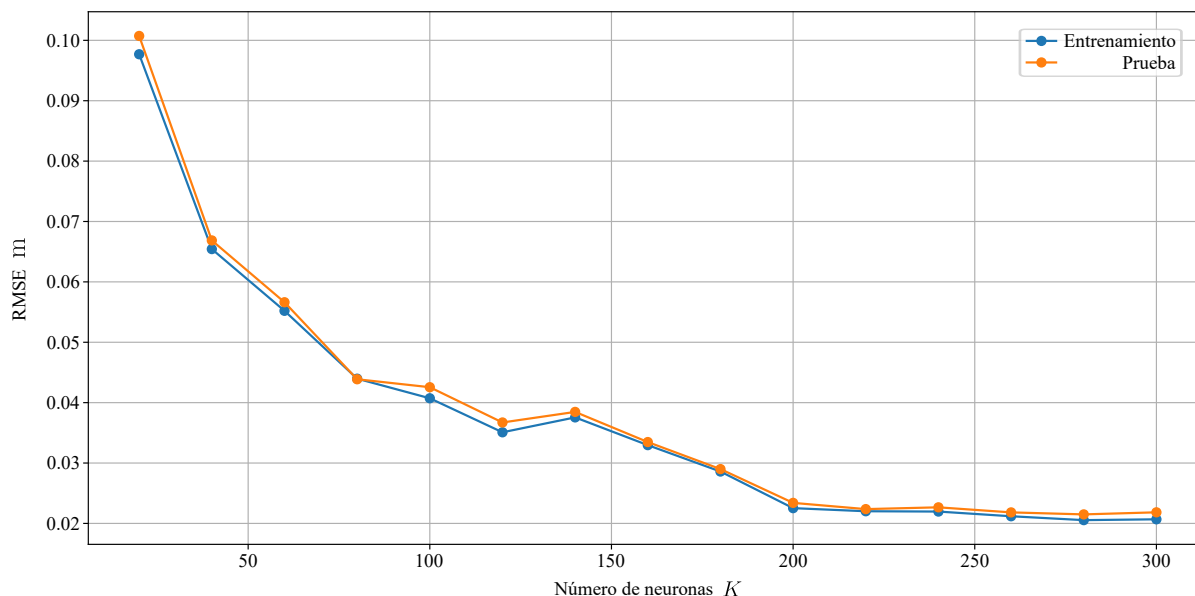


Figura 5.3: Curva de RMSE en función del número de neuronas K .

El comportamiento de las curvas de la Figura 5.3 proporciona información importante sobre el entrenamiento; si $K \in [20, 60)$, el error indica subajuste. Cuando el valor de K aumenta, el error disminuye, lo que implica un mayor costo computacional. Además, se observa la ausencia de sobreajuste, ya que el RMSE de entrenamiento es aproximadamente igual al de prueba para todos los valores de K .

La elección del número de neuronas se toma en $K = 200$; a partir de este punto, el error es bajo e indica precisión en la predicción; además, tomar un valor mayor no disminuye significativamente el error, pero el costo computacional será relevante. De este modo, se mantiene una complejidad moderada del modelo y se evita una carga de trabajo excesiva en el sistema.

La Figura 5.4 muestra la curva de aprendizaje para $K = 200$. Este gráfico ilustra la presencia o ausencia de sobreajuste o subajuste y señala si el entrenamiento es adecuado. El eje horizontal muestra la proporción de datos de entrenamiento y el eje vertical, el RMSE.

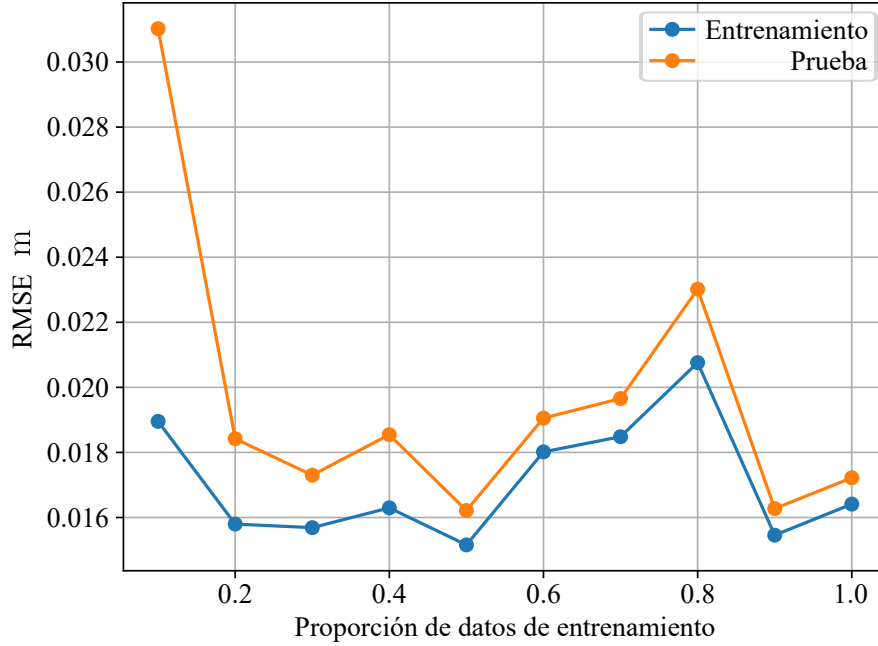


Figura 5.4: Curva de aprendizaje del RMSE en función de los datos de entrenamiento.

La curva de aprendizaje de la Figura 5.4 refuerza la ausencia de sobreajuste al mantener la curva de entrenamiento cercana a la de prueba. De modo que el modelo neuronal RBF con 200 neuronas mantiene un equilibrio entre el ajuste y la capacidad de funcionar ante nuevas entradas. Esta característica es primordial para el sistema propuesto, ya que la tarea de navegación del robot móvil debe estimar posiciones a partir de observaciones que no estuvieron presentes durante el entrenamiento.

Desde la perspectiva de la arquitectura sensoriomotora, el aprendizaje de la red neuronal puede interpretarse como una representación espacial interna del entorno de trabajo. La cual permite transformar la información visual proporcionada por la cámara en información espacial para el sistema de control y así dotar de movimiento al robot. En este sentido, la red neuronal no es únicamente un aproximador de funciones, sino que también constituye el enlace entre la percepción y la acción, así como el componente perceptivo del sistema sensoriomotor propuesto.

Desempeño del seguimiento visual

La evaluación de desempeño consiste en la toma de muestras de posición, considerando el marcador estático, como se aprecia en la Figura 5.5. En distintos puntos del espacio se toman 200 muestras en cada uno y se obtienen las métricas de error de la Tabla 5.1.



Figura 5.5: Seguimiento de marcador con la predicción neuronal.

Tabla 5.1: Métricas de error de seguimiento del marcador móvil (robot) cuando este se encuentra en posición estática. Método de aprendizaje de la representación espacial.

Posición P [m]	$\bar{e}_{euc}$ [m]	MSE	RMSE [m]	σ_x	σ_y
(0.5, 0.0)	0.01979	0.00039	0.01981	0.00107	0.000205
(1.0, 0.0)	0.05110	0.00262	0.05114	0.00228	0.000467
(0.0, 0.5)	0.05959	0.00355	0.05959	0.00006	0.000620
(0.0, 1.0)	0.05183	0.00269	0.05186	0.00023	0.001692
(0.5, 0.5)	0.00200	4×10^{-6}	0.00200	0.00028	0.000370
(0.5, 1.0)	0.01392	0.00019	0.01392	0.00019	0.00003
(1.0, 0.5)	0.01133	0.00013	0.01133	0.00034	0.00040
(1.0, 1.0)	0.00944	0.00009	0.00945	0.00018	0.00040
(1.4, 0.0)	0.02768	0.00077	0.02775	0.00206	0.00036
(1.4, 0.5)	0.04118	0.00170	0.04118	0.00035	0.00049
(1.4, 1.0)	0.00753	0.00006	0.00754	0.00020	0.00047

Con respecto de la Tabla 5.1, la posición P es la coordenada en metros del marcador móvil, $\bar{e}_{euc}$ es el promedio del error euclidiano de posición, MSE es el error cuadrático medio, RMSE es la raíz del error cuadrático medio y, por último, σ_x y σ_y son las desviaciones estándar a lo largo de los ejes x y y , respectivamente. El error euclidiano es mayor cuando el marcador se sitúa en los límites del espacio visual, ya que se aleja del conjunto de datos utilizado por la red neuronal. Bajo estas condiciones, se obtiene un valor máximo de 0.059 m. El RMSE también alcanza un máximo de 0.059 m. Este resultado se observa en las regiones de menor precisión del espacio de trabajo y

ocurre únicamente cerca de los bordes. Para las regiones intermedias del espacio visual, el error disminuye considerablemente, con valores de aproximadamente 0.01-0.02 m, y la precisión promedio de todo el espacio visual es de un RMSE de 0.027 m. La desviación estándar reducida en ambos ejes indica una baja dispersión de las estimaciones, lo que sugiere un comportamiento consistente del método de predicción.

Es importante mencionar que estos resultados corresponden a la caracterización del error del método de localización y no al error de posición durante la operación del vehículo terrestre. En particular, el valor máximo de 0.059 m representa la precisión de la estimación de la posición en determinadas regiones del espacio visual.

El error obtenido demuestra que la representación espacial aprendida proporciona estimaciones de posición precisas y consistentes para la tarea de navegación. En particular, la baja dispersión de las estimaciones y la capacidad de generalización permiten utilizar la información visual sin preocuparse por el sistema de control. Desde la inspiración sensoriomotora, estos resultados validan la función exteroceptiva del método de aprendizaje de la representación espacial, al transformar la información proveniente del sistema de visión en una representación espacial que cierra el ciclo percepción-acción del robot móvil.

5.1.2. Método de marcador ArUco

Los marcadores ArUco permiten obtener la posición y la orientación relativas con respecto de la cámara. Por ello, los resultados se obtienen mediante dos marcadores, uno ubicado en el robot y otro en un extremo del espacio de trabajo. Este último sirve como punto de origen del marco de referencia fijo. La Figura 5.6 muestra la vista de la cámara; se observan los marcadores ArUco y el espacio de trabajo del robot. Los números 50 y 23 mostrados en la imagen son los identificadores de los marcadores del marco de referencia fijo y móvil, respectivamente; en ellos, el eje x está en rojo, el eje y en verde y el eje z en azul.



Figura 5.6: Vista de la cámara y presencia de marcadores ArUco.

La estimación de la pose de los marcadores ArUco requiere los parámetros intrínsecos y de distorsión del sensor RGB de la cámara. Los parámetros se recuperan mediante la interfaz de programación *pyrealsense2*. Así, la estimación de la pose se basa de la información del fabricante.

Se realizan pruebas de rastreo del marcador ArUco del robot para analizar el error y la repetibilidad. La comparación se realiza entre los datos obtenidos mediante el seguimiento con marcadores y mediante la medición empírica. La Tabla 5.2 presenta las métricas de error de seguimiento del segundo marcador cuando este se encuentra estático.

Tabla 5.2: Métricas de error de seguimiento del marcador móvil (robot) cuando este se encuentra en posición estática. Método de marcador ArUco.

Posición P [m]	$\bar{e}_{euc}$ [m]	MSE	RMSE [m]	σ_x	σ_y
(0.5, 0.0)	0.034	0.0035	0.059	0.058	0.007
(1.0, 0.0)	0.059	0.014	0.119	0.117	0.014
(0.0, 0.5)	0.059	0.0036	0.060	0.029	0.0027
(0.0, 1.0)	0.085	0.0079	0.089	0.046	0.007
(0.5, 0.5)	0.050	0.0032	0.056	0.039	0.006
(0.5, 1.0)	0.08	0.0068	0.083	0.041	0.0083
(1.0, 0.5)	0.099	0.028	0.168	0.156	0.017
(1.0, 1.0)	0.113	0.024	0.155	0.138	0.017

La Tabla 5.2 presenta los siguientes hallazgos: el error euclidiano aumenta a medida que el marcador móvil se aleja del fijo, alcanzando un máximo de 0.113 m en la coordenada más alejada. Por otro lado, el error RMSE alcanza un máximo de 0.168 m y, en promedio, es de 0.099 m, lo que

indica que no es posible obtener una alta precisión con los marcadores ArUco. Por lo tanto, no es posible localizar al robot con una precisión inferior a 0.05 m. Otro dato importante a señalar es la desviación estándar del eje x con respecto de la del eje y ; se aprecia que hay mayor confiabilidad en la medición del eje y . Lo anterior puede deberse a factores que van desde la resolución proyectada en el eje x hasta distorsiones derivadas de la calibración de la cámara.

5.2. Simulación del método de aprendizaje de la representación espacial

Como se describe en la sección anterior, la red neuronal RBF forma parte del sistema de percepción visual y se encarga de estimar la posición del robot a partir de las coordenadas en píxeles. De este modo, la simulación incorpora la incertidumbre de dicha estimación mediante un sensor virtual basado en los errores observados durante el entrenamiento de la red.

Las métricas de error del entrenamiento son las siguientes:

$$\begin{aligned}
 \mu &= \begin{bmatrix} 0.000716 \\ 0.000233 \end{bmatrix} \text{ m} \\
 \Sigma &= \begin{bmatrix} 0.000449 & 0.000128 \\ 0.000128 & 0.000286 \end{bmatrix} \\
 RMSE &= 0.0271 \text{ m} \\
 \bar{e}_{eucMax} &= 0.390 \text{ m} \\
 \bar{e}_{eucMean} &= 0.017 \text{ m}
 \end{aligned} \tag{5.1}$$

donde μ es la media del error de predicción sobre el conjunto de datos de prueba, Σ es la matriz de covarianza del error, $\bar{e}_{eucMax}$ es el error euclidiano máximo y $\bar{e}_{eucMean}$ es el error euclidiano promedio.

Las métricas de error no presentan sesgo, lo que indica un buen desempeño del estimador. La covarianza indica una mayor dispersión del error en el eje x que en el eje y . Además, los términos no diagonales sugieren una correlación entre ambos errores. El valor de RMSE coincide con el promedio obtenido en las pruebas de desempeño experimentales, lo que indica que el modelo estadístico reproduce adecuadamente el comportamiento observado del sistema de percepción. La Figura 5.7 muestra la distribución espacial del error.

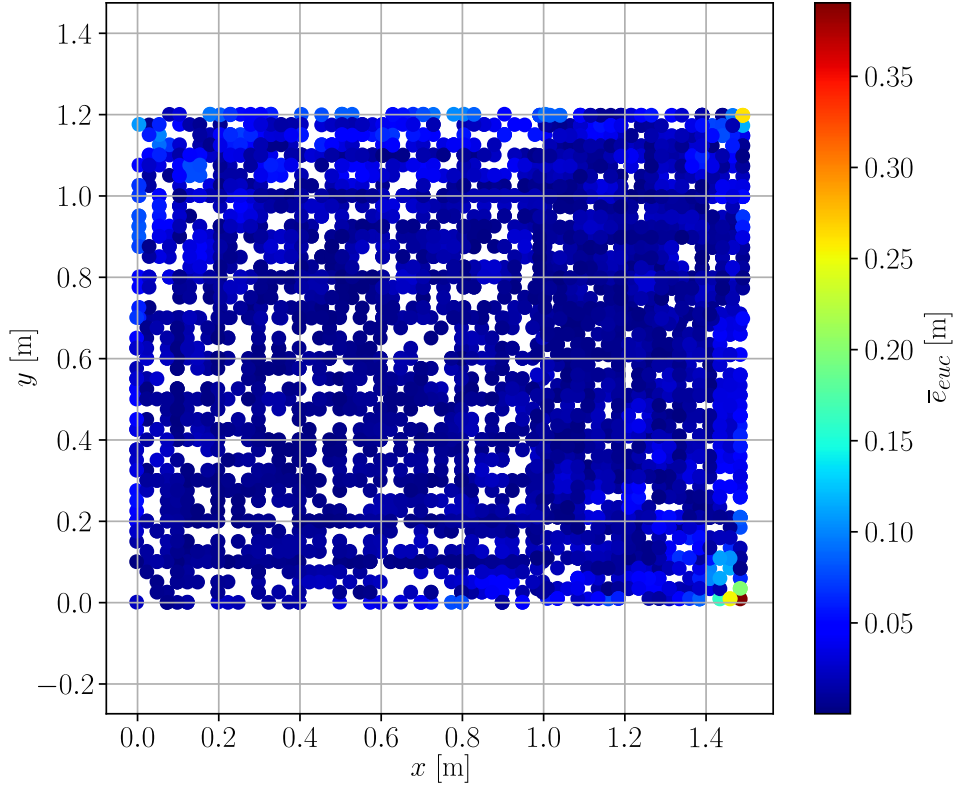


Figura 5.7: Distribución del error en función de la posición.

La Figura 5.7 muestra un hallazgo relevante: el error depende de la posición; es decir, no sigue una distribución fija, sino que varía en función de ella. Se observan los valores máximos en las esquinas más alejadas del conjunto de datos, en las coordenadas $(1.5, 1.2)$ y $(1.5, 0.0)$.

El sistema de adquisición se modela mediante un sensor virtual cuya incertidumbre depende de la posición del robot, a partir de la distribución de errores del entrenamiento de la red neuronal RBF.

Dado que el error depende de la posición, presenta una distribución asimétrica y no resulta conveniente modelarlo con una distribución gaussiana. La propuesta, entonces, es emplear *bootstrap* para preservar la correlación entre los ejes x y y , así como cualquier asimetría direccional del error.

La razón de emplear *bootstrap* es evitar asumir un modelo probabilístico específico del error. Así, las muestras del error se generan directamente a partir de los errores experimentales obtenidos durante el entrenamiento, preservando su distribución, asimetrías y correlación entre los errores en ambos ejes.

El procedimiento para incorporar el sensor virtual dentro del lazo de control se resume en el diagrama de flujo de la Figura 5.8.

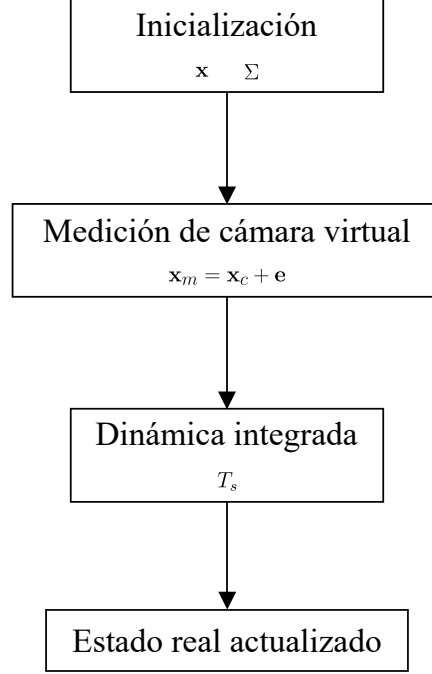


Figura 5.8: Diagrama de flujo de la simulación del sensor de percepción virtual.

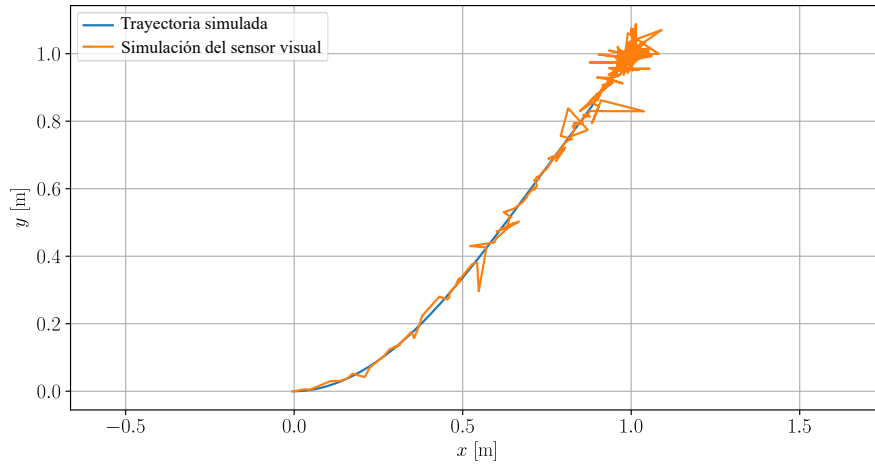
Con respecto de la Figura 5.8, la simulación se inicializa con el estado del robot ($\mathbf{x}$) y las estadísticas de error de la red RBF obtenidas durante el entrenamiento (Σ). En cada ciclo de $T_s = 30$ ms (para mantener la consistencia con las pruebas experimentales), la posición real ($\mathbf{x}_c$) del robot se obtiene a partir del modelo dinámico. A partir de esta posición, se genera la medición del sensor virtual mediante:

$$\mathbf{x}_m = \mathbf{x}_c + \mathbf{e}$$

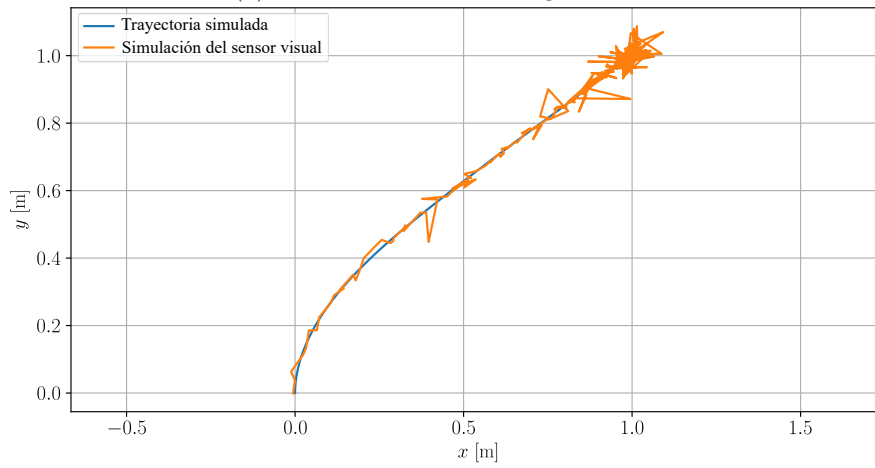
donde $\mathbf{e}$ es la muestra aleatoria del error de percepción generada mediante *bootstrap* a partir de los errores experimentales obtenidos durante el entrenamiento de la red RBF. Así, $\mathbf{x}_m$ corresponde a la posición que el sistema de visión hipotéticamente reporta para la posición real $\mathbf{x}_c$.

El valor de $\mathbf{x}_m$ se mantiene constante durante el siguiente intervalo de muestreo mediante un retenedor de orden cero y se proporciona al controlador. El cual calcula las señales de control utilizadas para integrar la dinámica del robot (reportada en la sección 4.1.1). Para finalizar, en cada ciclo se actualiza el estado real del robot y se genera una nueva muestra de error mediante *bootstrap* para obtener la siguiente medición del sensor virtual. De esta manera, la simulación involucra la interacción entre la dinámica, el sistema de percepción y el controlador y permite reconstruir la trayectoria real a partir de la estimada por el sistema de visión. En el anexo B se muestra el código de programación que se emplea en la simulación.

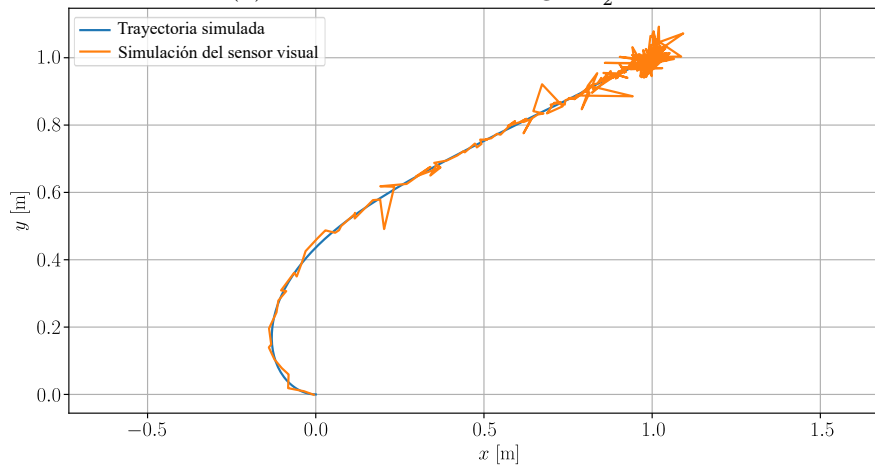
Se realizan tres pruebas simuladas variando la condición inicial de orientación. En todas las pruebas la condición inicial de posición es (0,0) y la posición deseada es (1.0,1.0). La Figura 5.9 muestra las gráficas de las tres simulaciones.



(a) Condición inicial de ángulo 0 rad



(b) Condición inicial de ángulo $\frac{\pi}{2}$ rad



(c) Condición inicial de ángulo π rad

Figura 5.9: Simulación.

La Figura 5.9 muestra tres gráficas correspondientes a cada prueba simulada. La señal corres-

pendiente a la trayectoria que sigue el robot simulado es de color azul y la trayectoria observada por el sistema de visión virtual, que contempla el error añadido, se muestra en naranja. El gráfico 5.9a muestra el resultado con una orientación inicial del robot de 0 rad. En la Figura 5.9b, el robot parte con una orientación de $\frac{\pi}{2}$ rad. Por último, en la respuesta 5.9c el robot comienza con una orientación de π rad.

En las tres gráficas se aprecia, como se esperaba, el comportamiento imperfecto del sensor virtual debido a las métricas de error del entrenamiento neuronal. A lo largo del recorrido en las tres simulaciones, el comportamiento del robot converge a la posición deseada incluso con mediciones imperfectas del sistema de visión; sin embargo, un aspecto relevante es que cuando el robot virtual alcanza su posición final, la cámara continúa enviando información con errores y se observa una medición ruidosa. Dado que se añade ruido, este comportamiento final es normal y previsto. En las pruebas experimentales se espera que dicho efecto sea significativamente menor, ya que una vez que el robot permanece estático, las variaciones de la medición se limitan al ruido propio del sistema de visión.

Los resultados de esta sección sugieren que la preservación de la correlación entre los errores de ambos ejes y de la variabilidad del error mediante el método *bootstrap* permite reproducir fielmente el comportamiento observado experimentalmente en el sistema de percepción visual. A diferencia de un modelo basado en ruido gaussiano, el sensor virtual conserva las características estadísticas aprendidas por la red RBF. Esta estrategia permite percibir en condiciones más cercanas a las observadas en el sistema real, donde la incertidumbre sensorial depende del contexto y de la posición del robot en el espacio de trabajo.

5.3. Control de posición con seguimiento visual

Después de observar el desempeño de ambos métodos de seguimiento, así como la simulación del método de aprendizaje de la representación espacial, en la presente sección se implementan experimentalmente los controladores de la sección 4.3, que también incluyen el controlador de la sección 4.1.2 para el control de la velocidad de las ruedas. El objetivo es realizar un seguimiento continuo del robot móvil y comparar el método propuesto con el de los marcadores ArUco. En el anexo C se presenta el código de programación embebido en el microcontrolador Raspberry Pi Pico.

5.3.1. Control de posición para llegar a la meta basado en representación espacial

El método de seguimiento visual propuesto permite seguir la posición del robot móvil con una precisión de aproximadamente 0.03 m. La orientación del robot móvil se obtiene con ayuda de dos marcadores, cuyas coordenadas en píxeles se conocen. Un marcador se encuentra en el frente del robot y otro en la parte trasera. De esta manera, se traza un vector cuyo ángulo coincide con el de la posición angular del robot móvil. La Figura 5.10 ilustra el método para determinar la posición angular.

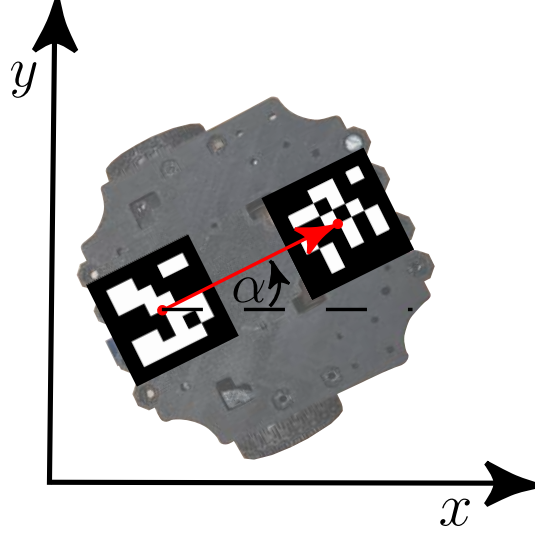


Figura 5.10: Obtención de la posición angular del robot móvil mediante dos marcadores.

Considerando el análisis de estabilidad del controlador de posición y la condición de (4.88). Las ganancias del controlador son $k_\alpha = 5.0$ y $k_\rho = 0.8$. El periodo de muestreo de todas las pruebas experimentales es $t_s = 30$ ms.

Las Figuras 5.11, 5.13 y 5.15 presentan tres pruebas experimentales bajo la misma condición inicial de posición $x = 0.5$ m y $y = 0$ m; no obstante, la condición inicial de orientación varía: $\alpha = 0, \frac{\pi}{2}, \pi$ rad. En todas las ilustraciones, se aprecian la posición en x y y , la orientación α y los esfuerzos de control de la rueda derecha u_r y de la izquierda u_l .

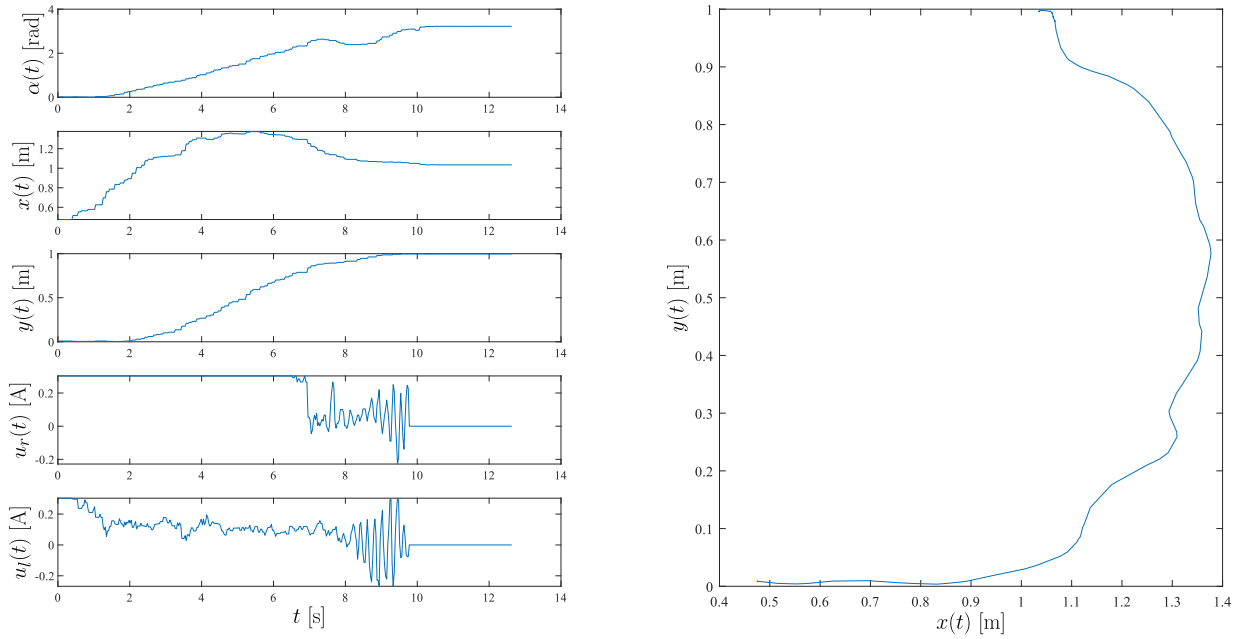


Figura 5.11: Prueba experimental del controlador de posición de llegada a la meta, con orientación inicial de 0 rad. Método de aprendizaje de la representación espacial.

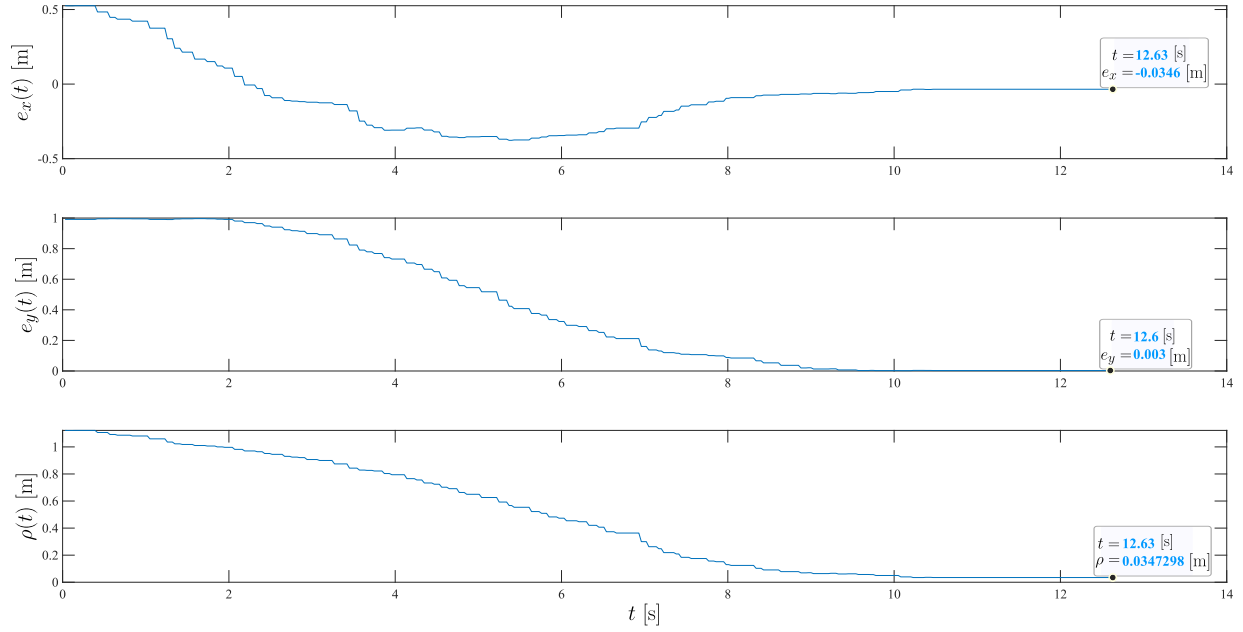


Figura 5.12: Gráficas de error de posición en los ejes x y y , y de la norma del error ρ . La prueba experimental es con $\alpha = 0$ rad.

La Figura 5.12 muestra las gráficas de los errores de posición en ambos ejes y de su norma. Los errores de estado estacionario experimentales son:

$$e_x = -0.0346 \text{ m}$$

$$e_y = 0.003 \text{ m}$$

$$\rho = 0.035 \text{ m}$$

Estos resultados indican que el error en estado estacionario permanece por debajo de 0.05 m y se encuentra dentro del rango de precisión experimental del sistema de seguimiento visual.

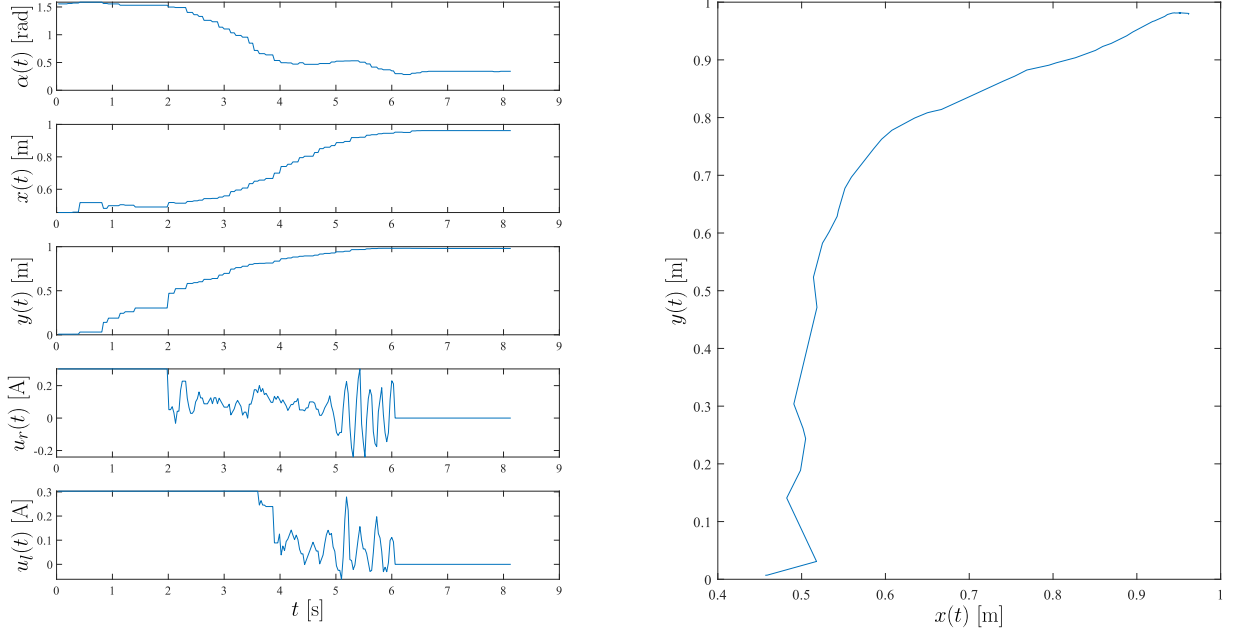


Figura 5.13: Prueba experimental del controlador de posición de llegada a la meta, con una orientación inicial de $\frac{\pi}{2}$ rad. Método de aprendizaje de la representación espacial.

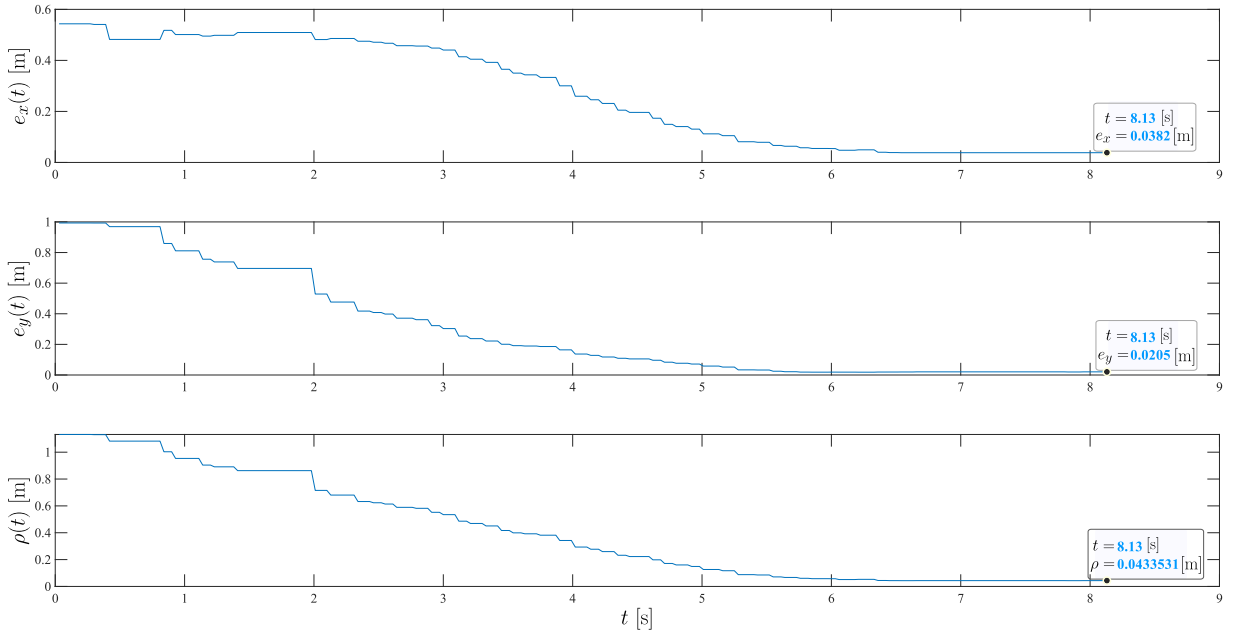


Figura 5.14: Gráficas de error de posición en los ejes x y y , y de la norma del error ρ . La prueba experimental es con $\alpha = \frac{\pi}{2}$ rad.

Las gráficas de la Figura 5.14 presentan los siguientes errores en estado estacionario:

$$e_x = 0.0382 \text{ m}$$

$$e_y = 0.0205 \text{ m}$$

$$\rho = 0.0433 \text{ m}$$

Nuevamente, los errores se encuentran dentro del rango de precisión esperado.

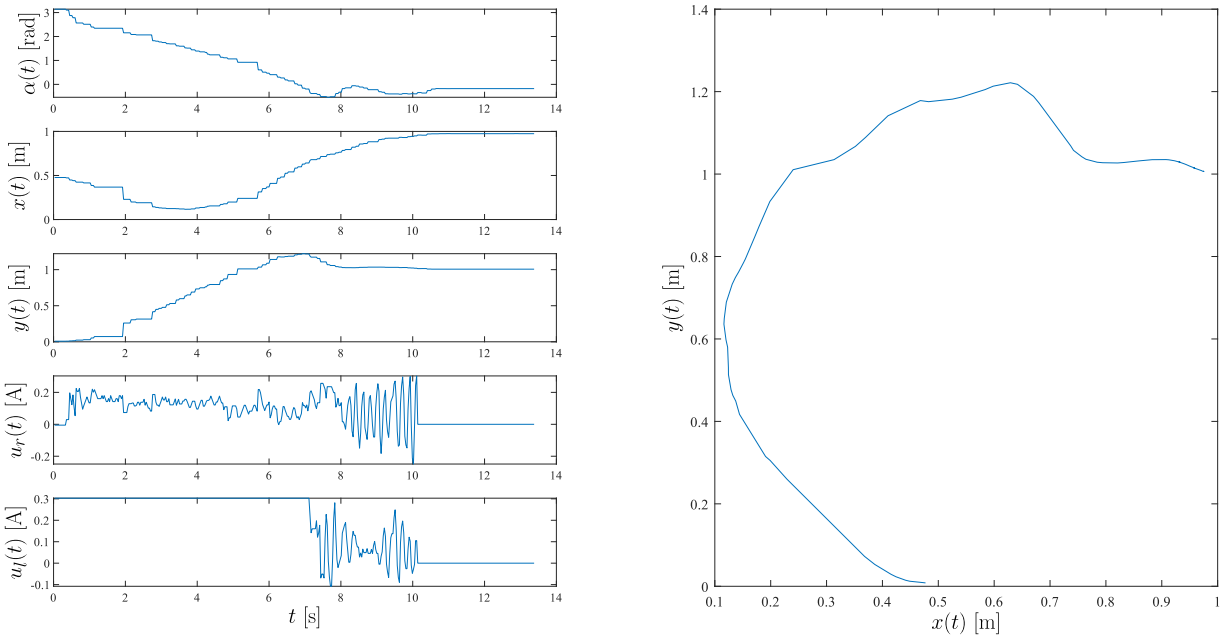


Figura 5.15: Prueba experimental del controlador de posición de llegada a la meta, con orientación inicial de π rad. Método de aprendizaje de la representación espacial.

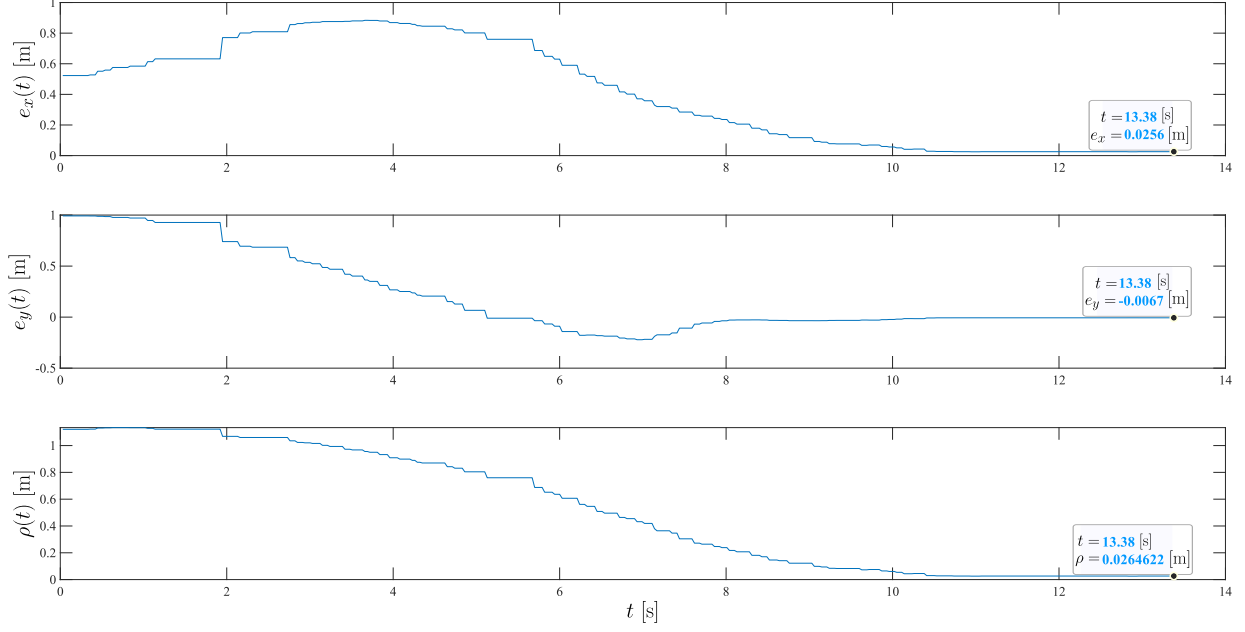


Figura 5.16: Gráficas de error de posición en los ejes x y y , y de la norma del error ρ . La prueba experimental es con $\alpha = \pi$ rad.

Los errores en estado estacionario de la Figura 5.16 son:

$$\begin{aligned} e_x &= 0.0256 \text{ m} \\ e_y &= -0.0067 \text{ m} \\ \rho &= 0.0265 \text{ m} \end{aligned}$$

Como se observa en las tres pruebas experimentales, el error absoluto traslacional permanece por debajo de 0.05 m en estado estacionario, lo que cumple con el criterio de desempeño deseado. Considerando que el sistema de seguimiento visual presenta un RMSE promedio de aproximadamente 0.027 m, una parte del error en estado estacionario puede atribuirse a la incertidumbre de la localización. Sin embargo, debe notarse que el error no depende únicamente de la precisión del seguimiento visual, sino también de las ganancias del controlador y de la dinámica no modelada del robot.

Los resultados obtenidos al ejecutar el controlador de posición para alcanzar la meta permiten evaluar la hipótesis, ya que se establece como criterio un error de traslación absoluto máximo inferior a 0.05 m. Las Figuras 5.12, 5.14 y 5.16 muestran los errores de posición de los ejes x y y , así como el error euclidiano, que se encuentran por debajo de 0.05 m. El sistema de control, en conjunto con el método de seguimiento visual, mantiene el error de posición dentro del límite, aun cuando el método de localización presenta errores mayores en determinadas zonas del espacio de trabajo.

Los resultados experimentales muestran que la representación espacial aprendida proporciona estimaciones de posición con suficiente precisión como para ser ingresadas en el sistema de control y permitir que el robot móvil alcance la posición objetivo bajo distintas condiciones iniciales de orientación. Así, la información visual se transforma en una representación espacial funcional capaz

de guiar el movimiento del robot, integrando el sistema de percepción con el controlador de posición. El sistema establece un vínculo entre la percepción y la acción, validando experimentalmente la inspiración de la arquitectura sensoriomotora propuesta y verificando que la representación espacial es funcional para el control del robot.

Todas las pruebas experimentales de la presente sección se realizan con base en el código proporcionado en el anexo D.

5.3.2. Control de posición para llegar a la meta empleando el método de los marcadores ArUco

Al realizar pruebas experimentales bajo las mismas condiciones que en el método de aprendizaje de la representación espacial, los resultados se presentan en las Figuras 5.17, 5.19 y 5.21. El método de los marcadores ArUco permite obtener directamente el ángulo del robot, por lo que basta con usar el método mostrado en la sección 4.2.2.

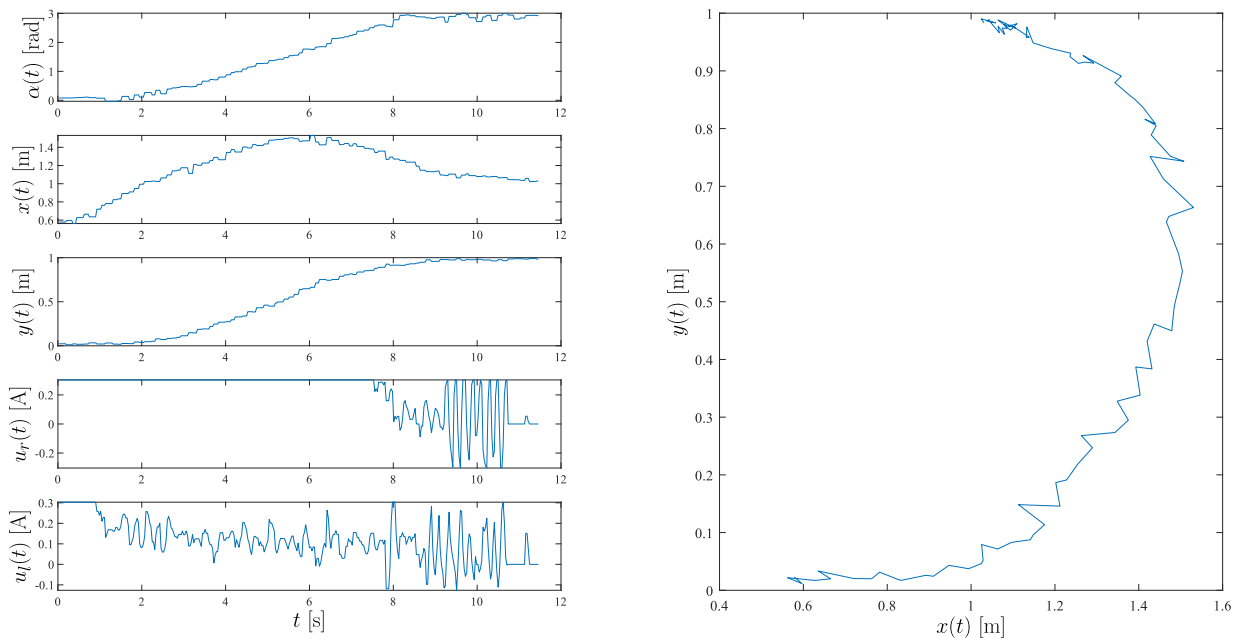


Figura 5.17: Prueba experimental del controlador de posición de llegada a la meta, con orientación inicial de 0 rad. Método de los marcadores ArUco.

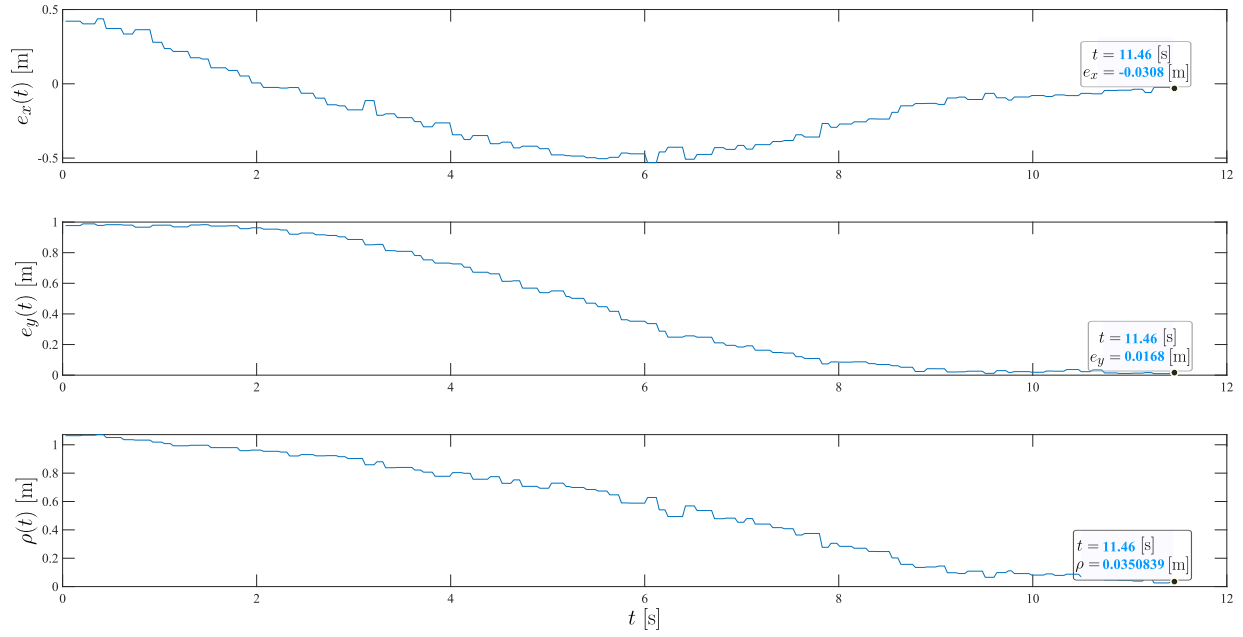


Figura 5.18: Gráficas de error de posición en los ejes x y y y de la norma del error ρ , obtenidas mediante seguimiento visual basado en marcadores ArUco. La prueba experimental se realiza con $\alpha = 0$ rad.

La Figura 5.18 exhibe los siguientes errores de estado estacionario:

$$e_x = -0.0308 \text{ m}$$

$$e_y = 0.0168 \text{ m}$$

$$\rho = 0.035 \text{ m}$$

Estos errores con el método de marcadores ArUco se mantienen por debajo del rango deseado de precisión experimental.

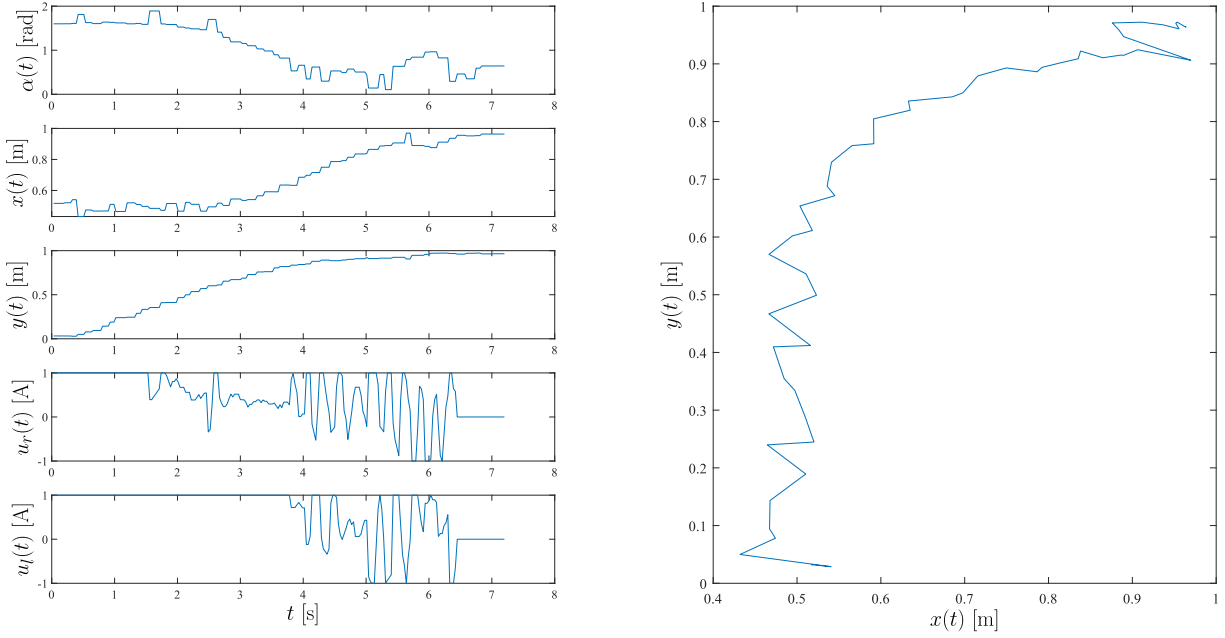


Figura 5.19: Prueba experimental del controlador de posición de llegada a la meta, con una orientación inicial de $\frac{\pi}{2}$ rad. Método de los marcadores ArUco.

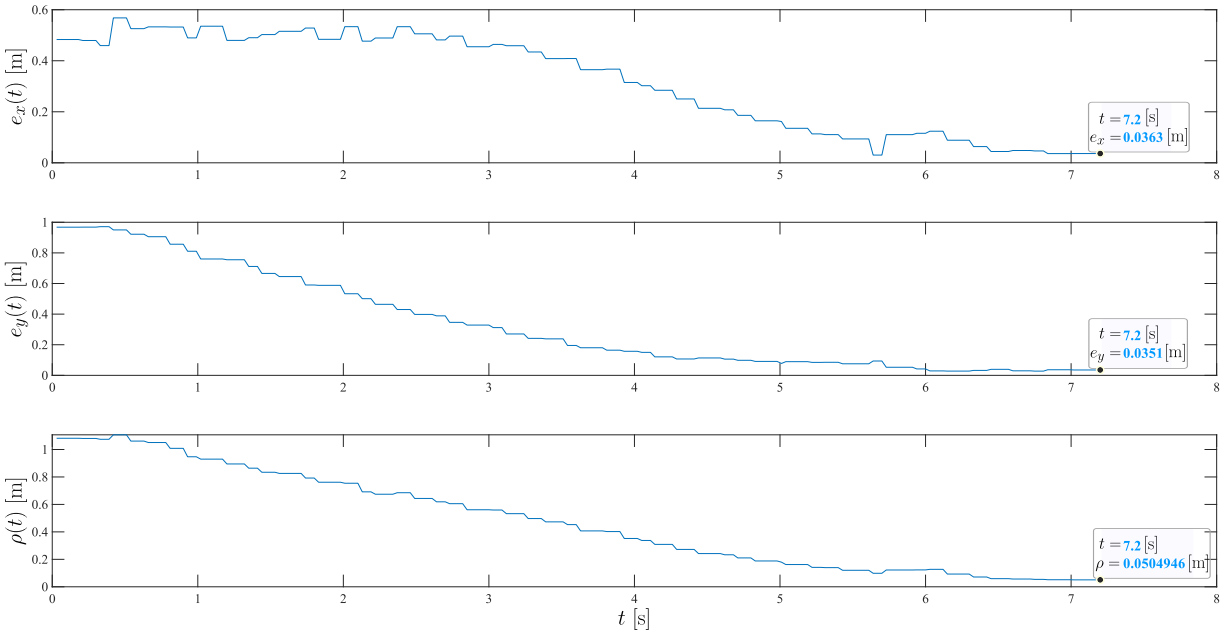


Figura 5.20: Gráficas de error de posición en los ejes x y y y de la norma del error ρ , obtenidas mediante seguimiento visual basado en marcadores ArUco. La prueba experimental se realiza con $\alpha = \frac{\pi}{2}$ rad.

Los errores de estado estacionario de la Figura 5.20 son:

$$e_x = 0.0363 \text{ m}$$

$$e_y = 0.0351 \text{ m}$$

$$\rho = 0.0505 \text{ m}$$

Se observa que los errores en los ejes se encuentran dentro del rango deseado, a diferencia del error absoluto, que supera 0.05 m.

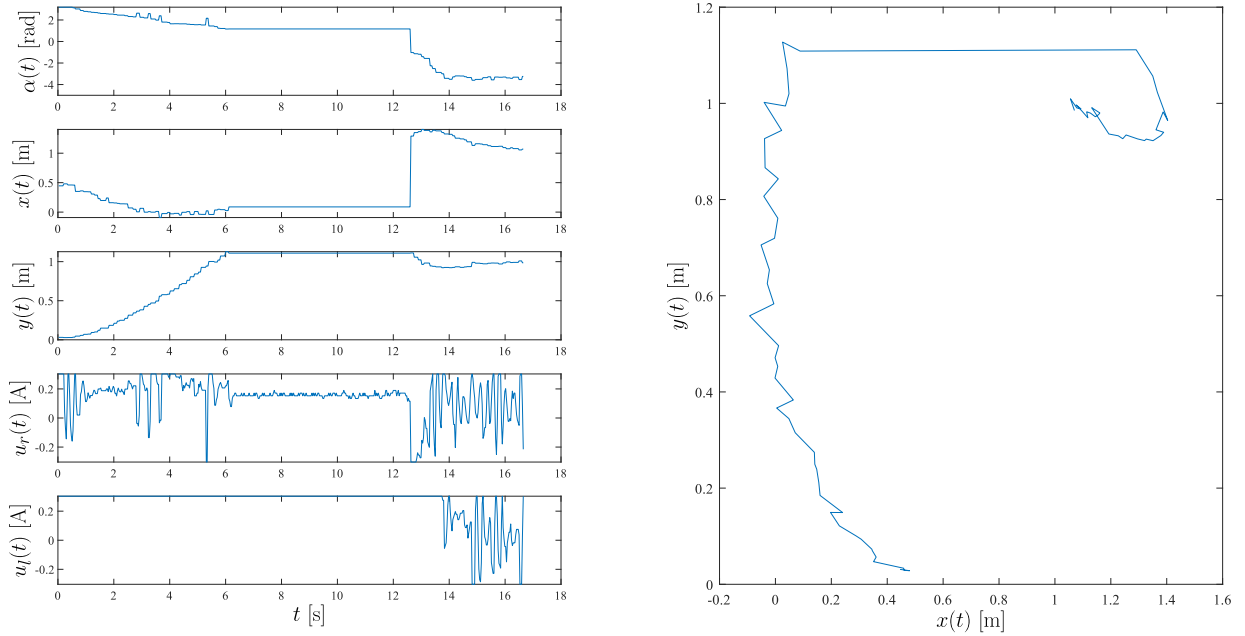


Figura 5.21: Prueba experimental del controlador de posición de llegada a la meta, con orientación inicial de π rad. Método de los marcadores ArUco.

En las gráficas de la Figura 5.21 se observa un comportamiento que no corresponde al movimiento del robot. En el intervalo de tiempo aproximado de $6 < t < 13$ segundos, el sistema de seguimiento deja de localizar al robot, el robot sale del espacio visual y las gráficas muestran el último dato válido. Es por esto que los estados $x(t), y(t), \alpha(t)$ parecen constantes, lo cual no es físicamente cierto. Este comportamiento probablemente se debe al ruido que introduce el método de seguimiento, lo que impide al controlador corregir a tiempo la trayectoria del robot móvil y provoca su desaparición del espacio de trabajo. No obstante, el controlador es lo suficientemente robusto como para que, en cuanto el robot regrese al espacio visual, alcance la referencia deseada.

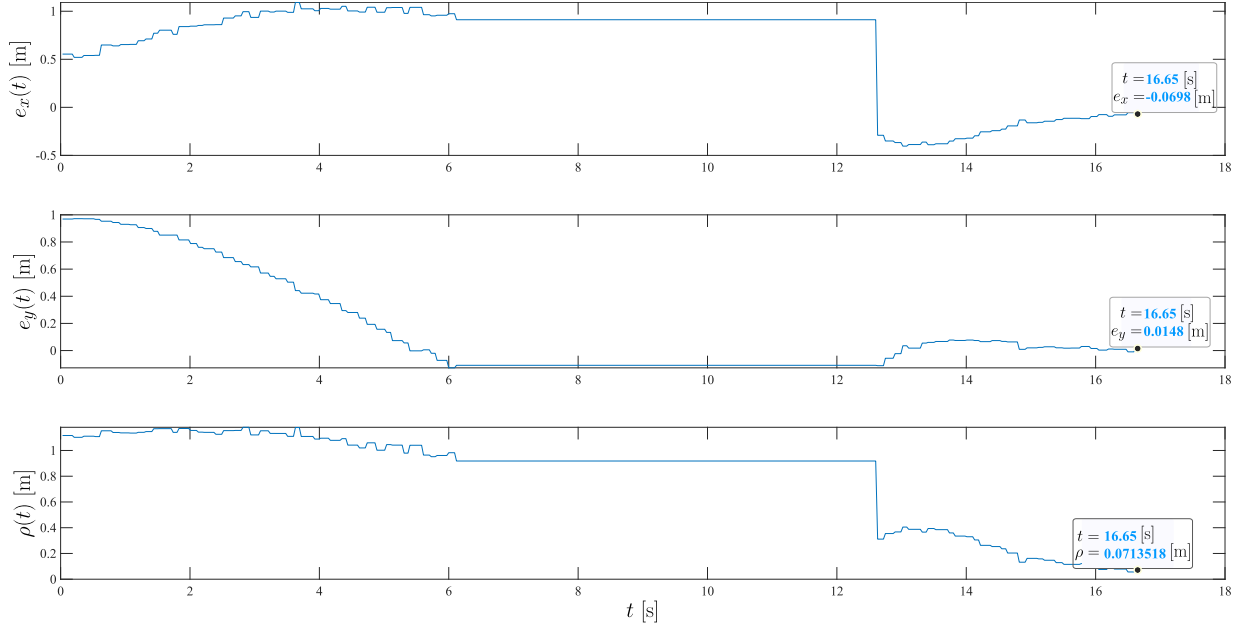


Figura 5.22: Gráficas de error de posición en los ejes x y y y de la norma del error ρ , obtenidas mediante seguimiento visual basado en marcadores ArUco. La prueba experimental se realiza con $\alpha = \pi$ rad.

La Figura 5.20 presenta los siguientes errores de estado estacionario:

$$\begin{aligned} e_x &= -0.07 \text{ m} \\ e_y &= 0.0148 \text{ m} \\ \rho &= 0.0714 \text{ m} \end{aligned}$$

En esta prueba experimental, el error en el eje x y el error absoluto superan los límites permisibles. Exhiben el mismo comportamiento interrumpido descrito en la Figura anterior. Otro dato importante es que el tiempo de recorrido es de aproximadamente 17 s, lo cual es mayor que en la prueba experimental del método propuesto.

Las tres pruebas experimentales presentan diferentes valores del error en estado estacionario; algunos se encuentran dentro del rango deseado. Este sistema de seguimiento visual presenta un RMSE promedio de aproximadamente 0.099 m; una parte considerable del error en estado estacionario se atribuye a la incertidumbre de la localización. Aunque la precisión del seguimiento visual es baja, las ganancias del controlador y la dinámica no modelada del robot también inciden en los resultados.

Las pruebas experimentales del controlador de posición con seguimiento visual basado en marcadores ArUco se realizan con el código de programación que se presenta en el anexo E.

5.4. Discusión de los resultados

A lo largo de este capítulo se presentan las métricas de error, la simulación del método de seguimiento visual propuesto y las pruebas experimentales del aprendizaje de la representación

espacial y del método basado en marcadores ArUco. La Tabla 5.3 presenta la comparación de los errores promedio de ambos métodos.

Tabla 5.3: Comparación de ambos métodos con los promedios de los errores obtenidos.

Método	$\bar{e}_{euc}$ [m]	$\overline{MSE}$	$\overline{RMSE}$ [m]	$\bar{\sigma}_x$	$\bar{\sigma}_y$
Aprendizaje	0.0269	0.0011	0.0269	0.0007	0.0005
ArUco	0.0724	0.0114	0.099	0.078	0.0099

El hallazgo más relevante que muestra la Tabla 5.3 es que, en todas las métricas de error, el método propuesto presenta un error menor; se observan mejoras en la precisión y en la reducción del ruido, lo que se traduce en una mejor estimación de la posición del robot móvil. La variabilidad de la medición se reduce y se mejora la robustez de la localización.

Las gráficas experimentales de ambos métodos presentan errores en estado estacionario similares; sin embargo, la propuesta muestra una disminución del RMSE y de la desviación estándar. Se aprecia una mayor estabilidad y consistencia en la estimación de la posición al producir señales suaves. Las gráficas del método de marcadores ArUco presentan oscilaciones que indican la presencia de ruido y reducen la confiabilidad de la estimación de la posición.

Con respecto de la simulación y las gráficas experimentales del método de aprendizaje propuesto, se aprecia un comportamiento experimental más suave y con menos ruido en comparación con las gráficas simuladas. La suposición de que la disminución del error en la posición final del robot fue acertada; las pruebas experimentales reducen el error final observado en las señales de la cámara virtual.

El método de aprendizaje emplea una red neuronal que produce una estimación más suave debido a la naturaleza interpolante del modelo aprendido. De este modo, se convierte en un método alternativo de mapeo plano para entornos estructurados con cámara fija, que evita la calibración explícita, reduce la acumulación del error numérico, la complejidad experimental y el ruido en la predicción de la posición.

A continuación se presenta la Tabla 5.4 que muestra las diferencias entre el método propuesto y los trabajos del estado del arte.

Tabla 5.4: Comparación del método propuesto con otras metodologías.

Característica	[16]	[18]	Propuesta
Visión global	✓	✓	✓
Cámara externa	✓	✓	✓
Estimación de posición	✓	✓	✓
Empleo de marcador visual	✓	✓	✓
Transformación imagen-espacio	✓	✓	✓
Método de transformación	Homografía	Trans. de perspectiva	Red neuronal RBF
Modelo de percepción aprendido	-	-	✓
Evaluación del error de localización	✓	✓	✓
Análisis de la dependencia espacial del error	-	-	✓
Consideración de incertidumbre del sensor	-	-	✓
Modelo de la incertidumbre	-	-	✓
Simulación considerando incertidumbre del sensor	-	-	✓
Control de movimiento basado en información visual	✓	✓	N/A
Planificación de trayectoria mediante visión	-	✓	N/A
Controlador de posición y trayectoria independiente del método de percepción	-	-	✓

✓: sí se aborda.

-: no se reporta o no constituye una aportación al trabajo.

N/A: No aplica.

Como se aprecia en la Tabla 5.4, los trabajos [16] y [18] emplean sistemas de visión global para obtener información espacial del robot móvil y realizar tareas de guiado. Ambos trabajos requieren transformaciones proyectivas para convertir las coordenadas de imagen en coordenadas espaciales. En contraste, la propuesta utiliza una red neuronal de función de base radial para establecer un mapeo directo de coordenadas a partir de datos experimentales, al considerar el sistema de visión como un sensor externo.

Adicionalmente, una diferencia radica en la caracterización de la incertidumbre asociada al proceso de estimación. Se considera una distribución del error en función de la posición del robot en el espacio de trabajo y no un único valor.

La diferencia principal respecto de los trabajos revisados no radica en el empleo de la visión global para el guiado de robots móviles, sino en el desarrollo de un modelo de percepción basado en el aprendizaje para la estimación espacial de la posición y en la caracterización de su incertidumbre.

5.5. Control de seguimiento de trayectoria con rastreo visual

En esta sección se presenta una extensión de los resultados que muestra la implementación de un controlador de seguimiento de trayectoria, en el que el seguimiento visual está presente en todo momento para asegurar que el robot móvil pueda moverse a lo largo de una trayectoria deseada. Las pruebas experimentales se realizan únicamente mediante el seguimiento visual propuesto para mostrar el alcance del sistema.

La primera prueba experimental consiste en una circunferencia. La trayectoria se define mediante la ecuación (5.2).

$$\begin{aligned}\omega &= \frac{2\pi}{T} \\ x_{ref} &= x_o + r \cos(\omega t) \\ y_{ref} &= y_o + r \sin(\omega t) \\ \alpha_{ref} &= \arctan 2(\dot{y}_{ref}, \dot{x}_{ref})\end{aligned}\tag{5.2}$$

donde x_{ref} y y_{ref} son las posiciones de la trayectoria de referencia en los ejes x y y , respectivamente; x_o y y_o son las coordenadas del centro de la circunferencia; r es el radio de la circunferencia; ω es la frecuencia angular; y T corresponde al periodo.

Al considerar la ley de control (4.107), con $K_x = 1.2$, $K_y = 1$ y $K_\alpha = 1$. Además, los parámetros de la trayectoria son $x_o = 0.7$ m, $y_o = 0.5$ m, $r = 0.2$ m, $T = 15$ s. La prueba experimental se ilustra en la Figura 5.23.

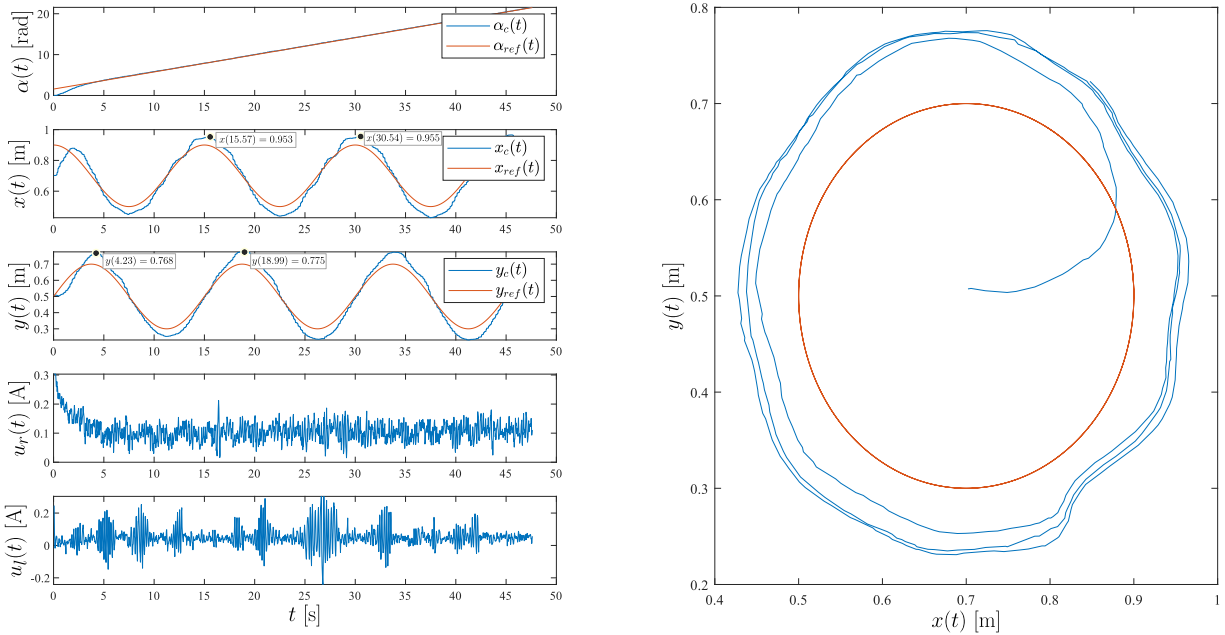


Figura 5.23: Gráficas de la prueba experimental del controlador de seguimiento de trayectoria circular, utilizando la estimación de posición basada en aprendizaje.

La Figura 5.23 muestra, a la izquierda, las gráficas de los estados del robot móvil y de los

esfuerzos de control, y, a la derecha, el recorrido planar del robot. En todos los gráficos, las señales de referencia son de color naranja y las señales reales obtenidas experimentalmente, de color azul.

La gráfica $\alpha(t)$ muestra una rápida convergencia de la orientación del robot hacia la orientación de referencia en aproximadamente 3 segundos. En cuanto a la gráfica de $x(t)$, la señal del robot imita la forma, el periodo y la frecuencia de la señal de referencia y converge en aproximadamente 5 segundos; sin embargo, la amplitud es ligeramente mayor que 0.9 m. Según los *Datatip*, la amplitud es de ≈ 0.95 m, lo que representa un error de 0.05 m. Por otro lado, la gráfica $y(t)$ presenta un comportamiento similar al de la anterior; la diferencia radica en que la amplitud es aún mayor. Los *Datatip* mencionan ≈ 0.77 m, es decir, un error de 0.07 m.

La gráfica del movimiento en el plano muestra el movimiento real del robot en el espacio; el robot sigue la trayectoria circular. Existe un error evidente que se observa en las gráficas individuales. El robot converge hacia la trayectoria, lo que, hasta este punto, parece un desempeño adecuado en el seguimiento visual.

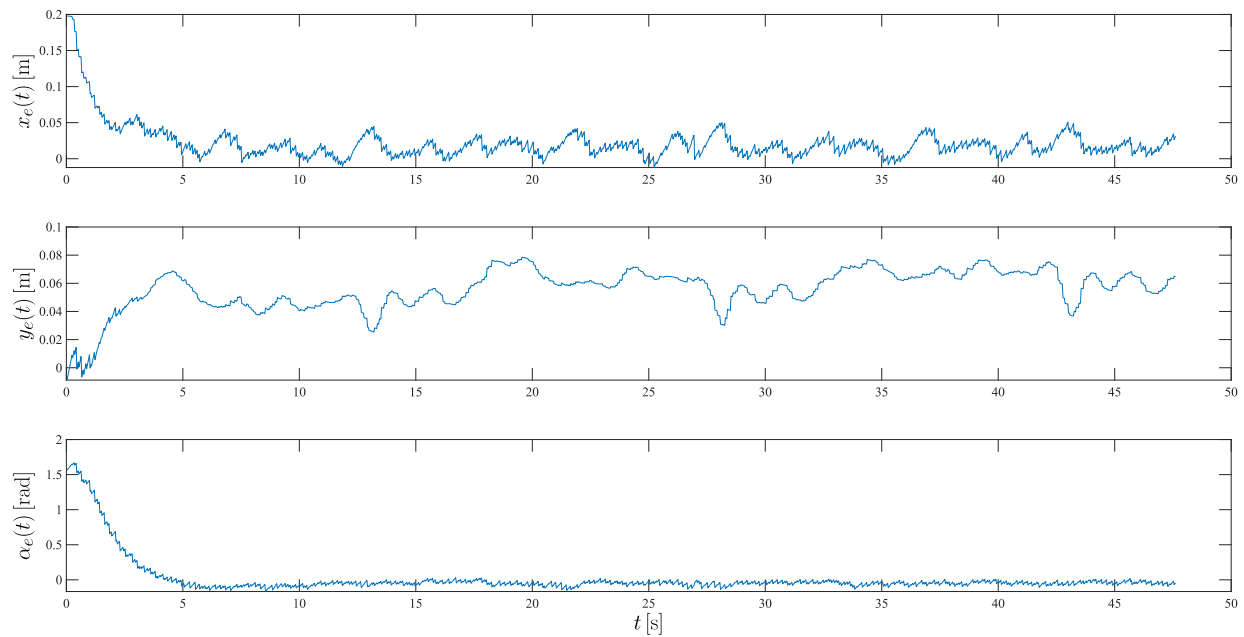


Figura 5.24: Gráficas de error de la prueba experimental del controlador de seguimiento de trayectoria circular.

La Figura 5.24 presenta tres gráficos de error: de posición en el eje x , de posición en el eje y y de orientación. La señal x_e ilustra la convergencia del error, con oscilaciones pequeñas alrededor de cero. Se aprecia un error acotado, lo que indica estabilidad.

La Tabla 5.5 presenta las métricas de error obtenidas durante el seguimiento de la trayectoria circular.

Tabla 5.5: Métricas de error de la prueba experimental del seguimiento de trayectoria circular.

Métrica	x_e [m]	y_e [m]	α_e [rad]	e_{euc} [m]
MAE	0.0218	0.0559	0.1124	0.0638
RMSE	0.0325	0.0577	0.2781	0.0662
Máximo	0.1978	0.0785	1.6690	0.1979
ME	0.0214	0.0558	0.0164	-

Con respecto de la Tabla 5.5, MAE es el error absoluto medio, Máximo es el error máximo obtenido, ME es el error medio con signo y $e_{euc} = \sqrt{x_e^2 + y_e^2}$ es el error euclidiano de posición. Se observa que el error absoluto medio en el eje y es considerablemente mayor que en el eje x ; sin embargo, el RMSE en el eje y sugiere un comportamiento relativo uniforme. En contraste, en el eje x se observa una diferencia considerable entre MAE y RMSE, con un error máximo elevado. Este comportamiento también se observa en el error de orientación; sin embargo, al analizar los datos de la prueba experimental, estos valores corresponden al instante inicial ($t = 0.03$ s), en el que el robot móvil parte de la condición inicial, lo que permite concluir que estos valores máximos no son concluyentes respecto del desempeño del controlador de seguimiento de trayectoria.

La segunda prueba experimental consiste en la trayectoria lemniscata de Bernoulli. Esta trayectoria se define mediante la ecuación (5.3).

$$\begin{aligned}
 \omega &= \frac{2\pi}{T} \\
 x_{ref} &= 0.8 + A \sin(\omega t) \\
 y_{ref} &= 0.7 + B \sin(2\omega t) \\
 \alpha_{ref} &= \arctan 2(\dot{y}_{ref}, \dot{x}_{ref})
 \end{aligned} \tag{5.3}$$

Siendo los parámetros de la trayectoria $A = 0.2$, $B = 0.2$ y $T = 25$. La prueba experimental se ilustra en la Figura 5.25.

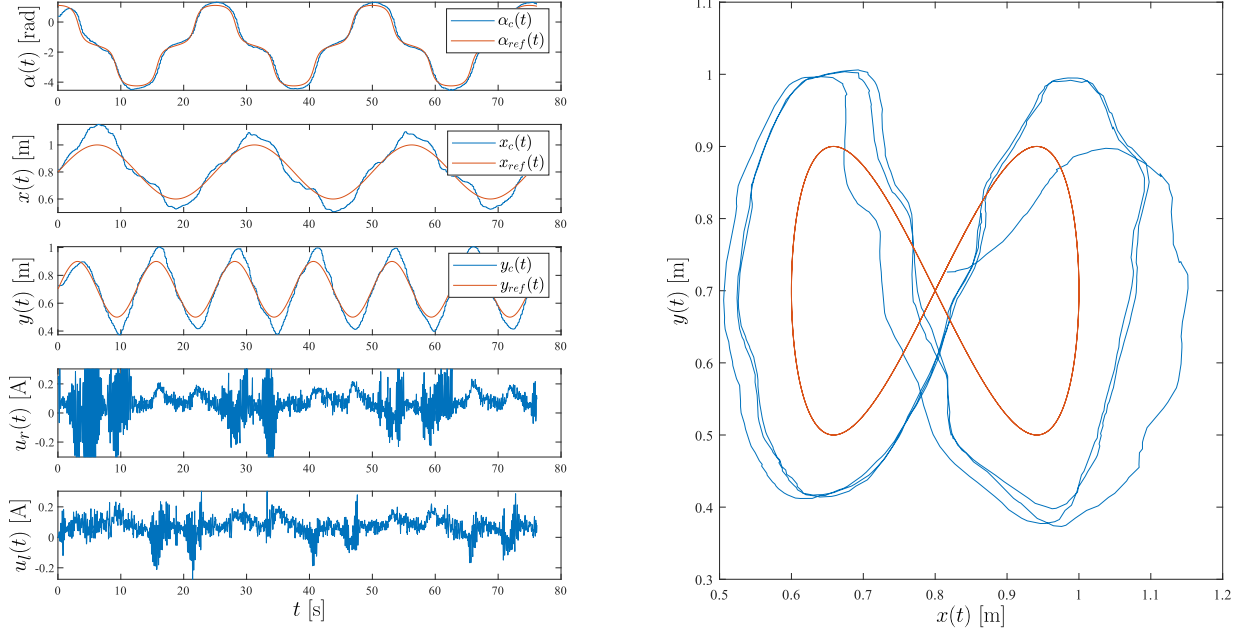


Figura 5.25: Gráficas de la prueba experimental del controlador de seguimiento de trayectoria lemniscata de Bernoulli.

Con respecto de la Figura 5.25, la gráfica $\alpha(t)$ muestra la convergencia de la orientación del robot α_c hacia la orientación de referencia α_{ref} ; ambas coinciden en cuanto a forma, periodo y frecuencia; sin embargo, hay instantes en los que el robot pierde ligeramente la forma. La gráfica de $x(t)$ también muestra un comportamiento convergente; en este caso, la señal del robot presenta una forma distinta, atribuible principalmente a la complejidad de la trayectoria. La gráfica $y(t)$ vuelve a coincidir en forma, periodo y frecuencia; sin embargo, la amplitud es mayor que la de la referencia.

Por último, en la gráfica del movimiento planar del robot se observa que este es capaz de seguir la trayectoria deseada. Se observa un error que se aprecia con mayor detalle en las gráficas individuales, pero la convergencia del robot hacia la trayectoria se traduce en un desempeño idóneo del controlador.

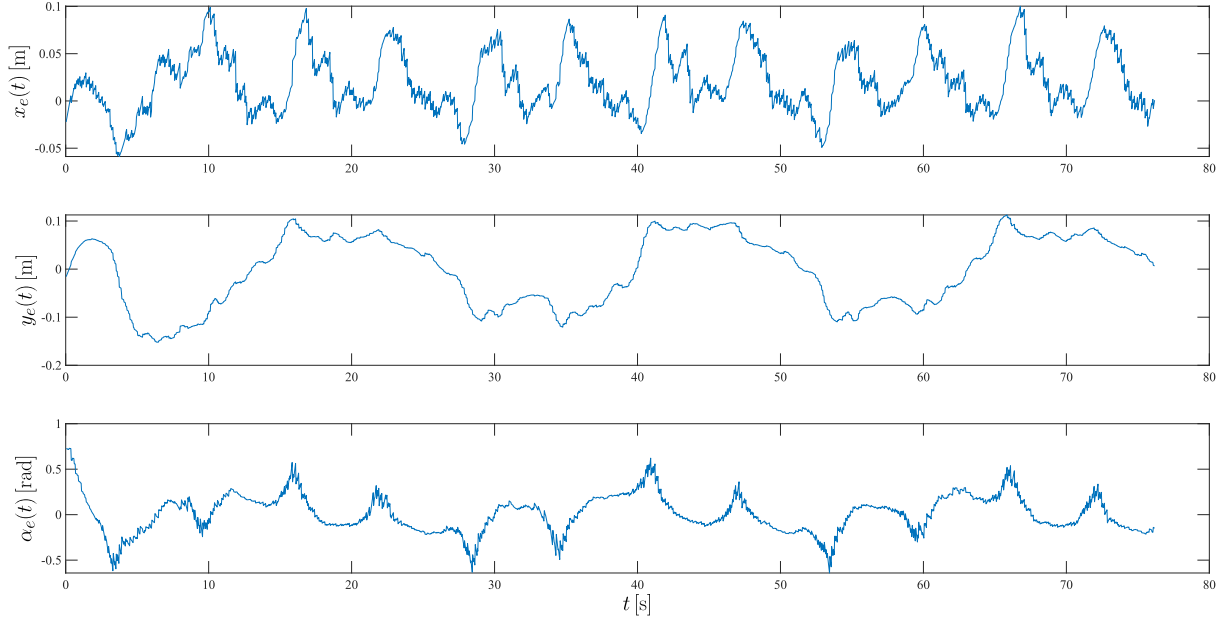


Figura 5.26: Gráficas de error de la prueba experimental del controlador de seguimiento de la trayectoria lemniscata de Bernoulli.

La Figura 5.26 muestra las gráficas de error. La señal x_e ilustra un error oscilante pero acotado y, como se mencionó anteriormente, dicha oscilación puede deberse a la complejidad de la trayectoria. La señal y_e presenta un error suave, oscilante y acotado. Por último, la señal α_e representa el error de orientación, que, nuevamente, está acotado.

Los errores de las pruebas experimentales son acotados y pueden atribuirse a diversos factores, desde desalineaciones mecánicas hasta incertidumbre paramétrica, dado que se trata de un controlador basado únicamente en el modelo cinemático. Sin embargo, dado que en todos los resultados se observa un comportamiento acotado, se garantiza un funcionamiento estable.

La Tabla 5.6 muestra las métricas de error obtenidas durante la prueba experimental de seguimiento de la trayectoria lemniscata de Bernoulli.

Tabla 5.6: Métricas de error de la prueba experimental del seguimiento de la trayectoria lemniscata de Bernoulli.

Métrica	x_e [m]	y_e [m]	α_e [rad]	e_{euc} [m]
MAE	0.0284	0.0654	0.1666	0.0743
RMSE	0.0375	0.0729	0.2066	0.0820
Máximo	0.1000	0.1521	0.7293	0.1583
ME	0.0198	-0.0015	0.0098	-

La Tabla 5.6 muestra nuevamente una diferencia considerable en el valor del error absoluto medio entre los ejes y y x . El RMSE en ambos ejes y en la orientación es mayor que el MAE e indica la presencia de desviaciones de mayor magnitud durante el seguimiento. En esta prueba experimental, los errores máximos no se presentan al inicio, sino en un instante aproximado a

$t = 6.5$ s; sin embargo, este comportamiento puede atribuirse a una mayor complejidad geométrica de la trayectoria, dada por la curvatura variable, en contraste con la trayectoria circular. Si bien la trayectoria de lemniscata presenta errores de seguimiento superiores a los obtenidos con la circunferencia, ambas pruebas experimentales demuestran la factibilidad de emplear la información espacial proporcionada por el sistema de visión basado en el aprendizaje en tareas de control de seguimiento de trayectoria.

Conclusiones

6.1. Conclusión

A lo largo de este trabajo se diseña, implementa y valida un sistema de percepción visual basado en una red neuronal de función de base radial, aplicado a un controlador de posición y de seguimiento de trayectoria de un robot móvil con ruedas de configuración diferencial. Emulando el papel del sistema sensoriomotor humano.

Los resultados muestran que la red RBF, entrenada con un conjunto de aproximadamente 20000 correspondencias píxel-metro, alcanza una precisión de 0.027 m, con un error máximo de 0.059 m localizado en la periferia del espacio visual, donde la densidad de datos de entrenamiento es menor. Este resultado confirma que el desempeño del sistema de percepción visual está directamente relacionado con la cobertura del conjunto de datos de entrenamiento y sugiere que uno de los trabajos futuros es aumentar la densidad muestral en las proximidades de los bordes del área de trabajo del robot.

La comparación del método propuesto con el método basado en marcadores ArUco permite resaltar su desempeño y aporta evidencia sobre las ventajas de evitar la estimación explícita de los parámetros extrínsecos e intrínsecos de la cámara (calibración explícita). Además de simplificar la implementación práctica, reducir el ruido de medición y mejorar la confiabilidad de la estimación de la posición.

El error del sensor visual no sigue una distribución uniforme ni simétrica en el espacio de trabajo, por lo que se emplea un remuestreo *bootstrap* para modelarlo; de este modo se preservan la correlación entre los ejes x y y y la asimetría observada experimentalmente. Esta estructura de incertidumbre permite construir un sensor visual realista para la etapa de simulación, cuyo comportamiento es coherente con lo observado posteriormente en las pruebas experimentales.

A partir de los resultados experimentales obtenidos durante la implementación del controlador de posición para alcanzar la meta, se considera cumplida la hipótesis planteada. El error de traslación durante la operación del robot se mantiene por debajo de 0.05 m, cumpliendo así el límite establecido. Aun cuando en la caracterización del método de localización se obtiene un error máximo de 0.059 m en determinadas regiones del espacio visual, este valor corresponde al error de estimación de la posición y no al error de posición del vehículo en lazo cerrado.

Finalmente, la integración del sensor visual basado en RBF dentro del lazo cerrado de control de posición (y también el control de seguimiento de trayectoria), permite al robot móvil alcanzar la posición deseada de manera satisfactoria, validando experimentalmente el planteamiento conceptual

referente al sistema sensoriomotor, es decir: que un sistema de percepción visual basado en el entrenamiento de una red neuronal de función de base radial, cumple con una tarea análoga al de la conciencia espacial dentro del sistema sensoriomotor biológico, sirviendo como referencia confiable para la planificación y ejecución de movimiento. Los resultados obtenidos respaldan la viabilidad de esta analogía como marco de diseño para sistemas de percepción y control en la robótica móvil.

6.2. Impactos

A continuación, se detallan y exploran los impactos más significativos que el proyecto genera, ofreciendo una perspectiva integral de sus efectos y consecuencias.

6.2.1. Impacto científico

Se desarrolla una metodología de autoajuste de la localización para dispositivos UGV empleando inteligencia artificial basada en una característica humana: el sistema sensoriomotor. Y así disminuir el error máximo absoluto de traslación, además de evitar la estimación explícita de los parámetros extrínsecos e intrínsecos y de la homografía, a cambio de un conjunto de entrenamiento específico.

6.2.2. Impacto tecnológico

La metodología se centra en un sistema de estimación de coordenadas para la portabilidad y migración del algoritmo a otros tipos de UGV, lo que abre la posibilidad de escalar esta metodología a diferentes sistemas que requieran la localización adecuada y precisa en el espacio, lo que aporta robustez a los UGV requeridos para la realización de tareas diversas.

6.2.3. Impacto social

La metodología propuesta sugiere incrementar el uso de los dispositivos UGV en aplicaciones de los sectores civil, agrícola y militar, entre otros, sujeto al escalamiento y a la validación de la plataforma.

6.3. Productos logrados

Las respectivas referencias de los productos logrados son:

- Artículos científicos
 - Congreso internacional: [53]
 - Revista local: [54]

6.4. Trabajo a futuro

Más allá de los resultados obtenidos, algunas posibles direcciones del trabajo futuro son:

- Realizar pruebas experimentales considerando un espacio de trabajo más amplio y una periferia mayor, lo que permita reevaluar las métricas de error.

- Probar la metodología en una plataforma de mayor consumo para impulsar la mejora de la electrónica.
- Extender el alcance a entornos dinámicos en los que el sistema de visión pueda estar en movimiento, añadiendo sensores externos que permitan reforzar la estimación de la posición del robot móvil.
- Realizar pruebas que involucren casos de oclusión, cambios de iluminación, diferentes alturas y rotaciones de la cámara, vibración, distintos tamaños de imagen, entre otros. Con la intención de ampliar el alcance del presente trabajo a diferentes condiciones.

Bibliografía

- [1] C. Okonkwo and I. Awolusi, “Environmental sensing in autonomous construction robots: Applicable technologies and systems,” *Automation in Construction*, vol. 172, p. 106075, 2025.
- [2] D. Zhang, Y. Jiang, J. Liu, G. Xiong, and S. Zou, “Design, protection, perception and planning of mobile robots in nuclear power plants,” *Progress in Nuclear Energy*, vol. 186, p. 105821, 2025.
- [3] P.-C. Shih, Y.-C. Chiang, J.-J. Hu, K.-T. Lin, and L.-C. Fu, “Autonomous exploration of mobile robot equipped with lidar for high-quality reconstruction in large-scale indoor environments,” *Mechatronics*, vol. 112, p. 103409, 2025.
- [4] D. Li, D. Tang, C. Liu, Z. Zhang, and L. Wang, “Research on dynamic obstacle avoidance for industrial agvs using decay model-based multi-objective q-learning,” *Knowledge-Based Systems*, vol. 333, p. 115000, 2026.
- [5] F. Driewer, M. Sauer, and K. Schilling, “Mixed reality for teleoperation of mobile robots in search and rescue scenarios,” *IFAC Proceedings Volumes*, vol. 39, no. 3, pp. 267–272, 2006. 12th IFAC Symposium on Information Control Problems in Manufacturing.
- [6] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer Handbooks, Springer Cham, 2016.
- [7] R. Siegwart, I. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots, second edition*. Intelligent Robotics and Autonomous Agents series, MIT Press, 2011.
- [8] W. Lei, C. Zhang, Z. Jin, and J. Chen, “A review on magnetic-assisted localization for mobile robots,” *Measurement*, vol. 246, p. 116643, 2025.
- [9] G. Conte and P. Doherty, “An integrated uav navigation system based on aerial image matching,” in *2008 IEEE Aerospace Conference*, pp. 1–10, 2008.
- [10] A. S. Silva Oliveira, M. C. dos Reis, F. A. X. da Mota, M. E. M. Martinez, and A. R. Alexandria, “New trends on computer vision applied to mobile robot localization,” *Internet of Things and Cyber-Physical Systems*, vol. 2, pp. 63–69, 2022.
- [11] X. Liang, H. Wang, Y.-H. Liu, Z. Liu, B. You, Z. Jing, and W. Chen, “Purely image-based pose stabilization of nonholonomic mobile robots with a truly uncalibrated overhead camera,” *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 724–742, 2020.

- [12] S. Mehtatt, W. Dixon, D. Mac Arthur, and C. Crane, “Visual servo control of an unmanned ground vehicle via a moving airborne monocular camera,” in *2006 American Control Conference*, pp. 6 pp.–, 2006.
- [13] A. Zdešar, I. Škrjanc, and G. Klančar, “Visual trajectory-tracking model-based control for mobile robots,” *International Journal of Advanced Robotic Systems*, vol. 10, no. 9, p. 323, 2013.
- [14] E. Enikov and J. Escareno, “Application of sensory body schemas to path planning for micro air vehicles (mavs),” vol. 1, pp. 25–31, 01 2015.
- [15] M. Dirik, A. F. Kocamaz, and E. Dönmez, “Visual servoing based control methods for nonholonomic mobile robot,” *Journal of Engineering Research*, vol. 8, no. 2, pp. 95–113, 2020.
- [16] Q. Wang, J. Liang, and Z. Rao, “Research and design of trackless agv system based on global vision,” in *2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pp. 387–391, 2020.
- [17] J. Franco-Robles, J. Escareno, D. Soto-Guerrero, and O. Labbani-Igbida, “Feedforward formation control based on self-organized body-schema,” in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1299–1307, 2021.
- [18] Z. Zheng and Y. Lu, “Research on agv trackless guidance technology based on the global vision,” *Science progress*, vol. 105, p. 368504221103766, 07 2022.
- [19] J. Franco-Robles, J. Escareno, and O. Labbani-Igbida, “Two-level sensorimotor learning for leader-follower consensus control,” in *2023 European Control Conference (ECC)*, pp. 1–8, 2023.
- [20] J. Franco-Robles, J. Escareno, K. A. Camarillo-Gómez, G. I. Pérez-Soto, and O. Labbani-Igbida, “Bio-inspired sensorimotor control based on multimodal sensor fusion for mas optimal consensus,” *Aerospace Science and Technology*, vol. 168, p. 110741, 2026.
- [21] J. Ni, J. Hu, and C. Xiang, “A review for design and dynamics control of unmanned ground vehicle,” *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 235, p. 095440702091209, 04 2020.
- [22] Q. Liu, S. Yuan, and Z. Li, “A survey on sensor technologies for unmanned ground vehicles,” 2020.
- [23] C. Ersü, E. Petlenkov, and K. Janson, “A systematic review of cutting-edge radar technologies: Applications for unmanned ground vehicles (ugvs),” *Sensors*, vol. 24, no. 23, 2024.
- [24] J. E. G. Lopera, J. A. T. Rangel, Á. J. S. Vergel, O. López-Bustamante, and D. G. Ibarra, “Vehículos terrestres no tripulados, sus aplicaciones y tecnologías de implementación,” *ingeniare*, no. 30, pp. 47–71, 2021.
- [25] S. Stasinopoulos, M. Zhao, and Y. Zhong, “Simultaneous localization and mapping for autonomous bicycles,” *International Journal of Advanced Robotic Systems*, vol. 14, p. 172988141770717, 05 2017.

- [26] ROS.org, “Pioneer 3-dx.” <https://robots.ros.org/pioneer-3-dx/>, 2025. Accessed: 2025-06-16.
- [27] Carnegie Mellon University, “Palm pilot robot kit (pprk).” <https://www.cs.cmu.edu/~pprk/>, 2000. Accessed: 2025-06-16.
- [28] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, “Stanley: The robot that won the darpa grand challenge,” *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
- [29] J. E. Mendoza, *Body Schema*, pp. 427–428. New York, NY: Springer New York, 2011.
- [30] H. HEAD and G. HOLMES, “SENSORY DISTURBANCES FROM CEREBRAL LESIONS¹,” *Brain*, vol. 34, pp. 102–254, 11 1911.
- [31] M. Hübsch, “Understanding the concepts peripersonal space, body schema and body image,” 2012.
- [32] S. Ulbrich, *Sensorimotor Learning for an Artificial Body Schema on Humanoid Robots*. PhD thesis, 2014.
- [33] D. Wolpert, J. Diedrichsen, and J. Flanagan, “Principles of sensorimotor learning,” *Nature reviews. Neuroscience*, vol. 12, pp. 739–51, 12 2011.
- [34] M. Hoffmann, H. Marques, A. Arieta, H. Sumioka, M. Lungarella, and R. Pfeifer, “Body schema in robotics: A review,” *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 4, pp. 304–324, 2010.
- [35] O. Sigaud, C. Salauin, and V. Padois, “On-line regression algorithms for learning mechanical models of robots: A survey,” *Robotics and Autonomous Systems*, vol. 59, no. 12, pp. 1115–1129, 2011.
- [36] L. Rouhiainen, “Inteligencia artificial,” *Madrid: Alienta Editorial*, pp. 20–21, 2018.
- [37] M. J. Orr *et al.*, “Introduction to radial basis function networks,” 1996.
- [38] D. Broomhead and D. Lowe, “Radial basis functions, multi-variable functional interpolation and adaptive networks,” *ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (UNITED KINGDOM)*, vol. RSRE-MEMO-4148, 03 1988.
- [39] J. Park and I. W. Sandberg, “Universal approximation using radial-basis-function networks,” *Neural Computation*, vol. 3, no. 2, pp. 246–257, 1991.
- [40] “An improved fast training algorithm for rbf networks using symmetry-based fuzzy c-means clustering,” *MATEMATIKA*, vol. 24, p. 141–148, Dec. 2008.
- [41] C. Dash, A. K. Behera, S. Dehuri, and S.-B. Cho, “Radial basis function neural networks: A topical state-of-the-art survey,” *Open Computer Science*, vol. 6, 05 2016.
- [42] R. e Manual, “Turtlebot3,” 2025. Accessed: 2025-01-16.

- [43] Y. Li and Y. Liu, *Dynamics and Control for Nonholonomic Mobile Modular Manipulators*. 12 2006.
- [44] B. C. Kuo, *Sistemas de control automático*. Pearson Educación, 1996.
- [45] Q. Que and M. Belkin, “Back to the future: Radial basis function network revisited,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 8, pp. 1856–1867, 2020.
- [46] OpenCV Documentation, *ArUco Marker Detection*. Open Source Computer Vision Library (OpenCV), 2018. Accessed: 2026-02-18.
- [47] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge books online, Cambridge University Press, 2003.
- [48] E. Marchand, H. Uchiyama, and F. Spindler, “Pose estimation for augmented reality: A hands-on survey,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, 01 2016.
- [49] A. Hemami, M. G. Mehrabi, and R. M. H. Cheng, “Optimal kinematic path tracking control of mobile robots with front steering,” *Robotica*, vol. 12, no. 6, p. 563–568, 1994.
- [50] A. WICKENS, “Dynamics of actively guided vehicles,” *Vehicle System Dynamics*, vol. 20, no. 3-4, pp. 219–242, 1991.
- [51] M. R. Spiegel and O. G. Ribero, *Manual de fórmulas y tablas matemáticas*. McGraw-Hill Madrid, 1970.
- [52] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, “A stable tracking control method for a non-holonomic mobile robot,” in *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, pp. 1236–1241 vol.3, 1991.
- [53] A. I. de la Fuente-Sánchez, G. I. Pérez Soto, and J. Franco-Robles, “Learning-based visual tracking for mobile robot localization using radial basis function networks,” in *12th Scientific Conference ENTREPRENEURSHIP-ENGINEERING-MANAGEMENT*, (Zrenjanin, Serbia), p. ..., ..., May 2026.
- [54] A. I. de la Fuente-Sánchez, G. I. Pérez Soto, and J. Franco-Robles, “Tendencias en control aplicado a robots móviles mediante visión externa,” *Perspectivas de la Ciencia y la Tecnología*.

Código de adquisición de datos del espacio de trabajo

El anexo A presenta el código de programación para la adquisición de datos del espacio de trabajo, que, al final, almacena la información en un archivo de texto.

```
1 import cv2
2 import numpy as np
3
4 cap = cv2.VideoCapture(1)
5
6 nx = 11
7 ny = 8
8 l = 0.10
9 subdivisions = 4
10
11 dataset = []
12
13 ORIGIN_PX = np.array([87.0, 456.0])
14
15 while True:
16     ret, frame = cap.read()
17     if not ret:
18         break
19
20     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
21     #Funcion para detectar bordes en el patron de referencia
22     retval, corners = cv2.findChessboardCorners(gray, (nx, ny), None)
23
24     if retval:
25         criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
26         corners = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)
27         corners_grid = corners.reshape(ny, nx, 2)
28
29         #Espejo de grilla de posiciones en caso de ser necesario
30         corners_grid = corners_grid[::-1, :]      # espejo vertical
31
```

```

32
33     dist_x_total = 0.0
34     count_x = 0
35     for r in range(ny):
36         d = np.linalg.norm(corners_grid[r, -1] - corners_grid[r, 0])
37         dist_x_total += d
38         count_x += 1
39     s_x = ((nx - 1) * 1) / (dist_x_total / ny) #factor de escala en x
40
41     dist_y_total = 0.0
42     count_y = 0
43     for c in range(nx):
44         d = np.linalg.norm(corners_grid[-1, c] - corners_grid[0, c])
45         dist_y_total += d
46         count_y += 1
47     s_y = ((ny - 1) * 1) / (dist_y_total / nx) #factor de escala en y
48
49     corner00_px = corners_grid[0, 0] #origen detectado por software
50     delta_px = corner00_px - ORIGIN_PX
51
52     offset_X = delta_px[0] * s_x #compensacion de distancia en x
53     offset_Y = -delta_px[1] * s_y #compensacion de distancia en y
54
55     dataset_frame = []
56
57     for r in range(ny - 1):
58         for c in range(nx - 1):
59
60             P00 = corners_grid[r,      c      ]
61             P10 = corners_grid[r,      c + 1]
62             P01 = corners_grid[r + 1, c      ]
63             P11 = corners_grid[r + 1, c + 1]
64
65             W00 = np.array([c          * 1,  r          * 1])
66             W10 = np.array([(c + 1) * 1,  r          * 1])
67             W01 = np.array([c          * 1,  (r + 1) * 1])
68             W11 = np.array([(c + 1) * 1,  (r + 1) * 1])
69
70             for i in range(subdivisions + 1):
71                 for j in range(subdivisions + 1):
72
73                     u = i / subdivisions
74                     v = j / subdivisions
75
76                     P = (1-u)*(1-v)*P00 + u*(1-v)*P10 + \
77                        (1-u)*v      *P01 + u*v      *P11
78
79                     W = (1-u)*(1-v)*W00 + u*(1-v)*W10 + \
80                        (1-u)*v      *W01 + u*v      *W11
81
82                     px, py = float(P[0]), float(P[1])

```

```

83
84         X = -(float(W[0]) - offset_X)
85         Y = -(float(W[1]) - offset_Y)
86
87         dataset_frame.append([px, py, X, Y])
88         cv2.circle(frame, (int(px), int(py)), 2, (0, 0, 255), -1)
89
90         is_corner = (i % subdivisions == 0) and (j % subdivisions ==
0)
91         if is_corner:
92             cv2.putText(frame, f"X:{X:.2f}",
93                         (int(px) + 4, int(py) + 12),
94                         cv2.FONT_HERSHEY_SIMPLEX,
95                         0.28, (255, 255, 0), 1)
96             cv2.putText(frame, f"Y:{Y:.2f}",
97                         (int(px) + 4, int(py) + 24),
98                         cv2.FONT_HERSHEY_SIMPLEX,
99                         0.28, (255, 255, 0), 1)
100
101         # Ubicación del Origen físico
102         cv2.circle(frame, (87, 456), 6, (0, 255, 0), -1)
103         cv2.putText(frame, "X:0.00", (94, 465),
104                     cv2.FONT_HERSHEY_SIMPLEX, 0.32, (0, 255, 0), 1)
105         cv2.putText(frame, "Y:0.00", (94, 465),
106                     cv2.FONT_HERSHEY_SIMPLEX, 0.32, (0, 255, 0), 1)
107
108         cv2.drawChessboardCorners(frame, (nx, ny), corners, retval)
109         cv2.putText(frame,
110                     f"Frame: {len(dataset_frame)} pts | Total: {len(dataset)} pts",
111                     (20, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)
112
113         cv2.imshow("frame", frame)
114         key = cv2.waitKey(1) & 0xFF
115
116         #Instruccion que guarda los datos observados en la imagen
117         if key == ord('s') and retval:
118             dataset.extend(dataset_frame)
119             print(f"Frame guardado. Total acumulado: {len(dataset)} puntos")
120
121         #Guardado del dataset
122         if key == ord('w') and len(dataset) > 0:
123             np.savetxt("dataset_pixel_metros.txt",
124                       np.array(dataset),
125                       header="pixel_x pixel_y X_real Y_real")
126             print(f"Dataset guardado con {len(dataset)} puntos")
127
128         if key == 27:
129             break
130
131     cap.release()
132     cv2.destroyAllWindows()

```

Código de simulación

El anexo B muestra el código de programación para la simulación de los movimientos del robot móvil y la estimación de la posición de un sensor visual virtual. Al final se obtienen las gráficas referentes a la trayectoria seguida por el robot y a la trayectoria reportada por la cámara virtual.

```
1 from math import cos, sin, atan2, pi, sqrt
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from scipy.integrate import solve_ivp
5
6 #Parámetros del robot
7 m = 0.680
8 R = 0.035
9 Iw = 0.0001
10 Ia = 0.001
11 w = 0.0875
12 mg = 0.034
13 d = 0.015
14
15 #Condiciones iniciales
16 theta_d0 = 0.0
17 theta_i0 = 0.0
18 thetadot_d0 = 0.0
19 thetadot_i0 = 0.0
20 xi0 = 0.0
21 yi0 = 0.0
22 alpha0 = pi
23 ci = np.array([theta_d0, theta_i0, thetadot_d0, thetadot_i0, xi0, yi0, alpha0, 0.0,
24               0.0])
25
26 #Ganancias
27 k_rho, k_alpha = 0.8, 5.0
28 kp = np.diag([0.8, 0.8])
29 ki = np.diag([0.2, 0.2])
30 kv = np.diag([0.02, 0.02])
31
32 xi_des, yi_des = 1.0, 1.0
```

```

32
33 data = np.load("modelo_error_rbf.npz")
34 error_samples = data["error"]          # (4100, 2) errores reales de validación
35
36 rng = np.random.default_rng(42)
37
38 def sample_camera_noise():
39     idx = rng.integers(0, error_samples.shape[0])
40     return error_samples[idx]
41
42 #Simulación del sensor visual virtual (posición estimada por el RBF).
43 def camera_measure(x_m, y_m):
44     ex, ey = sample_camera_noise()
45     return x_m + ex, y_m + ey
46
47
48 def ed(t, x, xi_meas, yi_meas):
49     theta_d, theta_i = x[0], x[1]
50     thetadot_d, thetadot_i = x[2], x[3]
51     xi, yi, alpha = x[4], x[5], x[6]
52     integral = x[7:9]
53
54     exi = xi_des - xi_meas
55     eyi = yi_des - yi_meas
56     rho = sqrt(exi**2 + eyi**2)
57     angle2goal = atan2(eyi, exi)
58     ealpha = angle2goal - alpha
59     ealpha = (ealpha + np.pi) % (2*np.pi) - np.pi
60
61     xr_dot = k_rho * rho
62     alpha_dot = k_alpha * ealpha
63
64     thetadot_d_ref = (xr_dot + (w*alpha_dot)) / R
65     thetadot_i_ref = (xr_dot - (w*alpha_dot)) / R
66     thetadot_ref = np.array([thetadot_d_ref, thetadot_i_ref])
67     thetadot = np.array([thetadot_d, thetadot_i])
68
69     #control de velocidad PI de llantas
70     e_omega = thetadot_ref - thetadot
71     tau = kp @ e_omega + integral
72     integral_dot = ki @ e_omega
73
74
75     #Modelo dinámico
76     M11 = (m*R*R/4) + ((Ia*R*R)/(4*w*w)) + Iw
77     M12 = (m*R*R/4) - ((Ia*R*R)/(4*w*w))
78     M = np.array([[M11, M12], [M12, M11]])
79
80     alphas_dot_real = R*(thetadot_d - thetadot_i) / (2*w)
81
82     C12 = mg*d*R*R*alphas_dot_real / (2*w)

```

```

83     C = np.array([[0, C12], [-C12, 0]])
84
85     xr_dot_real = R*(thetadot_d + thetadot_i) / 2
86     xi_dot_real = xr_dot_real * cos(alpha)
87     yi_dot_real = xr_dot_real * sin(alpha)
88
89     q_ddot = np.linalg.solve(M, tau - C @ thetadot)
90
91     return np.hstack([thetadot_d, thetadot_i, q_ddot[0], q_ddot[1], xi_dot_real,
92                       yi_dot_real, alphas_dot_real, integral_dot[0], integral_dot[1]])
93
94 dt_cam = 0.030 #Periodo de muestreo de 30 ms
95 t0, tf = 0.0, 15.0
96 n_frames = int(round((tf - t0) / dt_cam))
97
98 x_current = ci.copy()
99 xi_meas, yi_meas = camera_measure(x_current[4], x_current[5])
100
101 t_all, y_all = [], []
102 meas_hist = [[t0, xi_meas, yi_meas]]
103
104 t = t0
105 for k in range(n_frames):
106     t_next = min(t + dt_cam, tf)
107
108     sol = solve_ivp(lambda tt, xx: ed(tt, xx, xi_meas, yi_meas), (t, t_next),
109                     x_current, method='RK45', t_eval=np.linspace(t, t_next, 5))
110
111     t_all.append(sol.t)
112     y_all.append(sol.y)
113
114     x_current = sol.y[:, -1] #actualización del estado real
115     t = t_next
116
117     xi_meas, yi_meas = camera_measure(x_current[4], x_current[5])
118     meas_hist.append([t, xi_meas, yi_meas])
119
120 #Concatenación de resultados
121 t_full = np.concatenate(t_all)
122 y_full = np.concatenate(y_all, axis=1)
123 theta_d, theta_i = y_full[0], y_full[1]
124 xi, yi, alpha = y_full[4], y_full[5], y_full[6]
125
126 meas_hist = np.array(meas_hist)
127
128 plt.figure(figsize=(10, 5))
129 plt.plot(xi, yi)
130 plt.plot(meas_hist[:, 1], meas_hist[:, 2])
131 plt.grid(True)
132 plt.legend()

```

```
132 plt.axis("equal")
133 plt.show()
134
135 rho_err = np.sqrt((xi_des - xi)**2 + (yi_des - yi)**2)
136 plt.figure(figsize=(10, 5))
137 plt.plot(t_full, rho_err)
138 plt.grid(True)
139 plt.legend()
140 plt.show()
```

Código de microcontrolador Raspberry Pi Pico

El anexo C describe el programa embebido, escrito en C++, del microcontrolador Raspberry Pi Pico. Este código de programación contiene las funciones de detección de información proveniente de los encoders de los motorreductores, así como el envío y la recepción de datos entre el dispositivo y la computadora para la correcta actuación de los motores.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <inttypes.h>
4  #include "pico/stdlib.h"
5  #include "hardware/timer.h"
6  #include "hardware/pwm.h"
7  #include "hardware/pio.h"
8  #include "math.h"
9  #include "quadrature_encoder.pio.h"
10 #include "hardware/uart.h"
11
12 //Parámetros exclusivos de ambos motores
13 #define PPR 2552 //2552 pulsos por revolucion
14 #define VM 12
15
16 //Valores para calculos
17 #define pi 3.141593
18
19 //Periodo de muestreo
20 #define Ts 0.01 //En segundos
21
22 //GPIO
23 #define ENCA_R 14
24 #define ENCB_R 15
25 #define DIR_R 10
26 #define ENCA_L 16
27 #define ENCB_L 17
28 #define DIR_L 9
29 #define LED 21
```

```

30
31 //PWM
32 #define PWM_R 13
33 #define PWM_L 12
34
35 //UART
36 #define UART_TX 4
37 #define UART_RX 5
38
39 //Variables para calculos
40 float esc = 2*pi/PPR; //Escalamiento para convertir las cuentas del encoder en
    radianes
41 float escs = 65535/VM; //Escalamiento de voltaje en bits
42 float p_R = 0.0, p_L = 0.0, p_1_R = 0.0, p_1_L = 0.0, d_R = 0.0, d_L = 0.0, v_R =
    0.0, v_L = 0.0;
43 float pwmf_R = 0.0, pwmf_L = 0.0;
44 uint16_t pwm_cmd_R = 0, pwm_cmd_L = 0;
45 bool new_cmd = false;
46 float time = 0; //Variable encargada de contar el tiempo
47 uint slice_R, channel_R, slice_L, channel_L; //Variable para modificar ciclo de
    trabajo de PWM
48
49 //Uso de PIO para lectura de encoders
50 PIO pio = pio0;
51 uint sm_R, sm_L;
52
53 int32_t count_R, count_L;
54
55 int cont = 0;
56 bool flag = false;
57
58 // Enviar datos binarios en lugar de texto
59 typedef struct {
60     float v_R;
61     float v_L;
62 } sensor_data_t;
63
64 void set_pin(void){
65     //Pines de dirección y led de status del robot
66     gpio_init(DIR_R);
67     gpio_init(DIR_L);
68     gpio_init(LED);
69     gpio_set_dir(DIR_R,GPIO_OUT);
70     gpio_set_dir(DIR_L,GPIO_OUT);
71     gpio_set_dir(LED, GPIO_OUT);
72
73     //Pines para PWM y configuración para frecuencia de 1kHz
74     gpio_set_function(PWM_R, GPIO_FUNC_PWM);
75     slice_R = pwm_gpio_to_slice_num(PWM_R);
76     channel_R = pwm_gpio_to_channel(PWM_R);
77     pwm_set_wrap(slice_R,65535); //Para resolución de 16 bits en PWM

```

```

78     pwm_set_clkdiv(slice_R, 1.907f); //Para una frecuencia de 1000 Hz
79     pwm_set_chan_level(slice_R, channel_R, 0);
80     pwm_set_enabled(slice_R, true);
81
82     gpio_set_function(PWM_L, GPIO_FUNC_PWM);
83     slice_L = pwm_gpio_to_slice_num(PWM_L);
84     channel_L = pwm_gpio_to_channel(PWM_L);
85     pwm_set_wrap(slice_L, 65535); //Para resolución de 16 bits en PWM
86     pwm_set_clkdiv(slice_L, 1.907f); //Para una frecuencia de 1000 Hz
87     pwm_set_chan_level(slice_L, channel_L, 0);
88     pwm_set_enabled(slice_L, true);
89
90     //Pines de PIO para lectura de encoders de ruedas
91     pio_add_program(pio, &quadrature_encoder_program);
92     //Descripción de encoders
93     sm_R = pio_claim_unused_sm(pio, true);
94     quadrature_encoder_program_init(pio, sm_R, ENCA_R, 0);
95     sm_L = pio_claim_unused_sm(pio, true);
96     quadrature_encoder_program_init(pio, sm_L, ENCA_L, 0);
97
98     //Inicialización y pines para UART Bluetooth
99     uart_init(uart1, 115200);
100    gpio_set_function(UART_TX, GPIO_FUNC_UART);
101    gpio_set_function(UART_RX, GPIO_FUNC_UART);
102 }
103
104 //Función para leer UART
105 void read_uart_commands() {
106     static uint8_t buffer[4];
107     static int idx = 0;
108
109     while (uart_is_readable(uart1)) {
110         buffer[idx++] = uart_getc(uart1);
111
112         if (idx == 4) {
113             pwm_cmd_L = buffer[0] | (buffer[1] << 8);
114             pwm_cmd_R = buffer[2] | (buffer[3] << 8);
115             new_cmd = true;
116             idx = 0;
117         }
118     }
119 }
120
121 bool repeating_callback(struct repeating_timer *t) {
122     static uint8_t send_counter = 0;
123
124     //Para control de posición
125     count_R = quadrature_encoder_get_count(pio, sm_R);
126     count_L = quadrature_encoder_get_count(pio, sm_L);
127     p_R = (count_R*esc); //Posicion en radianes
128     p_L = (count_L*esc);

```

```

129     d_R = p_R - p_1_R; //delta en radianes
130     d_L = p_L - p_1_L;
131     p_1_R = p_R;
132     p_1_L = p_L;
133     v_R = d_R / Ts; // Velocidad en radianes por segundo
134     v_L = d_L / Ts;
135
136     send_counter++;
137     if (send_counter >= 3) {
138         sensor_data_t data = {v_R, v_L};
139         uart_write_blocking(uart1, (uint8_t*)&data, sizeof(data));
140         send_counter = 0;
141     }
142     read_uart_commands();
143     if (new_cmd) {
144         pwmf_L = (float)(pwm_cmd_L & 0x7FFF);
145         pwmf_R = (float)(pwm_cmd_R & 0x7FFF);
146
147         // Extraer dirección del bit de signo
148         bool dir_L = (pwm_cmd_L & 0x8000) != 0;
149         bool dir_R = (pwm_cmd_R & 0x8000) != 0;
150
151         // Aplicar dirección
152         gpio_put(DIR_L, dir_L ? 1 : 0);
153         gpio_put(DIR_R, dir_R ? 0 : 1);
154
155         // Aplicar PWM
156         pwm_set_chan_level(slice_L, channel_L, (uint16_t)pwmf_L);
157         pwm_set_chan_level(slice_R, channel_R, (uint16_t)pwmf_R);
158
159         new_cmd = false;
160     } else {}
161     return true;
162 }
163
164
165
166 int main()
167 {
168     stdio_init_all();
169     set_pin();
170
171     // Dar tiempo al Bluetooth para inicializar
172     sleep_ms(2000);
173
174     // Limpiar buffer de cualquier dato residual
175     while (uart_is_readable(uart1)) {
176         uart_getc(uart1);
177     }
178
179     // Indicar que está esperando

```

```

180     uint8_t start_byte = 0;
181     while(start_byte != 'a') {
182         // Parpadear LED mientras espera
183         gpio_put(LED, 1);
184         sleep_ms(200);
185         gpio_put(LED, 0);
186         sleep_ms(200);
187
188         // Revisar si llegó algo
189         if(uart_is_readable(uart1)) {
190             uart_read_blocking(uart1, &start_byte, 1);
191         }
192     }
193
194     struct repeating_timer timer;
195     add_repeating_timer_ms(10, repeating_callback, NULL, &timer);
196     gpio_put(LED, 1);
197
198     while (true) {
199
200         sleep_ms(5);
201     }
202 }

```

Código de procesamiento para el método de aprendizaje de la representación espacial

El anexo D muestra el código de programación que se implementa en la computadora para relacionar la información procedente del microcontrolador y de la cámara cenital con el lazo de control de posición y devolver el esfuerzo de control para la actuación adecuada de los motorreductores del vehículo terrestre. En este caso se emplea el método de aprendizaje de la representación espacial.

```
1 import serial
2 import math
3 import numpy as np
4 import cv2
5 import struct
6 import threading
7 import time as time_module
8
9 puerto = 'COM5' #bluetooth es COM5
10 baudrate = 115200
11 ser = serial.Serial(puerto, baudrate, timeout=0.1)
12
13 print("Puerto abierto, esperando inicialización del Bluetooth...")
14
15 # Limpiar buffers de Python
16 ser.reset_input_buffer()
17 ser.reset_output_buffer()
18 time_module.sleep(0.5)
19 ser.reset_input_buffer() # Limpiar de nuevo
20
21 #almacenamiento de datos para graficas
22 f = open("datos.txt","w")
23
24 ref = 1.570796 #90 grados
25 Ts = 0.03 #Periodo de muestreo de 10 ms
26 tiempo = 0.0
```

```

27 iM = 3.3
28 escs = 32767 / iM #escalamiento a 15 bits para utilizar el ultimo bit como el signo
    32768
29
30 #Parámetros del robot
31 Rr = 0.035
32 w = 0.0875
33
34 flag = 0
35
36 #Variables para el aruco
37 ARUCO_DICT = cv2.aruco.DICT_6X6_250
38 dictionary = cv2.aruco.getPredefinedDictionary(ARUCO_DICT)
39 detector = cv2.aruco.ArucoDetector(dictionary)
40
41 cap = cv2.VideoCapture(1)
42
43 #variables compartidas entre hilos
44 yaw_global = 0.0
45 x_global = 0.0
46 y_global = 0.0
47 yaw_lock = threading.Lock()
48 running = True
49
50 #Carga del entrenamiento neuronal
51 data = np.load("rbf_model.npz")
52 centers = data["centers"]
53 sigma = data["sigma"]
54 W = data["W"]
55 K = centers.shape[0]
56
57 #Funciones de predicción
58 def rbf(x, c, s):
59     return np.exp(-np.linalg.norm(x-c)**2 / (2*s**2))
60 def predict(pixel):
61     phi = np.zeros(K)
62     for j in range(K):
63         phi[j] = rbf(pixel, centers[j], sigma)
64     return phi @ W
65
66 yaw_prev = 0
67 yaw_offset = 0
68
69 #función de detección de marcadores Aruco
70 def detect_markers(image):
71     global yaw_global, yaw_prev, yaw_offset, x_global, y_global
72
73     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
74     corners, ids, _ = detector.detectMarkers(gray)
75
76     if ids is not None:

```

```

77     if len(ids) >= 2:
78         for i in range(len(ids)):
79             if ids[i][0] == 23:
80                 xA, yA = corners[i][0].mean(axis=0)
81             elif ids[i][0] == 50:
82                 xB, yB = corners[i][0].mean(axis=0)
83
84             # Dibujar
85             cv2.circle(image, (int(xA), int(yA)), 5, (0,255,0), -1)
86             cv2.circle(image, (int(xB), int(yB)), 5, (0,255,0), -1)
87             cv2.circle(image, (87, 456), 10, (255,0,0), -1)
88             cv2.line(image,(int(xA),int(yA)),(int(xB),int(yB)),(0,255,0),1)
89             #Obtener el ángulo del robot absoluto
90             vA = np.array([xA,yA])
91             vB = np.array([xB,yB])
92             v = vB - vA
93             yaw = np.arctan2(-v[1],v[0])
94             # Hacer el ángulo continuo
95             if yaw_prev is not None:
96                 diff = yaw - yaw_prev
97                 if diff > np.pi:
98                     yaw_offset -= 2 * np.pi
99                 elif diff < -np.pi:
100                     yaw_offset += 2 * np.pi
101             yaw_prev = yaw
102             yaw_continuous = yaw + yaw_offset
103
104             #predecir posicion
105             pixel = np.array([xA, yA])
106             x, y = predict(pixel)
107             with yaw_lock:
108                 yaw_global = yaw_continuous
109                 x_global = x
110                 y_global = y
111     return image
112
113 #hilo de detección de marcadores
114 def aruco_thread():
115
116     global running
117     frame_count = 0
118
119     while running:
120         ret, frame = cap.read()
121         if not ret:
122             time_module.sleep(0.05)
123             continue
124
125         frame_count += 1
126         if frame_count % 2 == 0: # Procesar solo 1 de cada 2 frames
127             marker_image = detect_markers(frame)

```

```

128         else:
129             marker_image = frame
130
131             cv2.imshow('Markers', marker_image)
132
133             #Cerrar la ventana con 'q'
134             if cv2.waitKey(1) & 0xFF == ord('q'):
135                 running = False
136                 break
137             time_module.sleep(0.033) #Dar tiempo a otros hilos
138
139 # Iniciar hilo de ArUco
140 aruco_thread_obj = threading.Thread(target=aruco_thread, daemon=True)
141 aruco_thread_obj.start()
142
143 cont = 0
144 alpha = omega_R = omega_L = 0.0
145 proporcional = derivativa = integral= 0.0
146
147 #Ganancias de controlador
148 k_rho = 0.8 #ganancia proporcional cinematica
149 k_alpha = 5.0 #ganancia proporcional de orientación del robot
150
151 kp = 0.8 #ganancias para control PI de velocidades
152 ki = 0.2
153
154 x_des = float(input('Ingresa la posición deseada en x: '))
155 y_des = float(input('Ingresa la posición deseada en y: '))
156
157 start = input('Inserte cualquier caracter para iniciar el control')
158 if(start):
159     ser.write(b'a')
160     flag_global = True
161     time_module.sleep(0.2)
162     ser.reset_input_buffer() # Limpiar basura inicial
163
164 alpha_des_prev = 0
165 alpha_des_offset = 0
166 e_alpha_prev = 0
167 e_alpha_offset = 0
168 try:
169     while running:
170         # Esperar a que haya exactamente 8 bytes
171         while ser.in_waiting < 8 and running:
172             time_module.sleep(0.001)
173
174         if not running:
175             break
176
177         if ser.in_waiting >= 8:
178             try:

```

```

179         datos = ser.read(8)
180
181         #obtención de velocidades reales provenientes del microcontrolador
182         omega_R, omega_L = struct.unpack('<ff', datos)
183
184         #ley de control de posicion
185         e_x = x_des - x_global
186         e_y = y_des - y_global
187         rho = np.sqrt(e_x**2 + e_y**2)
188         if rho > 0.05:
189             alpha_des = np.arctan2(e_y,e_x)
190
191         e_alpha = alpha_des - yaw_global
192         e_alpha = (e_alpha + np.pi) % (2*np.pi) - np.pi
193
194         x_dot_r = k_rho * rho
195         alpha_dot = k_alpha * e_alpha
196
197         theta_dot_ref_r = (w*alpha_dot + x_dot_r) / Rr
198         theta_dot_ref_l = (-w*alpha_dot + x_dot_r) / Rr
199         st = " "
200
201
202         if(abs(e_x)< 0.05 and abs(e_y)<0.05): #and abs(e_alpha) < 0.1):
203             u_L = 0
204             u_R = 0
205         else:
206             #control de velocidad PI de llantas
207             e_omega_r = theta_dot_ref_r-omega_R
208             e_omega_l = theta_dot_ref_l-omega_L
209             proporcional_r = kp*e_omega_r
210             proporcional_l = kp*e_omega_l
211             if (integral_r < iM) and (integral_r > -iM):
212                 integral_r = integral_r + ki * Ts * e_omega_r
213             else:
214                 if integral_r >= iM:
215                     integral_r = iM
216                 if integral_r <= -iM:
217                     integral_r = -iM
218
219             if (integral_l < iM) and (integral_l > -iM):
220                 integral_l = integral_l + ki * Ts * e_omega_l
221             else:
222                 if integral_l >= iM:
223                     integral_l = iM
224                 if integral_l <= -iM:
225                     integral_l = -iM
226
227             u_L=proporcional_r+integral_r
228             u_R=proporcional_l+integral_l
229

```

```

230
231     u_R = np.fmax(np.fmin(u_R, iM), -iM)
232     u_L = np.fmax(np.fmin(u_L, iM), -iM)
233
234     curr_R = u_R / 3.3 #es la corriente de consumo de los motores
235     curr_L = u_L / 3.3
236
237     pwmf_R = math.fabs(u_R)*escs
238     pwmf_L = math.fabs(u_L)*escs
239
240     if(u_R < 0):
241         pwmf_R = int(pwmf_R) | 32768
242     if(u_L < 0):
243         pwmf_L = int(pwmf_L) | 32768
244
245     ser.write(struct.pack('<HH', int(pwmf_L), int(pwmf_R)))
246     tiempo = tiempo + Ts
247
248     f.write(f"{tiempo:.4f} {yaw_global:.4f} {x_global:.4f} {y_global:.4f}
{curr_L:.4f} {curr_R:.4f}\n")
249     except struct.error as e:
250         print(f"Error decodificando: {e}")
251         ser.reset_input_buffer() # Limpiar buffer corrupto
252         continue
253 except KeyboardInterrupt:
254     running = False
255
256 finally:
257     ser.close()
258     f.close()
259     cap.release()
260     cv2.destroyAllWindows()

```

Código de procesamiento para el método de marcadores ArUco

El anexo E realiza una tarea similar a la del anexo mostrado anteriormente, pero utiliza el método de seguimiento basado en marcadores ArUco.

```
1 import serial
2 import math
3 import numpy as np
4 import cv2
5 from cv2 import aruco
6 import struct
7 import threading
8 import time as time_module
9
10 puerto = 'COM5' #bluetooth es COM5
11 baudrate = 115200
12 ser = serial.Serial(puerto, baudrate, timeout=0.1)
13
14 print("Puerto abierto, esperando inicialización del Bluetooth...")
15
16 # Limpiar buffers de Python
17 ser.reset_input_buffer()
18 ser.reset_output_buffer()
19 time_module.sleep(0.5)
20 ser.reset_input_buffer() # Limpiar de nuevo
21
22 #almacenamiento de datos para gráficas
23 f = open("datos.txt","w")
24
25 ref = 1.570796 #90 grados
26 Ts = 0.03 #Periodo de muestreo de 10 ms
27 tiempo = 0.0
28 iM = 3.3
29 escs = 32767 / iM #escalamiento a 15 bits para utilizar el ultimo bit como el signo
    32768
30
```

```

31 #Parámetros del robot
32 Rr = 0.035
33 w = 0.0875
34
35 flag = 0
36
37 #Variables para el aruco
38 ARUCO_DICT = cv2.aruco.DICT_6X6_250
39 dictionary = cv2.aruco.getPredefinedDictionary(ARUCO_DICT)
40 #MATRIZ DE PARAMETROS INTRINSECOS Y COEFICIENTES DE DISTORSION, REALSENSE
41 mtx = np.array([[1.39169873e+03, 0.00000000e+00, 9.86262329e+02],
42                 [0.00000000e+00, 1.39233923e+03, 5.51184204e+02],
43                 [0.00000000e+00, 0.00000000e+00, 1.00000000e+00]])
44 dist = np.array([[0.],[ 0.],[ 0.],[ 0.],[ 0.],[ 0.],[ 0.],[ 0.],[ 0.],[ 0.],[ 0.],[
45                 0.],[ 0.],[ 0.]])
46
47 params = cv2.aruco.DetectorParameters()
48 cap = cv2.VideoCapture(1)
49
50 #variables compartidas entre hilos
51 flag_global = False
52 yaw_global = 0.0
53 x_global = 0.0
54 y_global = 0.0
55 yaw_lock = threading.Lock()
56 running = True
57
58 def rotationMatrixtoEulerAngles(R):
59     sy = np.sqrt(R[0,0]**2 + R[1,0]**2)
60     singular = sy < 1e-6
61     if not singular:
62         yaw = np.arctan2(R[1,0], R[0,0])
63     else:
64         yaw = 0
65
66     return -yaw + (np.pi/2)
67
68 def angulo_continuo(ang_prev, ang_act, ang_offset):
69
70     if ang_act is None:
71         return None, ang_prev, ang_offset
72     if ang_prev is None:
73         ang_prev = ang_act
74
75     diff = ang_act - ang_prev
76
77     if diff > np.pi:
78         ang_offset -= 2 * np.pi
79     elif diff < -np.pi:
80         ang_offset += 2 * np.pi

```

```

81
82     ang_prev = ang_act
83     ang_continuous = ang_act + ang_offset
84     return ang_continuous, ang_prev, ang_offset
85
86 yaw_aruco_prev = 0
87 yaw_aruco_offset = 0
88 def detect_markers(image):
89     global yaw_global, flag_global, yaw_aruco_prev, yaw_aruco_offset, x_global,
90     y_global
91     gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
92     marker_corners, marker_ids, _ = cv2.aruco.detectMarkers(
93         gray,
94         dictionary,
95         parameters=params)
96
97     if marker_ids is not None:
98
99         aruco.drawDetectedMarkers(image, marker_corners, marker_ids)
100
101         rvecs, tvecs, _ = aruco.estimatePoseSingleMarkers(
102             marker_corners,
103             0.12,
104             mtx,
105             dist
106         )
107
108         if len(marker_ids) >= 2:
109             for i in range(len(marker_ids)):
110                 if marker_ids[i][0] == 50:
111                     tvec_origen = tvecs[i][0]
112                     rvec_origen = rvecs[i][0]
113                 else:
114                     tvec_robot = tvecs[i][0]
115                     rvec_robot = rvecs[i][0]
116
117
118
119             R1, _ = cv2.Rodrigues(rvec_origen)
120             R2, _ = cv2.Rodrigues(rvec_robot)
121             yaw = rotationMatrixtoEulerAngles(R2)
122
123             if yaw_aruco_prev is not None:
124                 diff = yaw - yaw_aruco_prev
125                 if diff > np.pi:
126                     yaw_aruco_offset -= 2 * np.pi
127                 elif diff < -np.pi:
128                     yaw_aruco_offset += 2 * np.pi
129
130             yaw_aruco_prev = yaw

```

```

131         yaw_continuous = yaw + yaw_aruco_offset
132
133         p_rel = R1.T @ (tvec_robot - tvec_origen)
134
135         x, y, _ = p_rel
136         with yaw_lock:
137             yaw_global = yaw_continuous
138             x_global = x
139             y_global = y
140
141     for rvec, tvec in zip(rvecs, tvecs):
142         cv2.drawFrameAxes(
143             image,
144             mtx,
145             dist,
146             rvec,
147             tvec,
148             0.1
149         )
150
151     return image
152
153 def aruco_thread():
154
155     global running
156     frame_count = 0
157
158     while running:
159         ret, frame = cap.read()
160         if not ret:
161             time_module.sleep(0.05)
162             continue
163
164         frame_count += 1
165         if frame_count % 2 == 0: # Procesar solo 1 de cada 2 frames
166             marker_image = detect_markers(frame)
167         else:
168             marker_image = frame
169
170         cv2.imshow('Markers', marker_image)
171
172         #cerrar la ventana con 'q'
173         if cv2.waitKey(1) & 0xFF == ord('q'):
174             running = False
175             break
176         time_module.sleep(0.033) #Dar tiempo a otros hilos
177
178 # Iniciar hilo de ArUco
179 aruco_thread_obj = threading.Thread(target=aruco_thread, daemon=True)
180 aruco_thread_obj.start()
181

```

```

182 cont = 0
183 alpha = alpha_imu = omega_R = omega_L = 0.0
184 proporcional = derivativa = integral= 0.0
185
186 #Ganancias de controlador
187 k_rho = 0.8 #ganancia proporcional cinematica
188 k_alpha = 5.0 #ganancia proporcional de orientación del robot
189
190 kp = 0.8 #ganancias para control PI de velocidades
191 ki = 0.2
192
193 x_des = float(input('Ingrese la posición deseada en x: '))
194 y_des = float(input('Ingrese la posición deseada en y: '))
195
196 start = input('Inserte cualquier caracter para iniciar el control')
197 if(start):
198     ser.write(b'a')
199     flag_global = True
200     time_module.sleep(0.2)
201     ser.reset_input_buffer() # Limpiar basura inicial
202
203 alpha_des_prev = 0
204 alpha_des_offset = 0
205 e_alpha_prev = 0
206 e_alpha_offset = 0
207 try:
208     while running:
209         # ESPERAR activamente a que haya exactamente 12 bytes
210         while ser.in_waiting < 12 and running:
211             time_module.sleep(0.001)
212
213         if not running:
214             break
215
216         if ser.in_waiting >= 8:
217             try:
218                 datos = ser.read(8)
219
220                 #obtención de velocidades reales provenientes del microcontrolador
221                 omega_R, omega_L = struct.unpack('<ff', datos) #línea.split('\t')
222
223                 #ley de control de posición
224                 e_x = x_des - x_global
225                 e_y = y_des - y_global
226                 rho = np.sqrt(e_x**2 + e_y**2)
227                 if rho > 0.05:
228                     alpha_des = np.arctan2(e_y,e_x)
229
230                 e_alpha = alpha_des - yaw_global
231                 e_alpha = (e_alpha + np.pi) % (2*np.pi) - np.pi
232

```

```

233 x_dot_r = k_rho * rho
234 alpha_dot = k_alpha * e_alpha #+ k_beta*beta
235
236 theta_dot_ref_r = (w*alpha_dot + x_dot_r) / Rr
237 theta_dot_ref_l = (-w*alpha_dot + x_dot_r) / Rr
238 st = " "
239
240 if(abs(e_x)< 0.05 and abs(e_y)<0.05): #and abs(e_alpha) < 0.1):
241     u_L = 0
242     u_R = 0
243 else:
244     e_omega_r = theta_dot_ref_r-omega_R
245     e_omega_l = theta_dot_ref_l-omega_L
246     proporcional_r = kp*e_omega_r
247     proporcional_l = kp*e_omega_l
248     if (integral_r < iM) and (integral_r > -iM):
249         integral_r = integral_r + ki * Ts * e_omega_r
250     else:
251         if integral_r >= iM:
252             integral_r = iM
253         if integral_r <= -iM:
254             integral_r = -iM
255
256     if (integral_l < iM) and (integral_l > -iM):
257         integral_l = integral_l + ki * Ts * e_omega_l
258     else:
259         if integral_l >= iM:
260             integral_l = iM
261         if integral_l <= -iM:
262             integral_l = -iM
263
264     u_L=proporcional_r+integral_r
265     u_R=proporcional_l+integral_l
266
267 u_R = np.fmax(np.fmin(u_R, iM), -iM)
268 u_L = np.fmax(np.fmin(u_L, iM), -iM)
269
270 curr_R = u_R / 3.3 #es la corriente de consumo de los motores
271 curr_L = u_L / 3.3
272
273 pwmf_R = math.fabs(u_R)*escs
274 pwmf_L = math.fabs(u_L)*escs
275
276 if(u_R < 0):
277     pwmf_R = int(pwmf_R) | 32768
278 if(u_L < 0):
279     pwmf_L = int(pwmf_L) | 32768
280
281 ser.write(struct.pack('<HH', int(pwmf_L), int(pwmf_R)))
282 tiempo = tiempo + Ts
283

```

```
284         f.write(f"{tiempo:.4f} {yaw_global:.4f} {x_global:.4f} {y_global:.4f}
{curr_L:.4f} {curr_R:.4f}\n")
285     except struct.error as e:
286         print(f"Error decodificando: {e}")
287         ser.reset_input_buffer() # Limpiar buffer corrupto
288         continue
289 except KeyboardInterrupt:
290     running = False
291
292 finally:
293     ser.close()
294     f.close()
295     cap.release()
296     cv2.destroyAllWindows()
```