UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA

FACULTAD DE INGENIERÍA

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

NAVEGACIÓN AUTÓNOMA DE UN VEHÍCULO AÉREO NO TRIPULADO USANDO APRENDIZAJE POR REFUERZO

POR:

RAYDESEL ARIEL SÁNCHEZ MONTES

TESIS PRESENTADA COMO REQUISITO PARA OBTENER EL GRADO DE

MAESTRO EN INGENIERÍA EN COMPUTACIÓN

CHIHUAHUA, CHIH., MÉXICO **NOVIEMBRE 2023**

"Navegación autónoma de un vehículo aéreo no tripulado usando aprendizaje por refuerzo". Tesis, presentada por Raydesel Ariel Sánchez Montes como requisito parcial para obtener el grado de Maestría en Ingeniería en Computación, ha sido aprobado y aceptado por:

M.I. Fabián Vinicio Hernández Martínez
Director de la Facultad de Ingeniería

Dr. Fernando Martínez Reyes
Secretario de Investigación y Posgrado

M.S.I. Karina Requena Yáñez
Coordinadora Académica

Dr. Alain Manzo Martínez
Director de Tesis

Noviembre 2023

COMITÉ
Dr. Alain Manzo Martínez
Dr. Luis Fernando Gaxiola Orduño
Dr. Fernando Martínez Reyes
Dra. Graciela María de Jesús Ramírez Alonso

UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA

31 de octubre de 2023.

ING. RAYDESEL ARIEL SÁNCHEZ MONTES
Presente. -

En atención a su solicitud relativa al trabajo de tesis para obtener el grado de Maestro en Ingeniería en Computación, nos es grato transcribirle el tema aprobado por esta Dirección, propuesto y dirigido por el director **Dr. Alain Manzo Martínez** para que lo desarrolle como tesis, con el título **"NAVEGACIÓN AUTÓNOMA DE UN VEHÍCULO AÉREO NO TRIPULADO USANDO APRENDIZAJE POR REFUERZO"**.

Índice de Contenido

UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA

Capítulo 3: Metodología

3.1 Construir entornos

3.2 Generar la trayectoria

3.3 Seguimiento de la trayectoria en simulación

3.4 Métricas de error

Capítulo 4: Experimentos

4.1 Experimento para comparar la relación entre el RMSE y la velocidad del cuadricóptero en un entorno circular con Bézier y B-spline

4.2 Experimento para obtener la distancia mínima en un entorno cuadrado para Bézier y B-spline

4.3 Experimento para obtener el grado de la curva para B-spline

4.4 Experimento para la comparación de tres métodos para generar trayectoria en entornos interiores

Capítulo 5: Conclusiones y recomendaciones

5.1 Conclusiones

5.2 Recomendaciones

5.3 Trabajos futuros

A T E N T A M E N T E

"naturam subiecit aliis"

FACULTAD DE INGENIERÍA U.A.CH.

EL DIRECTOR

M.I. FABIÁN VINICIO HERNÁNDEZ MARTÍNEZ

SECRETARIO DE INVESTIGACIÓN Y POSGRADO

DR. FERNANDO MARTÍNEZ REYES

DIRECCIÓN

Agradecimientos

Quiero expresar mi profundo agradecimiento al Consejo Nacional de Humanidades, Ciencia y Tecnología (CONAHCYT) por el apoyo económico brindado durante mi Maestría. Sin su generosidad y respaldo, este logro habría sido mucho más difícil de alcanzar. Su inversión en la educación y la investigación es fundamental para el desarrollo de la ciencia y la tecnología en nuestro país.

A usted, Alain, mi director de Tesis, tengo mucho que agradecer. Gracias por darme la oportunidad de participar en este trabajo de investigación. Esta experiencia ha sido tremenda, enorme y sin duda valiosa para mí. Sus consejos expertos, su paciencia y su dedicación fueron fundamentales en la elaboración de este documento.

También quiero agradecer a los profesores y compañeros que me proporcionaron los medios y los recursos necesarios para llevar a cabo esta investigación. Su apoyo y colaboración fueron esenciales en mi camino académico.

Esta tesis no habría podido finalizarse sin la paciencia y la colaboración de todas las personas que me brindaron su apoyo y ánimo cuando más lo necesitaba, mostrándome su cariño incondicional.

Por último, pero no menos importante, quiero expresar mi agradecimiento especial a mi familia. Ellos son testigos de la evolución de mi vida y del esfuerzo que he realizado para llegar hasta aquí. Sin su valioso apoyo, de ninguna manera habría podido lograrlo. Su amor y aliento constante son mi mayor fortaleza.

Este logro es el resultado del esfuerzo conjunto de muchas personas, y por eso les estoy profundamente agradecido a todos. Sin su apoyo, orientación y cariño, este trabajo no habría sido posible. Gracias de todo corazón.

Resumen

En esta tesis, se aborda el tópico de la navegación autónoma de vehículos aéreos no tripulados (UAVs), mediante la optimización del algoritmo de aprendizaje por refuerzo Aprendizaje-Q para la generación autónoma de trayectorias en entornos de interiores como bodegas, casas, farmacias, etc.

La navegación autónoma de UAVs es un campo de investigación relevante en la actualidad, con diversas aplicaciones. Sin embargo, la planificación de trayectorias en entornos de interiores, con obstáculos y limitaciones como paredes, objetos, pasillos estrechos y cambios bruscos de dirección, es un desafío significativo. Superar este desafío es crucial para permitir la navegación autónoma segura y eficiente.

El objetivo general de esta tesis es, desarrollar un modelo computacional que permita la simulación de la navegación autónoma de un UAV, usando técnicas de aprendizaje por refuerzo.

La metodología se centra en discretizar entornos interiores y en la adaptación de dos métodos: muestreo de Thompson y distribución Gaussiana, para modificar la política utilizada en el Aprendizaje-Q para la selección de la acción que llevara al estado siguiente utilizando la ecuación de la diferencia temporal. Se diseñaron y utilizaron cinco entornos de interiores basados en planos arquitectónicos para evaluar el rendimiento del algoritmo.

Los resultados muestran una mejora en cuanto al tiempo para la convergencia con el algoritmo Aprendizaje-Q optimizado.

En conclusión, las trayectorias generadas mostraron un vuelo libre de colisiones que cumplen con los objetivos de navegación. Además, la optimización del algoritmo Aprendizaje-Q permite una exploración y explotación de recompensas más eficiente.

Palabras clave: Navegación autónoma, Entornos interiores, Vehículos aéreos no tripulados, Aprendizaje por refuerzo, Algoritmo Aprendizaje-Q.

Índice de Contenido

Índice de Tablas

Índice de Figuras

Capítulo 1: Introducción

Los humanos siempre han soñado con construir seres artificiales que se encarguen de tareas tediosas, molestas o peligrosas, que los entretengan y que estén sujetos a las órdenes del hombre (Nüchter 2012).

Robots móviles que controlados a distancia por un operador realizan tareas como la inspección de tuberías o la fotografía aérea, no son autónomos, sino que utilizan sus sensores para dar a su operador acceso remoto a lugares peligrosos, distantes o inaccesibles (Bestaoui Sebbane 2018).

Algunos robots son semiautónomos, realizando subtareas automáticamente. El piloto automático de un dron estabiliza el vuelo mientras el humano elige la trayectoria. Un robot en una tubería puede controlar su movimiento dentro de la misma mientras el humano busca los defectos que hay que reparar (Joseph and Cacace 2021).

Los robots móviles totalmente autónomos no dependen de un operario, sino que toman decisiones por sí mismos y realizan tareas, como transportar material mientras navegan por un terreno incierto (paredes, puertas dentro de los edificios, cruces en las calles, etc.) y en un entorno en constante cambio (gente caminando, coches circulando por las calles, etc.) (Mordechai and Francesco 2018).

Quizás el robot móvil autónomo que más publicidad está recibiendo estos días es el coche auto conducido. Su desarrollo es extremadamente difícil debido a la gran complejidad del entorno incierto del tráfico motorizado y a los estrictos requisitos de seguridad (Gupta et al. 2021).

Gran parte de la investigación y el desarrollo de la robótica se centra hoy en hacer que los robots sean más autónomos, mejorando los sensores y permitiendo un control más inteligente del robot. Unos sensores de calidad pueden percibir los detalles de situaciones más complejas, pero para manejar estas situaciones, el control del comportamiento del robot debe ser muy flexible y adaptable. La visión, en particular, es un campo de investigación muy activo porque las cámaras son baratas y la información que pueden adquirir es muy rica.

Se están haciendo esfuerzos para que los sistemas sean más flexibles, de modo que puedan aprender a adaptarse a nuevas situaciones (Mordechai and Francesco 2018).

En años recientes el uso de vehículos aéreos no tripulados (UAVs por sus siglas en ingles *unmanned aerial vehicles*) se ha hecho más frecuente en actividades como la agricultura de precisión (Niu et al. 2019; Tello Vargas and Herrera Victorio 2019), la entrega de mercancía (Miranda et al. 2022), carreras de drones (Song et al. 2020), búsqueda de personas en zonas peligrosas (Hodge, Hawkins, and Alexander 2021), tareas interiores (Mesa Santana 2019), Limpieza de edificios (Plaza Rey 2017), etc. Independientemente del tipo de tarea, la planificación autónoma de la trayectoria es siempre la clave para completar la tarea (Zhang, Li, and Dong 2022).

La tesis se enfoca en desarrollar un modelo computacional que permite la planificación autónoma de la trayectoria en dos dimensiones, para un UAV en un entorno simulado usando técnicas de aprendizaje por refuerzo. Además, se llevan a cabo pruebas en diferentes entornos para evaluar la precisión del UAV durante el vuelo y optimizar las trayectorias, tomando en cuenta las limitaciones del UAV para mejorar su capacidad de navegación autónoma para alcanzar un punto de destino de manera eficiente y precisa.

El aprendizaje por refuerzo (RL por sus siglas en ingles *Reinforcement Learning*) (Sutton and Barto 2018) es un método computacional para entender y automatizar el aprendizaje orientado a objetivos y la toma de decisiones. Aprende a asignar estados a acciones para obtener la política óptima. Al agente no se le dice qué acciones son correctas, sino que tiene que probar y equivocarse para descubrir qué acciones producirán el mayor rendimiento. La diferencia entre los algoritmos de RL y otros métodos de computación es que RL hace hincapié en que el agente aprende a través de la interacción con el entorno directamente, sin supervisión ni modelos del entorno.

Una buena forma de entender el RL es considerar algunos ejemplos:

- Un robot móvil decide entre si debe entrar a limpiar una nueva habitación o empezar a buscar la forma de volver a su estación para recargar batería. Su

decisión se basa en el nivel de carga actual y qué tan rápido y fácil fue encontrar la estación de batería en el pasado (Sutton and Barto 2018).

- Un dron autónomo, diseñado para tareas en interiores, establece maniobras y trayectorias a seguir dentro de una casa, experimentando con varias formas de seguir una trayectoria: volando a cierta altura, dando vuelta con cierto ángulo, e incluso dando vuelta sobre sí mismo, enriquece su criterio sobre cómo debe de ser seguida una trayectoria o si hay otras trayectorias más sencillas de recorrer, para realizar un monitoreo más rápido y efectivo de la casa. (Mesa Santana 2019; Tiemann, Ramsey, and Wietfeld 2018).

1.1 Definición del Problema

El problema de la navegación autónoma en interiores se refiere a la capacidad de un UAV para moverse de forma autónoma y segura en entornos cerrados, sin la necesidad de intervención humana. En este tipo de entornos, existen obstáculos y limitaciones que deben ser considerados en la planificación y seguimiento de la trayectoria del UAV, como paredes, objetos, pasillos estrechos y cambios bruscos de dirección.

La planificación autónoma de la trayectoria se refiere a el proceso mediante el cual el UAV toma una secuencia de decisiones de manera autónoma, con el fin de generar un conjunto de coordenadas que aseguren la ausencia de colisiones con obstáculos en el entorno y la consecución de cumplir un objetivo prestablecido, ya sea alcanzar un punto de destino o explorar un entorno desconocido, con el fin de maximizar su cobertura de información (Zhou, X., Yi, Z., Liu, Y., Huang, K., Huang 2020). Por lo tanto la planificación autónoma de la trayectoria implica la consideración de factores, como el tiempo de vuelo, el consumo de energía y los límites de maniobrabilidad de la plataforma utilizada (Cabreira, Brisolara, and Ferreira Paulo 2019; Nex et al. 2022; H. Wang et al. 2015).

.

1.2 Antecedentes

En los últimos años, se han realizado muchos trabajos utilizando RL para resolver problemas de navegación autónoma en UAVs:

Las entregas de paquetes a domicilio son comunes en todo el mundo, los sistemas de entrega con drones son de gran interés, ya que pueden permitir entregas más rápidas y baratas. En (Miranda et al. 2022), se presenta un sistema de navegación para drones autónomos, aplicado a la entrega de paquetes a domicilio. El sistema genera una trayectoria entre un origen y un destino, y controla al dron para que siga esta trayectoria basándose en su localización, obtenida mediante GPS, 9DoF IMU y barómetro. En la fase de aterrizaje se utiliza una técnica de visión por computadora para la detección de marcas *ArUco*; además, la estimación del software del dron se fusiona con dispositivos *ultra-wide band* (UWB), utilizando un algoritmo *extended Kalman filter* para mejorar la precisión del aterrizaje. También se implementa un método *vector field-based* para que el dron siga la trayectoria de forma suave, reduciendo las vibraciones que podrían dañar la carga transportada. Experimentos en un entorno real validan la estrategia y permiten evaluar el rendimiento de las técnicas implementadas.

Uno de los problemas para lograr que los sistemas de navegación autónoma en UAV's tengan éxito es la evasión de obstáculos estáticos, una aproximación para solucionar este problema la propone Cerón Contreras (2022). Específicamente se utiliza el algoritmo de *Deep Q-Network* (DQN) para entrenar un UAV, que se desplaza desde un punto aleatorio A hasta un punto aleatorio B evitando obstáculos estáticos. Para ello se usa un UAV prediseñado en el software de simulación de robots *Coppelia Sim*. Al cuadricóptero se le integró una cámara de tipo *Fisheye* que permite capturar cuadro a cuadro una imagen de profundidad de 360° que se usa como entrada al sistema en conjunto con la posición actual del cuadricóptero y la del punto objetivo.

La importancia de una vigilancia precisa y multifacética es bien conocida para identificar los problemas a tiempo y evitar que se agraven, en (Hodge et al., 2021), se describe un algoritmo de navegación que usa los datos de los sensores a bordo del dron para guiarlo

al sitio del problema. Se utiliza el algoritmo de *proximal policy optimisation deep reinforcement learning* junto con *incremental curriculum learning and long short-term memory neural,* para implementar un algoritmo de navegación adaptativa. Se evalúan diferentes configuraciones con técnicas heurísticas para demostrar su precisión y eficacia. Finalmente, se considera como la seguridad del dron puede ser asegurada en un entorno real.

En las carreras de drones, el objetivo es recorrer un conjunto de puntos de ruta lo más rápido posible. Song et al. (2021) presentan un método para generación de trayectoria en tiempo mínimo para drones de carreras utilizando las técnicas de *deep reinforcement learning* y *relative gate observations,* este enfoque puede calcular de forma adaptativa trayectorias en tiempo óptimo para diseños de pista aleatorios, presentando una ventaja computacional sobre otros métodos similares, se evalúa en un conjunto de pistas de carreras en simulación y en el mundo real, alcanzando velocidades de hasta 17 m/s con un dron físico.

El uso de estos drones se ve restringido por su limitada potencia y capacidad de cálculo, abordando esta problemática Anwar & Raychowdhury (2020) presentan un enfoque basado en *Transfer Learning* para reducir la computación requerida a bordo para entrenar una *deep neural network* para navegación autónoma a través del *Deep Reinforcement Learning*. Una librería de meta-entornos realistas en 3D es diseñada manualmente utilizando *Unreal Gaming Engine*.

En (C. Wang et al. 2019), se propone un método basado en *deep reinforcement learning* que permite a los UAVs ejecutar tareas de navegación en entornos complejos. El problema es formulado como un *partially observable Markov decision process* (POMDP) y resuelto por un novedoso algoritmo de *deep reinforcement learning* diseñado sobre dos teoremas *strictly proved policy gradient* en el *act-critic framework*. Los resultados de los experimentos demuestran que este método puede permitir a los vehículos aéreos no tripulados realizar de forma autónoma la navegación en un entorno complejo virtual a gran escala.

1.3 OBJETIVOS

Desarrollar un modelo computacional que permita la simulación de la navegación autónoma de un vehículo aéreo no tripulado, usando técnicas de aprendizaje por refuerzo. Objetivos específicos:

- Revisión de la literatura y estado actual del tema con respecto al estado del arte.
- Hacer una búsqueda de las herramientas computacionales que sirven para la simulación de la navegación de vehículos aéreos no tripulados.
- Aprender a utilizar el entorno de programación para modelar el vehículo aéreo no tripulado.
- Implementar los algoritmos computacionales que permitan el modelado del vehículo aéreo no tripulado.
- Implementar los algoritmos de aprendizaje por refuerzo para probar el modelo sobre distintas trayectorias de navegación.
- Probar el vehículo aéreo no tripulado de manera simulada.
- Reportar los resultados obtenidos y realizar las conclusiones del trabajo

1.4 JUSTIFICACIÓN

El uso de drones se ha hecho común en múltiples aplicaciones, siendo usados como sistema de grabación, vigilancia, carreras, etc. Los sistemas de navegación autónoma pueden expandir las aplicaciones de los drones, realizando tareas automáticas, evitando colisiones y obedeciendo a las instrucciones del operador del sistema. Por lo tanto, este tipo de drones se puede utilizar en seguridad civil, periodismo, fotografía profesional, entre otros (Ramos Catari 2017).

Aunque hay muchos avances en los sistemas de navegación autónoma, todavía hay varios problemas abiertos en investigación, en cuanto a la cantidad de sensores a utilizar para lograr un vuelo autónomo (Padhy et al. 2018), la precisión con la que el dron sigue una trayectoria (Hodge, Hawkins, and Alexander 2021), superar la media de vuelo seguro (Anwar

and Raychowdhury 2020). Un reto a destacar por su dificultad es transferir el aprendizaje logrado en simulación a un dron físico, ya que es difícil conseguir una simulación que pueda generar datos comparables a los que se pueden obtener en el mundo real (Anwar and Raychowdhury 2020; Hwangbo et al. 2017; Song et al. 2021). Otro reto presente es lograr reducir el tiempo que el dron tarda adquiriendo la experiencia necesaria para llevar a cabo una tarea (Jang and Choi 2022). Todos estos obstáculos deben eliminarse para que la navegación autónoma tenga éxito. Por lo tanto, los desafíos de la navegación autónoma en UAV's deben ser abordados con sistemas eficientes para esta tarea.

El RL tiene muchas ventajas para resolver problemas de navegación autónoma en UAVs: (1) En primer lugar, en lugar de requerir modelos precisos del robot y modelos de los sensores, el RL recibe las observaciones originales obtenidas por los sensores directamente, y las mapea a las acciones a realizar, lo cual es un método de toma de decisiones de extremo a extremo y evita la incertidumbre causada por el ruido de los sensores. (2) Entonces, RL no depende del modelo del entorno y es adecuado para resolver problemas de planificación de trayectorias en entornos dinámicos. (3) Por último, en comparación con otros métodos basados en el aprendizaje, como el aprendizaje supervisado, RL puede aprender directamente a través de la interacción con el entorno, en lugar de las etiquetas de entrenamiento (Zhang, Li, and Dong 2022).

1.5 APORTACIONES

Las contribuciones de esta investigación son significativas y pueden resumirse de la siguiente manera:

Optimización del Algoritmo Aprendizaje-Q

Se realizo una optimización del algoritmo de aprendizaje por refuerzo Aprendizaje-Q para la generación autónoma de trayectorias en entornos interiores, donde, se han adaptado dos métodos: muestreo de Thompson y distribución Gaussiana, para modificar la política utilizada en el Aprendizaje-Q para la selección de la acción que llevara al estado siguiente

utilizando la ecuación de la diferencia temporal. Esta optimización permite una exploración y explotación de recompensas que demuestra un aumento del 81% en eficiencia, en términos de iteraciones para alcanzar la convergencia, en comparación con el método convencional.

Modelo Computacional para Navegación Autónoma 2D

Se ha logrado desarrollar un modelo computacional que permite la simulación de la navegación autónoma en 2D de un cuadricóptero en entornos estáticos interiores. Este modelo es una herramienta valiosa que facilita la investigación y el desarrollo de algoritmos de navegación autónoma. Además, el modelo se ha optimizado con una configuración que permite la generación de trayectorias libres de colisiones que cumplen de manera precisa con los objetivos de navegación. Esto es esencial para garantizar la seguridad y la precisión en la navegación de vehículos aéreos no tripulados en entornos interiores.

Validación de Trayectorias en Entornos Simulados

El modelo computacional desarrollado ofrece la posibilidad de probar y validar trayectorias en entornos simulados antes de implementarlas en situaciones reales. Esto es fundamental para reducir riesgos y garantizar un rendimiento óptimo de las trayectorias de vuelo en entornos interiores, lo que puede tener aplicaciones significativas en la navegación autónoma de drones y vehículos aéreos en espacios cerrados.

En resumen, estas contribuciones no solo mejoran significativamente el rendimiento del algoritmo de Aprendizaje-Q, sino que también proporcionan herramientas y métodos para la navegación autónoma en entornos interiores, lo que puede tener un impacto positivo en la seguridad y la eficiencia de los sistemas de vehículos aéreos no tripulados en aplicaciones diversas.

Capítulo 2: Marco Teórico

En este capítulo se presenta el marco teórico que sienta las bases sobre la simulacion de la navegación autónoma en interiores para UAVs. Primero, con el objetivo de proporcionar los conceptos teóricos relacionados con los UAVs en entornos interiores simulados, se define la simulación de UAV's y se introduce el cuadricóptero como el tipo de UAV que se utiliza en esta tesis, seguido por un análisis matemático y dinámico del cuadricóptero, que incluye su modelado utilizando quaternions. Además, se profundiza en los componentes de un cuadricóptero, incluyendo los sensores, la unidad de medición inercial, así como el tiempo de vuelo y la velocidad máxima que los UAVs pueden alcanzar en entornos interiores.

Luego, con el objetivo de proporcionar los conceptos teóricos relacionados con la navegación en entornos de interiores, se describe la localización que implica determinación de la posición a partir de un ángulo y una distancia, así como también la localización a través de la determinación de la posición por triangulación. Luego se describe la representación de los mapas discretos y continuos. También, se presenta un análisis de la interpolación, incluyendo curvas de Bézier y B-splines.

A continuación, con el objetivo de proporcionar los conceptos teóricos relacionados con el aprendizaje por refuerzo, se describe la exploración y explotación de la recompensa, la recompensa retardada, los procesos de decisión de Markov, las políticas y funciones de valor y los algoritmos Aprendizaje-Q y muestreo de Thompson. Además, dado que en esta tesis se adapta una distribución Gaussiana al algoritmo Aprendizaje-Q, se proporciona una sesión adicional dedicada a la teoría de la distribución Gaussiana.

Ahora, con el objetivo de proporcionar los conceptos teóricos relacionados con las métricas de error para evaluar la precisión durante el vuelo del UAV, se describen las métricas de porcentaje de error y la raíz del error cuadrático medio.

Por último, se presentan se presentan las herramientas utilizadas para realizar esta investigación.

Este marco teórico es esencial para la comprensión y ejecución de la metodología presentada en el capítulo siguiente, donde se describe en detalle el simulador, el algoritmo de Aprendizaje-Q, la generación de la trayectoria y la medición del error.

2.1 SIMULACIÓN DE UAV'S

Para la planificación, ejecución y seguimiento de misiones de UAV's se utilizan plataformas de software que permiten la realización de funcionalidades complejas del UAV, utilizando entornos y técnicas de visualización e interacción en tiempo real con un conjunto multidimensional de datos del UAV (Bestaoui Sebbane 2018).

El simulador de UAV's que se utiliza en esta tesis es Robot Operating System (ROS)-GAZEBO (Joseph and Cacace 2021; ROS 2022) el cual es una colección de paquetes de software de código abierto especializado en aplicaciones robóticas. Uno de sus principales propósitos es proporcionar comunicación entre el usuario, el sistema operativo del ordenador y los equipos externos al mismo. Este equipo puede incluir sensores, cámaras y robots. La ventaja de ROS es la abstracción del hardware y su capacidad para controlar un robot en general, un UAV en particular, sin que el usuario tenga que conocer todos los detalles del robot. Los UAV pueden simularse utilizando el simulador 3D Gazebo que incluye la física del UAV mientras se mueve en un entorno simulado. El módulo de detección de colisiones de Gazebo utiliza la propiedad de colisión para identificar los límites del objeto.

2.2 CUADRICÓPTERO

El Cuadricóptero forma parte de una amplia categoría de drones que utilizan cuatro rotores como potencia directa de vuelo (Fairchild and Harman 2016; He and Zhao 2014). Tiene seis grados de libertad (6 GDL: x, y, z, roll, pitch, yaw) y cuatro variables controlables, lo que lo convierte en un sistema dinámicamente inestable. El número de variables controlables es igual al número de propulsores que afectan a la posición y altitud del

cuadricóptero en el espacio. El cuadricóptero no puede moverse de forma traslacional sin la rotación alrededor de uno de los ejes, es decir, sin la inclinación del cuadricóptero. Para ello, es necesario aumentar o disminuir el empuje de uno o dos propulsores. Si el cambio de empuje se produce en un solo propulsor, eso podría causar inestabilidad en el giro alrededor del eje de rotación Z_b. Para lograr un vuelo estable, es necesario combinar varios sensores de alta precisión con un algoritmo de control rápido y robusto (Benic, Piljek, and Kotarski 2016).

Las únicas partes móviles del cuadricóptero son las hélices que se fijan en un eje de los propulsores. El cuadricóptero puede tener una configuración en cruz (+) (los ejes X_b e Y_b están orientados en las direcciones de los propulsores), o puede tener una configuración en X (los ejes XB e YB están orientados en las direcciones entre los propulsores). Para la modelización matemática posterior, se asume la configuración +. La estructura del cuadricóptero es una construcción simétrica, ligera y delgada que conecta mecánicamente los propulsores. El propulsor y la hélice están directamente conectados, siendo todos los ejes del propulsor fijos y paralelos. La rotación de la hélice provoca un flujo de aire en la dirección negativa del eje Z_b, lo que resulta en un empuje en la dirección positiva del eje Z_b. Se supone que la estructura del cuadricóptero es rígida. Lo único que influye directamente en el movimiento del cuadricóptero son las RPM de cada motor (Benic, Piljek, and Kotarski 2016).

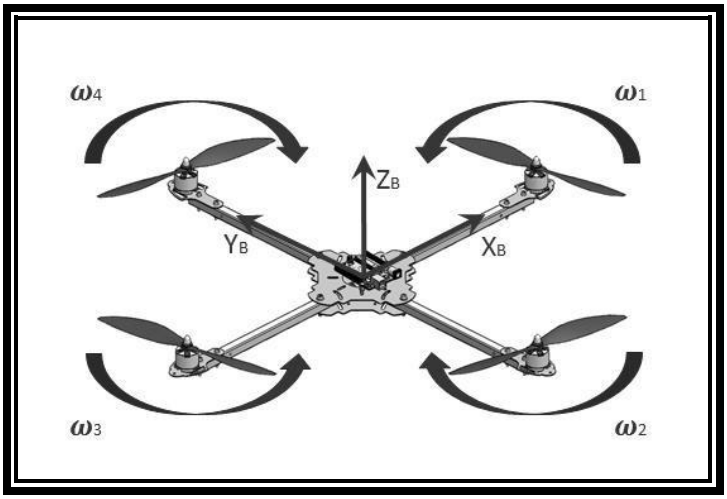


Figura 2.1 Estructura de un Cuadricóptero (Benic, Piljek, and Kotarski 2016).

2.3 Modelo Matemático de un Cuadricóptero

El modelo matemático describe el movimiento y el comportamiento del cuadricóptero con respecto a los valores de entrada del modelo y las influencias externas sobre el cuadricóptero. Mediante el uso de un modelo matemático, es posible predecir la posición y la altitud del cuadricóptero conociendo las cuatro velocidades angulares de las hélices, es decir, permite la simulación por ordenador del comportamiento del cuadricóptero en diferentes condiciones (Benic, Piljek, and Kotarski 2016).

La simulación por ordenador es un método relativamente sencillo y barato para la verificación del algoritmo de control. Un modelo matemático más detallado describe el comportamiento del cuadricóptero con mayor precisión, pero también requiere más recursos computacionales, lo que conlleva un mayor tiempo de simulación o incluso la imposibilidad de completarla con éxito. En función de los requisitos, es necesario encontrar un equilibrio entre la complejidad del modelo y la precisión (Benic, Piljek, and Kotarski 2016).

Para el modelo matemático, el movimiento del cuadricóptero depende directamente de las velocidades angulares de las hélices. La estructura del cuadricóptero con las hélices son los únicos elementos que se tendrán en cuenta en este modelo matemático (Figura 2.1) (Benic, Piljek, and Kotarski 2016).

2.3.1 Sistema de coordenadas

Para obtener el modelo matemático, es necesario definir dos sistemas de coordenadas:

- Marco fijo terrestre (E-frame, $\mathcal{F}^E$)
- Marco fijo del cuerpo (B-frame, $\mathcal{F}^B$)

Algunas propiedades físicas del cuadricóptero se miden en $\mathcal{F}^E$ (ángulos de *roll-pitch-yaw*, velocidades angulares), mientras que algunas propiedades se miden en $\mathcal{F}^B$ (aceleraciones lineales) (Benic, Piljek, and Kotarski 2016).

$\mathcal{F}^E$ es el sistema de coordenadas inerciales donde la dirección positiva del eje Z_E está en la dirección de la tierra. La posición del cuadricóptero $\boldsymbol{\xi}$ y la altitud $\boldsymbol{\eta}$ se definen en $\mathcal{F}^E$ (Benic, Piljek, and Kotarski 2016).

$\mathcal{F}^B$ se fija en el cuerpo del cuadricóptero. La dirección positiva del eje X_B pasa por el propulsor 1 que se encuentra en la parte delantera del cuadricóptero. La dirección positiva del eje Y_B pasa por el propulsor 4 que se encuentra en el lado izquierdo del cuadricóptero. El eje Z_B es perpendicular a los ejes X_B y Y_B y su dirección positiva está en la dirección de las fuerzas de empuje de los propulsores. Se asume que el origen de $\mathcal{F}^B$ coincide con el centro de gravedad del cuadricóptero. Las velocidades lineales $\mathbf{v}^{\mathrm{B}}$, las velocidades angulares $\boldsymbol{\omega}^{\mathrm{B}}$, las fuerzas $\mathbf{f}^{\mathrm{B}}$ y los torques $\boldsymbol{\tau}^{\mathrm{B}}$ se definen en $\mathcal{F}^B$ (Benic, Piljek, and Kotarski 2016).

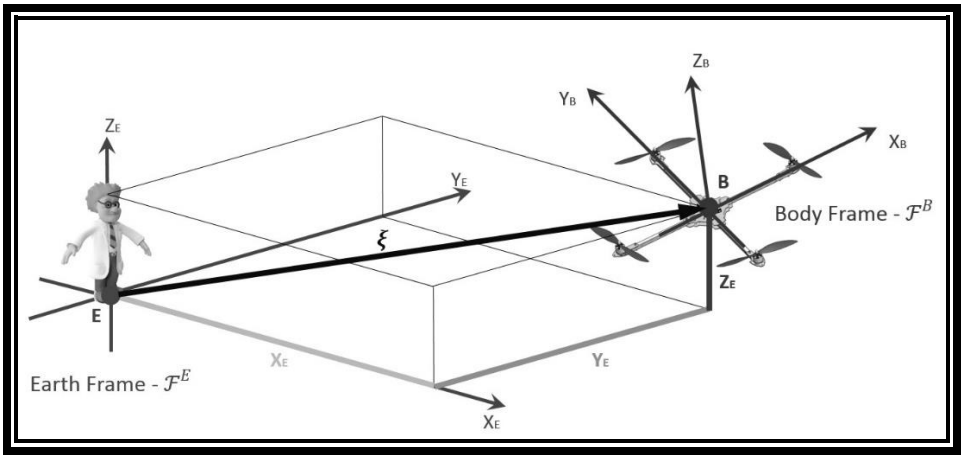


Figura 2.2 Marco Terrestre y Marco del Cuerpo (Benic, Piljek, and Kotarski 2016).

La posición del cuadricóptero se define con el vector **ξ** entre los orígenes de $\mathcal{F}^E$ y $\mathcal{F}^B$ (Figura 2.2)

$$\boldsymbol{\xi} = [X \;\; Y \;\; Z]^T. \tag{2.1}$$

La altitud del cuadricóptero **η** se define con la orientación de $\mathcal{F}^B$ con respecto al $\mathcal{F}^E$. La orientación se define con tres rotaciones consecutivas alrededor de la $\mathcal{F}^E$. Se aplica el orden *roll-pitch-yaw*

$$\boldsymbol{\eta} = [\phi \; \theta \; \psi]^T. \tag{2.2}$$

Las ecuaciones de movimiento son más adecuadas para formularlas con respecto al $\mathcal{F}^B$ por varias razones: la matriz de inercia del sistema es invariable en el tiempo, la simplificación de las ecuaciones debido a la simetría del marco del cuadricóptero, las mediciones de los sensores se convierten fácilmente en $\mathcal{F}^B$ y la simplificación de las ecuaciones de las variables de control (Benic, Piljek, and Kotarski 2016).

2.3.2 Cinemática del cuadricóptero

La cinemática de un cuerpo rígido de 6 GDL está dada por:

$$\dot{\boldsymbol{\varepsilon}} = \boldsymbol{J}\, \boldsymbol{v}, \tag{2.3}$$

donde $\dot{\boldsymbol{\varepsilon}}$ es el vector de velocidad generalizado en $\mathcal{F}^B$, $\boldsymbol{v}$ es el vector de velocidad generalizado en $\mathcal{F}^B$, y $\boldsymbol{J}$ es la matriz de rotación y transformación generalizada. $\boldsymbol{\varepsilon}$ se compone de la posición del cuadricóptero $\boldsymbol{\xi}$ y de la altitud $\boldsymbol{\eta}$

$$\boldsymbol{\varepsilon} = [\boldsymbol{\xi} \;\; \boldsymbol{\eta}]^T = [X \;\; Y \;\; Z \;\; \phi \;\; \theta \;\; \psi]^T. \tag{2.4}$$

El vector de velocidad generalizado en $\mathcal{F}^B$, se define de la misma manera:

$$\boldsymbol{v} = [\mathbf{v}^{\mathrm{B}} \;\; \boldsymbol{\omega}^{\mathrm{B}}]^T = [u \;\; v \;\; w \;\; p \;\; q \;\; r]^T. \tag{2.5}$$

La matriz de rotación y transformación generalizada transfiere las velocidades de $\mathcal{F}^B$ a $\mathcal{F}^E$, que es una forma más natural de observar el movimiento del cuadricóptero. Consta de cuatro submatrices

$$\boldsymbol{J} = \begin{bmatrix} \mathbf{R} & \mathbf{0}_{3x3} \\ \mathbf{0}_{3x3} & \mathbf{T} \end{bmatrix}. \tag{2.6}$$

R es la matriz de rotación:

$$\mathbf{R} = \begin{bmatrix} \cos\psi\cos\theta & cos\psi\, sin\theta\, sin\phi - sin\psi\, cos\phi & cos\psi\, sin\theta\, cos\phi + sin\psi\, sin\phi \\ \sin\psi\cos\theta & sin\psi\, sin\theta\, sin\phi + cos\psi\, cos\phi & sin\psi\, sin\theta\, cos\phi - cos\psi\, sin\phi \\ -sin\theta & cos\theta\, sin\phi & cos\theta\, cos\phi \end{bmatrix}. \tag{2.7}$$

Debido a la necesidad de transformar los valores medidos de un sistema de coordenadas a otro, se introduce la matriz de rotación que por multiplicación matricial, transfiere el vector de velocidad lineal de un sistema de coordenadas a otro. La matriz **R** es la matriz ortogonal.

Los ángulos y las velocidades angulares se miden en $\mathcal{F}^E$. La matriz **T** es la matriz de transformación que transfiere las velocidades angulares de $\mathcal{F}^B$ a $\mathcal{F}^E$

$$\mathbf{T} = \begin{bmatrix} 1 & sin\phi\, tan\theta & cos\phi\, tan\theta \\ 0 & cos\phi & -sin\phi \\ 0 & sin\phi/cos\theta & cos\phi/cos\theta \end{bmatrix}. \tag{2.8}$$

Para transferir las velocidades angulares de $\mathcal{F}^E$ a $\mathcal{F}^B$, el vector de velocidad angular en $\mathcal{F}^E$ tiene que ser multiplicado por la matriz inversa de la matriz de transformación (Benic, Piljek, and Kotarski 2016).

2.3.3 Modelado dinámico con quaternions

El estado de traslación y rotación de cualquier cuerpo rígido puede expresarse como $x \coloneqq \left[\xi\ \dot{\xi}\ \boldsymbol{q}\ \omega\right]^T$ donde $\xi \in \mathrm{R}^3$ simboliza la posición del cuerpo en el marco inercial, $\dot{\xi}$ su velocidad, $\boldsymbol{q} = q_0 + [\,q_1\ \ q_2\ \ q_3\,]^T$ define la orientación del vehículo respecto al marco inercial, representada como un quaternion unitario (Sciavicco and Siciliano 2001) y $\omega = \left[\omega_x\ \omega_y\ \omega_z\right]^T$ representa la velocidad de rotación en el marco del cuerpo situado en su centro de masa. Por lo tanto, siguiendo las ecuaciones de movimiento de Newton, el modelo dinámico de cualquier cuerpo rígido expresado con quaternions unitarios es

$$\dot{\mathrm{x}} = \begin{bmatrix} \dot{\xi} \\ \ddot{\xi} \\ \dot{\boldsymbol{q}} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \dot{\xi} \\ m^{-1}F_I \\ \frac{1}{2}\boldsymbol{q} \otimes \omega \\ J^{-1}(\tau - \omega \times J\omega) \end{bmatrix}, \tag{2.9}$$

donde J es la matriz de inercia, τ representa el torque total, ambos con respecto al sistema de referencia del cuerpo, y F_I define la fuerza total aplicada al cuerpo en el sistema de coordenadas inerciales (Abaunza, Castillo, and Lozano 2018).

Aplicando un mapeo logarítmico a la altitud, la variable de estado del cuerpo se define como $\mathrm{y} = [\ \xi\ \ \dot{\xi}\ \ \bar{\alpha}\ \ \dot{\alpha}\ ]^T$ el modelo dinámico se convierte en

$$\dot{\mathrm{y}} = \begin{bmatrix} \dot{\xi} \\ \ddot{\xi} \\ \dot{\bar{\alpha}} \\ \ddot{\bar{\alpha}} \end{bmatrix} = \begin{bmatrix} \dot{\xi} \\ m^{-1}F_I \\ \dot{\bar{\alpha}} \\ J^{-1}(\tau - \dot{\bar{\alpha}} \times J\dot{\bar{\alpha}}) \end{bmatrix}. \tag{2.10}$$

Las ecuaciones (2.9) y (2.10) pueden ser usadas para describir cualquier sistema mecánico incluyendo UAV's. En la siguiente subsección se describe la dinámica de un cuadricóptero usando este enfoque (Abaunza, Castillo, and Lozano 2018).

2.3.4 Vector de rotación

Considerando $\bar{p} \in \mathbb{R}^3$ como un vector 3D en un primer marco de referencia (las coordenadas terrestres), y $\bar{p}'$ como el mismo vector, pero ahora respecto a un nuevo marco de referencia (las coordenadas del cuerpo del cuadricóptero), entonces $\bar{p}$ puede ser transformado en $\bar{p}'$ usando un producto doble quaternions como

$$\bar{p}' = \boldsymbol{q}^{-1} \otimes \bar{p} \otimes \boldsymbol{q} = \boldsymbol{q}^{*} \otimes \bar{p} \otimes \boldsymbol{q}, \tag{2.11}$$

Donde el quaternion $\boldsymbol{q}$ representa la rotación del segundo marco de referencia con respecto al primero. Esta rotación no afecta a la magnitud del vector (Abaunza, Castillo, and Lozano 2018).

2.3.5 Modelo dinámico del cuadricóptero

Si se considera que el conjunto de los componentes mecánicos de un cuadricóptero es simétrico, y algunos efectos como el aleteo de las palas y la desalineación en los ejes de los motores pueden considerarse lo suficientemente pequeños, entonces podemos suponer que las fuerzas y torques que actúan sobre el cuadricóptero son sólo las ilustradas en la Figura 2.3 (Abaunza, Castillo, and Lozano 2018).

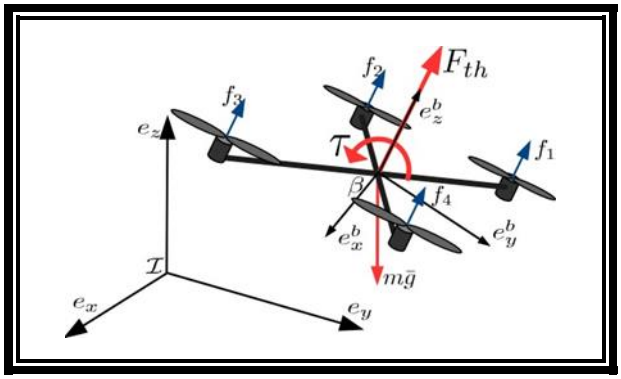


Figura 2.3 Cuerpo libre del cuadricóptero (Abaunza, Castillo, and Lozano 2018).

Aquí, se consideran dos marcos de referencia. $I = \left[e_x\ e_y\ e_z\right]^T$ define las coordenadas inerciales fijas, y $\beta = \left[e_x^b\ e_y^b\ e_z^b\right]^T$ representa el marco del cuerpo móvil situado en el centro de masa del vehículo, $F_{th} = [\,0\ \ 0\ \ \sum_{i=1}^{3} fi]^T$ simboliza la fuerza total de empuje generada por las hélices en rotación, y el torque total viene dado por

$$\tau = \begin{bmatrix} l(f_1 + f_4 - f_2 - f_3) \\ l(f_1 + f_2 - f_3 - f_4) \\ \sum_{i=1}^{4} (-1)^{i+1}\ \tau_i \end{bmatrix}, \tag{2.12}$$

Donde l define la distancia perpendicular entre motores y ejes, e_x^b o e_y^b. Obsérvese que tanto F_{th} como τ actúan sobre β, pero aplicando una rotación de quaternions con (2.11), es fácil cambiar su marco de referencia a I. Así, el modelo dinámico del cuadricóptero se expresa como

$$\dot{x}_{cuad} = \frac{d}{dt}\begin{bmatrix} \xi \\ \dot{\xi} \\ \boldsymbol{q} \\ \omega \end{bmatrix} = \begin{bmatrix} \dot{\xi} \\ \boldsymbol{q} \otimes \dfrac{F_{th}}{m} \otimes \boldsymbol{q}^* + \bar{g} \\ \dfrac{1}{2}\boldsymbol{q} \otimes \omega \\ J^{-1}(\tau - \omega \times J\omega) \end{bmatrix}, \tag{2.13}$$

Donde $\bar{g} = [\,0\;\;0\;\;g\,]^T$ corresponde al vector de gravedad. Definiendo $F_{th}^I = \boldsymbol{q} \otimes F_{th} \otimes \boldsymbol{q}^* + m\bar{g}$, el modelo puede ser representado usando la notación eje-ángulo (Carino, Abaunza, and Castillo 2015)

$$\dot{x}'_{cuad} = \frac{d}{dt}\begin{bmatrix}\xi \\ \dot{\xi} \\ \bar{\alpha} \\ \dot{\bar{\alpha}}\end{bmatrix} = \begin{bmatrix}\dot{\xi} \\ m^{-1}F_{th}^I \\ \dot{\bar{\alpha}} \\ J^{-1}(\tau - \dot{\bar{\alpha}} \times J\dot{\bar{\alpha}})\end{bmatrix}. \tag{2.14}$$

En (2.13) y (2.14) se observa que las dinámicas de rotación y traslación del cuadricóptero están acopladas, debido a la orientación de F_{th}^I que depende de la altitud $\boldsymbol{q}$ del cuadricóptero. Sin embargo, utilizando un enfoque apropiado y algunas propiedades de los quaternions unitarios, el cuadricóptero puede ser fácilmente estabilizado a pesar de su naturaleza subactuada (Abaunza, Castillo, and Lozano 2018).

2.3.6 Conversión de quaternions a representación eje-ángulo

La notación eje-ángulo es algo intuitiva y similar a los quaternions, pero en 3D. Se puede convertir de eje-ángulo a quaternions y viceversa. La ecuación (2.15) convierte el vector eje-ángulo en un quaternion (Yousuf et al. 2015).

$$\begin{aligned} q1 &= \sin\left(\frac{\alpha}{2}\right)\cos\left(\beta_x\right) \\ q2 &= \sin\left(\frac{\alpha}{2}\right)\cos\left(\beta_y\right) \\ q3 &= \sin\left(\frac{\alpha}{2}\right)\cos\left(\beta_z\right) \\ q4 &= \cos\left(\frac{\alpha}{2}\right) \end{aligned} \tag{2.15}$$

El quaternion es normalizado aplicando la ecuación (2.16).

$$q1^2 + q2^2 + q3^2 + q4^2 = 1 \tag{2.16}$$

Finalmente, la ecuación (2.17) convierte el quaternion a un vector eje-ángulo.

$$\begin{aligned} \alpha &= 2\cos^{-1}(q4) \\ x &= \frac{q1}{\sqrt{1-q4^2}} \end{aligned} \tag{2.17}$$

$$y = \frac{q2}{\sqrt{1 - q4^2}}$$
$$z = \frac{q3}{\sqrt{1 - q4^2}}$$

2.4 COMPONENTES DE UN CUADRICÓPTERO

Los principales componentes para el vuelo de un cuadricóptero son el cuerpo, los motores y las hélices. El cuerpo contiene los circuitos de control del motor, del vuelo, los circuitos de comunicación y la batería. En vuelo, el cuadricóptero informa su estado e información de vuelo al dispositivo de control en tierra utilizando la telemetría. En el caso del cuadricóptero, la telemetría es la transmisión inalámbrica de varios parámetros, como el estado de la batería y la posición y orientación de la nave. En la nave, hay un conjunto de unidades de medición llamadas sensores que miden los parámetros que son codificados y transmitidos al dispositivo de control en tierra (Fairchild and Harman 2016).

2.4.1 Comprensión de los sensores del cuadricóptero

El circuito del controlador de vuelo a bordo recibe información de los sensores que proporcionan datos sobre la nave en vuelo. Algunos de los posibles sensores que determinan la posición, la altitud y la dirección de vuelo son (Fairchild and Harman 2016):

- Un giroscopio que determina la posición de la nave incluyendo su pitch y roll. Esto indica el movimiento de rotación de la nave.

- Un acelerómetro que determina la tasa de cambio de velocidad de la nave con respecto a los tres ejes.

- Un altímetro o barómetro que determina la altitud de la nave sobre el suelo. A bajas altitudes, se puede utilizar un sensor de sonar descendente para determinar altitudes de varios metros.

- Un magnetómetro que sirve de brújula para indicar la dirección de la nave utilizando el campo magnético de la Tierra como referencia.

El acelerómetro y el magnetómetro necesitan ser calibrados para inicializar sus lecturas a las condiciones en las que tendrán lugar los vuelos. Por lo tanto, para cada cuadricóptero es importante seguir cuidadosamente las instrucciones del fabricante para configurar la nave antes de los vuelos (Fairchild and Harman 2016).

2.4.2 Unidad de Medición Inercial

La Unidad de Medición Inercial (IMU por sus siglas en ingles *Inertial Measurement Unit*) es una combinación de giroscopio y acelerómetro. Esta unidad indicará la información completa sobre las características de vuelo del cuadricóptero. Normalmente, la unidad medirá la aceleración y la orientación del dron en las tres dimensiones. Estos sensores permiten el vuelo en interiores y exteriores. Sin embargo, todos los sensores antes mencionados sufren ligeros errores que pueden acumularse durante el vuelo, por lo que es necesario tener precaución al volar en espacios reducidos (Fairchild and Harman 2016).

2.4.3 Tiempo de vuelo

La velocidad de vuelo estándar para los drones en interiores es de 5-10 $\frac{m}{s}$ para permitir una adecuada toma de muestras y su duración de vuelo suele ser de 15-30 minutos en drones operados con baterías, mientras que los drones de alas fijas necesitan velocidades de vuelo de 80 $\frac{km}{h}$ o más. Los drones con híbridos de gasolina pueden aumentar significativamente su tiempo de vuelo hasta 2-3 horas, pero requieren una planificación cuidadosa en términos de reserva de espacio aéreo y pronóstico del tiempo. Los drones en interiores tienen velocidades de vuelo más bajas que los drones al aire libre debido a los desafíos adicionales de los espacios cerrados, pero algunos drones de última generación son capaces de evitar obstáculos y maniobrar con precisión en espacios reducidos (Nex et al. 2022).

2.5 LOCALIZACIÓN

La navegación por odometría es propensa a errores y sólo puede dar una estimación de la postura real del robot. Además, cuanto más se mueva el robot, mayor será el error en la estimación de la postura, especialmente en el rumbo. La odometría en un robot puede ser

comparada con caminar con los ojos cerrados, contando los pasos hasta llegar a nuestro destino. Al igual que con la odometría, cuanto más caminemos, más incertidumbre tendremos sobre nuestra ubicación. Incluso cuando contamos los pasos, debemos abrir los ojos de vez en cuando para reducir la incertidumbre de nuestra ubicación. Para un robot, ¿qué significa "contar pasos" y "abrir los ojos de vez en cuando"? Significa que para moverse en distancias cortas la odometría es buena, pero cuando se mueve en distancias más largas, el robot debe determinar su posición en relación con una referencia externa llamada punto de referencia. Este proceso se llama localización (Mordechai and Francesco 2018).

2.5.1 Determinación de la posición a partir de un ángulo y una distancia

La Figura 2.4 muestra la geometría de un robot respecto a un objeto. En el diagrama, el objeto está representado por el punto grande situado en el origen (x_0, y_0) de un sistema de coordenadas. El *azimuth* del robot θ es el ángulo entre el norte y la dirección de avance del robot; puede medirse con una brújula. Se utiliza un escáner láser para medir la distancia s al objeto y el ángulo ϕ entre la dirección de avance del robot y el objeto (Mordechai and Francesco 2018). Las coordenadas relativas Δx y Δy pueden determinarse por simple trigonometría:

$$\Delta x = s\,sin(\theta - \varphi), \Delta y = s\,cos(\theta - \varphi). \tag{2.18}$$

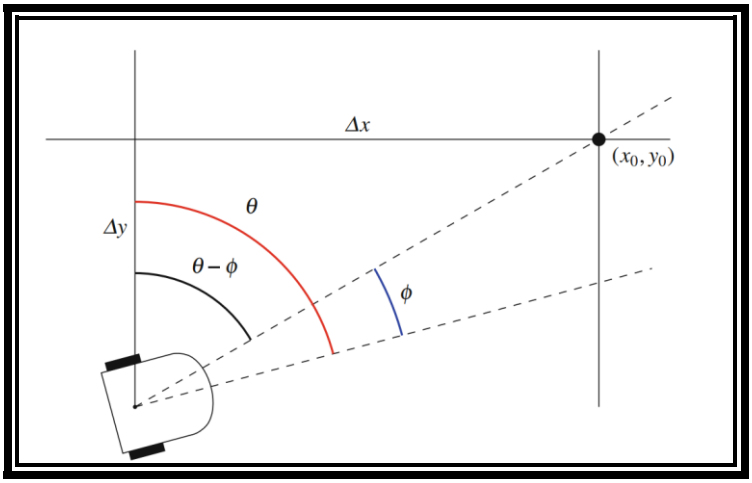


Figura 2.4 Determinación de la posición a partir de un ángulo y una distancia (Mordechai and Francesco 2018).

2.5.2 Determinación de la posición por triangulación

La triangulación se utiliza para determinar las coordenadas cuando es difícil o imposible medir las distancias. Se utilizaba mucho en la topografía antes de que existieran los láseres, porque era imposible medir distancias largas con precisión. El principio de la triangulación es que a partir de dos ángulos de un triángulo y la longitud del lado correspondiente se pueden calcular las longitudes de los otros lados. Una vez conocidas éstas, se puede calcular la posición relativa de un objeto lejano (Mordechai and Francesco 2018).

La Figura 2.5 muestra al robot midiendo los ángulos α y β del objeto desde dos posiciones separadas por una distancia c. la distancia puede medirse mediante odometría a medida que el robot se desplaza de una posición a otra, aunque esto puede ser menos preciso (Mordechai and Francesco 2018).

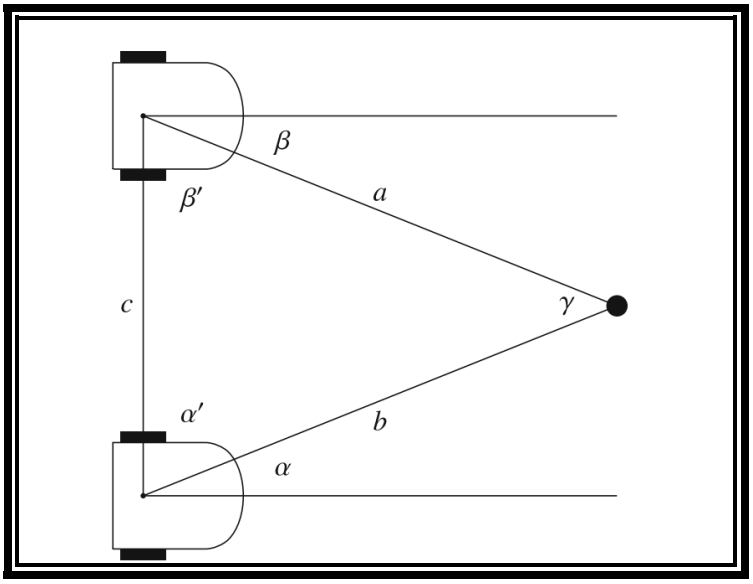


Figura 2.5 Triangulación (Mordechai and Francesco 2018).

Las longitudes a y b se calculan mediante la ley de los senos:

$$\frac{a}{sin\alpha'} = \frac{b}{sin\beta'} = \frac{c}{sin\gamma}, \tag{2.19}$$

Donde $\alpha' = 90^\circ - \alpha$, $\beta = 90^\circ - \beta$ son los ángulos interiores del triángulo. Para utilizar la ley, necesitamos *c*, que se ha medido, y γ que es:

$$\gamma = 180^\circ - \alpha' - \beta' = 180^\circ - (90^\circ - \alpha) - (90^\circ - \beta) = \alpha + \beta\,. \tag{2.20}$$

De la ley de senos:

$$b = \frac{csin\beta'}{sin\gamma} = \frac{csin(90^\circ - \beta)}{sin(\alpha + \beta)} = \frac{c\cos\beta}{sin(\alpha + \beta)}. \tag{2.21}$$

Un cálculo similar da como resultado *a* (Mordechai and Francesco 2018)

2.6 MAPAS

Un robot puede utilizar su capacidad de detectar obstáculos para localizarse, basándose en la información sobre la posición de los obstáculos u otra información sobre el

entorno. Esta información la proporciona normalmente un mapa (Mordechai and Francesco 2018).

2.6.1 Mapas discretos y continuos

En robótica, existen dos técnicas para almacenar mapas en memoria, los mapas discretos (también llamados mapas de cuadrícula) y los mapas continuos (Mordechai and Francesco 2018).

La Figura 2.6a muestra un mapa cuadriculado de 8 × 8 con un objeto triangular. La ubicación del objeto se almacena como una lista de las coordenadas de cada celda de la cuadrícula cubierta por el objeto (Mordechai and Francesco 2018).

El objeto de la figura está formado por las celdas:

(5, 3), (5, 4), (5, 5), (4, 5), (5, 6), (4, 6), (3, 6).

La Figura 2.6b muestra un mapa continuo del mismo objeto. En lugar de almacenar las posiciones del objeto, se almacenan las coordenadas de las posiciones de los límites:

A = (6, 3), B = (3, 7), C = (6, 7).

Los mapas discretos no son muy precisos: es difícil reconocer el objeto de la Figura 2.6a como un triángulo. Para mejorar la precisión, hay que utilizar una cuadrícula más fina: 16 × 16 o incluso 256 × 256. Por supuesto, a medida que aumenta el número de puntos de la cuadrícula, también debe aumentar el tamaño de la memoria del robot. Además, hay que utilizar un ordenador más potente para procesar las celdas de la cuadrícula (Mordechai and Francesco 2018).

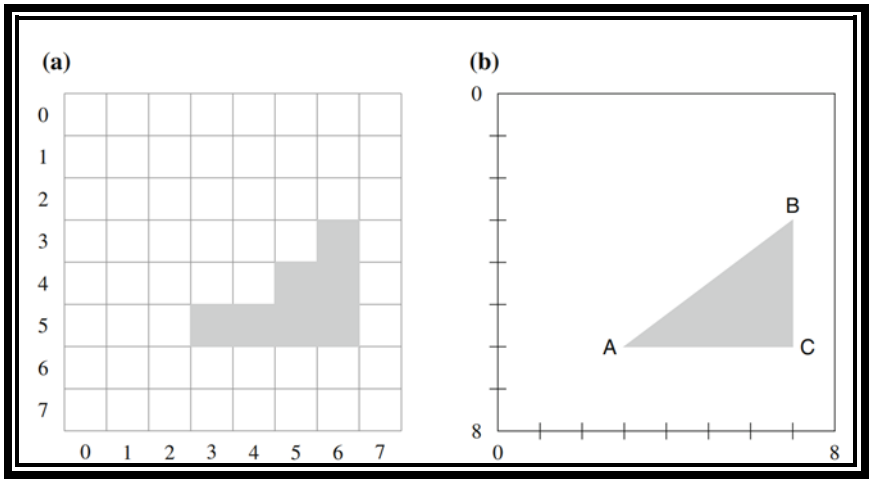


Figura 2.6 **(a)** Un mapa discreto de las celdas ocupadas de un objeto, (**b**) Un mapa continuo del mismo objeto (Mordechai and Francesco 2018).

Los robots móviles tienen limitaciones de peso, coste, capacidad de la batería, etc., por lo que, las cuadrículas muy finas pueden no ser prácticas (Mordechai and Francesco 2018).

Si los objetos del entorno son pocos y tienen una forma sencilla, un mapa continuo es más eficiente, además de ser más preciso. En la Figura 2.6b, tres pares de números representan el triángulo con mucha más precisión que los siete pares del mapa discreto. Además, es fácil calcular si un punto está en el objeto o no utilizando geometría analítica. Sin embargo, si hay muchos objetos o si éstos tienen formas muy complejas, los mapas continuos ya no son eficientes ni en memoria ni en la cantidad de cómputo necesario. El objeto de la Figura 2.6b está limitado por líneas rectas, pero si los límites se describen con curvas de alto orden, el cálculo se volvería difícil. Considerando un mapa con 32 objetos de tamaño uno, ninguno de los cuales se toca. El mapa discreto tendría 32 coordenadas, mientras que el mapa continuo necesitaría almacenar las coordenadas de las cuatro esquinas de cada objeto (Mordechai and Francesco 2018).

En robótica móvil, los mapas discretos se utilizan comúnmente para representar mapas del entorno (Mordechai and Francesco 2018).

2.6.2 El contenido de las celdas de un mapa discreto

En un mapa discreto cada celda almacena un número y tenemos que decidir qué codifica un número (Mordechai and Francesco 2018).

La codificación más sencilla consiste en asignar un bit a cada celda. Un valor de 1 indica que existe un objeto en esa celda y un valor de 0 indica que la celda está vacía. En la Figura 2.6a, el gris representa el valor 1 y el blanco el valor 0 (Mordechai and Francesco 2018).

Sin embargo, los sensores no son precisos y puede ser difícil saber si una celda está ocupada por un objeto o no. Por lo tanto, tiene sentido asignar una probabilidad a cada celda indicando la certeza de que el objeto está en esa celda. La Figura 2.7 es una copia de la Figura 2.6a con las probabilidades de cada casilla. Las celdas sin número tienen una probabilidad de 0 (Mordechai and Francesco 2018).

Se puede observar que las celdas con una probabilidad de al menos 0,7 son las que se consideran en la Figura 2.6a como celdas ocupadas por el objeto. Por supuesto, somos libres de elegir otro umbral, por ejemplo 0,5, en cuyo caso se considera que hay más celdas ocupadas. En este ejemplo, sabemos que el objeto es triangular, por lo que podemos ver que un umbral de 0,5 hace que el objeto sea más grande de lo que realmente es, mientras que el umbral 0,7 da una mejor aproximación (Mordechai and Francesco 2018).

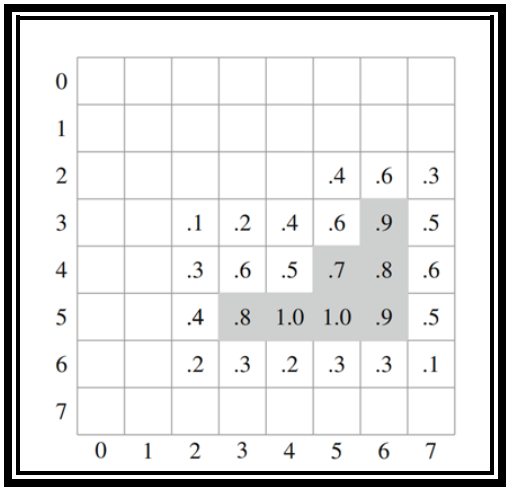


Figura 2.7 Mapa probabilístico cuadriculado (Mordechai and Francesco 2018)

2.7 INTERPOLACIÓN

En modelación geométrica, así como también en otras aplicaciones con frecuencia hay que encontrar expresiones analíticas, usualmente de curvas de las cuales no se conoce una descripción matemática o ésta es muy complicada. En este caso se mide o se evalúa la curva en un conjunto de puntos y se construye una aproximación o interpolación. Este capítulo describe algunas de las técnicas (Paluszny, Prautzsch, and Boehm 2002).

2.7.1 Curvas de Bézier

Las curvas de Bézier fueron inventadas en 1962 por el ingeniero francés Pierre Bézier para diseñar carrocerías de automóviles. Hoy en día, las curvas de Bézier se utilizan ampliamente en los gráficos por ordenador y la animación (Sederberg 2012). Una curva de Bézier de grado *n* puede representarse como

$$P_{[t_0,t_1]}(t) = \sum_{i=0}^{n} B_i^n(t) P_i \tag{2.22}$$

Donde P_i son puntos de control tal que $P(t_0) = P_0$ y $P(t_1) = P_n$, $B_i^n(t)$ es un polinomio de Bernstein dado por

$$B_i^n(t) = \binom{n}{i} \left(\frac{t_1 - t}{t_1 - t_0}\right)^{n-i} \left(\frac{t - t_0}{t_1 - t_0}\right)^{i}, \qquad i \in \{0,1,\dots,n\} \tag{2.23}$$

Las curvas de Bézier tienen propiedades útiles para la planificación de trayectorias (Choi, Curry, and Elkaim 2008):

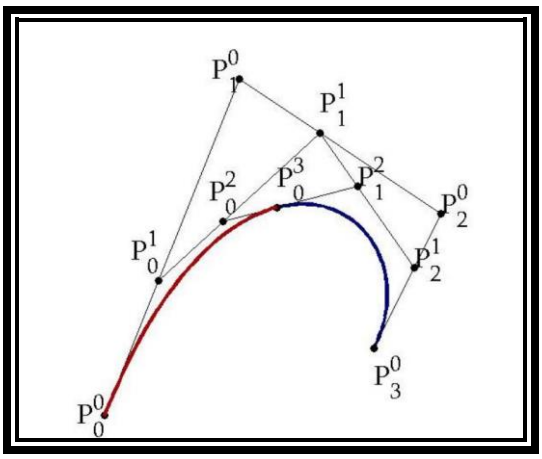


Figura 2.8 Subdivisión de una curva cúbica de Bézier con $\tau = .4$ por el Algoritmo de Casteljau (Choi, Curry, and Elkaim 2008).

1. Siempre pasan por P_0 y P_n.
2. Siempre son tangentes a las líneas que conectan $P_0 \rightarrow P_1$ y $P_n \rightarrow P_{n-1}$ en P_0 y P_n respectivamente.
3. Siempre se encuentran dentro del casco convexo formado por sus puntos de control.

2.7.2 B-splines

Los splines son curvas polinómicas por trozos continuamente diferenciables hasta un orden prescrito (Paluszny, Prautzsch, and Boehm 2002). El ejemplo más sencillo es el spline C^0, o sea, lineal por trozos. Este spline es simplemente una poligonal en el plano o en el espacio. Otro ejemplo son los splines cúbicos C^1 y C^2 construidos en la Figura 2.9.

La herramienta de los splines se desarrolla para solventar las limitaciones de las curvas de Bézier: falta de control local, la laboriosidad requerida para imponer continuidad C^2 y el hecho de que el número de puntos de control de una curva de Bézier impone su grado (Paluszny, Prautzsch, and Boehm 2002).

En analogía a la representación de Bézier de curvas polinómicas también es conveniente expresar un spline $s(u)$ como una combinación afín de ciertos puntos de control $\boldsymbol{c_i}$, estos son:

$$s(u) = \sum \boldsymbol{c_i} N_i^n(u) \tag{2.24}$$

Donde los $N_i^n(u)$ son funciones polinómicas por trozos con soporte finito (se anulan fuera de un intervalo finito) y satisfacen ciertas condiciones de continuidad. Shoenberg introdujo el nombre de B-splines para estas funciones (Schoenberg 1967).

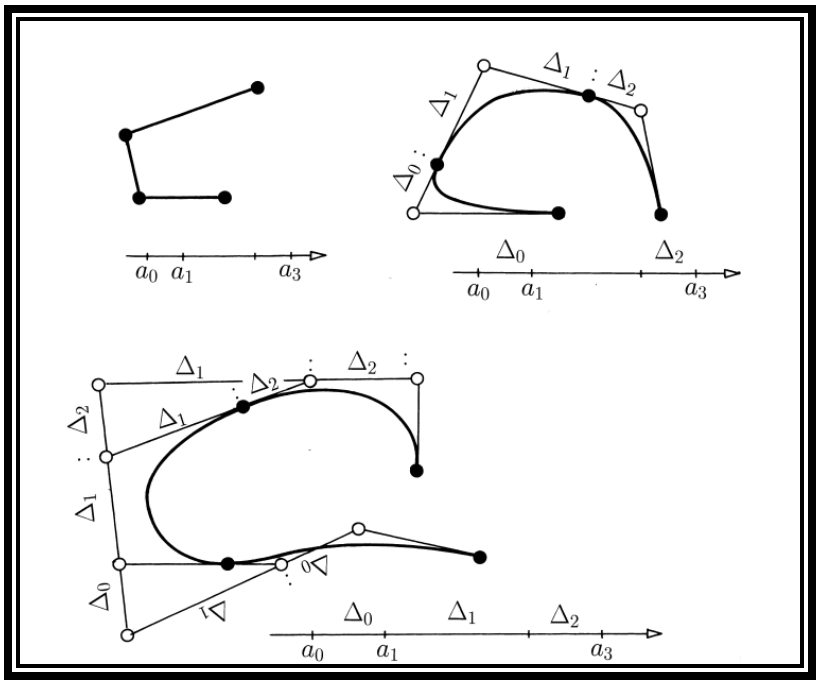


Figura 2.9 Splines paramétricos de grado 1, 2 y 3(Paluszny, Prautzsch, and Boehm 2002).

2.7.3 El Algoritmo de De Boor

El Algoritmo de De Boor proporciona un método para evaluar una curva B-spline, es decir, encontrar el punto en la curva B-spline correspondiente a un valor de parámetro dado. Cada punto en una curva B-spline $P(t)$ tiene una representación polar $P(t, t, \ldots, t)$, que se puede encontrar insertando el valor del nodo t n veces. El algoritmo de De Boor se utiliza para calcular esta representación polar y, a partir de ella, obtener el punto correspondiente en

la curva B-spline. La Figura 2.10 ilustra el proceso del Algoritmo de De Boor (De Boor 1972).

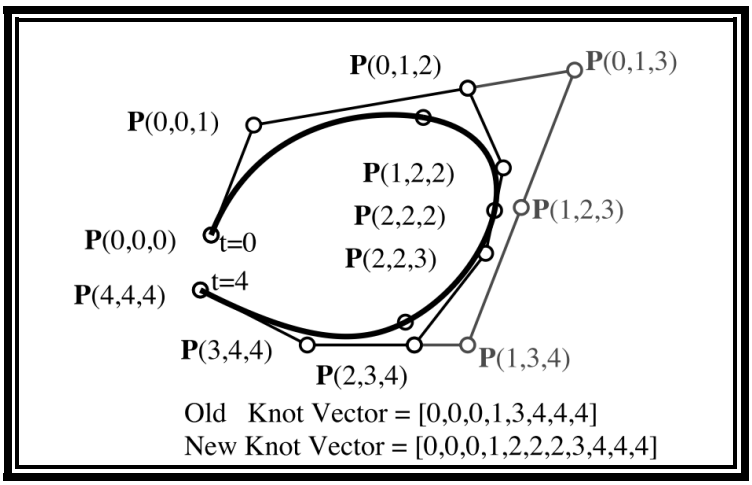


Figura 2.10 Algoritmo de De Boor (Sederberg 2012).

2.8 APRENDIZAJE POR REFUERZO

El aprendizaje por refuerzo es el problema al que se enfrenta un agente que debe aprender un comportamiento a través de interacciones de ensayo y error con un entorno dinámico (Littman, Kaelbling, and Moore 1996).

2.8.1 Exploración y explotación de la recompensa

La exploración y la explotación representan un dilema central en el aprendizaje por refuerzo. La exploración se refiere a la toma de acciones que el agente desconoce o ha explorado poco, con el propósito principal de aprender sobre el entorno. Esto es fundamental para descubrir las acciones óptimas y adaptarse a entornos cambiantes. Sin embargo, la exploración, conlleva el riesgo de obtener recompensas inmediatas más bajas debido a la toma de acciones subóptimas (Sewak 2019).

Por otro lado, la explotación se centra en seleccionar acciones que el agente cree que son las mejores según su conocimiento actual. Esto busca maximizar recompensas inmediatas y es eficiente en situaciones donde el agente tiene información precisa. Sin

embargo, una excesiva explotación puede llevar al estancamiento y a decisiones subóptimas si el conocimiento inicial es inexacto (Sewak 2019).

Los algoritmos de aprendizaje por refuerzo buscan un equilibro entre exploración y explotación que se adapte en el tiempo. La elección de algoritmos como ϵ-*greedy* se presenta como una solución que permite controlar este equilibrio mediante un valor constante de ϵ. La determinación del valor óptimo de ϵ se basa en la naturaleza del proceso de decisión de Markov subyacente, con entornos más deterministas favoreciendo una menor exploración. La capacidad de adaptar este equilibrio a lo largo del tiempo se convierte en una estrategia esencial para lograr un aprendizaje y toma de decisiones efectivos en el aprendizaje por refuerzo (Sutton and Barto 2018).

2.8.2 Recompensa retardada

En el caso general del problema de aprendizaje por refuerzo, las acciones del agente determinan no sólo su recompensa inmediata, sino también (al menos probabilísticamente) el siguiente estado del entorno. Estos entornos pueden considerarse como redes de problemas *bandit*, pero el agente debe tener en cuenta el siguiente estado, así como la recompensa inmediata, cuando decida qué acción tomar. El modelo de optimización a largo plazo que utiliza el agente determina exactamente cómo debe tener en cuenta el valor del futuro. El agente tendrá que ser capaz de aprender del refuerzo retardado: puede tomar una larga secuencia de acciones, recibiendo un refuerzo insignificante y, finalmente, llegar a un estado con un refuerzo elevado. El agente debe ser capaz de aprender cuáles de sus acciones son deseables basándose en una recompensa que puede tener lugar arbitrariamente en el futuro. Los problemas con refuerzo retardado se modelan bien como Procesos de Decisión de Markov (PDM) (Littman, Kaelbling, and Moore 1996).

2.8.3 Procesos de decisión de Markov

Los PDM pretenden ser un planteamiento sencillo del problema de aprender de la interacción para conseguir un objetivo. El que aprende y toma las decisiones se llama agente. Lo que está en interacción con el agente, es decir, todo lo que está fuera de él, se llama

entorno. Estos interactúan continuamente, el agente seleccionando acciones y el entorno respondiendo a estas acciones y presentando nuevas situaciones al agente. El entorno también da lugar a recompensas, valores numéricos especiales que el agente trata de maximizar en el tiempo a través de su elección de acciones (Sutton and Barto 2018).

Más concretamente, el agente y el entorno interactúan en cada una de las secuencias de pasos temporales discretos, $t = 0, 1, 2, 3, \ldots$. En cada paso de tiempo t, el agente recibe alguna representación del *estado* del entorno $S_t \in \mathcal{S}$, y sobre esa base selecciona una *acción*, $At \in \mathcal{A}(s)$. Un paso de tiempo después, en parte como consecuencia de su acción, el agente recibe una recompensa numérica recompensa, $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$, y se encuentra en un nuevo estado, S_{t+1} (Sutton and Barto 2018).

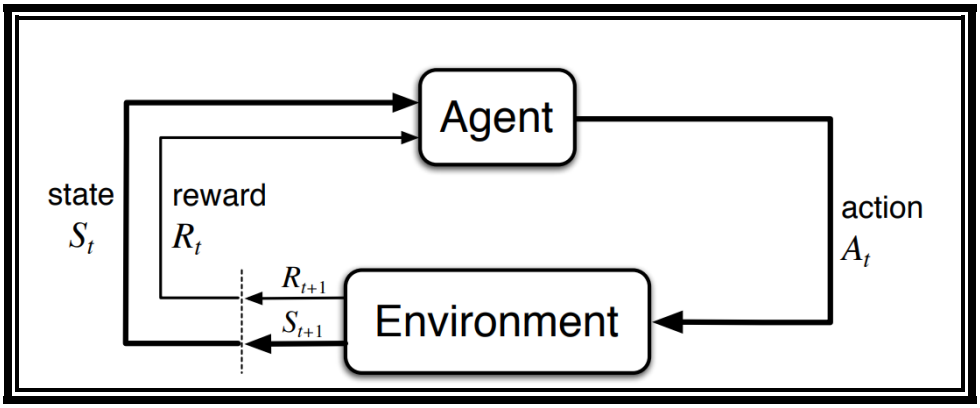


Figura 2.11 La interacción agente-entorno en un proceso de decisión de Markov (Sutton and Barto 2018).

2.8.4 Políticas y funciones de valor

Casi todos los algoritmos de aprendizaje por refuerzo implican la estimación de *funciones de valor* que estiman *que tan bueno* es para el agente estar en un estado determinado (o que tan bueno es realizar una acción determinada en un estado determinado). La noción de "qué tan bueno" se define aquí en términos de las recompensas futuras que se pueden esperar, o, para ser precisos, en términos de rendimiento esperado. Por supuesto, las recompensas que el agente puede esperar recibir en el futuro dependen de las acciones que

realice. En consecuencia, las funciones de valor se definen con respecto a determinadas formas de actuar, denominadas políticas (Sutton and Barto 2018).

Formalmente, una política es un mapeo de estados a probabilidades de seleccionar cada acción posible. Si el agente sigue la política π en el instante t, entonces $\pi(a|s)$ es la probabilidad de que la acción $At = a$ sí el estado $St = s$. Al igual que la probabilidad p, la política π es una función ordinaria; el "|" en medio de $\pi(a|s)$ sólo recuerda que define una distribución de probabilidad sobre la acción $a \in \mathcal{A}(s)$ para cada estado $s \in \mathcal{S}$. Los métodos de aprendizaje por refuerzo especifican cómo se modifica la política del agente como resultado de su experiencia (Sutton and Barto 2018).

2.8.5 Aprendizaje-Q

Es una técnica de Aprendizaje por refuerzo introducida a finales de la década de los ochenta por Watkins (Watkins 1989). Está basada en diferencias temporales (TD) que se definen como:

La diferencia temporal en el momento *t*, denotada por $TD_t(s_t, a_t)$, es la diferencia entre:

1. $R(s_t, a_t) + \gamma \max_a \left(Q\left(s_{t+1,} a\right)\right)$, es la recompensa $R(s_t, a_t)$ obtenida al realizar la acción a_t en el estado s_t, mas el valor-Q de la mejor acción realizada en el futuro estado s_{t+1}, descontado por un factor $\gamma \in [0,1]$, llamada el factor de descuento.
2. $Q(s_t, a_t)$, es el valor-Q de la acción a_t realizada en el estado s_t.

Esto lleva a:

$$TD_t(s_t, a_t) = R(s_t, a_t) + \gamma \max_a\big(Q(s_{t+1}, a)\big) - Q(s_t, a_t) \tag{2.25}$$

En términos generales Aprendizaje-Q busca estimar una función de valor-acción conocida como $Q(s, a)$, donde $Q(s, a)$ es "el valor Q de la acción realizada en el estado *s*". la cual representa el valor esperado de la recompensa acumulada para ese par de estado-acción (s, a)

$$Q: (s \epsilon S, a \epsilon A) \rightarrow Q(s,a) \epsilon \boldsymbol{R} \tag{2.26}$$

, de esta manera, si se conoce el valor de la recompensa para cada par estado-acción, es posible maximizar la recompensa del agente siguiendo la serie de acciones que maximizan la función Q. Para esto, Aprendizaje-Q actualiza los valores Q de la siguiente manera con la ecuación de Bellman:

$$Q_t(s_t, a_t) = \mathrm{Q}_{\mathrm{t}-1}(s_\mathrm{t}, a_\mathrm{t}) + \alpha \mathrm{TD}_\mathrm{t}(s_\mathrm{t}, a_\mathrm{t}) \tag{2.27}$$

Donde:

- r es la recompensa percibida por tomar la acción A_t en el estado S_t.
- $\propto$ es la tasa de aprendizaje (importancia que se le da a cada actualización).
- γ es la tasa de descuento (equilibrio entre recibir recompensas inmediatas o a largo plazo).

Adicionalmente, dado que Aprendizaje-Q aproxima directamente la función de valor, independiente de la política (π) que se esté usando, este se conoce como una técnica de control *off-policy*. Esto permite utilizar acciones exploratorias a través de políticas $\epsilon - greedy$. De esta manera, se puede ajustar el parámetro ϵ para determinar la probabilidad de que el agente explore el entorno (tomando una acción aleatoria) o explote (sea *greedy* con respecto a) la función Q estimada.

2.8.6 Algoritmo muestreo de Thompson

El algoritmo de muestreo de Thompson, también conocido como muestreo posterior y ajuste de probabilidades, fue propuesto por primera vez en 1933 por Thompson (Thompson 1933, 1935). Este algoritmo se utiliza en problemas de toma de decisiones en tiempo real, como el aprendizaje por refuerzo, donde se deben equilibrar la explotación del conocimiento disponible para maximizar el rendimiento inmediato y la exploración para adquirir nueva información que pueda mejorar el rendimiento futuro (Russo et al. 2018).

El algoritmo muestreo de Thompson utiliza una distribución de probabilidad para modelar la incertidumbre sobre las posibles recompensas de cada opción y selecciona la opción con la muestra más alta en cada iteración. Con el tiempo, las distribuciones de probabilidad se ajustan y se hace una toma de decisiones más precisa a medida que se recopila más información.

La distribución Beta (Evans, Hastings, and Peacock 2000) es comúnmente utilizada en el contexto del algoritmo muestreo de Thompson y se define por dos parámetros, α y β. En cada iteración, la distribución Beta se actualiza para reflejar la nueva información recopilada sobre las recompensas de cada opción, lo que permite una toma de decisiones más precisa a medida que se ajusta (Ponteves 2019).

El Algoritmo de muestreo de Thompson sigue los pasos descritos a continuación y mostrados en la Tabla 2.1.

1. En cada iteración t, se procede a muestrear una estimación de los parámetros del modelo, representados por la variable aleatoria θˆ, de la distribución de probabilidad p.
2. Luego, se selecciona la acción x_t que maximiza la esperanza condicional de la recompensa $r(y_t)$ dada la acción x_t y la estimación θˆ, es decir, xt ← $\text{argmax}_{x\in\chi}\, \mathbb{E}_{q\theta\hat{}}[\text{r(yt)}|\text{xt} = \text{x}]$.
3. A continuación, se aplica la acción x_t y se observa la recompensa $r(y_t)$.
4. Finalmente, se actualiza la distribución de probabilidad p, que representa la incertidumbre sobre los parámetros del modelo, utilizando la regla de Bayes con la nueva observación y_t y la acción tomada x_t. La distribución resultante es utilizada en la próxima iteración para muestrear una nueva estimación de los parámetros del modelo.

Algoritmo muestreo de Thompson(χ, p, q, r) (Russo et al. 2018)
1: **for** $t = 1, 2, \ldots$ ***do***

2:	Sample $\hat{\theta} \sim p$	#muestrear modelo:
3:	xt $\leftarrow$ argmax$_{x \in \chi}$ $\mathbb{E}_{q\theta}$[r(yt)|xt = x]	#selecciona y aplica acción:
	Apply xt and observe yt	
4:	$p \leftarrow \mathbb{P}_{p,q}(\theta \in \cdot \mid xt, yt)$	#actualiza distribución:
5:	**end for**	

Tabla 2.1 Algoritmo muestreo de Thompson

2.9 DISTRIBUCIÓN GAUSSIANA

La forma más simple de la distribución gaussiana es la distribución gaussiana unidimensional estándar (Figura 2.12), que puede ser descrita mediante la función de densidad de probabilidad (pdf):

$$p(x) = \phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{2.28}$$

donde $\frac{1}{\sqrt{2\pi}}$ asegura la normalización, es decir, $\int \mathbb{R}\, p(x)dx = 1$. Esta distribución se centra alrededor de $x = 0$ y la tasa de decaimiento o "ancho" de la curva es 1.

De manera más general, podemos aplicar traslaciones y escalas para obtener una distribución gaussiana que se centre en cualquier $\mu \in R$ y con cualquier ancho $\sigma > 0$. La función de densidad de probabilidad (pdf) es:

$$p(x) = \frac{1}{\sigma}\phi\left(\frac{x - \mu}{\sigma}\right) = \frac{1}{\sqrt{2\pi}\sigma} exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \tag{2.29}$$

Técnicamente, μ se llama media y σ^2 se llama varianza, μ es el pico/moda de la densidad, y también es la media y la mediana de la distribución debido a la simetría de la densidad alrededor de μ. Si una variable aleatoria $\mathcal{X}$ tiene esta densidad, entonces:

$$\mathcal{X} \sim \mathcal{N}(\mu, \sigma^2) \tag{2.30}$$

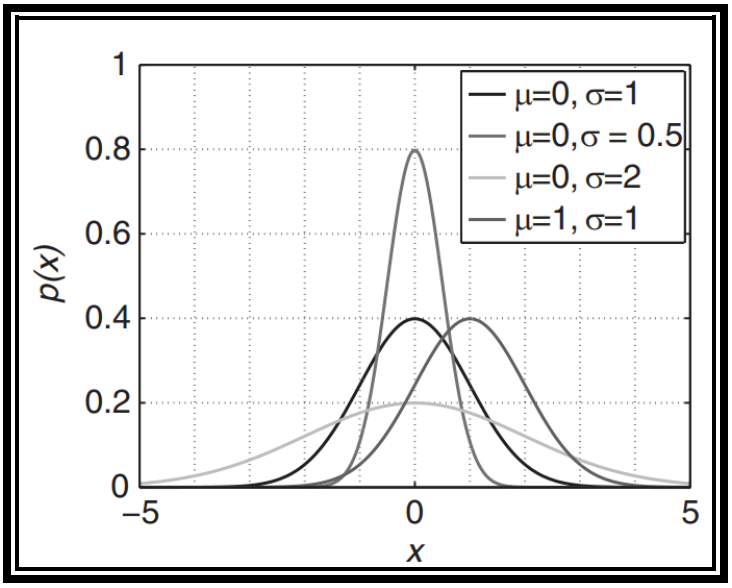


Figura 2.12 Distribución Gaussiana (Sammut and Geoffrey I. 2010).

2.10 MÉTRICAS DE ERROR

En esta sección, se abordan métricas para evaluar el error de las predicciones generadas por un modelo en comparación con los valores reales. Estas métricas son utilizadas para cuantificar la precisión de las predicciones generadas por un modelo en comparación con los valores reales.

2.10.1 Porcentaje del error

El porcentaje de error se utiliza para medir la precisión del modelo en términos de cuanto se desvían sus predicciones de los valores reales. Se evalúa aplicándolo a datos de prueba para los cuales se conocen las etiquetas de clase (valores Y). Este indicador se obtiene midiendo la discrepancia entre las predicciones del modelo y los valores reales, expresando esta diferencia como una proporción con respecto a la magnitud del valor real. La fórmula que se emplea para calcular el porcentaje de error se expresa de la siguiente manera:

$$\%error = \frac{|\lambda(x) - y|}{|y|} \times 100 \tag{2.31}$$

En esta fórmula, 'y' representa el valor objetivo real para la instancia de prueba, mientras que '$\lambda(x)$' representa la estimación objetivo generada por el modelo para la misma instancia de prueba (Sammut and Geoffrey I. 2010).

2.10.2 Raíz del error cuadrático medio

La raíz del error cuadrático medio (RMSE por sus siglas en inglés *Root Mean Square Error*) es una métrica de evaluación del modelo que se utiliza a menudo para cuantificar la novedad del estado en el seguimiento de trayectorias de UAV's (Castillo García, Munoz Hernandez, and Gil 2017; Krukhmalev and Pshikhopov 2017). La raíz del error cuadrático medio de un modelo con respecto a un conjunto de pruebas es la raíz cuadrada de la media de los errores de predicción al cuadrado sobre todas las instancias del conjunto de pruebas. El error de predicción es la diferencia entre el valor real y el valor predicho para una instancia.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(y_i - \lambda(x_i)\right)^2}{n}} \tag{2.32}$$

donde y_i es el valor objetivo verdadero para la instancia de prueba x_i, $\lambda(x_i)$ es el valor objetivo predicho para la instancia de prueba x_i, y n es el número de instancias de prueba (Géron 2019).

2.11 HERRAMIENTAS

En esta sección, se presentan las herramientas utilizadas para realizar los experimentos de esta tesis.

2.11.1 Robot Operating System

Robot Operating System (ROS) es un *framework* que proporciona diversas herramientas y bibliotecas para escribir software de robótica. Ofrece varias funciones potentes para ayudar a los desarrolladores en tareas como el paso de mensajes, la computación distribuida, la reutilización de código y la implementación de algoritmos de última generación para aplicaciones robóticas. El proyecto ROS fue iniciado en 2007 por Morgan Quigley y su desarrollo continuó en Willow Garage, un laboratorio de investigación

robótica para el desarrollo de hardware y software de código abierto para robots. El objetivo de ROS era establecer una forma estándar de programar robots y, al mismo tiempo, ofrecer componentes de software listos para usar que pudieran integrarse fácilmente con aplicaciones robóticas personalizadas (Joseph and Cacace 2021).

Comprensión de la computación gráfica de ROS:

La computación en ROS se realiza utilizando una red de nodos ROS. Esta red de nodos se denomina grafo de cálculo. Los conceptos principales en el grafo de cálculo son:

Nodos: Los nodos son los procesos que tienen computación. Cada nodo ROS está escrito utilizando las librerías cliente ROS. Usando las APIs de las librerías cliente, podemos implementar diferentes funcionalidades de ROS, como los métodos de comunicación entre nodos, lo cual es particularmente útil cuando los diferentes nodos del robot deben intercambiar información entre ellos. Uno de los objetivos de los nodos ROS es construir procesos simples en lugar de un gran proceso con todas las funcionalidades deseadas. Al ser estructuras simples, los nodos ROS son fáciles de depurar.

Maestro: El maestro ROS proporciona los procesos de registro y búsqueda de nombres para el resto de los nodos. Los nodos no podrán encontrarse entre sí, intercambiar mensajes o invocar servicios sin un maestro ROS. En un sistema distribuido, debemos ejecutar el maestro en un ordenador; entonces, los otros nodos remotos pueden encontrarse entre sí comunicándose con este maestro.

Tópicos: Cada mensaje en ROS se transporta utilizando buses llamados tópicos. Cuando un nodo envía un mensaje a través de un tópico, podemos decir que el nodo está publicando en el tópico. Cuando un nodo recibe un mensaje a través de un tópico, entonces podemos decir que el nodo se está suscribiendo a un tópico. Cada tópico tiene un nombre único, y cualquier nodo puede subscribirse a él y publicar datos a través de él siempre que tenga el tipo de mensaje adecuado.

Bagfiles: ROS proporciona un sistema de registros para almacenar datos, de los sensores, que son necesarios para desarrollar y probar algoritmos robóticos. Estos registros

son llamados *bagfiles* y son características muy útiles cuando trabajamos con mecanismos robóticos complejos.

2.11.2 Gazebo

Gazebo es un simulador para la simulación robótica compleja en interiores y exteriores. Podemos simular robots complejos, sensores y una gran variedad de objetos 3D. Gazebo ya cuenta con modelos de simulación de robots populares, sensores y una gran variedad de objetos 3D en su repositorio. Gazebo está perfectamente integrado con ROS gracias a la interfaz ROS-Gazebo, que expone el control completo de Gazebo en ROS (Joseph and Cacace 2021).

2.11.3 RotorS

El objetivo del *framework* de simulación RotorS es simular la dinámica del UAV con sensores básicos (como IMU, GPS, etc.) y hélices. Este *framework* proporciona un conjunto de archivos de configuración y modelos en forma de paquetes ROS para simular diferentes tipos de UAV. Además de los modelos estándar, RotorS permite a los desarrolladores configurar nuevos sistemas multirrotor desde cero. Este paquete ROS implementa tanto sensores como mecanismos en forma de plugins de Gazebo que se pueden montar en el multirrotor (Furrer et al. 2016; Joseph and Cacace 2021).

El simulador RotorS está dividido en diferentes paquetes, como se muestra en la siguiente figura:

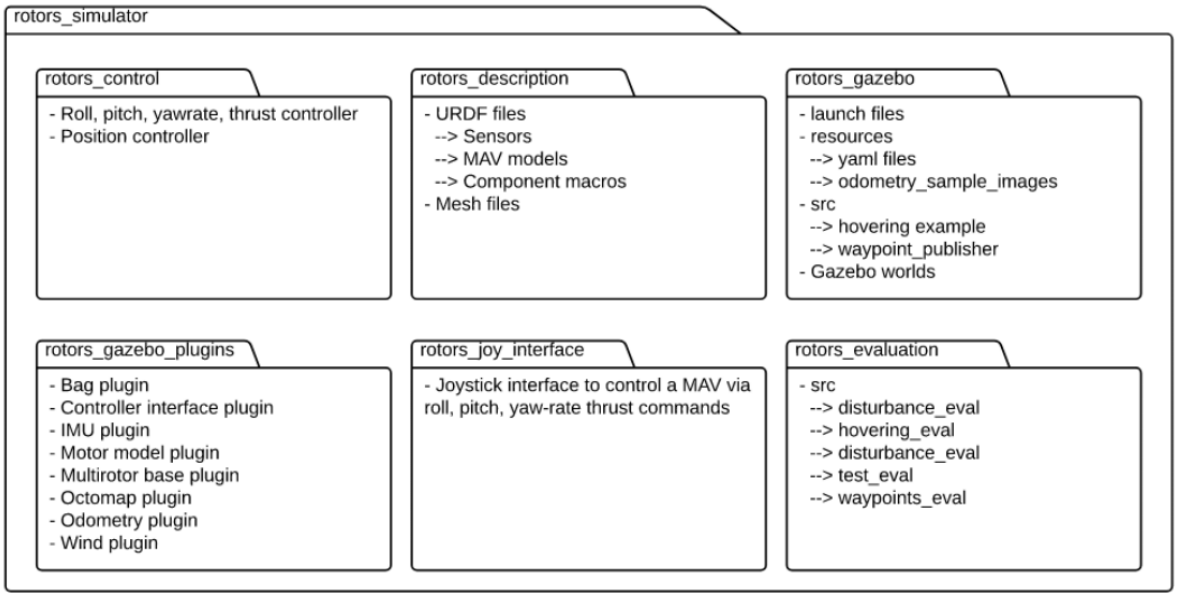


Figura 2.13 Paquetes del simulador RotorS (Furrer et al. 2016).

Los principales elementos de RotorS se detallan en la siguiente lista:

- *rotors_description*: El paquete *rotors_description* contiene los archivos *xacro* y los modelos 3D de los componentes involucrados en la simulación (sensores, bastidores del UAV, etc.).

- *rotors_control*: Este paquete contiene un conjunto de controladores de bajo nivel para el UAV que generan las velocidades de las hélices en función de la entrada de posición deseada.

- *rotors_gazebo_plugins*: Este paquete contiene un conjunto de *plugins* de Gazebo que se utilizan para simular los sensores y las hélices del UAV. Todos los modelos incluirán los siguientes plugins:

Plugin IMU: Esto simula el sensor inercial.

Plugin del modelo del motor: Simula la dinámica de los motores montados en el el UAV.

Plugin de la base del multirrotor: Este plugin calcula y aplica las fuerzas y pares a la base del UAV, basándose en las velocidades de los motores.

Plugin de odometría: Este plugin simula un sensor de odometría para transmitir

la posición y orientación del UAV.

Interfaz Gazebo-ROS de los rotores: Este plugin representa la capa de comunicación entre los mensajes ROS de RotorS y la escena de simulación de Gazebo. Si no se carga este plugin, no se podrá utilizar ningún tópico ROS para comandar el robot. Además, sólo se puede cargar una instancia de este plugin. Por esta razón, es conveniente cargar este plugin en el archivo de mundo de Gazebo.

- ***rotors_gazebo***: Este paquete contiene el archivo de lanzamiento para iniciar los diferentes modelos en el simulador Gazebo.

La combinación de los elementos incluidos en los paquetes RotorS permite crear nuevos robots, o modificar los ya implementados, añadiendo sensores o cambiando sus parámetros dinámicos (como la masa o la inercia del robot).

El simulador RotorS posee varios modelos de UAV con diferentes configuraciones. Sin embargo, es posible añadir nuevos modelos de robot con el número de motores que desee, colocados en cualquier lugar de la estructura base del robot. Para importar un nuevo modelo en el marco de RotorS necesitamos definir el archivo *xacro* que contiene todas las articulaciones, enlaces y sensores del multirrotor. El paquete rotors_description contiene un conjunto de archivos *xacro* para implementar diferentes macros para simplificar la creación del UAV. En particular, los siguientes se incluirán en el cuadricóptero para los experimentos de esta investigación:

- El archivo *multirotor_base.xacro* representa el elemento principal del UAV.

- El archivo *component_snippets.xacro* contiene las macros para varios componentes relacionados con la simulación (sensores, motores, etc.).

Nos referiremos a estos archivos para construir el modelo de cuadricóptero IRIS utilizado en este proyecto (Joseph and Cacace 2021).

Con esto ya es posible controlar al UAV en simulación (Figura 2.14) para probar el algoritmo de aprendizaje por refuerzo “Aprendizaje-Q”.

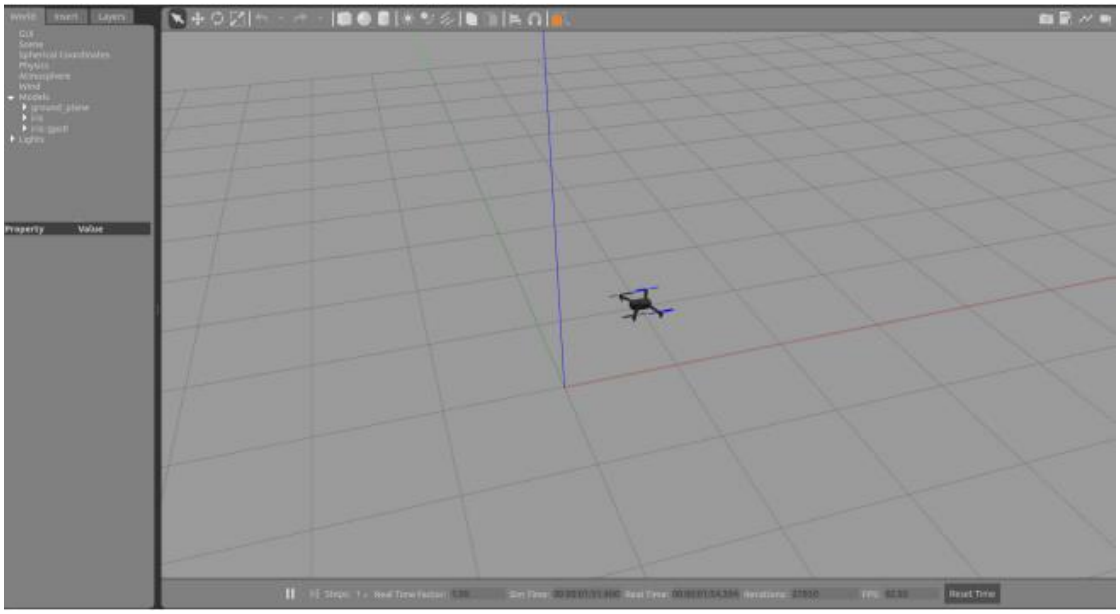


Figura 2.14 Cuadricóptero IRIS simulado en ROS-Gazebo

Capítulo 3: Metodología

En este capítulo, se detalla la metodología utilizada para desarrollar un modelo computacional que permita la simulación en Gazebo de la navegación autónoma de un cuadricóptero. A través de la optimización del algoritmo de aprendizaje por refuerzo Aprendizaje-Q para la generación autónoma de trayectorias.

En la sección 3.1 se describe el proceso para representar mapas continuos del simulador Gazebo como mapas discretos, para poder obtener estados, acciones y recompensas para entrenar el algoritmo de Aprendizaje-Q.

La sección 3.2 describe el proceso para generar trayectorias con la optimización del algoritmo Aprendizaje-Q que adapta dos métodos: muestreo de Thompson y distribución Gaussiana, para modificar la política utilizada en el Aprendizaje-Q para la selección de la acción que llevara al estado siguiente utilizando la ecuación de la diferencia temporal. También, se describe el proceso para suavizar trayectorias con los algoritmos de interpolación Bézier y B-spline. Además, se explica el cálculo del ángulo *yaw* y el retardo temporal entre la publicación de los puntos de ruta necesario para regular la velocidad del cuadricóptero.

Finalmente, en la sección 3.3 se describe el proceso de publicación de la trayectoria en el simulador Gazebo para realizar la navegación autónoma con el cuadricóptero, incluyendo la captura y análisis de los datos de odometría y la evaluación de la precisión de la trayectoria utilizando métricas de error.

3.1 CONSTRUIR ENTORNOS

En esta sección se describe el proceso para representar mapas continuos del simulador Gazebo como mapas discretos, para poder obtener estados, acciones y recompensas para entrenar el algoritmo de Aprendizaje-Q.

3.1.1 Los estados

El estado es el valor numérico que se le asigna a la localización donde el agente está en un momento determinado, esto sirve para simplificar la entrada para las ecuaciones matemáticas, además, para poder dar como entrada estos estados al cuadricóptero en simulación, también se le asigna una coordenada a cada estado, en la Tabla 3.2 se muestra un ejemplo de este mapeo localización-estado-coordenada y en la Figura 3.15 se muestra un entorno con localizaciones.

Localizacion	Estado
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8
J	9
K	10
L	11

Tabla 3.2 Mapeo localización-estado.

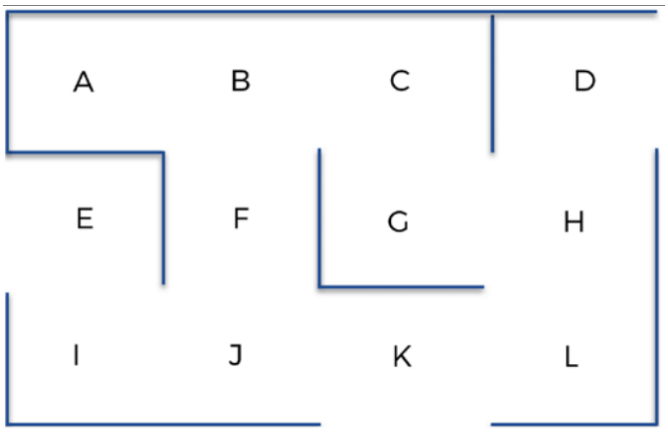


Figura 3.15 Entorno con localizaciones (Ponteves 2019).

3.1.2 Las acciones

Las acciones son los siguientes movimientos que el agente puede hacer para ir de una ubicación a la siguiente. Por ejemplo, si el agente está en la ubicación J; las posibles acciones que el agente puede realizar son ir a I, a F, o a K. cómo se usan ecuaciones matemáticas, se codifican estas acciones con los mismos estados que para las localizaciones, como se muestra en la Tabla 3.3.

Acciones	0	1	2	3	4	5	6	7	8	9	10	11

Tabla 3.3 Lista de acciones.

3.1.3 La recompensa

Por último, se define una función de recompensa R que toma como entrada un estado s y una acción a, y devuelve una recompensa numérica r que el agente obtendrá al realizar la acción a en el estado s:

$$R: (s, a) \rightarrow r \in \boldsymbol{R} \tag{3.1}$$

Dado que en hay un número discreto y finito de estados, así como un número discreto y finito de acciones, la mejor manera de construir la función de recompensa R es simplemente hacer una matriz.

Para este ejemplo, la función de recompensa es una matriz de exactamente 12 filas y 12 columnas (Tabla 3.4), donde las filas corresponden a los estados, y las columnas a las acciones. De esta manera, en la función R, s será el índice de la fila de la matriz, a será el índice de columna de la matriz, y r será la celda de índice (s, a) en la matriz.

Para construir esta matriz de recompensas, lo que hay que hacer primero es atribuir, para cada una de las celdas, una recompensa 0 a las acciones que el cuadricóptero no puede realizar, y una recompensa 1 a las acciones que el cuadricóptero puede realizar. Haciendo esto para cada una de las acciones, se obtiene una matriz de recompensas (Tabla 3.4).

	A	B	C	D	E	F	G	H	I	J	K	L
A	0	1	0	0	0	0	0	0	0	0	0	0
B	1	0	1	0	0	1	0	0	0	0	0	0
C	0	1	0	0	0	0	1	1	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	1	0	0	0
F	0	1	0	0	0	0	0	0	0	1	0	0
G	0	0	1	0	0	0	0	1	0	0	0	0
H	0	0	0	1	0	0	1	0	0	0	0	1
I	0	0	0	0	1	0	0	0	0	1	0	0
J	0	0	0	0	0	1	0	0	1	0	1	0
K	0	0	0	0	0	0	0	0	0	1	0	1
L	0	0	0	0	0	0	0	1	0	0	1	0

Tabla 3.4 Matriz de recompensa.

Se puede observar que la matriz de recompensa de la Tabla 3.4 está diseñada para acciones de movimientos horizontales y verticales. Sin embargo, para poder realizar movimientos con diagonales en el entorno mostrado en la Figura 3.15, es necesario agregar recompensa en las acciones correspondientes a los movimientos en diagonales. Para ello, se presenta la Tabla 3.5, donde se incluyen las recompensas para las acciones de movimientos diagonales, y se muestran los estados que ahora son alcanzables mediante estas acciones de movimiento. En particular, las acciones de movimiento en diagonales que se han añadido son:

$(A \rightarrow F),\ (B \rightarrow G),\ (C \rightarrow HF),\ (D \rightarrow G),\ (E \rightarrow J),\ (F \rightarrow IKCA),\ (G \rightarrow LBD),\ (H \rightarrow KC),$
$(I \rightarrow F),\ (J \rightarrow E),\ (K \rightarrow FH),\ (L \rightarrow G)$

	A	B	C	D	E	F	G	H	I	J	K	L
A	0	1	0	0	0	1	0	0	0	0	0	0
B	1	0	1	0	0	1	1	0	0	0	0	0
C	0	1	0	0	0	1	1	1	0	0	0	0
D	0	0	0	0	0	0	1	0	0	0	0	0
E	0	0	0	0	0	0	0	0	1	1	0	0
F	1	1	1	0	0	0	0	0	1	1	1	0
G	0	1	1	1	0	0	0	1	0	0	0	1
H	0	0	1	1	0	0	1	0	0	0	1	1
I	0	0	0	0	1	1	0	0	0	1	0	0
J	0	0	0	0	1	1	0	0	1	0	1	0
K	0	0	0	0	0	1	0	1	0	1	0	1

L	0	0	0	0	0	0	1	1	0	0	1	0

Tabla 3.5 Matriz de recompensa con diagonales.

3.1.4 Construir entornos de interiores

En esta sección, se detallan los procesos (Figura 3.16) para construir entornos de interiores inspirados en lugares como bodegas, oficinas o casas. Primero, se utiliza como referencia un plano arquitectónico del lugar. Luego, utilizando un software de edición de imágenes, como Photoshop, se construye un mapa binario con una resolución de HxW pixeles. Donde cada píxel en este mapa binario se redimensionará según una resolución especifica en el mapa del simulador Gazebo.

Para ilustrar este proceso, en la Figura 3.17, se puede observar que cada píxel en el mapa equivale a un metro cuadrado en la representación espacial de la malla del simulador Gazebo (Figura 3.23).

Finalmente, el mapa binario se exporta en formato PNG para ser utilizado en la construcción del entorno discreto en el simulador Gazebo.

Figura 3.16 Diagrama a bloques mapa binario.

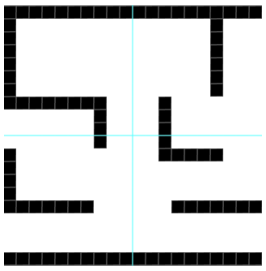


Figura 3.17 Mapa binario $20x20$ pixeles.

Ahora con la finalidad de extraer del PNG estados, acciones y recompensas para el algoritmo de Aprendizaje-Q se siguen los siguientes procesos (Figura 3.18):

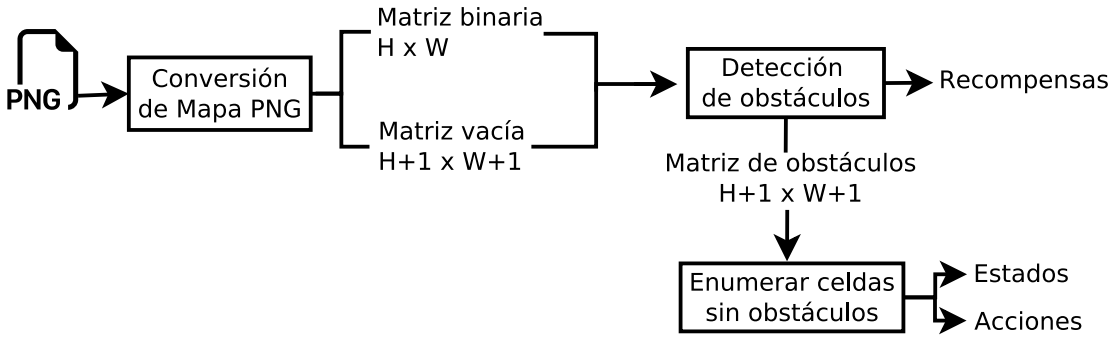


Figura 3.18 Diagrama a bloques PNG como entrada para Aprendizaje-Q.

Conversion de Mapa PNG

El proceso de "Conversión de Mapa PNG", tiene como objetivo obtener dos representaciones del mapa PNG:

1. **Matriz Binaria**: una matriz binaria con las mismas dimensiones HxW, donde las celdas con valor de cero representan espacios con obstáculos como paredes, muebles, mesas, lámparas, puertas, ventanas, estanterías, etc. Y las celdas con valor de uno representan espacios sin obstáculos.
2. **Matriz Vacía**: Ahora para poder representar cada celda de la matriz binaria, como un punto de intersección entre cuatro celdas, se inicializa una matriz vacía, con las dimensiones HxW de la matriz binaria, incrementadas en más uno, es decir, $H + 1\ x\ W + 1$. Esto permite que el centro del marco fijo del cuadricóptero se posicione en los puntos de intercesión de la matriz vacía (Figura 3.19).

Detección de obstáculos

El proceso de "Detección de obstáculos", tiene como objetivo detectar la precencia de obstáculos en los vértices de las celdas de la matriz binaria. Una vez detectados los obstáculos, se procede a actualizar el estado de cada vértice en la celda correspondiente de la matriz vacía, indicando de esta manera la presencia o ausencia de obstáculos en esas

ubicaciones específicas (Figura 3.19). Al final de este proceso, se obtiene la matriz de obstáculos y la recompensa.

Enumerar celdas sin obstáculos

El proceso de "Enumerar celdas sin obstáculos", tiene como objetivo asignar números únicos a las celdas que no tienen obstáculos de la matriz de obstáculos. Al final de este proceso, se obtiene una matriz de estados que proporciona una identificación única para cada celda sin obstáculos en la matriz (Figura 3.19). También se obtienen las acciones que son el enlistado de los estados enumerados.

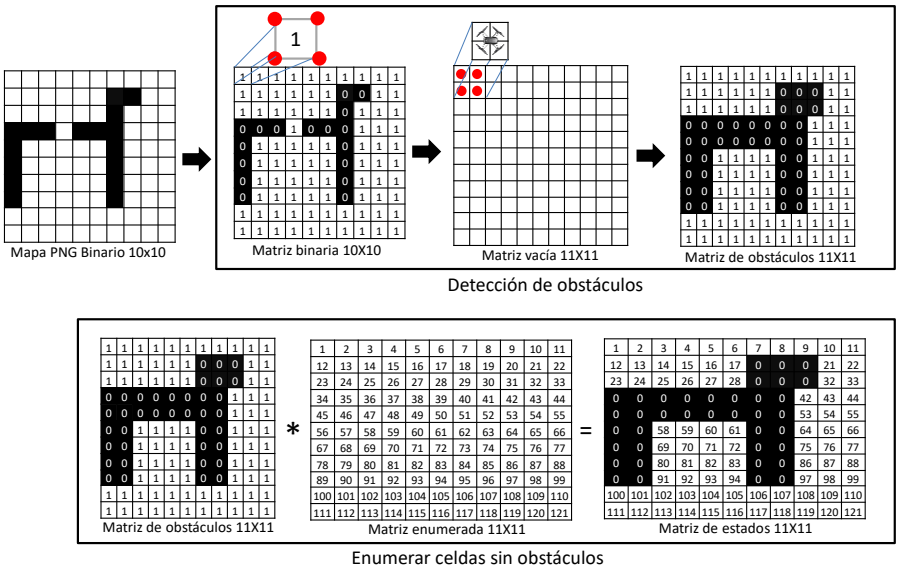


Figura 3.19 PNG como entrada para Aprendizaje-Q.

3.1.5 Recompensa modificada con ventanas de proximidad al obstáculo

La "Recompensa modificada con ventanas de proximidad al obstáculo" es una estrategia que permite al algoritmo de Aprendizaje-Q considerar la proximidad a los obstáculos en la exploración y explotación de la recompensa en el entorno. En esta técnica, se utiliza la fórmula (3.2) para ajustar la recompensa de cada estado según su proximidad a los obstáculos detectados en el área de una ventana de tamaño NxN,

$$\mathrm{R}_{ventana_{NxN}} = \mathrm{R}_{\max} - \frac{N_{ceros}}{\acute{A}rea_{ventana} - camino} \tag{3.2}$$

Donde:

- NxN denota una matriz cuadrada con N siendo un número impar.
- R_{max} es el valor máximo de la recompensa para un estado diferente al destino.
- N_{ceros} es el número de ceros equivalentes a los obstáculos detectados en la ventana del estado que se está evaluando.
- $\acute{A}rea_{ventana}$ representa el área de la ventana NxN.
- $camino$ es el número de celdas en una matriz cuadrada NxN, con N siendo un número impar, que corresponde al camino más corto posible desde el borde de la matriz hasta el centro de la matriz, es decir $camino = \frac{N-1}{2} + 1$.

La variable $camino$ se establece debido a que para que un estado sea válido en el entorno, debe existir al menos un camino que lo conecte con otros estados.

En la Figura 3.20, inciso A), se representa un entorno mediante una matriz de dimensiones $11x11$. En este entorno, el valor máximo de la recompensa (R_{max}) es igual a 2. Además, se destaca la presencia de un obstáculo en el centro, el cual ocupa un espacio de $5x5$ celdas y está representado por valores de recompensa igual a 0. El estado actual en evaluación, que se denota como S_t, se encuentra marcado en gris en la figura.

En la Figura 3.20, inciso B), se presenta una ventana cuadrada de dimensiones $3x3$. Esta ventana representa un fragmento del entorno del inciso A) y se utiliza para evaluar el estado actual S_t marcado en gris. Se observa que el numero de ceros detectados en la ventana es $N_{ceros} = 2$, el área de la ventana $Area_{ventana} = 9$, el número de celdas que corresponde a un $camino = 2$.

Entonces la recompensa para el estado actual S_t se calcula al sustituir estas variables en la fórmula (3.2), como se muestra a continuación:

$$R_{ventana3x3} = 2 - \frac{2}{9-2} = \frac{12}{7}$$

La evaluación completa para todos los estados del entorno del inciso A) se encuentra en la parte izquierda etiquetada como $3x3$ en la Figura 3.21. se puede observar cómo las celdas a una distancia de una celda del obstáculo tienen una recompensa de $\frac{11}{7}$ cuando hay 3 obstáculos en la evaluación de la fórmula (3.2), una recompensa de $\frac{12}{7}$ cuando hay 2 obstáculos y una recompensa de $\frac{13}{7}$ cuando hay 1 obstáculo.

Ahora, en el inciso C) de la Figura 3.20 se muestra el ejemplo para la ventana $5x5$ se repite el proceso del inciso B), y de igual forma el procedimiento completo se muestra en la parte derecha de la Figura 3.21 con la etiqueta $5x5$.

Por último, para el inciso D) como detalle a destacar se puede observar que la ventana $7x7$ presenta valores vacíos. Esto ocurre debido a que esta parte de la ventana se encuentra fuera de los bordes del entorno del inciso A). Por lo tanto, estos valores vacíos no influyen en la evaluación del estado en cuestión, ya que están más allá de los límites del entorno definido en el inciso A).

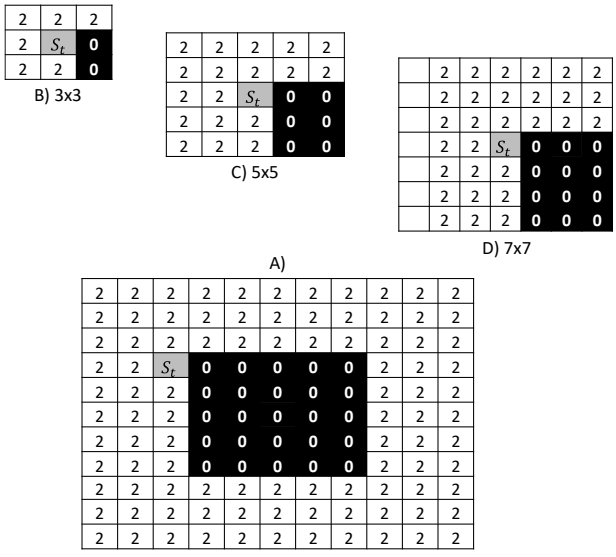


Figura 3.20 Ventanas.

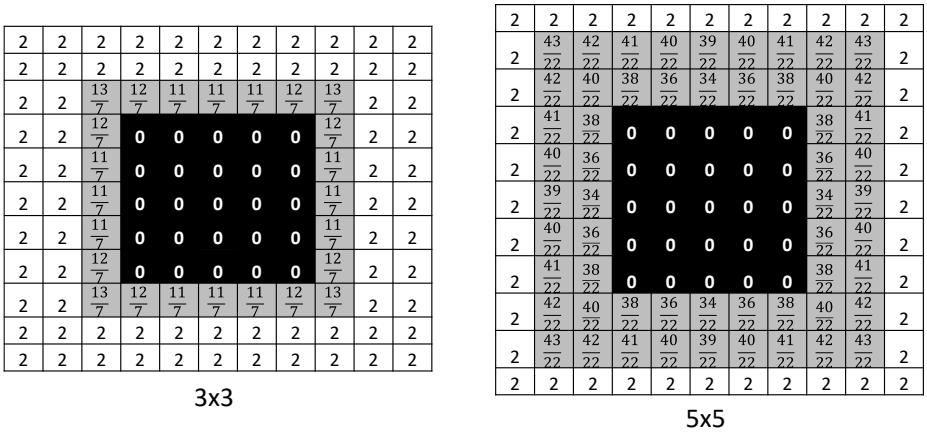


Figura 3.21 Recompensa modificada con ventanas.

3.1.6 Exportar mapa binario a Gazebo

Si bien el cuadricóptero navega por un número finito de estados, Gazebo es un entorno continuo en el que existen una amplia variedad de posiciones. Para poder aplicar el algoritmo de Aprendizaje-Q en este entorno, es necesario realizar alguna forma de discretización.

Una forma de discretizar el entorno de Gazebo es mediante el uso de una malla, como se muestra en la Figura 3.23. La malla divide el espacio continuo de posiciones en una cuadricula finita de celdas discretas, lo que permite representar el espacio de estados de manera discreta.

Para exportar un mapa binario a Gazebo (Figura 3.22), se emplea el nodo *map_server*. Este nodo recibe como entrada el mapa binario en formato PNG y un archivo de configuración en el que se especifica la resolución de metro por píxel. Además, se utiliza el nodo *map2gazebo*, que se suscribe al tópico */map* del nodo *map_server* para exporta una malla 3D con obstáculos correspondientes a las celdas ocupadas del mapa. El resultado final se puede observar en la Figura 3.23, donde se muestran los píxeles ocupados extruidos hacia arriba, generando los obstáculos en el entorno de simulación de Gazebo (Curtis 2020).

En este trabajo, se ha utilizado el cuadricóptero IRIS, que tiene un marco fijo de cuarenta y cinco centímetros cuadrados. Por lo tanto, todos los entornos son generados con

una resolución de cincuenta centímetros por píxel. Además, se ha implementado una matriz de obstáculos (Figura 3.19) para asegurar que todos los estados en los que el cuadricóptero se puede mover tengan una distancia mínima del centro de su marco fijo de un pixel de resolución, es decir, cincuenta centímetros.

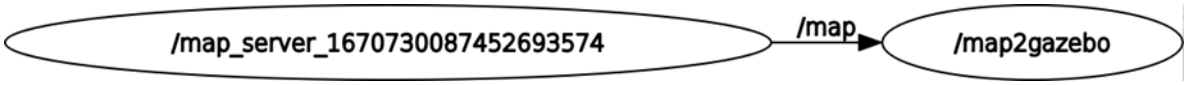


Figura 3.22 Nodos y tópico para cargar el mapa binario a Gazebo en formato world.

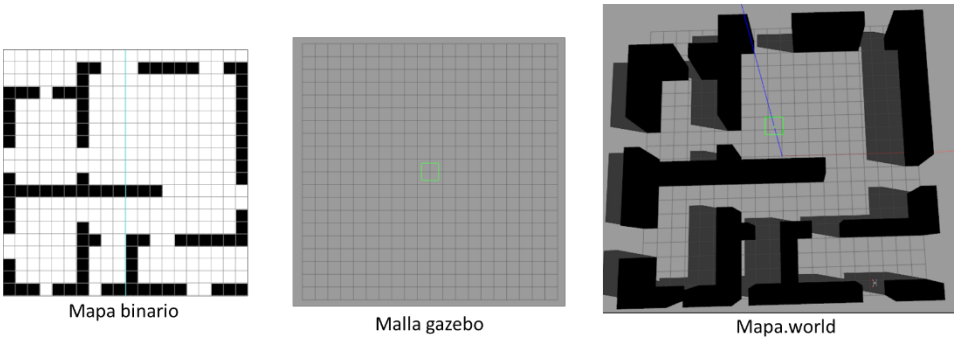


Figura 3.23 Exportar Mapa binario a Gazebo.

3.2 GENERAR LA TRAYECTORIA

En esta sección se presenta el proceso que se siguen para generar los puntos de ruta para controlar el vuelo de un cuadricóptero en simulación desde un punto de inicio hasta un punto final. Este proceso se describe en el diagrama a bloques de la Figura 3.24, donde primero, se genera una ruta de coordenadas discretas utilizando el algoritmo de Aprendizaje-Q. Luego, se aplican técnicas de interpolación para obtener una trayectoria continua en el espacio cartesiano del simulador. Además, se calcula el ángulo yaw para cada punto y se genera un vector de retardo que establece el intervalo de tiempo entre la publicación de los puntos de ruta de la trayectoria en la simulación. Por último, se crea un archivo de texto que contiene el vector de retardo, coordenadas x, y, z, donde z se mantiene constante con un valor de 2, y *yaw* en el orden correspondiente.

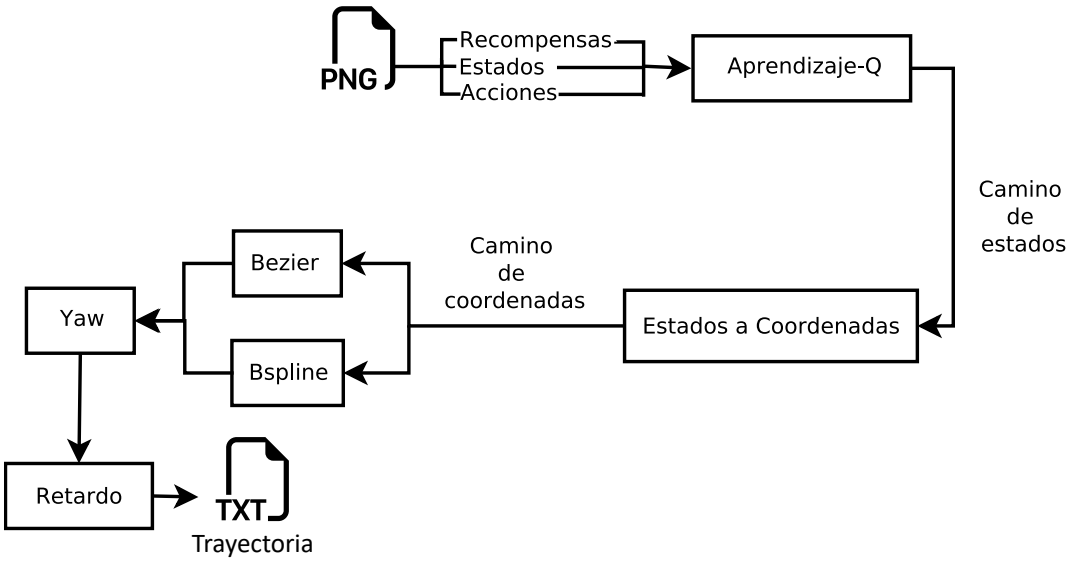


Figura 3.24 Diagrama a bloques Aprendizaje-Q e interpolación.

A continuación, en esta sección se detalla en mayor profundidad cada uno de los bloques del diagrama de la Figura 3.24.

3.2.2 Algoritmo Aprendizaje Q

El Algoritmo de Aprendizaje-Q, utiliza un modo de entrenamiento, durante el cual los parámetros que son aprendidos se denominan valores Q, y un modo de inferencia, cuando se tiene un modelo completamente entrenado para hacer predicciones.

En la primera parte de esta sección se presentan los pasos del entrenamiento. Donde se explora cómo se aprenden los valores Q y cómo se ajustan a lo largo del proceso de aprendizaje. Cabe resaltar que en el paso dos del entrenamiento, se aborda la estrategia para elegir la acción a_t a partir de un estado s_t, y se presenta el método convencional el cual es un enfoque exploratorio comúnmente utilizado como estrategia RL.

Luego, como modificación al paso dos del algoritmo se introducen dos métodos que se adaptan al problema de un PDM discreto y combinan la exploración y la explotación. El primero adapta el algoritmo de muestreo de Thompson y el segundo adapta una distribución Gaussiana.

Finalmente, se aborda el modo de inferencia del algoritmo. En esta fase, se explora cómo el modelo completamente entrenado se utiliza para realizar predicciones y tomar decisiones. Este modo representa la culminación del proceso de entrenamiento y prepara el algoritmo para aplicaciones prácticas.

Entrenamiento

inicialización:

Para todas las parejas de estados *s* y acciones *a*, los valores Q se inicializan a 0 y el valor $R(s, a)$ del destino se inicializa en 1000.

Se inicializa el número de iteraciones por época igual al número de acciones a que puede llevar a un siguiente estado posible, es decir, tal que $R(s_t, a_t) > 0$.

Mientras la diferencia en el paso seis sea mayor que 0, en cada iteración $t \geq 1$, los siguientes pasos se repiten hasta alcanzar el número de iteraciones por época:

1. Se selecciona un estado aleatorio s_t de entre los posibles estados.
2. A partir de ese estado, se realiza una acción aleatoria a_t que puede llevar a un siguiente estado posible, es decir, tal que $R(s_t, a_t) > 0$.
3. Se alcanza el siguiente estado s_{t+1} y se obtiene la recompensa $R(s_t, a_t)$.
4. Se calcula la diferencia temporal con (2.25).
5. Se actualiza el valor Q aplicando la ecuación de Bellman con (2.27).
6. Al final de cada época se obtiene la diferencia entre la tabla Q original (sin los valores actualizados) y la nueva tabla Q que incorpora los valores actualizados.

Al final de este proceso, se han obtenido valores Q que ya no se actualizan, señalando la convergencia de la tabla Q en este entrenamiento.

Entrenamiento con Muestreo de Thompson Adaptado (MTA)

La siguiente modificación propuesta en el paso dos del entrenamiento del algoritmo de Aprendizaje-Q utiliza un enfoque similar al del algoritmo muestreo de Thompson (Tabla 2.1) para seleccionar la próxima acción. En este caso, en lugar de seleccionar una acción

aleatoria de las disponibles, se lleva a cabo el siguiente procedimiento en el paso dos del entrenamiento:

1. Desde el estado actual s_t , para cada acción a_t que puede llevar a un siguiente estado posible, es decir, tal que $R(s_t, a_t) > 0$, se calcula la cantidad obstáculos (β) y estados validos (α) que se deben agregar a la distribución Beta para modelar la recompensa esperada de esa acción. Luego, se muestrea una distribución Beta para cada acción a_t y se selecciona la acción con la muestra más alta de la distribución Beta como la acción a tomar.

Para ilustrar este proceso, En la Figura 3.25 se presenta un entorno, donde los obstáculos se representan con celdas de color negro, mientras que los espacios libres se muestran con color blanco. La recompensa se distribuye utilizando una ventana de proximidad al obstáculo de $3x3$. Con esta distribución de recompensa, se utilizan ocho distribuciones Beta, una para cada posible acción a_t desde el estado actual s_t. Estas distribuciones Beta se pueden observar en la Figura 3.26, donde las celdas de color rojo, azul y verde de la Figura 3.25 se corresponden con distribuciones Beta del mismo color.

Cabe mencionar que, en la Figura 3.26 solo se visualizan tres distribuciones Beta, debido a que las distribuciones que modelan un mismo valor de recompensa se superponen.

Estas distribuciones Beta se utilizan para modelar la recompensa esperada en cada acción a_t. Si una distribución Beta esta sesgada hacia la derecha, significa que la acción a_t tiene una alta probabilidad de obtener recompensas altas. Por otro lado, si está sesgada hacia la izquierda, indica que la acción a_t tiene una alta probabilidad de obtener recompensas bajas.

En la Tabla 3.6. se detalla la configuración de estas distribuciones Beta, mostrando en la segunda columna el valor de α y β para cada una y sumando 1 para suavizar la distribución Beta. Si α es mayor que β, la distribución Beta se sesga hacia la derecha, y si β es mayor que α, se sesga hacia la izquierda.

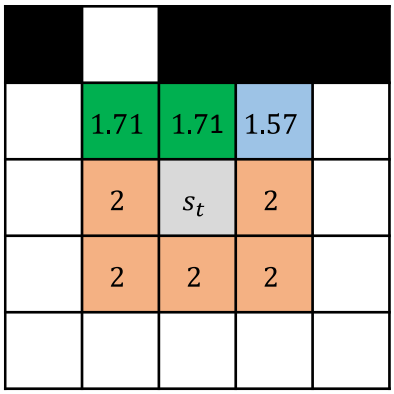


Figura 3.25 Distribución de la recompensa en el entorno con ventana 3x3.

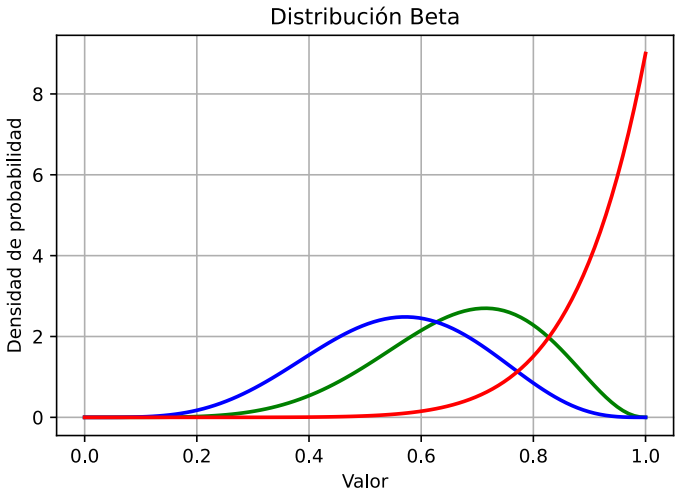


Figura 3.26 Distribuciones Beta.

Color	Distribución β	Recompensa
Rojo	$(7+1, 0+1)$	2
Verde	$(5+1, 2+1)$	1.714
Azul	$(4+1, 3+1)$	1.571

Tabla 3.6 Configuración de distribuciones Beta.

Para mostrar el funcionamiento del método MTA de selección de acciones, en cuanto a exploración y explotación de la recompensa en el entorno, en la Figura 3.27, se presenta la frecuencia relativa de acciones realizadas a partir de un estado s_t, durante 1681 iteraciones. Esto se hace en relación con la recompensa distribuida utilizando las ventanas de $3x3$ (Figura 3.25) y $7x7$.

Cuando la recompensa está distribuida utilizando la ventana $3x3$, en la parte izquierda de la Figura 3.27, se observa que cuando la recompensa para una acción a_t es igual a 1.57, la frecuencia relativa es de 0.059%. Cuando la recompensa aumenta ligeramente a 1.71, la frecuencia relativa acumulada, en dos celdas, es de 0.238%. Finalmente, con la máxima recompensa de 2, la frecuencia relativa acumulada, en seis celdas, es de 99.703%.

Estos resultados indican que, cuando se utiliza la ventana $3x3$, el método MTA tiende a explorar las acciones a_t que proporcionan la recompensa más alta, el 99.703% de las iteraciones. Por otro lado, MTA explora las acciones con recompensas más bajas el 0.297% de las iteraciones.

Por otro lado, cuando la recompensa está distribuida utilizando la ventana $7x7$, en la parte derecha de la Figura 3.27, se observa que cuando la recompensa para una acción a_t es igual a 1.78, la frecuencia relativa es de 0%. Cuando la recompensa aumenta ligeramente a 1.82, la frecuencia relativa es de 0.178%. A medida que la recompensa continúa aumentando a 1.84, 1.87, 1.89 y 1.91, se nota un aumento en la frecuencia relativa a 0.654%, 1.844%, 4.7% y 10.291% respectivamente.

La tendencia sigue siendo evidente cuando la recompensa alcanza 1.93, con una frecuencia relativa más alta de 26.889%. Finalmente, la máxima recompensa de 1.96, es donde la frecuencia relativa alcanza su máximo con 55.443%.

Estos resultados indican que, cuando se utiliza la ventana $7x7$, en cuanto a explotación, el método MTA tiende a explotar la acción a_t que proporciona la recompensa más alta, en la mayoría de las iteraciones, con una frecuencia relativa del 55.443%. Por otro lado, en cuanto a exploración, MTA explora gradualmente las acciones con recompensas más

bajas en un orden descendente, y en este caso, la exploración de la acción con la recompensa más baja es del 0%.

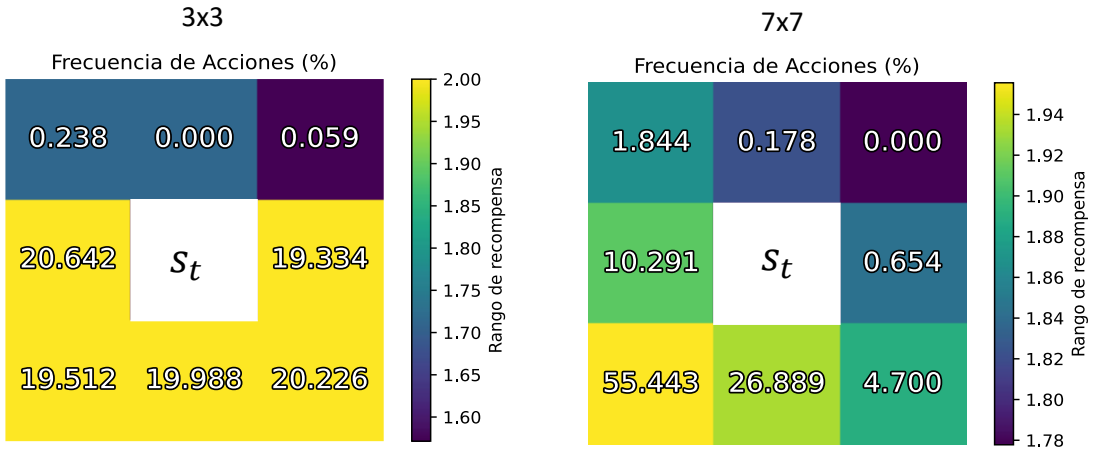


Figura 3.27 Frecuencia relativa de acciones en una época.

Entrenamiento con Distribución Gaussiana (DG)

La modificación propuesta en el paso dos del entrenamiento del algoritmo de Aprendizaje-Q utiliza una distribución gaussiana (Figura 2.12) para seleccionar la próxima acción.

La descripción del paso dos del algoritmo de Aprendizaje-Q ahora se vería de esta forma:

2.1 Se modela una distribución gaussiana $\mathcal{N}(\mu, \sigma)$ para el conjunto de estados en el entorno. La media μ se calcula a partir del estado actual s_t, seleccionando la acción a_t con la mayor recompensa. Si existen múltiples acciones con la máxima recompensa, se selecciona la primera acción que cumple con este criterio, siguiendo el orden de aparición en el vector de acciones.

2.2 El ancho σ de la gaussiana se determina como la distancia entre la media μ y la acción posible más alejada de la media desde el estado actual s_t.

2.3 A continuación, se multiplica el vector de recompensa de las posibles acciones a_t por la función de densidad de probabilidad (PDF) de la gaussiana. Este proceso

genera un nuevo vector que asigna una probabilidad a cada acción, reflejando cuán probable es cada acción dada la distribución gaussiana.

2.4 Luego, este vector de probabilidad se normaliza de manera que la suma total de las probabilidades sea igual a uno.

2.5 Finalmente, se selecciona la próxima acción de manera aleatoria, teniendo en cuenta las probabilidades asignadas a cada acción en el vector normalizado.

Para ilustrar este proceso, en la Figura 3.28a, se modela una distribución Gaussiana, donde el conjunto de estados validos es igual a 1123.

El estado actual s_t es igual a 286 y las posibles acciones, están definidas en el siguiente vector de acciones:

$$acciones=[244, 245, 246, 285, S_t, 287,326,327,328]$$

Donde 285 es la acción que se toma como media, por ser la primera ocurrencia con la máxima recompensa que es 2.

$$recompensa=[1.7, 1.7, 1.5, 2, 2, 2, 2, 2]$$

Nótese que la distancia entre la media $\mu(285)$ y la acción más alejada 328, es de $43\ estados$, por lo tanto, se selecciona un $\sigma = 43$ como el ancho de la distribución gaussiana. Al elegir un valor σ igual a la distancia entre la media y la acción más alejada, se logra una distribución gaussiana que se adapta a la dispersión de las acciones en el entorno. Esto significa que las probabilidades asignadas a las acciones reflejarán de manera más precisa la recompensa asociada a cada acción.

En la Figura 3.28b, se observa cómo se distribuye la recompensa al multiplicar el vector de recompensa por la PDF de la gaussiana. El resultado de esta multiplicación es el siguiente vector:

$$PDF * recompensa = [1.396, 1.414, 1.312, 2, 1.998, 1.629, 1.608, 1.587]$$

Para obtener el vector de probabilidades, se normaliza el vector $PDF * recompensa$. El vector normalizado se ve de la siguiente manera:

$$Probabilidad = [0.107\ 0.109\ 0.101\ 0.154\ 0.154\ 0.125\ 0.124\ 0.122]$$

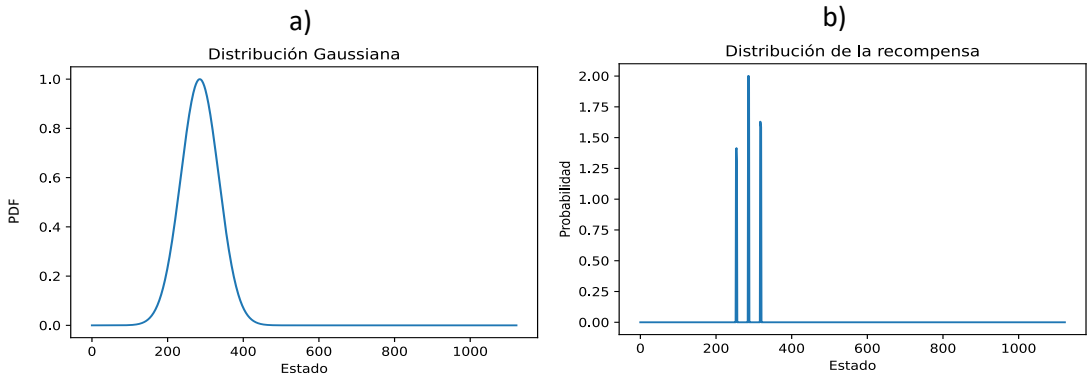


Figura 3.28 Distribución de la recompensa con una Gaussiana.

Finalmente, en la Figura 3.29, se representa la distribución de probabilidad en el entorno de acuerdo con la recompensa mostrada en la Figura 3.25, En esta representación:

La media de la está en la celda roja con la probabilidad más alta de 15.44%, debido a su recompensa y su ubicación central en relación con las otras celdas. El resto de las acciones obtienen su probabilidad de acuerdo con su recompensa y su proximidad a la media.

Donde:

- Las celdas verdes, con una recompensa de 1.7, tienen una probabilidad asignada del 10.78% cuando la distancia a la media es de 41 estados y ligeramente mayor 10.92% cuando la distancia a la media es de 40 estados.
- La celda azul, que tiene la menor recompensa de 1.5, tiene una probabilidad asignada del 10.13% y una distancia a la media de 39 estados.
- Las celdas rojas, tienen la recompensa más alta de 2, y sus probabilidades van de 12.25% a 15.43% de acuerdo con su distancia a la media, donde la mayor distancia es de 43 estados y la mínima de 2 estados.

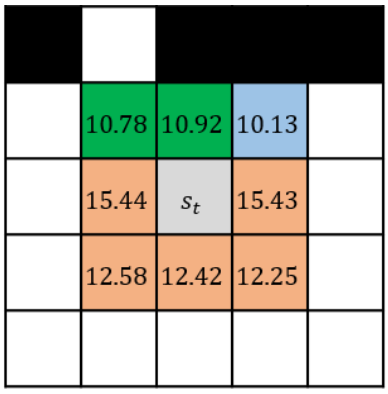


Figura 3.29 Probabilidad (%) para cada acción posible a_t desde un estado s_t.

Para mostrar el funcionamiento del método DG de selección de acciones, en cuanto a exploración y explotación de la recompensa en el entorno, en la Figura 3.30, se presenta la frecuencia relativa de acciones realizadas a partir de un estado s_t durante 1681 iteraciones. Esto se hace en relación con la recompensa distribuida utilizando las ventanas de $3x3$ (Figura 3.29) y $7x7$.

Cuando la recompensa está distribuida utilizando la ventana $3x3$, en la parte izquierda de la Figura 3.30, se observa que cuando la recompensa para una acción a_t es igual a 1.57, la frecuencia relativa es de 10.648%. Cuando la recompensa aumenta ligeramente a 1.71, la frecuencia relativa acumulada, en dos celdas, es de 20.583%. Finalmente, con la máxima recompensa de 2, la frecuencia relativa acumulada, en seis celdas, es de 68.769%.

Estos resultados indican que, cuando se utiliza la ventana $3x3$, el método DG tiende a explorar las acciones a_t que proporciona la recompensa más alta, el 68.769% de las iteraciones. Por otro lado, DG explora las acciones con recompensas más bajas el 31.231% de las iteraciones.

Cuando la recompensa está distribuida utilizando la ventana $7x7$, en la parte derecha de la Figura 3.30, Se puede observar que a medida que la recompensa varía, también lo hace la frecuencia relativa de la siguiente manera:

- Cuando la recompensa es igual a 1.78, la frecuencia relativa es del 7.258%.

- Un ligero aumento en la recompensa a 1.82 resulta en un aumento leve en la frecuencia relativa, que se incrementa a 7.311%.
- Un aumento adicional en la recompensa a 1.84 provoca un aumento significativo en la frecuencia relativa, alcanzando un 14.872%.
- Sin embargo, cuando la recompensa aumenta a 1.87, la frecuencia relativa disminuye a 7.139%.
- Un nuevo aumento en la recompensa a 1.89 conlleva un aumento en la frecuencia relativa a 17.311%.
- Luego, cuando la recompensa aumenta a 1.91, la frecuencia relativa disminuye nuevamente a 13.682%.
- Continuando con el patrón, cuando la recompensa llega a su máximo de 1.96, la frecuencia relativa disminuye a 15.110%.

Estos resultados indican que, en cuanto a explotación, el método DG tiende a explotar la acción a_t que proporciona la recompensa más alta, el 15.110% de las iteraciones. Por otro lado, en cuanto a exploración, DG explora gradualmente las acciones con recompensas más bajas en un orden descendente pero con un factor de aleatoriedad que hace que existan variaciones que permite que acciones con recompensas bajas sean consideradas e incluso tengan frecuencias relativas cercanas a la recompensa más alta, como es el caso de la acción con la recompensa de 1.84 donde donde frecuencia relativa es de 14.872% y la exploración de la acción con la recompensa más baja es del 7.258%.

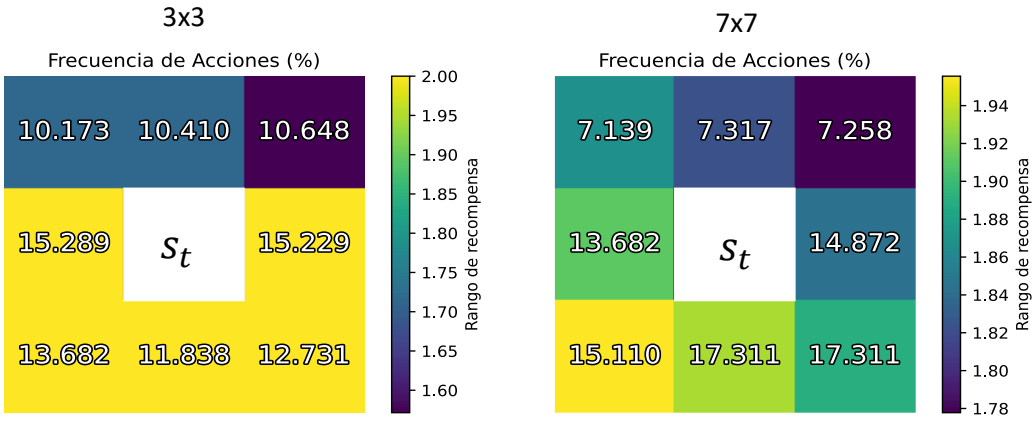


Figura 3.30 Frecuencia relativa de acciones en una época.

Modo inferencia

El entrenamiento está completo, y ahora comienza la inferencia. En este entorno, las predicciones que se van a hacer son las acciones por realizar para ir desde el principio (lugar C) hasta el final (lugar D). Cuando se obtiene un determinado estado s_t, se realiza la acción a_t que tenga el valor Q más alto para ese estado s_t:

$$a_t = \arg_a \max (Q(s_t, a)) \tag{3.3}$$

Al hacer esto en cada estado, se llega a un destino por la ruta más corta.

3.2.3 Algoritmo de Bézier

Para calcular Bézier (Sederberg 2012) se siguen los siguientes pasos:

1. Definir los puntos de control: El primer paso es definir los puntos de control que definen la forma de la curva. Estos puntos son las coordenadas de ruta 2D generadas por Aprendizaje-Q.
2. Definir el grado de la curva: El grado de la curva es el número de puntos de control menos uno. Por ejemplo, si hay cuatro puntos de control, el grado de la curva será tres.
3. Definir la función de base de Bézier: La función de base de Bézier para el i-ésimo punto de control de una curva de grado n se puede calcular utilizando la fórmula matemática (2.23).
4. Evaluar la curva de Bézier: Para evaluar la curva de Bézier, se calcula la suma ponderada de los puntos de control, donde los pesos se determinan a partir de las funciones de base de Bézier. La fórmula para calcular la curva de Bézier es (2.22), donde t es un parámetro que varía de 0 a 1.
5. Calcular los puntos de la curva de Bézier: La curva de Bézier se aproxima calculando los valores de $P_{[t_0,t_1]}$ para un conjunto discreto de valores de t. Cuanto más pequeño sea el intervalo entre los valores de t, más suave será la curva de Bézier.

3.2.4 Generar B-spline usando el algoritmo de De Boor

El algoritmo de De Boor (De Boor 1972) para generar una curva B-spline a partir de un conjunto de puntos consta de los siguientes pasos:

1. Definir los puntos de control: El primer paso es definir los puntos de control que definen la forma de la curva. Estos puntos son las coordenadas de ruta 2D generadas por Aprendizaje-Q.
2. Definir el grado de la curva: El grado de la curva B-spline es un parámetro que determina la complejidad de la curva. Seleccionar un grado bajo resulta en curvas suaves, mientras que un grado alto resulta en curvas más complejas.
3. Definir la función de base de B-spline: Las funciones de base B-spline se define recursivamente utilizando el algoritmo de De Boor. Cada función de base depende del grado de la curva y del número de puntos de control.
4. Evaluar la curva B-spline: Para evaluar la curva de B-spline, se calcula la suma ponderada de los puntos de control, donde los pesos se determinan a partir de las funciones de base B-spline. La fórmula para calcular la curva B-spline es (2.24).
5. Calcular los puntos de la curva B-spline: La curva B-spline se aproxima calculando los valores de s(u) para un conjunto discreto de valores de u. Cuanto más pequeño sea el intervalo entre los valores de u, más suave será la curva B-spline.

3.2.5 Cálculo del ángulo yaw

El cálculo del ángulo *yaw* se realiza con referencia al marco de referencia fijo del cuerpo del cuadricóptero, considerándolo como un plano cartesiano (ver Figura 3.31). En primer lugar, se debe identificar el cuadrante en el que se encuentra el vector de dirección del cuadricóptero y luego se aplican diferentes funciones trigonométricas para calcular el *yaw* correspondiente a cada cuadrante. A continuación, se detalla el método utilizado para obtener el ángulo *yaw* en cada uno de los cuadrantes.

Cuadrante 1: El ángulo de *yaw* en el primer cuadrante se calcula como el ángulo entre el vector de dirección y el eje x positivo. Para encontrar este ángulo, se utiliza la formula

$arctan\left(\frac{\Delta y}{\Delta x}\right)$, donde Δy es el cambio en la coordenada y y Δx es el cambio en la coordenada x.

Cuadrante 2: En el segundo cuadrante se calcula como el ángulo entre el vector de dirección y el eje x negativo. Para encontrar este ángulo, se utiliza la fórmula $180 - arctan\left(\frac{|\Delta y|}{|\Delta x|}\right)$.

Cuadrante 3: En el tercer cuadrante se calcula como el ángulo entre el vector de dirección y el eje x negativo, pero esta vez el ángulo es negativo. Para encontrar este ángulo, se utiliza la fórmula $-180 + arctan\left(\frac{\Delta y}{\Delta x}\right)$.

Cuadrante 4: En el cuarto cuadrante se calcula como el ángulo entre el vector de dirección y el eje x positivo. Para encontrar este ángulo, se utiliza la fórmula $-arctan\left(\frac{\Delta y}{\Delta x}\right)$.

La función trigonométrica *arctan* produce un ángulo en radianes, por lo que para obtener el resultado en grados, es necesario convertir el valor usando la fórmula $yaw_{grados} = yaw_{radianes} \times \frac{180}{\pi}$.

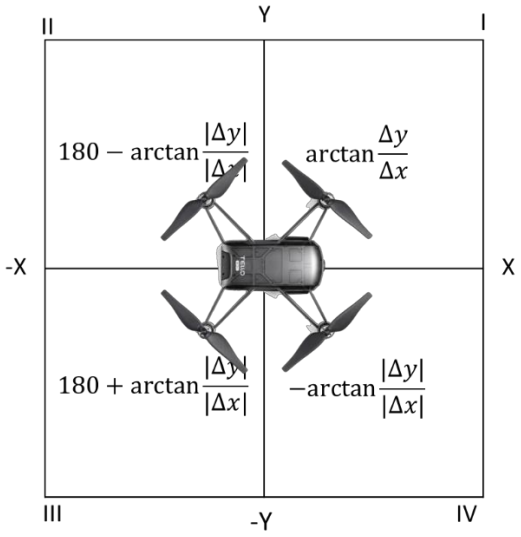


Figura 3.31 ángulo *yaw* en el plano cartesiano

3.2.6 Retardo temporal entre las publicaciones de puntos de ruta

El retardo se utiliza para establecer el intervalo de tiempo entre las publicaciones de los puntos de ruta, lo que a su vez afecta la velocidad a la que el cuadricóptero sigue la

trayectoria en la simulación. Al establecer un valor adecuado para el retardo se consigue una velocidad a la que el cuadricóptero sigue la trayectoria de manera suave y consistente.

Por lo tanto, este retardo debe elegirse teniendo en cuenta la velocidad máxima del dron y la complejidad de la ruta a seguir, para que el dron pueda seguir correctamente la trayectoria en la simulación.

3.3 SEGUIMIENTO DE LA TRAYECTORIA EN SIMULACIÓN

En esta sección se describe el proceso de seguimiento de trayectorias en simulación, que abarca la publicación de los puntos de ruta y la captura de datos de odometría para medir la precisión del seguimiento mediante la métrica de RMSE. El proceso se ilustra en la Figura 3.32.

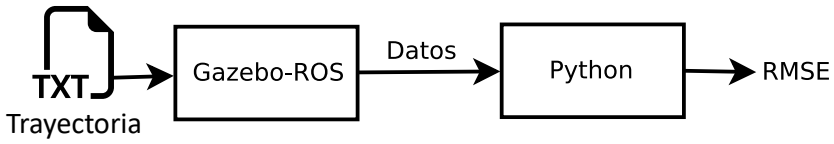


Figura 3.32 Diagrama a bloques RMSE.

3.3.2 Publicación de los puntos de ruta en el controlador del cuadricoptero

En este proceso, la ruta generada en formato txt, se utiliza como entrada al nodo ROS *waypoint_publisher_file,* el cual se encarga de publicar los puntos de ruta en el controlador del cuadricóptero IRIS simulado en Gazebo. El archivo de texto (Tabla 3.7) contiene el vector de retardo, coordenadas, x, y, z, y yaw, que conforman los puntos de ruta de la trayectoria. Además, la Figura 3.33 muestra cómo se realiza esta publicación de los puntos de ruta en el controlador del cuadricóptero IRIS.

Retardo	X	Y	Z	Yaw
0.1	-8	8	3	-81.9455
0.1	-7.9653	7.7547	3	-63.5389
0.1	-7.8722	7.5678	3	-46.1898
0.1	-7.7351	7.4249	3	-33.1621
0.1	-7.5658	7.3142	3	-24.6546
0.1	-7.3739	7.2261	3	-19.5255
0.1	-7.1672	7.1528	3	-16.6806
0.1	-6.9517	7.0883	3	-15.3452
0.1	-6.732	7.028	3	-14.9934
0.1	-6.5112	6.9688	3	-15.2582
0.1	-6.2912	6.9088	3	-15.8769
0.1	-6.0732	6.8468	3	-16.6627

Tabla 3.7 Puntos de ruta de la trayectoria.

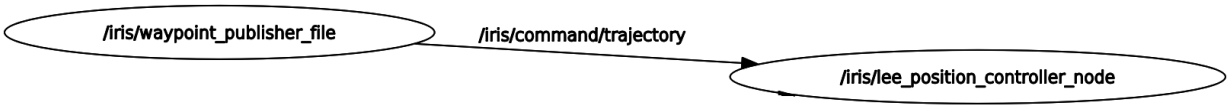


Figura 3.33 Nodos ROS para la publicación de puntos de ruta.

3.3.3 Captura y análisis de los datos de Odometría

Durante la simulación, se capturaron datos de odometría en formato *bagfile* de ROS que proporcionan información precisa sobre la posición y orientación del cuadricóptero. La Tabla 3.8 muestra las coordenadas x, y, z en las columnas 2 a 4, mientras que las columnas 5 a 9 contienen la orientación en forma de quaternions. Para analizar la orientación en forma de vector eje-ángulo, se utilizó la ecuación (2.17). Es importante destacar que estos datos de odometría son generados por el modelo matemático del simulador y no presentan ruido.

Time	position.x	position.y	position.z	orientation.x	orientation.y	orientation.z	orientation.w
21.217	-5.374116	5.5	1.977984315	-1.23E-08	4.11E-08	1.84E-09	1
21.217	-5.374116	5.5	1.977984315	-1.23E-08	4.12E-08	1.85E-09	1
21.217	-5.374116	5.5	1.977984315	-1.22E-08	4.13E-08	1.85E-09	1
21.217	-5.374116	5.5	1.977984315	-1.22E-08	4.15E-08	1.85E-09	1
21.217	-5.374116	5.5	1.977984315	-1.21E-08	4.16E-08	1.86E-09	1
21.217	-5.374116	5.5	1.977984315	-1.21E-08	4.17E-08	1.86E-09	1
21.217	-5.374116	5.5	1.977984315	-1.20E-08	4.19E-08	1.86E-09	1
21.217	-5.374116	5.5	1.977984315	-1.19E-08	4.20E-08	1.87E-09	1
21.218	-5.374116	5.5	1.977984315	-1.19E-08	4.21E-08	1.87E-09	1
21.219	-5.374116	5.5	1.977984315	-1.18E-08	4.22E-08	1.87E-09	1
21.22	-5.374116	5.5	1.977984315	-1.18E-08	4.24E-08	1.88E-09	1

Tabla 3.8 Datos de odometría.

3.4 MÉTRICAS DE ERROR

En esta sección, abordaremos las métricas de error utilizadas para evaluar la precisión de la navegación autónoma del UAV durante el seguimiento de la trayectoria y medir la suavidad de la trayectoria generada por diferentes métodos. A continuación, se describe la metodología para calcularlas.

3.4.1 RMSE

Para evaluar la precisión de la navegación autónoma del UAV durante el seguimiento de la trayectoria para los ejes x e y, y para el ángulo *yaw*, se utilizó la métrica RMSE, la cual se calculó mediante la comparación de la posición u orientación esperada según la trayectoria generada con la posición u orientación observada en los datos de odometría.

Para calcular el RMSE para el ángulo yaw de la trayectoria se utiliza la ecuación (2.32). Sin embargo, para los ejes x e y, la ecuación se adapta de la siguiente manera:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}((x_i - \lambda(X_i))^2 + (y_i - \lambda(Y_i))^2)}{n}} \tag{3.4}$$

donde x_i e y_i son el valor objetivo verdadero para las instancias de prueba X_i e Y_i, $\lambda(X_i)$ y $\lambda(Y_i)$ son el valor objetivo predicho para las instancias de prueba X_i e Y_i, y n es el número de instancias de prueba.

3.4.2 Porcentaje de error

Para evaluar la precisión de la navegación autónoma del UAV durante el seguimiento de la trayectoria para los ejes x e y, y para el ángulo *yaw*, se utilizó la métrica de porcentaje del error, la cual se calculó mediante la comparación de la posición u orientación esperada según la trayectoria generada con la posición u orientación observada en los datos de odometría.

Para calcular el porcentaje de error para el ángulo yaw de la trayectoria se utiliza la ecuación (2.31). Sin embargo, para los ejes x e y, la ecuación se adapta de la siguiente manera:

$$\%error = \frac{(|\lambda(X) - x|) - (|\lambda(Y) - y|)}{|x| + |y|} \times 100 \tag{3.5}$$

donde x e y son el valor objetivo verdadero para las instancias de prueba X e Y, mientras que $\lambda(X)$ y $\lambda(Y)$ son el valor objetivo predicho para las instancias de prueba X e Y.

3.4.3 Métricas de error para medir la suavidad de la trayectoria generada

Con el objetivo de medir la suavidad de la trayectoria generada y establecer una referencia que permita comparar la similitud entre las trayectorias generadas por diferentes métodos, calculamos el RMSE y el porcentaje de error para la trayectoria generada con B-spline de grado 10, tomando como referencia la trayectoria generada con B-spline de grado 1.

3.4.4 Submuestreo con B-spline usando el algoritmo de Dierckx

Dado que la trayectoria esperada se configura para mantenerse constante en mil puntos, mientras que la trayectoria observada contiene miles de puntos, porque la frecuencia de muestreo de la grabación de los datos de odometría en el tópico *iris/pose* es de mil muestras por segundo.

Con el objetivo de calcular el RMSE y el porcentaje de error, es necesario que el número de muestras en la trayectoria esperada (Tabla 3.7) coincida con el número de muestras en los datos de odometría (Tabla 3.8).

Para lograr esto, se llevó a cabo un submuestreo utilizando el algoritmo de Dierckx (Dierckx 1975), el cual se encuentra incluido en la librería de programación Python *scipy.interpolate.*

Se siguen estos pasos para realizar el submuestreo:

1. Se generan dos vectores de puntos, uno con el tamaño de los datos de odometría y otro con el tamaño de los datos esperados. Estos vectores se utilizan para evaluar el spline que se utiliza en la interpolación de los datos de odometría y en los esperados, respectivamente.
2. Se utilizó la función *splrep* de la misma librería para ajustar un spline a los datos de odometría. Luego, se evaluó el spline en el vector de los datos esperados generado en el paso anterior.

Capítulo 4: Experimentos

Este capítulo se centra en la realización de cuatro experimentos con el propósito de comprender la relación entre el error y la velocidad del cuadricóptero, así como investigar la influencia de otros factores, como el entorno, el método de interpolación y el grado de B-spline. Cabe destacar que al generar trayectorias con el algoritmo de Aprendizaje-Q, se lleva a cabo una comparativa no solo empleando el método convencional, sino que también se evalúan dos métodos: muestreo de Thompson y distribución Gaussiana, para modificar la política utilizada en el Aprendizaje-Q para la selección de la acción que llevara al estado siguiente utilizando la ecuación de la diferencia temporal.

4.1 Experimento para comparar la relación entre el RMSE y la velocidad del cuadricóptero en un entorno circular con Bézier y B-spline.

En esta sección, se presenta un experimento para comparar la precisión del cuadricóptero durante el seguimiento de trayectoria en un entorno circular utilizando dos métodos de interpolación, Bézier y B-spline de grado tres. Para ello, las trayectorias se generan seleccionando ochenta y ocho puntos de control alrededor del círculo, para luego interpolarlos usando mil puntos. La evaluación de la relación entre el RMSE y la velocidad del cuadricóptero en simulación se lleva a cabo mediante la disminución progresiva del retardo temporal, desde cincuenta hasta siete milisegundos. Con este experimento se determina cuál de los dos métodos de interpolación ofrece un mejor desempeño de precisión en el seguimiento de trayectoria en un entorno circular, considerando la velocidad del cuadricóptero.

Generación del entorno circular

El experimento se lleva a cabo en un entorno de $40x40$ pixeles e incluye un obstáculo con forma de círculo discretizado en su centro (Figura 4.34). Este círculo se creó a partir de un cuadrado de $10x10$ pixeles al que se le eliminaron las esquinas para crear una forma circular.

4.1.1 Relación entre el RMSE y la velocidad del cuadricóptero en un entorno circular con Bézier

Con el objetivo de explorar la relación entre el RMSE y la velocidad del cuadricóptero en un entorno circular con Bézier, se llevó a cabo la grabación de datos de odometría del cuadricóptero durante el seguimiento de la trayectoria alrededor de un obstáculo circular, con diez retardos temporales diferentes $50, 40, 30, 20, 15, 12, 10, 9, 8\ y\ 7$ milisegundos.

Generación de la trayectoria

La trayectoria alrededor de un obstáculo circular (Figura 4.34) se generó seleccionando ochenta y ocho puntos de control alrededor del círculo, para luego interpolarlos con Bézier usando mil puntos. La trayectoria resultante se desvía dieciséis puntos de control en cada esquina con curvas cerradas.

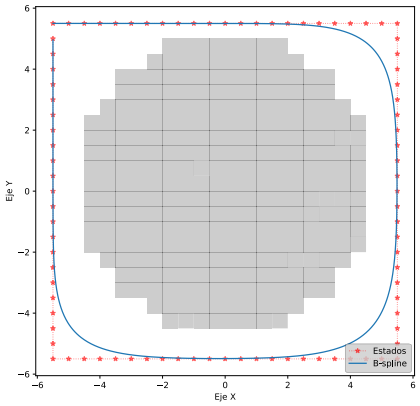


Figura 4.34 Entorno con un obstáculo circular y trayectoria generada con Bézier.

Análisis de la relación entre la velocidad y el retardo temporal de la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria

Durante el seguimiento de la trayectoria del cuadricóptero, se llevó a cabo un análisis de la velocidad en función del retardo. Los resultados se presentan en la Tabla 4.9, que muestra la velocidad máxima y media del cuadricóptero para diferentes valores de retardo, así como el incremento en velocidad media entre retardos.

Los datos indican que a medida que el retardo disminuye de cincuenta milisegundos a siete milisegundos, tanto la velocidad máxima como la media del cuadricóptero aumentan.

En particular, la velocidad máxima pasa de $0.870\frac{m}{s}$ a $5.612\frac{m}{s}$, mientras que la velocidad media aumenta de $0.820\frac{m}{s}$ a $5.101\frac{m}{s}$. Cabe mencionar que el cuadricóptero colisiona con un retardo menor a $7ms$.

Retardo(ms)	Velocidad media(m/s)	Velocidad maxima(m/s)
50	0.822733	0.870361
40	1.026406	1.089589
30	1.360483	1.447463
20	2.008926	2.166962
15	2.62268	2.870535
12	3.146976	3.581792
10	3.781933	4.259485
9	4.154583	4.682289
8	4.52867	5.095577
7	5.101043	5.612318

Tabla 4.9 Relación entre la velocidad y el retardo en la publicación de puntos de ruta con Bézier.

En la Figura 4.35 se muestra un gráfico que representa la velocidad (en metros sobre segundo) del cuadricóptero durante el seguimiento de la trayectoria en función del retardo medido en milisegundos. Se muestran diferentes curvas para diferentes velocidades del cuadricóptero y se observa que las curvas siguen una tendencia creciente a medida que disminuye el retardo. Además, se puede notar que las curvas reflejan las características de la trayectoria alrededor del obstáculo circular, mostrando una disminución de velocidad en las zonas donde la trayectoria se desvía más de la línea recta y una recuperación de velocidad en las zonas más rectas. También se puede observar que a medida que incrementa la velocidad la disminución de velocidad en estas zonas se aproxima más a $1\frac{m}{s}$.

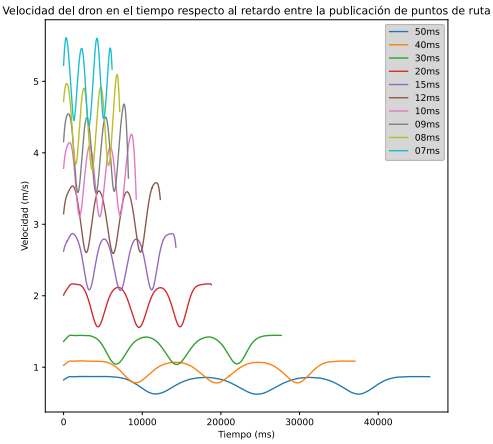


Figura 4.35 Velocidad del dron en el tiempo respecto al retardo con Bézier.

Análisis de la relación entre el RMSE y el retardo temporal de la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria

La Tabla 4.10 muestra la relación entre el retardo entre la publicación de puntos de ruta y el RMSE para el ángulo yaw. En el caso del ángulo yaw, el RMSE se mide en grados en el rango de 0° a 360°. Se observa que conforme disminuye el retardo de $50ms$ a $8ms$ el RMSE en el ángulo yaw aumenta de 5.688° a 33.474°, para luego decrementar a 30.796° con un retardo de $7ms$. Se encontró que el RMSE mínimo de 5.688° se obtiene con un retardo de $50ms$, mientras que el RMSE mayor de 33.474° se obtiene con un retardo de $8ms$.

Retardo(ms)	RMSE
50	**5.688154**
40	6.907722
30	9.083279
20	13.1766
15	17.1883
12	21.97799
10	26.43901
9	30.31053
8	33.47469
7	30.79693

Tabla 4.10 Relación entre el retardo y el RMSE para el ángulo yaw.

La Figura 4.36 muestra la comparación entre el ángulo yaw esperado y observado en los datos de odometría, en los valores de retardo donde el RMSE es más bajo y alto, $50ms$ y $8ms$ respectivamente. Se puede observar que en $50ms$ el ángulo yaw esperado y observado tienen un desfase poco pronunciado siendo el error promedio de $5.688°$, mientras que en $8ms$ el ángulo yaw observado se desfasa de lo esperado notoriamente siendo el error promedio de $33.474°$

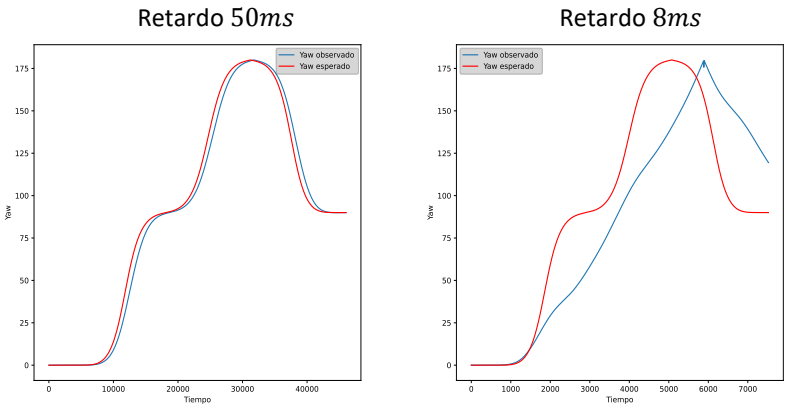


Figura 4.36 Trayectoria observada y esperada en el ángulo Yaw.

Por otro lado, para los ejes x e y el RMSE se mide en metros. Se observa que conforme aumenta la velocidad máxima de $50ms$ a $7ms$ el RMSE en la navegación autónoma del cuadricóptero durante seguimiento de la trayectoria incrementa de $0.028m$ a $0.469m$. Se encontró que el RMSE mínimo de $0.028m$ se obtiene con un retardo temporal de $50ms$, mientras que el RMSE mayor de $0.469m$ se obtiene con un retardo temporal de $7ms$.

Retardo(ms)	RMSE
50	**0.028719**
40	0.038388
30	0.056505
20	0.090461
15	0.145165
12	0.259695
10	0.290199
9	0.354426
8	0.329044
7	0.46915

Tabla 4.11 Relación entre la velocidad máxima y el RMSE para los ejes XY.

La Figura 4.37 muestra la comparación entre las trayectorias esperadas y observadas en los datos de odometría para los ejes x e y, en los valores donde el RMSE es más bajo y alto, $50ms$ y $7ms$ respectivamente. Se puede notar que en $50ms$ la trayectoria esperada y observada se desfasan $0.028m$, mientras que en $7ms$ la trayectoria observada se desfasa de lo esperado notoriamente $0.469m$ con curvas cerradas.

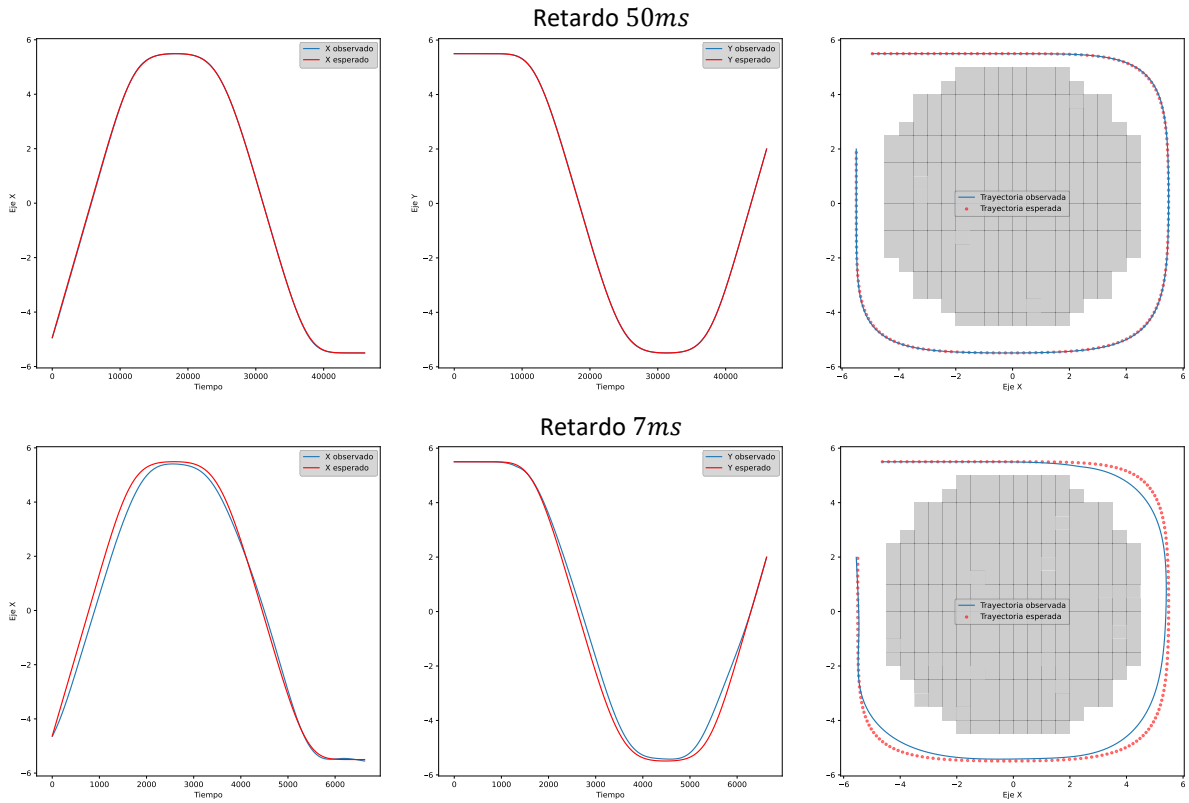


Figura 4.37 Trayectoria observada y esperada en los ejes XY.

4.1.2 Relación entre el RMSE y la velocidad del cuadricóptero en un entorno circular con B-spline de grado tres

Con el objetivo de explorar la relación entre el RMSE y la velocidad del cuadricóptero en un entorno circular con B-spline de grado tres, se llevó a cabo la grabación de datos de odometría del cuadricóptero durante el seguimiento de la trayectoria alrededor de un obstáculo circular con diez retardos temporales diferentes $50, 40, 30, 20, 15, 12, 10, 9, 8\ y\ 7$ milisegundos.

Generación de la trayectoria

La trayectoria alrededor de un obstáculo circular (Figura 4.38) se generó seleccionando ochenta y ocho puntos de control alrededor del círculo, para luego interpolarlos con B-spline de grado tres usando mil puntos. La trayectoria resultante se desvía de un punto de control en cada esquina con curvas poco pronunciadas.

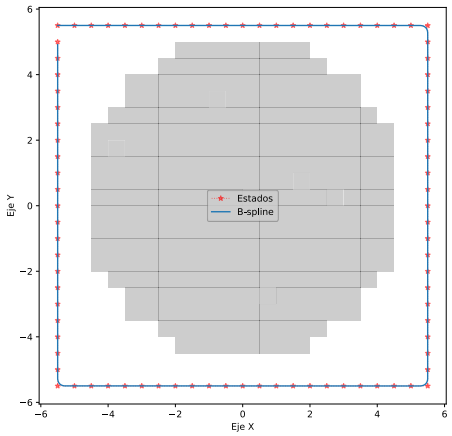


Figura 4.38 Entorno con un obstáculo circular y trayectoria generada con B-spline.

Análisis de la relación entre la velocidad y el retardo temporal de la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria

Durante el seguimiento de la trayectoria del cuadricóptero, se llevó a cabo un análisis de la velocidad en función del retardo entre la publicación de puntos de ruta. Los resultados se presentan en la Tabla 4.12, que muestra la velocidad máxima y media del cuadricóptero para diferentes valores de retardo.

Los datos indican que a medida que el retardo disminuye de cincuenta milisegundos a siete milisegundos, tanto la velocidad máxima como la media del cuadricóptero aumentan. En particular, la velocidad máxima pasa de $0.933\frac{m}{s}$ a $6.559\frac{m}{s}$, mientras que la velocidad media aumenta de $0.859\frac{m}{s}$ a $5.452\frac{m}{s}$. Cabe mencionar que el cuadricóptero colisiona con un retardo menor a $7ms$.

Retardo(ms)	Velocidad media(m/s)	Velocidad maxima(m/s)
50	0.859294	0.933697
40	1.073798	1.131079
30	1.428613	1.479995
20	2.135858	2.203196
15	2.805893	2.915126
12	3.431319	3.634142
10	4.037121	4.355896
9	4.406632	4.866254
8	4.7925	5.437905
7	5.452781	6.559884

Tabla 4.12 Relación entre la velocidad y el retardo en la publicación de puntos de ruta con B-spline.

En la Figura 4.39 se muestra un gráfico que representa la velocidad (en metros sobre segundo) del cuadricóptero durante el seguimiento de la trayectoria en función del retardo medido en milisegundos. Se muestran diferentes curvas para diferentes velocidades del cuadricóptero y se observa que las curvas siguen una tendencia creciente a medida que disminuye el retardo. Además, se puede notar que las curvas reflejan las características de la trayectoria alrededor del obstáculo circular, mostrando una disminución de velocidad en las zonas donde la trayectoria se desvía más de la línea recta y una recuperación de velocidad en las zonas más rectas. También se puede observar que a medida que incrementa la velocidad la disminución de velocidad en estas zonas se aproxima más a $1.5\frac{m}{s}$.

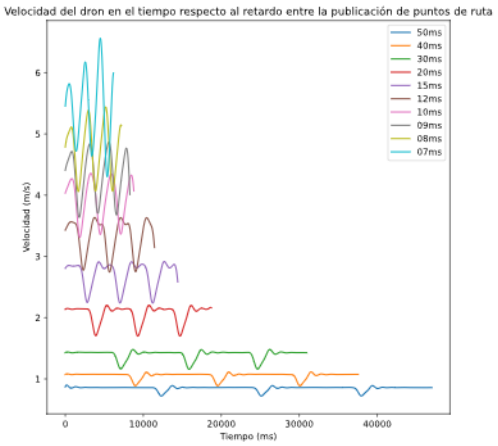


Figura 4.39 Velocidad del dron en el tiempo respecto al retardo con B-spline.

Análisis de la relación entre el RMSE y el retardo de la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria

La Tabla 4.13 muestra la relación entre el retardo entre la publicación de puntos de ruta y el RMSE para el ángulo yaw. En el caso del ángulo yaw, el RMSE se mide en grados en el rango de 0° a 360°. Se observa que conforme disminuye el retardo de $50ms$ a $8ms$ el RMSE en el ángulo yaw aumenta de 15.135° a 27.821°, para luego decrementar a 21.986° con un retardo de $7ms$. Se encontró que el RMSE mínimo de 15.688° se obtiene con un retardo de $50ms$, mientras que el RMSE mayor de 27.821° se obtiene con un retardo de $8ms$.

Retardo(ms)	RMSE
50	**15.1351**
40	16.01149
30	18.04247
20	20.54131
15	22.19435
12	23.69988
10	24.59877
9	27.76133
8	27.82151
7	21.98664

Tabla 4.13 Relación entre el retardo y el RMSE para el ángulo yaw.

La Figura 4.40 muestra la comparación entre el ángulo yaw esperado y el ángulo yaw observado en los datos de odometría del ángulo yaw, en los valores de retardo donde el RMSE es más bajo y alto, 50ms y 8ms respectivamente. Se puede observar que en 50ms la trayectoria esperada y observada tienen un desfase de 15.135° en promedio, mientras que en $8ms$ la trayectoria observada se desfasa de lo esperado notoriamente 27.821° en promedio.

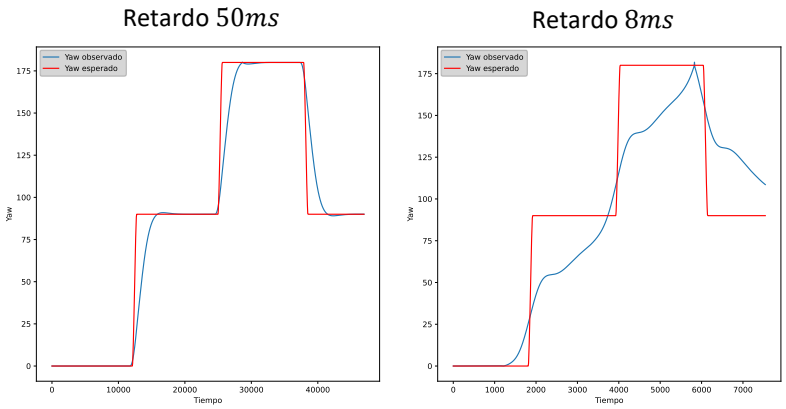


Figura 4.40 Trayectoria observada y esperada en el ángulo yaw.

Por otro lado, para los ejes x e y el RMSE se mide en metros. Se observa que en el decremento de retardo de $50ms$ a $40ms$ el RMSE decrementa de $0.105m$ *a* $0.040m$, para luego incrementar de $0.040m$ a $0.632m$ conforme disminuye el retardo de 40ms a 7ms. Se encontró que el RMSE mínimo de $0.040m$ se obtiene con un retardo temporal de $40ms$, mientras que el RMSE mayor de $0.632m$ se obtiene con un retardo temporal de $7ms$.

Retardo(ms)	RMSE
50	0.10575
40	**0.040039**
30	0.056593
20	0.095153
15	0.137935
12	0.195513
10	0.241449
9	0.295448
8	0.374477
7	0.632832

Tabla 4.14 Relación entre el retardo y el RMSE para los ejes XY.

La Figura 4.41 muestra la comparación entre las trayectorias esperadas y observadas en los datos de odometría para los ejes x e y, en los valores donde el RMSE es más bajo y alto, $40ms$ y $7ms$ respectivamente. Se puede observar que en $40ms$ la trayectoria esperada y observada se desfasan poco solo $0.040m$ en promedio, mientras que en $7ms$ la trayectoria

observada se desfasa de lo esperado notoriamente con curvas abiertas y deformes que tienen $0.632m$ de error en promedio.

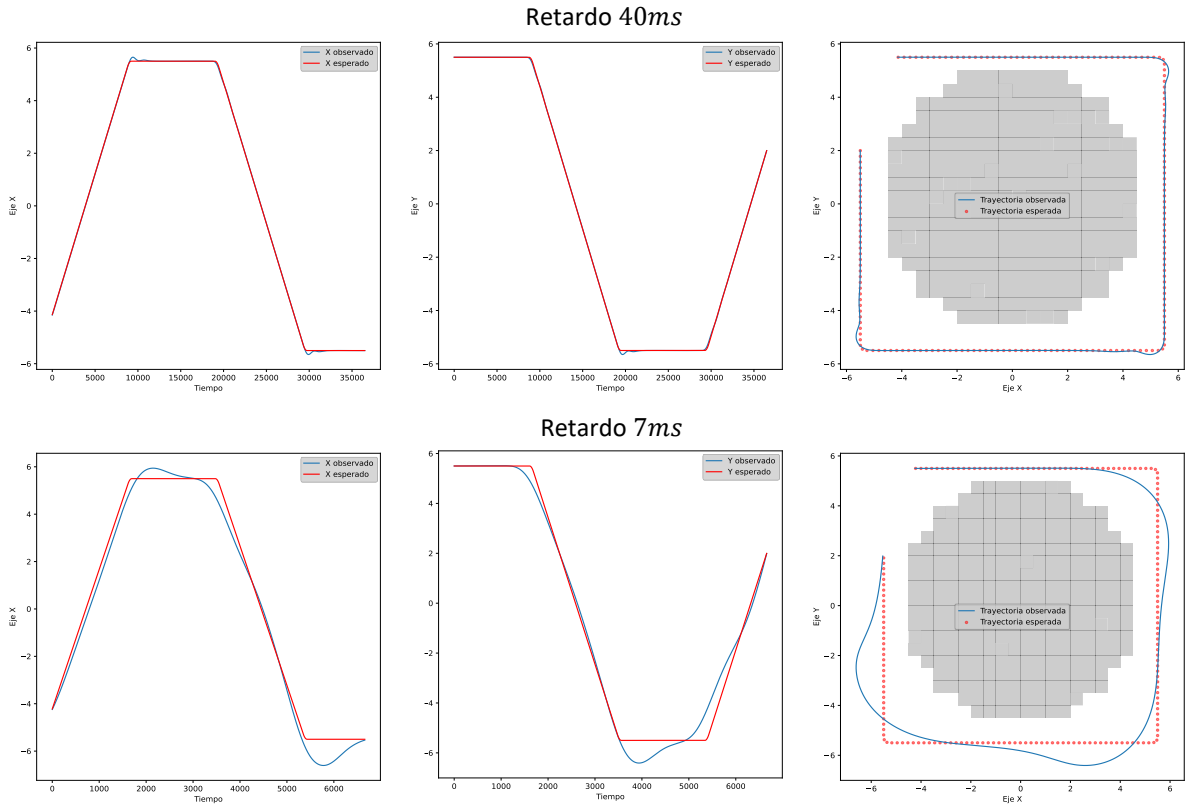


Figura 4.41 Trayectoria observada y esperada en los ejes XY.

4.1.3 Comparacion entre Bézier y B-spline

En la Figura 4.42, se presenta una comparación entre Bézier y B-spline en función de la velocidad máxima y el RMSE para el ángulo yaw en la parte izquierda, y para los ejes x e y en la parte derecha. A continuación, se presentan observaciones de esta comparación.

El RMSE para el ángulo yaw en la parte izquierda de la Figura 4.42, muestra que B-spline tiende a desviarse de Bézier mostrando entre $0.5\frac{m}{s}$ y $1.5\frac{m}{s}$ más en velocidad máxima. Además, cuando la velocidad máxima es inferior a $4\frac{m}{s}$, El RMSE muestra que B-spline se desvía en promedio $10°$ más en comparación con Bézier. Por otro lado, cuando la velocidad máxima supera a $4\frac{m}{s}$, El RMSE muestra que Bézier se desvía en promedio $5°$ más en

comparación con B-spline y alcanza la misma velocidad que B-spline de $5\frac{m}{s}$. Sin embargo, B-spline finaliza con una velocidad máxima $1\frac{m}{s}$ más alta y un RMSE 8° menor.

El RMSE para los ejes x e y en la parte derecha de la Figura 4.42, se destaca que B-spline comienza con un mayor RMSE, desviándose $0.08m$ más que Bezier de la trayectoria, pero luego se iguala a Bézier hasta velocidades superiores a $3\frac{m}{s}$. A partir de este punto, Bézier mantiene un RMSE $0.05m$ superior hasta velocidades ligeramente inferiores a $5\frac{m}{s}$. Sin embargo, a velocidades más altas B-spline finaliza con una velocidad máxima $1\frac{m}{s}$ más alta y un RMSE $0.1m$ mayor. Es notorio que la curva en esta gráfica sigue una tendencia exponencial.

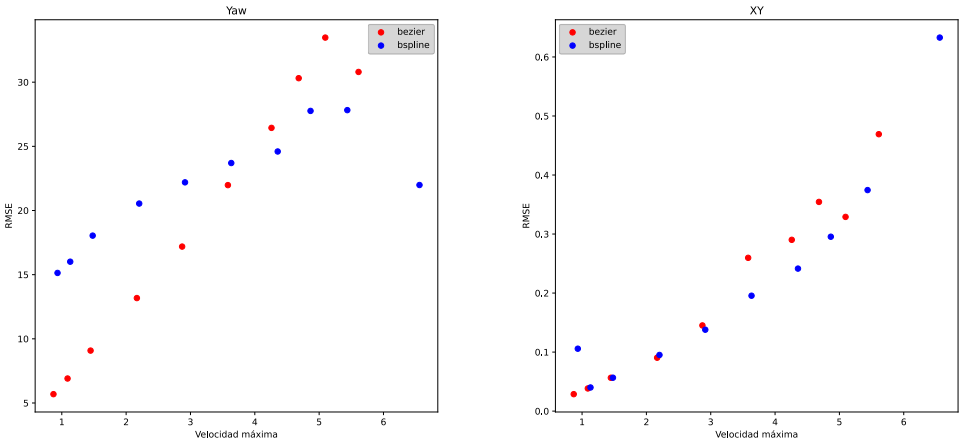


Figura 4.42 Relación entre la velocidad máxima y el RMSE para el ángulo yaw y los ejes XY.

4.1.4 Conclusión

En conclusión, el experimento realizado permitió comparar la precisión del seguimiento de trayectoria utilizando dos métodos de interpolación, Bézier y B-spline de grado tres, en un entorno circular. Para ello, las trayectorias se generan seleccionando ochenta y ocho puntos de control alrededor del círculo, para luego interpolarlos usando mil puntos y se evaluó la relación entre el RMSE y la velocidad del cuadricóptero en simulación mediante la disminución progresiva del retardo temporal.

Durante el análisis de la relación de la velocidad y el retardo temporal, se encontró que conforme el retardo temporal disminuía, la velocidad del cuadricóptero se incrementaba. Además, se observó que la trayectoria observada en ambos métodos se desvía de lo esperado notoriamente a partir de velocidades que están por arriba de $2.166\ \frac{m}{s}$ (decrementos a partir de 20ms). También se pudo observar que conforme aumenta la velocidad, la disminución de velocidad en curvas se aproxima a $1\frac{m}{s}$, este valor puede ser considerado como un límite de seguridad para el cuadricóptero durante el seguimiento de trayectorias con curvas.

Para las trayectorias generadas con Bézier en un entorno circular se encontró un buen control en el rango de velocidades de $0.870\frac{m}{s}$ a $5.09\frac{m}{s}$. Sin embargo, A partir de $5.612\frac{m}{s}$, Bézier mostro dificultades en el control, resultando en una pérdida de control al final de la trayectoria y una velocidad máxima antes de la colisión de $6.2\frac{m}{s}$. Adicionalmente, se observó que las trayectorias generadas con Bézier exhibieron curvas suaves que a medida que aumenta la velocidad a partir de $2.166\frac{m}{s}$ se vuelven progresivamente más cerradas y pronunciadas, generando una desviación de más de $0.1m$ con respecto a la trayectoria deseada. Estos resultados indican que Bézier puede proporcionar un rendimiento sólido en entornos con obstáculos circulares, siempre que la velocidad máxima se mantenga por debajo de $2.166\frac{m}{s}$.

Por otro lado, para las trayectorias generadas con B-spline de grado tres en un entorno circular, se observó un buen control en el rango de velocidades de $0.933\frac{m}{s}$ a $5.437\frac{m}{s}$. Sin embargo, partir de $6.559\frac{m}{s}$, B-spline mostro dificultades en el control, resultando en curvas distorsionadas por la velocidad y una velocidad máxima antes de la colisión de $7.518\frac{m}{s}$. Adicionalmente, Se observó que la falta de suavidad en la trayectoria generada con B-spline de grado tres causa una pérdida de control alrededor de las curvas, que a medida que aumenta la velocidad en un rango de $4.355\frac{m}{s}$ a $6.559\frac{m}{s}$, estas curvas se vuelven más amplias y deformes.

En base a los resultados de la comparación entre Bézier y B-spline, se pude concluir que, para este caso particular de seguimiento de trayectoria en un entorno circular, el método

de Bézier presenta una mejor precisión y menor error que el método de B-spline. Esto se debe a que Bézier muestra mayor suavidad en curvas, lo que facilita un seguimiento más preciso de la trayectoria.

Además, se observó que, en un entorno circular, que es un espacio con espacio para maniobrar en curvas, el RMSE en el ángulo yaw no afecta significativamente al cuadricóptero durante el seguimiento de la trayectoria.

4.2 EXPERIMENTO PARA OBTENER LA DISTANCIA MÍNIMA EN UN ENTORNO CUADRADO PARA BÉZIER Y B-SPLINE.

En esta sección, se presenta un experimento para determinar la distancia mínima y la velocidad máxima a la que se puede generar una trayectoria en un entorno con un obstáculo cuadrado en su centro, utilizando Bézier y B-spline de grado tres. Durante la navegación autónoma del cuadricóptero en el seguimiento de la trayectoria se capturaron datos de odometría para comparar la trayectoria observada con la esperada y calcular el RMSE.

Generación del entorno cuadrado

El experimento se lleva a cabo en un entorno de $40x40$ pixeles e incluye un obstáculo cuadrado de $10x10$ pixeles en su centro (Figura 4.44).

4.2.1 Obtener la distancia mínima en un entorno cuadrado para Bézier

Con el objetivo de obtener la distancia mínima a la que debe de estar el centro del marco fijo del cuadricóptero de un obstáculo, pada poder dar seguimiento a una trayectoria usando Bézier con un retardo de $20ms$ se generaron trayectorias hasta que la curva no colisiona con el obstáculo cuadrado.

El retardo temporal de $20ms$ es donde se obtiene un RMSE menor a $0.1m$ en el entorno circular para los ejes x e y. Además, solo se prueba este retardo temporal porque se observó en el entorno circular con Bézier que los retardos menores hacen que la curva se cierre.

Generación de la trayectoria

En la Figura 4.43 se presentan las trayectorias generadas a distancias de $0.5m$, $1m$ y $1.5m$ alrededor de un obstáculo cuadrado. Para generar estas trayectorias, se seleccionaron 88, 96 y 104 puntos de control alrededor del cuadrado, y se interpolo cada conjunto de puntos con Bézier usando mil puntos. Se observa que la única distancia en la que la trayectoria no colisiona con el obstáculo es $1.5m$. Por lo tanto, se concluye que la distancia mínima a la que se debe generar la trayectoria para usar Bézier es de $1.5m$.

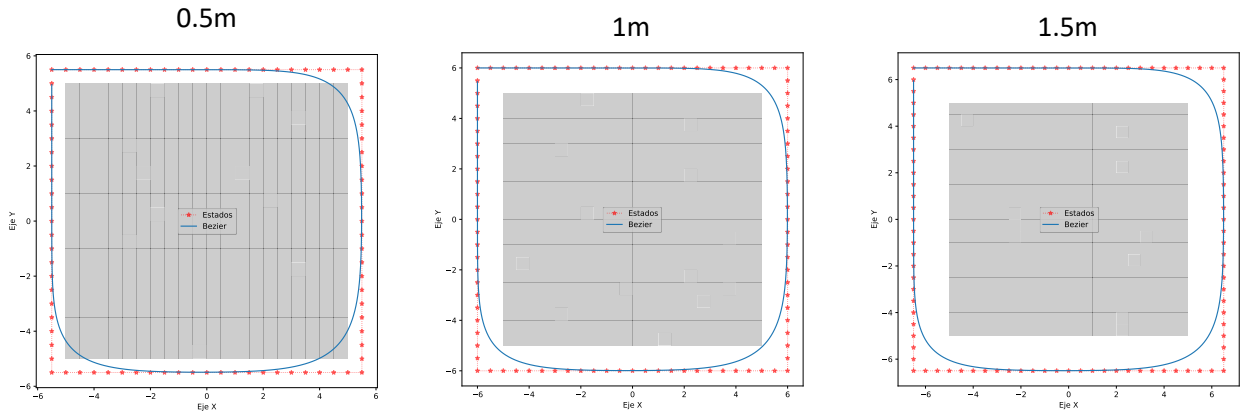


Figura 4.43 Entorno con un obstáculo cuadrado y trayectorias generadas con Bézier.

Análisis de la trayectoria observada

La trayectoria observada no muestra diferencia notoria respecto a la esperada en los ejes x e y, el RMSE indica que los valores de error en la navegación autónoma del cuadricóptero durante seguimiento de la trayectoria se desvían en promedio $0.036m$.

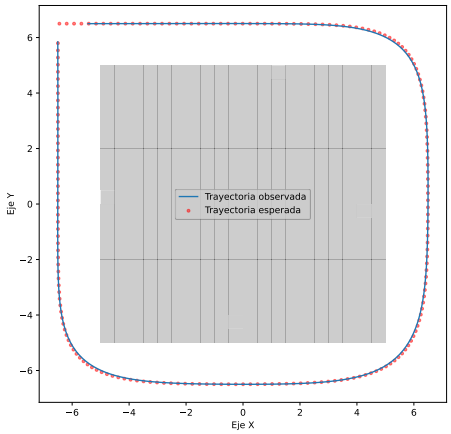


Figura 4.44 Trayectoria observada y esperada.

Análisis de la velocidad durante la trayectoria

En la Figura 4.45, se observa que la duración de la trayectoria es de $19{,}000ms$, la velocidad máxima de $2.582\frac{m}{s}$, la velocidad media de $2.199\frac{m}{s}$ y el decremento en las curvas es de aproximadamente $0.7\frac{m}{s}$.

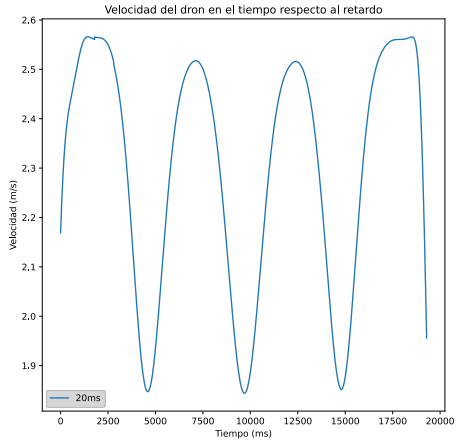


Figura 4.45 Velocidad durante la trayectoria.

4.2.2 Obtener la distancia mínima en un entorno cuadrado con B-spline.

Con el objetivo de obtener la velocidad máxima a la que se puede seguir una trayectoria a medio metro y a un metro del obstáculo utilizando B-spline de grado tres. En la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria se

capturaron datos de odometría, disminuyendo progresivamente el retardo temporal hasta que la velocidad hacia colisionar al cuadricóptero.

Generación de la trayectoria

Se generaron dos trayectorias (Figura 4.46), la primera para probar las velocidades que son posibles a una distancia de medio metro al obstáculo, y la segunda para probar las velocidades que son posibles a una distancia de un metro al obstáculo.

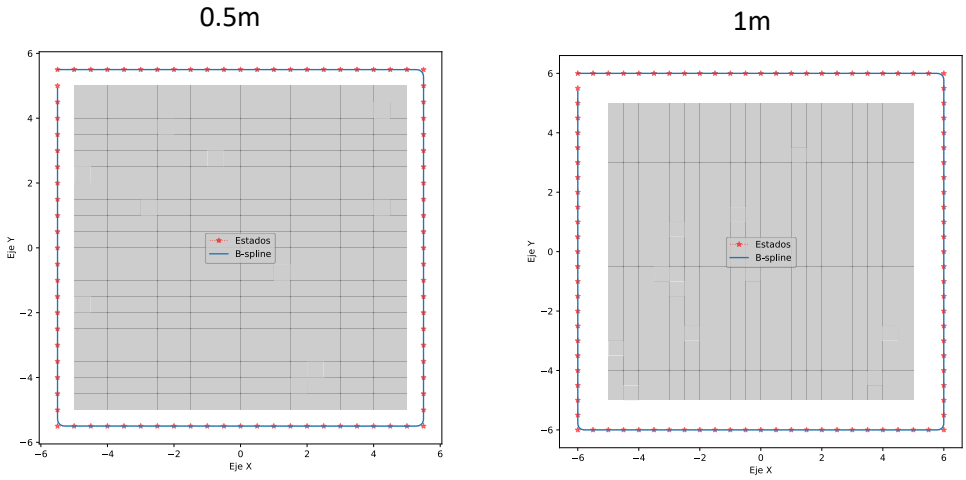


Figura 4.46 Entorno con un obstáculo cuadrado y trayectoria generada con B-spline.

Análisis de la relación entre la velocidad y el retardo temporal de la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria a medio metro del obstáculo

Durante el seguimiento de la trayectoria del cuadricóptero, se llevó a cabo un análisis de la velocidad en función del retardo entre la publicación de puntos de ruta. Los resultados se presentan en la Tabla 4.15, que muestra la velocidad máxima y media del cuadricóptero para diferentes valores de retardo.

Los datos indican que a medida que el retardo disminuye de cincuenta milisegundos a quince milisegundos, tanto la velocidad máxima como la media del cuadricóptero aumentan. En particular, la velocidad máxima pasa de $0.934\frac{m}{s}$ a $2.936\frac{m}{s}$, mientras que la

velocidad media aumenta de $0.859\frac{m}{s}$ a $2.811\frac{m}{s}$. Cabe mencionar que el cuadricóptero colisiona con retardos menores a $15ms$.

Retardo(ms)	Velocidad media(m/s)	Velocidad maxima(m/s)
50	0.859314	0.934844
40	1.073549	1.129932
30	1.430508	1.478056
20	2.134245	2.200991
15	2.811069	2.936901

Tabla 4.15 Relación entre la velocidad y el retardo durante la trayectoria.

En la Figura 4.47 se muestra un gráfico que representa la velocidad del cuadricóptero durante el seguimiento de la trayectoria en función del retardo medido en milisegundos. Se observa cómo las curvas de la trayectoria alrededor del obstáculo cuadrado se reflejan en la gráfica, mostrando un decremento de velocidad que, a medida que aumenta la velocidad del cuadricóptero, se aproxima a $0.5\frac{m}{s}$

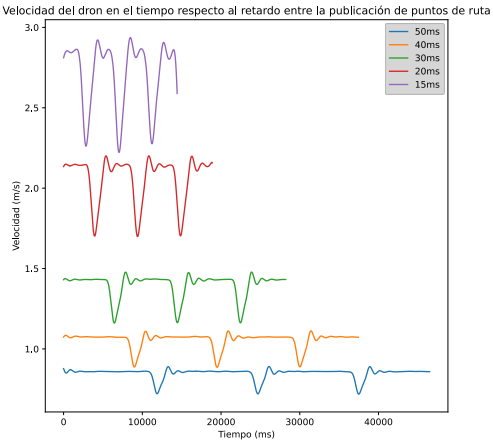


Figura 4.47 Velocidad del dron en el tiempo respecto al retardo con B-spline.

Análisis de la relación entre el RMSE y el retardo de la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria a medio metro del obstáculo

La Tabla 4.16 muestra la relación entre el retardo entre la publicación de puntos de ruta y el RMSE para el ángulo yaw. En el caso del ángulo yaw, el RMSE se mide en grados

en el rango de 0° a 360° y se observa que los valores promedio de error en la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria van de 14.369° a 22.283° de lo esperado conforme aumenta la velocidad máxima.

Retardo(ms)	RMSE
50	**14.36989**
40	16.07374
30	17.97042
20	20.66619
15	22.28302

Tabla 4.16 Relación entre el retardo y el RMSE para el ángulo yaw.

La Figura 4.48 muestra la comparación entre el ángulo yaw esperado y el observado en los datos de odometría, en los valores de retardo donde el RMSE es más bajo y alto, $50ms$ y $15ms$ respectivamente. Se puede observar que en $50ms$ la trayectoria esperada y observada tienen un desfase poco pronunciado, mientras que en $15ms$ la trayectoria observada se desfasa de lo esperado notoriamente.

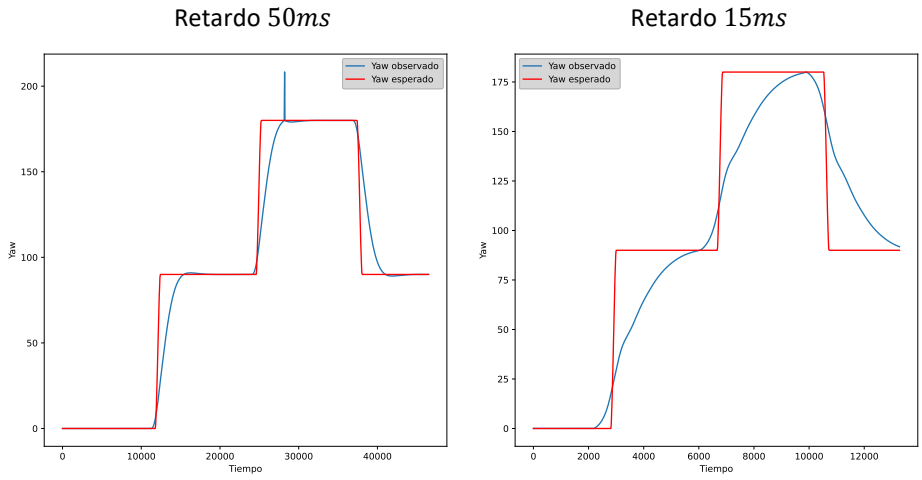


Figura 4.48 Trayectoria observada y esperada en el ángulo Yaw.

Por otro lado, para los ejes x e y el RMSE se mide en metros. En promedio, los valores de error en la navegación autónoma del UAV durante el seguimiento de la trayectoria se desvían de $0.031m$ a $0.140m$. Se encontró que el RMSE mínimo de $0.031m$ se obtiene con

un retardo de $50ms$, mientras que el RMSE mayor de $0.140m$ se obtiene con un retardo de $15ms$.

Retardo(ms)	RMSE
50	**0.031575**
40	0.041852
30	0.055571
20	0.095093
15	0.140861

Tabla 4.17 Relación entre el retardo y el RMSE en los ejes XY.

En la Figura 4.49 muestra la comparación entre las trayectorias esperadas y observadas en los datos de odometría para los ejes x e y, en los valores donde el RMSE es más bajo y alto, $50ms$ y $15ms$ respectivamente. Se puede observar que en $50ms$ la trayectoria esperada y observada se desfasan poco solo $0.031m$ en promedio, mientras que en $7ms$ la trayectoria observada se desfasa de lo esperado notoriamente con curvas que se cierran al inicio y se abren al final con $0.140m$ de error en promedio.

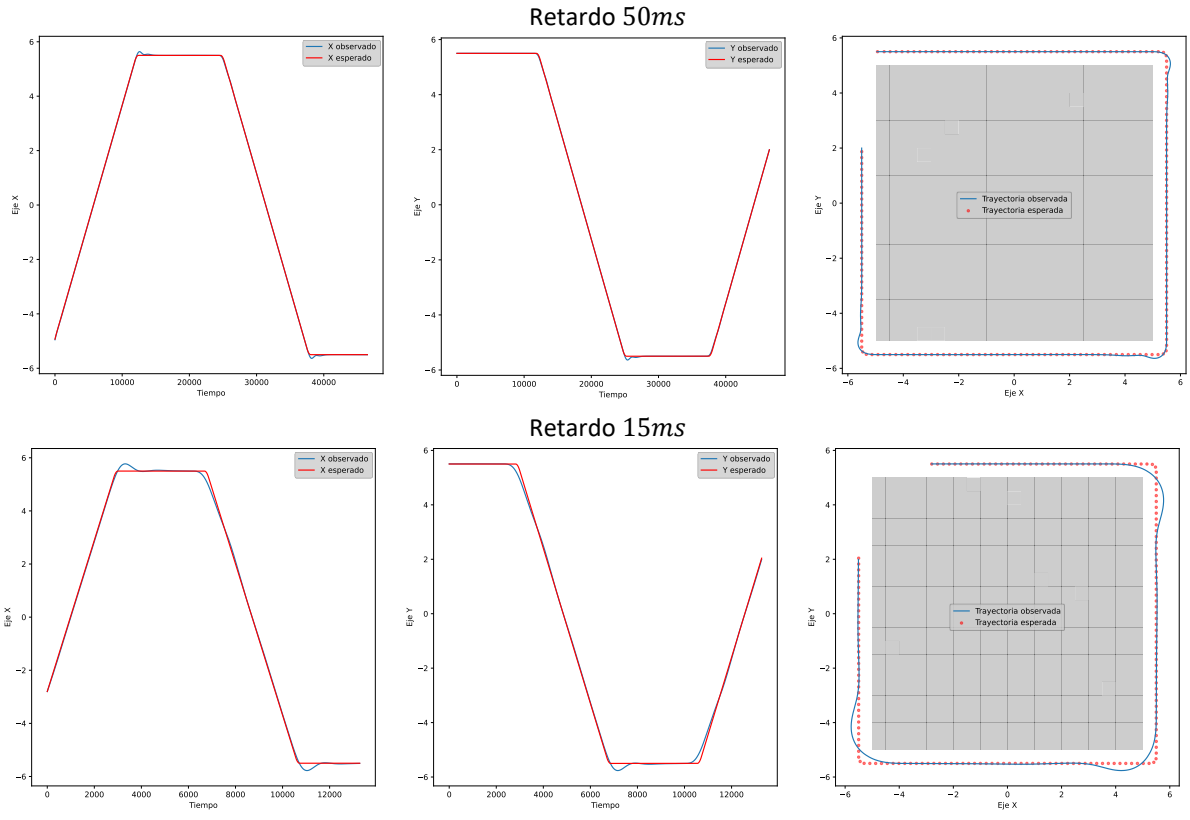


Figura 4.49 Trayectoria observada y esperada en los ejes XY.

Análisis de la relación entre la velocidad y el retardo temporal de la navegación autónoma del UAV durante el seguimiento de la trayectoria a un metro del obstáculo

Durante el seguimiento de la trayectoria del cuadricóptero, se llevó a cabo un análisis de la velocidad en función del retardo entre la publicación de puntos de ruta. Los resultados se presentan en la Tabla 4.18, que muestra la velocidad máxima y media del cuadricóptero para diferentes valores de retardo.

Los datos indican que a medida que el retardo disminuye de doce milisegundos a ocho milisegundos, tanto la velocidad máxima como la media del cuadricóptero aumentan. En particular, la velocidad máxima pasa de $3.979\frac{m}{s}$ a $5.911\frac{m}{s}$, mientras que la velocidad media aumenta de $3.758\frac{m}{s}$ a $5.227\frac{m}{s}$. Cabe mencionar que el cuadricóptero colisiona con retardos menores a $8ms$.

Retardo(ms)	Velocidad media(m/s)	Velocidad maxima(m/s)
12	3.758723	3.979526
10	4.378745	4.750859
9	4.826497	5.317858
8	5.227806	5.911128

Tabla 4.18 Relación entre la velocidad y el retardo durante la trayectoria.

En la Figura 4.50 se muestra un gráfico que representa la velocidad del cuadricóptero durante el seguimiento de la trayectoria en función del retardo medido en milisegundos. Se observa cómo las curvas de la trayectoria alrededor del obstáculo cuadrado se reflejan en la gráfica, mostrando un decremento de velocidad que, a medida que aumenta la velocidad del cuadricóptero, se aproxima a $1.5\frac{m}{s}$

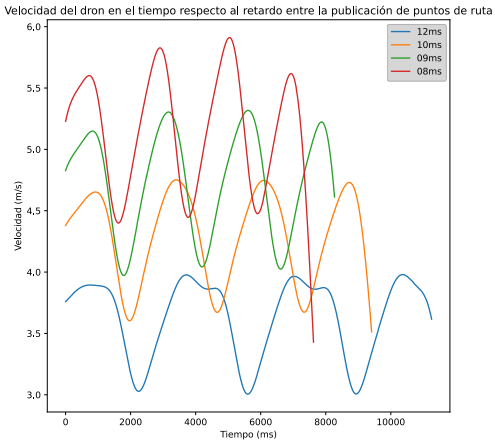


Figura 4.50 Velocidad del dron en el tiempo respecto al retardo con B-spline.

Análisis de la relación entre el RMSE y el retardo de la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria a un metro del obstáculo

La Tabla 4.19 muestra la relación entre la velocidad máxima y el RMSE para el ángulo yaw. En el caso del ángulo yaw, el RMSE se mide en grados en el rango de 0° a 360° y se observa que los valores promedio de error en la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria van de 23.399° a 27.264° de lo esperado conforme aumenta la velocidad máxima.

Retardo(ms)	RMSE
12	**23.39992**
10	24.12812
9	27.11199
8	27.2647

Tabla 4.19 Relación entre el retardo y el RMSE para el ángulo yaw.

La Figura 4.51 muestra la comparación entre el ángulo yaw esperado y observado en los datos de odometría, en los valores de retardo donde el RMSE es más bajo y alto, $12ms$ y $8ms$ respectivamente. Se puede observar que en $12ms$ el ángulo yaw esperado y observo tienen un desfase de $23.399°$ en promedio, mientras que en $8ms$ el ángulo yaw se desfasa de lo esperado $27.264°$ en promedio.

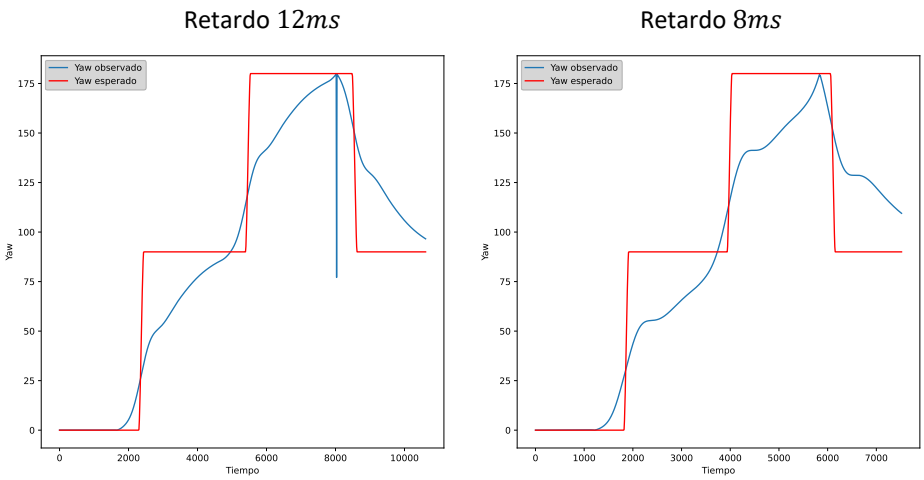


Figura 4.51 Trayectoria observada y esperada en el ángulo Yaw.

Por otro lado, para los ejes x e y el RMSE se mide en metros. En promedio, los valores de error en la navegación autónoma del UAV durante el seguimiento de la trayectoria se desvían de $0.213m$ a $0.415m$. Se encontró que el RMSE mínimo de $0.213m$ se obtiene con un retardo de $12ms$, mientras que el RMSE mayor de $0.415m$ se obtiene con un retardo de $8ms$.

Retardo(ms)	RMSE
12	**0.213267**
10	0.268014
9	0.318713
8	0.415173

Tabla 4.20 Relación entre el retardo y el RMSE en los ejes XY.

La Figura 4.52 muestra la comparación entre las trayectorias esperadas y observadas en los datos de odometría para los ejes x e y, en los valores donde el RMSE es más bajo y alto, $12ms$ y $8ms$ respectivamente. Se puede observar que en $12ms$ la trayectoria esperada y observada se desfasan poco solo $0.213m$ en promedio, mientras que en $8ms$ la trayectoria observada se desfasa de lo esperado notoriamente con curvas abiertas y deformes que tienen $0.415m$ de error en promedio.

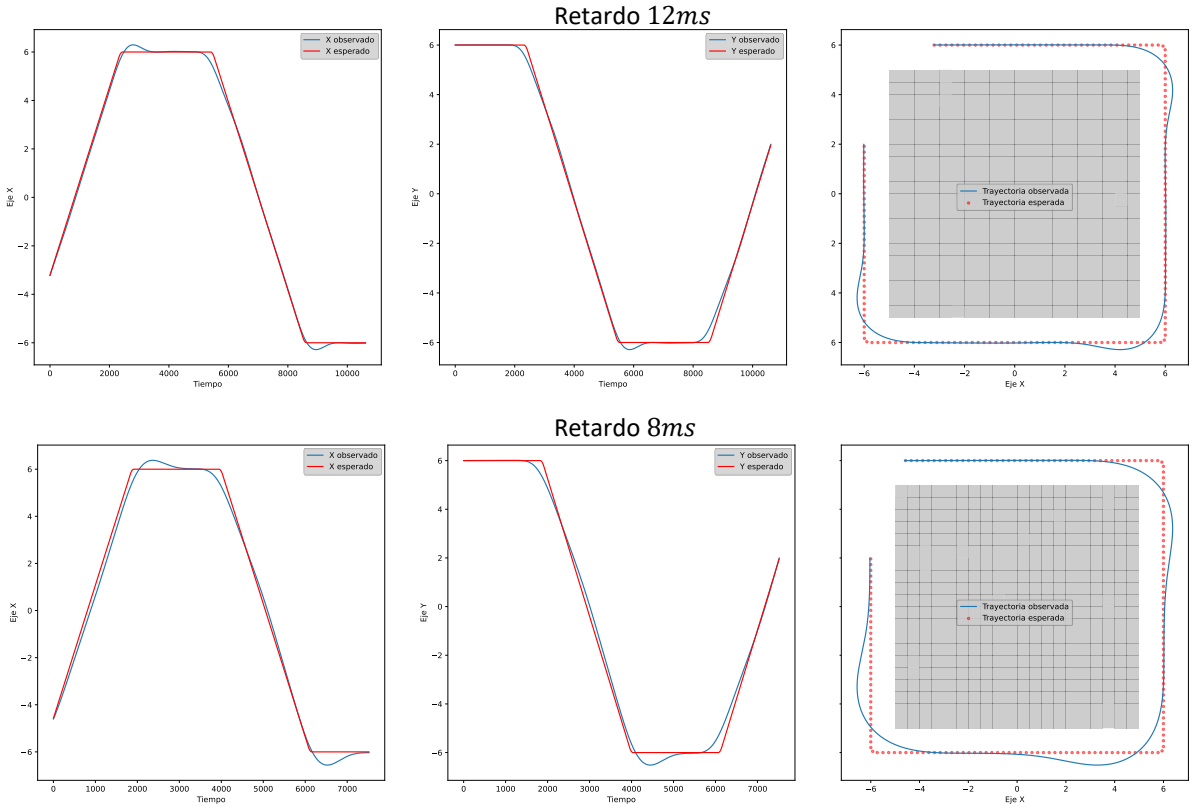


Figura 4.52 Trayectoria observada y esperada en los ejes XY.

4.2.3 Conclusión

Luego de analizar la distancia mínima en un entorno con un obstáculo cuadrado en su centro para Bézier y B-spline, se hacen las siguientes conclusiones:

Para las trayectorias generadas con Bézier a una velocidad máxima de $2.166\frac{m}{s}$ y con unas distancias al obstáculo de $0.5m$, $1.0m$ y $1.5m$, se observó que la distancia mínima a la que se deben generar trayectorias con Bézier es de $1.5m$, donde el RMSE es de $0.036m$. Para conseguir velocidades más altas a $2.166\frac{m}{s}$ se debe alejar más la trayectoria porque a mayor velocidad las curvas de Bézier son más cerradas.

Para las trayectorias generadas con B-spline con una distancia al obstáculo de medio metro, se observó que la velocidad máxima a la que se puede lograr un buen control es de $2.936\frac{m}{s}$, donde el RMSE es de $0.140m$. y con una distancia de un metro, se observó que la

velocidad máxima a la que se puede lograr un buen control es de $5.911\,\frac{m}{s}$, donde el RMSE es de $0.415m$.

En base a los resultados de la comparación entre Bézier y B-spline, se pude concluir que, para este caso particular de seguimiento de trayectoria en un entorno con un obstáculo cuadrado en su centro, el método de B-spline es mejor que Bézier, porque puede generar trayectorias a menor distancia del obstáculo ($0.5m$ y $1.0m$), mientras que Bézier solo puede generar trayectorias a distancia de $1.5m$ del obstáculo.

4.3 EXPERIMENTO PARA OBTENER EL GRADO DE LA CURVA PARA B-SPLINE

A continuación, se presenta un experimento para determinar el mejor grado de B-spline para generar trayectorias alrededor de un obstáculo cuadrado a un metro de distancia. En la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria se capturaron datos de odometría para comparar la trayectoria observada con la esperada y calcular el RMSE.

4.3.1 Comparación entre los grados de B-spline 3, 10, 20, 30, 40, 50 y 60

En esta sección, se presenta un experimento que tiene como objetivo probar los grados de B-spline 3, 10, 20, 30, 40, 50 y 60 para encontrar el grado que tiene el menor RMSE en el seguimiento de una trayectoria alejada un metro del obstáculo cuadrado con un retardo entre la publicación de puntos de $8ms$.

Se utiliza un retardo de $8ms$ porque a un metro de distancia al obstáculo, es el menor retardo temporal que se puede usar con B-spline como se observó en el experimento anterior.

Análisis de la relación entre el RMSE, el porcentaje de error y el grado de B-spline en la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria

La Tabla 4.21 presenta la relación entre el grado de B-spline, el RMSE y el porcentaje de error para el ángulo yaw, que se mide en grados en el rango de 0° a 360°. Para el RMSE, se observa que es más bajo en 40° y más alto en 20°. En cuanto al porcentaje de error, se

observa que es más bajo en 10° y más alto en 50°. La Figura 4.53 ilustra la comparación entre el ángulo yaw esperado y el observado para estos valores de error destacados.

Grado	RMSE	%error
3	29.60927	8.403217
10	28.38267	**7.757237**
20	31.23092	9.290233
30	20.98446	9.27363
40	**19.28969**	9.150964
50	22.724	13.19639
60	21.30391	12.22387

Tabla 4.21 Error en el ángulo Yaw de la trayectoria.

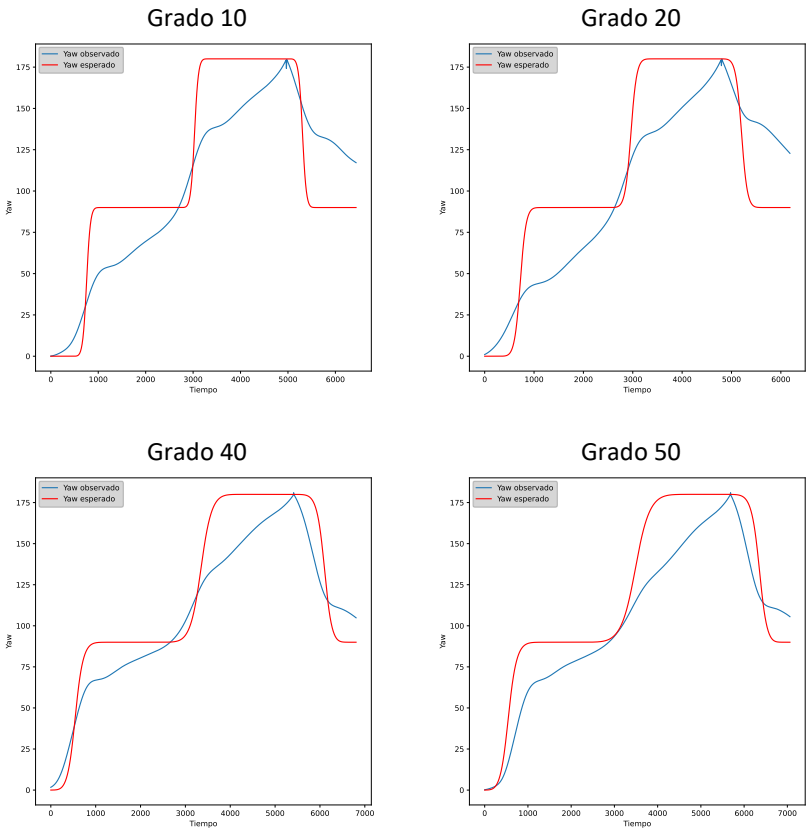


Figura 4.53 Trayectoria observada y esperada para el ángulo yaw.

Por otro lado, para los ejes x e y el RMSE se mide en metros. En promedio, los valores de error en la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria

se desvían de $0.377m$ a $0.926m$. Se encontró que el RMSE mínimo de $0.377m$ se obtiene con un grado 10°, mientras que el RMSE mayor de $0.926m$ se obtiene con un grado 50°.

El porcentaje de error en la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria va de 3.579% a 16.805%. Se encontró que el porcentaje de error mínimo de 3.579% se obtiene con un grado 10°, mientras que el porcentaje de error mayor de 16.805% se obtiene con un grado 50°.

La Figura 4.54 ilustra la comparación entre la trayectoria esperada y observada para estos valores de error destacados. Se puede observar que en 10° la trayectoria esperada y observada se desfasan poco solo $0.377m$ en promedio, mientras que en 50° la trayectoria observada se desfasa de lo esperado notoriamente con curvas abiertas y deformes que tienen $0.936m$ de error en promedio.

Grado	RMSE	%error
3	0.452946	4.374587
10	**0.37798**	**3.579408**
20	0.567771	6.166188
30	0.86247	11.99486
40	0.903297	12.89673
50	0.926271	16.80533
60	0.864094	16.66673

Tabla 4.22 Error en los ejes XY de la trayectoria.

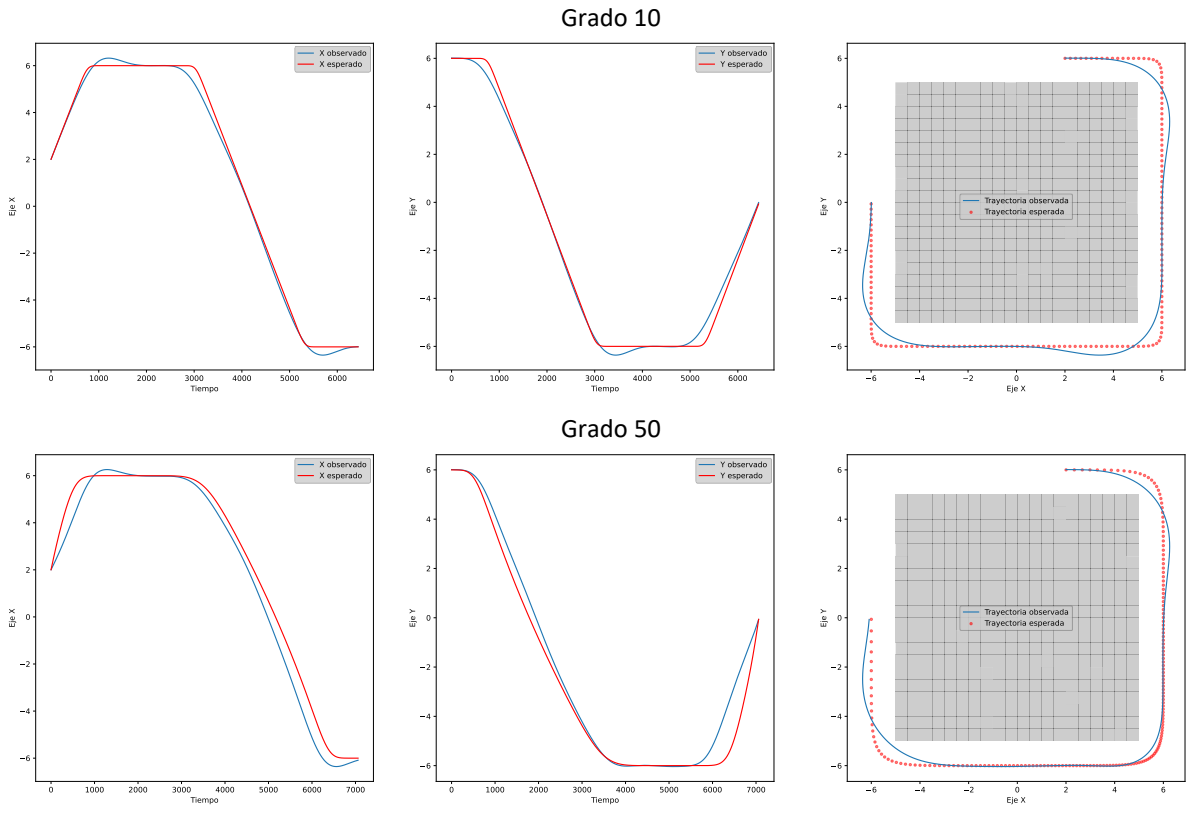


Figura 4.54 Trayectoria observa y esperada para los ejes XY.

4.3.2 Conclusión

Se observo que para el ángulo yaw el grado con el menor RMSE es cuarenta y el grado con el menor porcentaje de error es diez, para los ejes x e y el menor error en las dos métricas lo obtuvo grado diez, por lo que el mejor grado para suavizar trayectorias con B-spline es el grado diez.

4.4 EXPERIMENTO PARA LA COMPARACIÓN DE TRES MÉTODOS PARA GENERAR TRAYECTORIA EN ENTORNOS INTERIORES

El objetivo de este experimento es comparar diferentes métodos de Aprendizaje Q en la generación de trayectorias en entornos simulados de interiores.

En este experimento, se comparan tres métodos para modificar la política utilizada en el Aprendizaje-Q para la selección de la acción que llevara al estado siguiente utilizando la ecuación de la diferencia temporal: selección aleatoria (Convencional), muestreo de

Thompson adaptado (MTA) y distribución Gaussiana (DG). Además, se analiza el uso y no uso de diagonales, así como la incorporación de ventanas de distancia al obstáculo, las cuales poseen dimensiones de $1x1, 3x3, 5x5\ y\ 7x7$.

La ejecución de las pruebas se ha llevado a cabo empleando un factor de descuento (γ) de 0.99 y un factor de aprendizaje (α) de 0.9.

Estas configuraciones (Tabla 4.23) para la generación de trayectoria con diferentes métodos permite observar cómo cada método se comporta bajo distintas configuraciones del algoritmo y del espacio de búsqueda.

Método	Ventana	Diagonales	γ	α
Convencional	1x1, 3x3, 5x5 y 7x7	Sin y con	0.99	0.9
MTA	1x1, 3x3, 5x5 y 7x7	Sin y con	0.99	0.9
DG	1x1, 3x3, 5x5 y 7x7	Sin y con	0.99	0.9

Tabla 4.23 Configuración para la generación de trayectorias.

Generación de entornos de interiores

El experimento se lleva a cabo en cinco entornos inspirados en interiores. Cada entorno varió en número de estados y área en pixeles, así como en relación con el número y la disposición de obstáculos. La Tabla 4.24 muestra la configuración de cada entorno inspirado en interiores, en cuanto a número de estados y área en pixeles.

Entorno	Numero de estados	Área (pixeles)
Casa	1123	40x40
Supermercado	2687	45x100
Oficina	2938	44x88
Farmacia	3862	71x100
Bodega	4853	81x100

Tabla 4.24 Configuración de cada entorno, en cuanto a número de estados y área en pixeles.

Selección de los puntos de ruta de inicio y fin para las trayectorias

Con el propósito de generar cuatro trayectorias únicas en cada uno de los entornos, se han establecido cuatro puntos de inicio y fin distintos. Donde, cada trayectoria ha sido diseñada con el objetivo de llevar a cabo una tarea especifica de navegación en el entorno correspondiente.

Consideraciones en la comparación de tres métodos para generar trayectorias en entornos interiores

Debido a que profundizar en el análisis de todas las trayectorias resultaría en la generación de contenido redundante y extenso. A continuación, se presentan algunas consideraciones en cuanto al análisis realizado en cada entorno

- Para el análisis de cada entorno solo se detalla el proceso de generar la trayectoria más extensa en cada entorno.
- Dado que los resultados de generar estas trayectorias con diferentes métodos presentan el mismo patrón en todos los entornos.
    1. Se presentan únicamente tablas de resultados de comparación de tiempo y número de iteraciones para el entorno “casa”.
    2. Se presentan únicamente tablas que muestra las mejoras porcentuales en tiempo y número de iteraciones, aplicadas a los métodos DG y MTA. Estas tablas se centran en los entornos “casa” y “bodega”, cuidadosamente seleccionados debido a que representan los escenarios con el menor y el mayor número de estados, respectivamente.
- En relación con el análisis del vuelo del cuadricóptero durante el seguimiento de la trayectoria, se observa una consistencia en los resultados indicando que las mejores ejecuciones de vuelo se logran con ventanas de distancia al obstáculo $5x5$ y $7x7$:
    1. Por lo tanto, en esta sección, solo se presenta un análisis detallado del vuelo del cuadricóptero en el entorno “casa”. Para los demás entornos, se resumirán los resultados en tablas que corresponden a ventanas $5x5$ y $7x7$.

- Una constante en los resultados son las colisiones del cuadricóptero durante el seguimiento de trayectorias que incorporan diagonales.
    1. Por este motivo, se presentarán observaciones únicamente para las trayectorias con colisiones durante el vuelo en el entorno “casa”.
- Las tablas pormenorizadas en los experimentos se adjuntan al apéndice para su consulta.

4.4.1 Comparación de tres métodos para generar trayectorias en un entorno inspirado en una casa

El objetivo de este experimento es comparar diferentes métodos de Aprendizaje Q en la generación de trayectorias en un entorno simulado inspirado en una casa.

En las siguientes subsecciones, se lleva a cabo un análisis del tiempo y el número de iteraciones necesarios para que el algoritmo alcance la convergencia. Se explorará la relación entre el método empleado, el tamaño de ventana de proximidad a los obstáculos y el uso o no uso de diagonales.

Generación del entorno simulado inspirado en una casa

El experimento se lleva a cabo en un entorno inspirado en una casa (Figura 4.55). El entorno se ha construido sobre un *canvas* de $40x40\ pixeles$. Dentro de este espacio, se han identificado $1123\ pixeles$ como espacios libres y con líneas de $1\ píxel$ de grosor se han representado las paredes que definen los limites de las habitaciones y corredores.

Selección de los puntos de ruta de inicio y fin de las trayectorias

Para generar trayectorias, se definieron cuatro puntos de inicio y fin. La Figura 4.55 muestra los números correspondientes a cada trayectoria, representando el inicio con un círculo gris y el final con un círculo negro. Cada trayectoria se diseñó para realizar una tarea de navegación en este entorno:

1. La trayectoria uno inicia en el “Patio” y finaliza en la “Habitación 2”.
2. La trayectoria dos inicia en la “Sala” y finaliza en la “Habitación 2”.

3. La trayectoria tres inicia en la “Habitación 1” y finaliza en el “Baño”.
4. La trayectoria cuatro inicia en la “Cocina” y finaliza en el “Comedor”.

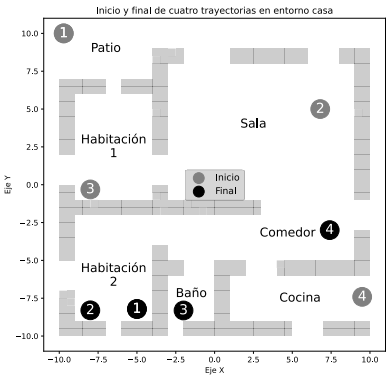


Figura 4.55 Inicio y final de cuatro trayectorias en entorno casa.

Análisis de los métodos en cuanto a la generación de la trayectoria más extensa

A continuación, se presentan tablas para cada método (Tablas 4.1 – 4.3) con los detalles respecto a la generación de las trayectorias más extensa, donde la trayectoria más extensa fue la uno que inicia en el “Patio” y finaliza en la “Habitación 2”.

En la primera columna se observa el tamaño de ventana de proximidad al obstáculo, en la segunda el número de iteraciones para la convergencia, en la tercera el tiempo de entrenamiento para la convergencia, en la cuarta el número de estados de la trayectoria, en la quinta la distancia en metros de la trayectoria, en la sexta columna el RMSE, en la séptima columna se muestra el porcentaje de error en la trayectoria.

Análisis de la trayectoria más extensa sin diagonales utilizando el método Convencional

En la Tabla 4.25 se presentan resultados en relación con la generación de la trayectoria más extensa con el método Convencional sin diagonales. Entre las diferentes configuraciones de ventanas de proximidad a los obstáculos, se destaca que el menor número de iteraciones requeridas para alcanzar la convergencia se obtiene con las dimensiones de ventana $1x1$ y $3x3$. Este proceso demanda 16,349,406 iteraciones, equivalente a 27.026

minutos, mientras que la mayor cantidad de iteraciones se presenta con la ventana $7x7$, totalizando 16,759,570 iteraciones en 27.704 minutos.

En términos de precisión, sin la incorporación de diagonales, el RMSE más bajo se obtiene con la ventana de $5x5$, presentando una desviación promedio de la trayectoria de 1.094 metros. Por otro lado, las ventanas de $1x1$ y $3x3$ registran el RMSE más alto, con una desviación promedio de 1.109 metros.

El porcentaje de error también se revela como una métrica relevante. Sin diagonales, las ventanas $1x1$ y $3x3$ muestran el menor porcentaje de error, con un 16.229% de desviación en la trayectoria. En contraposición, las ventanas $5x5$ y $7x7$ presentan el mayor porcentaje de error, alcanzando un 17.647% de desviación.

En cuanto al número de estados y la distancia, se destaca que, sin diagonales, todas las configuraciones de ventanas mantienen una distancia de 41 metros, correspondiente a un total de 82 estados.

Análisis de la trayectoria más extensa con diagonales utilizando el método Convencional

Cuando se consideran las diagonales como parte del método, se destacan observaciones notables (Tabla 4.25) en relación con la generación de la trayectoria más extensa. En particular, se observa que el menor número de iteraciones requerido para la convergencia se logra con la ventana de $3x3$, totalizando 23,676,885 iteraciones en 39.138 minutos. Por otro lado, se observa que la ventana de $5x5$ presenta el mayor número de iteraciones, con un total de 24,050,067 iteraciones en un periodo de 39.755 minutos.

La evaluación del RMSE con la inclusión de diagonales muestra que el valor más bajo se obtiene con la ventana de $7x7$, presentando una desviación promedio de la trayectoria de 1.336 metros. En contraste, la ventana de $3x3$ presenta el RMSE más alto, con una desviación promedio de 1.376 metros.

Otro aspecto que considerar es el porcentaje de error. En esta métrica, se destaca que la ventana de $1x1$ presenta el porcentaje de error más bajo, registrando un 24.592% de

desviación de la trayectoria. Por otro lado, la ventana $7x7$ presenta el porcentaje de error más alto, alcanzando un 28.669% de desviación de la trayectoria.

En lo que respecta al número de estados y la distancia, es notable que, con la incorporación de diagonales resulta en un número constante de 30.5 metros, equivalente 61 estados, para todas las ventanas.

Ventana	Iteraciones	Tiempo (min)	Número de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1X1	**16349406**	**27.026**	82	41	1.109	**16.229**
3x3	**16349406**	**27.026**	82	41	1.109	**16.229**
5x5	16475481	27.234	82	41	**1.094**	17.647
7x7	16759570	27.704	82	41	1.1	17.647
Con diagonales						
1X1	23804641	39.349	61	30.5	1.35	**24.592**
3x3	**23676885**	**39.138**	61	30.5	1.376	24.7
5x5	24050067	39.755	61	30.5	1.348	28.055
7x7	23966017	39.616	61	30.5	**1.336**	28.669

Tabla 4.25 Análisis de la trayectoria más extensa con diagonales utilizando el método Convencional.

Análisis de la trayectoria más extensa sin diagonales utilizando el método MTA

En la Tabla 4.26, se destacan observaciones notables en relación con la generación de la trayectoria más extensa con el método MTA sin diagonales. Específicamente, se evidencia que la menor cantidad de iteraciones necesaria para alcanzar la convergencia se obtiene utilizando una ventana de tamaño $3x3$, demandando 4,326,894 iteraciones, equivalentes a 11.553 minutos. Por otro lado, la ventana $1x1$ presenta el mayor número de iteraciones, totalizando 6,327,284 iteraciones durante un período de 16.894 minutos.

En la evaluación del RMSE, el porcentaje del error, la distancia y numero de estados sin la presencia de diagonales se señalan los mismos valores que en el método Convencional, lo que significa que la trayectoria es la misma.

Análisis de la trayectoria más extensa con diagonales utilizando el método MTA

Cuando se consideran las diagonales como parte del método, se destacan observaciones notables (Tabla 4.26) en relación con la generación de la trayectoria más extensa. En particular, se observa que el menor número de iteraciones requerido para la convergencia se logra con la ventana de $3x3$, totalizando $4{,}360{,}514$ iteraciones en 11.643 minutos. Por otro lado, se observa que la ventana de $5x5$ presenta el mayor número de iteraciones, con un total de $29{,}064{,}490$ iteraciones en un periodo de 77.602 minutos.

La evaluación del RMSE con la inclusión de diagonales muestra que el valor más bajo se obtiene con la venta de $5x5$, presentando una desviación promedio de la trayectoria de 1.244 metros. En contraste, las ventanas de $1x1$ y $3x3$, presentan el RMSE más alto, con una desviación promedio de 1.376 metros.

Otro aspecto que considerar es el porcentaje de error. En esta métrica, se destaca que las ventanas de $1x1$ y $3x3$, presentan el porcentaje de error más bajo, registrando un 24.7% de desviación de la trayectoria. Por otro lado, la ventana de $7x7$ presenta el porcentaje de error más alto, alcanzando un 29.277% de desviación de la trayectoria.

En lo que respecta al número de estados y la distancia, es notable que, con la incorporación de diagonales las ventanas de $1x1, 3x3$ y $7x7$ mantienen una distancia invariable de 30.5 metros, lo que corresponde a un total de 61 estados. En el caso de la ventana de $5x5$, la distancia recorrida se extiende a 31 metros, lo que equivale a 62 estados en total.

En la evaluación del RMSE y el porcentaje del error, al considerar la presencia de diagonales, se observan valores idénticos para la ventana $3x3$ con el método Convencional. Esto implica que las trayectorias son idénticas. Sin embargo, para las ventanas $1x1$, $5x5$ y $7x7$, las trayectorias difieren del método Convencional.

Ventana	Iteraciones	Tiempo (min)	Numero de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1x1	6327284	16.894	82	41	1.109	**16.229**

3x3	**4326894**	**11.553**	82	41	1.109	**16.229**
5x5	5653203	15.094	82	41	**1.094**	17.647
7x7	5938973	15.857	82	41	1.1	17.647
Con diagonales						
1x1	7616611	20.336	61	30.5	1.376	**24.7**
3x3	**4360514**	**11.643**	61	30.5	1.376	**24.7**
5x5	29064490	77.602	62	31	**1.244**	24.83
7x7	13348821	35.641	61	30.5	1.294	29.277

Tabla 4.26 Análisis de la trayectoria más extensa con diagonales utilizando el método MTA.

Análisis de la trayectoria más extensa sin diagonales utilizando el método DG

En la Tabla 4.27, se destacan observaciones notables en relación con la generación de la trayectoria más extensa con el método DG. Específicamente, se evidencia que la menor cantidad de iteraciones necesaria para alcanzar la convergencia se obtiene utilizando las ventanas $1x1$ y $3x3$, demandando $4{,}340{,}342$ iteraciones, correspondientes a 13.702 minutos. Por otro lado, la ventana $5x5$ presenta el mayor número de iteraciones, totalizando $4{,}484{,}908$ iteraciones durante un periodo de 14.159 minutos.

En la evaluación del RMSE, el porcentaje del error, la distancia y numero de estados sin la presencia de diagonales se señalan los mismos valores que en el método Convencional y en el método MTA, lo que significa que la trayectoria es la misma.

Análisis de la trayectoria más extensa con diagonales utilizando el método DG

Cuando se consideran las diagonales como parte del método, se destacan observaciones notables (Tabla 4.27) en relación con la generación de la trayectoria más extensa. En particular, se observa que el menor número de iteraciones requerido para la convergencia se logra con la ventana de $3x3$, totalizando $4{,}452{,}969$ iteraciones en 14.058 minutos. Por otro lado, se observa que la ventana de $7x7$ presenta el mayor número de iteraciones, con un total de $4{,}669{,}818$ en un periodo de 14.743 minutos.

La evaluación del RMSE con la inclusión de diagonales muestra que el valor más bajo se obtiene con la ventana de $7x7$, presentando una desviación promedio de la trayectoria de 1.344 metros. En contraste, la ventana de $1x1$, presenta el RMSE más alto, con una desviación promedio de 1.376 metros.

Otro aspecto que considerar es el porcentaje de error. En esta métrica, se destaca que la ventana de $3x3$, presenta el porcentaje de error más bajo, registrando un 24.385% de desviación de la trayectoria. Por otro lado, la ventana de $7x7$ presenta el porcentaje de error más alto, alcanzando un 28.79% de desviación de la trayectoria.

En lo que respecta al número de estados y la distancia, es notable que, con la incorporación de diagonales resulta en un número constante de 30.5 metros, equivalente 61 estados, para todas las ventanas.

En la evaluación del RMSE y el porcentaje del error, al considerar la presencia de diagonales, se observan valores idénticos tanto para la ventana $1x1$ con el método MTA como para la ventana $5x5$ con el método Convencional. Esto implica que las trayectorias son idénticas en ambos casos. Sin embargo, para las ventanas 3x3 y 7x7, las trayectorias difieren tanto de la selección aleatoria como del MTA.

Ventana	Iteraciones	Tiempo (min)	Numero de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1x1	**4340342**	**13.702**	82	41	1.109	**16.229**
3x3	**4340342**	**13.702**	82	41	1.109	**16.229**
5x5	4484908	14.159	82	41	**1.094**	17.647
7x7	4474822	14.127	82	41	1.1	17.647
Con diagonales						
1x1	4474822	14.127	61	30.5	1.376	24.7
3x3	**4452969**	**14.058**	61	30.5	1.346	**24.385**
5x5	4644603	14.663	61	30.5	1.348	28.055
7x7	4669818	14.743	61	30.5	**1.344**	28.79

Tabla 4.27 Análisis de la trayectoria más extensa con diagonales utilizando el método DG.

Comparación entre los métodos en términos de tiempo e iteraciones para las ventanas de proximidad al obstáculo

La Figura 4.56 presenta una comparación entre los métodos en relación con el tiempo e iteraciones requeridos para las distintas ventanas de proximidad al obstáculo, considerando tanto la inclusión como la exclusión de diagonales en el análisis.

A continuación, se detallan las observaciones específicas para cada grafica mostrada en la Figura 4.56.

Comparación entre los métodos cuando no se incorporan diagonales

Comparación en términos de tiempo (Fig. 4.3 arriba-izquierda)

Cuando no se incorporan diagonales, el método DG exhibe el menor tiempo para lograr la convergencia en las ventanas $1x1$, $5x5$ y $7x7$. Por otro lado, para la ventana $3x3$, el método MTA muestra ser la opción más eficiente en términos de tiempo para alcanzar la convergencia. Mientras tanto, el método convencional se posiciona como el método que demanda más tiempo en todas las ventanas.

Comparación en términos de iteraciones (Fig. 4.3 arriba-derecha)

Cuando no se incorporan diagonales, el método DG exhibe el menor número de iteraciones necesario para lograr la convergencia en las ventanas $1x1$, $5x5$ y $7x7$. Por otro lado, para la ventana $3x3$, el método MTA muestra ser la opción más eficiente en términos de numero de iteraciones para alcanzar la convergencia. Mientras tanto, el método Convencional se posiciona como el método que demanda mayor cantidad de iteraciones en todas las ventanas.

Comparación entre los métodos cuando se incorporan diagonales

Comparación en términos de tiempo (Fig. 4.3 abajo-izquierda)

Al considerar la inclusión de diagonales, se reafirma que el método DG permanece como la elección óptima en cuanto a tiempo requerido para lograr la convergencia en las ventanas de $1x1$, $5x5$ y $7x7$. Sin embargo, para la ventana $3x3$, el método MTA sigue siendo el más eficiente en términos de tiempo.

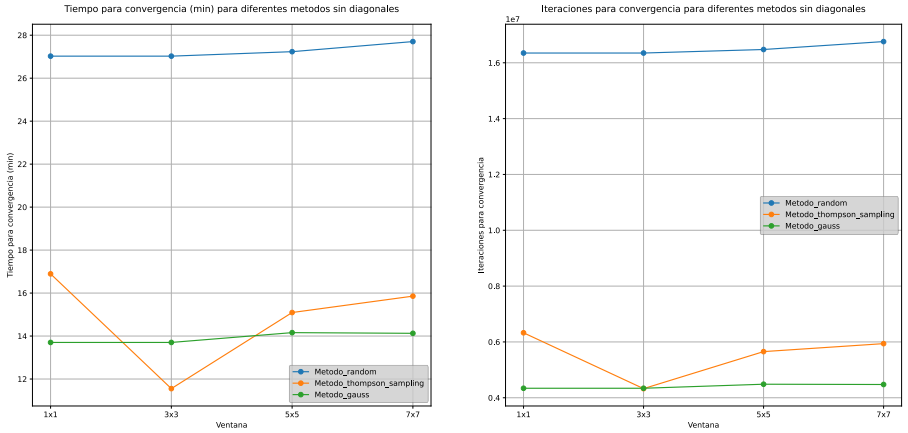
Por otro lado, se constata que el método Convencional es el que demanda más tiempo para la convergencia en las ventanas de $1x1$, $3x3$ y $7x7$, cuando se incluyen diagonales. Para la ventana $5x5$, el método MTA se posiciona como el método más lento.

Comparación en términos de iteraciones (Fig. 4.3 abajo-derecha)

Al considerar la inclusión de diagonales, se reafirma que el método DG permanece como la elección óptima a número de iteraciones requeridas para lograr la convergencia en las ventanas de $1x1$, $5x5$ y $7x7$. Sin embargo, para la ventana $3x3$, el método MTA sigue siendo el más eficiente en términos de iteraciones.

Por otro lado, se constata que el método Convencional es el que demanda mayor cantidad de iteraciones para la convergencia en las ventanas de $1x1$, $3x3$ y $7x7$, cuando se incluyen diagonales. Para la ventana $5x5$, el método MTA se posiciona como el método más exigente en términos de iteraciones.

Sin diagonales



Con diagonales

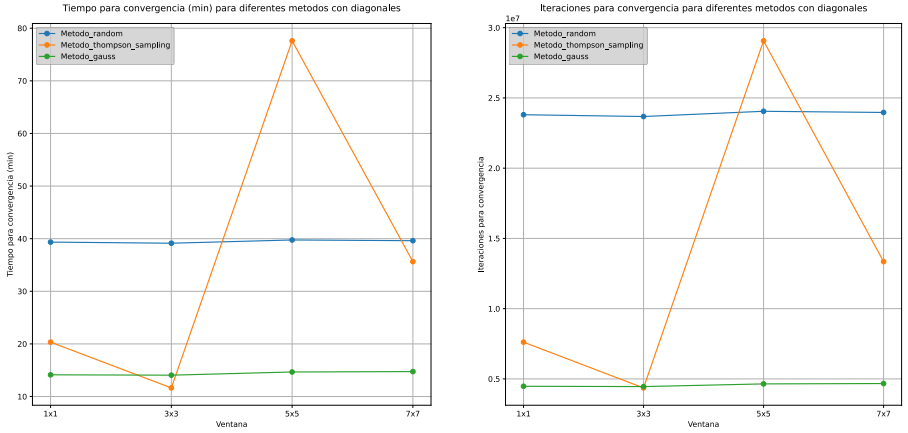


Figura 4.56 Comparación entre los métodos en términos de tiempo e iteraciones para las ventanas de proximidad al obstáculo.

A continuación, se presenta una tabla que muestra las mejoras porcentuales en tiempo y número de iteraciones para los métodos DG y MTA en comparación con el método convencional.

Cuando no se incorporan diagonales con el método DG, el mayor porcentaje de mejora, en cuanto a tiempo, se registra en las ventanas $1x1$ y $3x3$, con un 49.301%. En cuanto al número de iteraciones, la mejora es de 73.452% en las mismas ventanas de $1x1\ y\ 3x3$.

Por otro lado, cuando no se incorporan diagonales con el método MTA, en cuanto a tiempo, el mayor porcentaje de mejora es de 57.252% registrado con la ventana $3x3$, en cuanto a iteraciones la mejora es de 73.535% en la misma ventana $3x3$.

Cuando se incorporan diagonales con el método DG, en cuanto a tiempo, el mayor porcentaje de mejora es de 64.272% registrado con la ventana $3x3$, en cuanto a iteraciones la mejora es de 81.294% en la ventana $1x1$.

Por otro lado, cuando se incorporan diagonales con el método MTA, en cuanto a tiempo, el mayor porcentaje de mejora es de 70.251% con la ventana $3x3$, en cuanto a iteraciones la mejora es de 81.583% con la ventana $3x3$.

	DG		MTA	
	Tiempo	Iteraciones	Tiempo	Iteraciones
Ventana	Mejora (%)			
Sin diagonales				
1x1	**49.301**	**73.453**	37.490	61.300
3x3	**49.301**	**73.453**	**57.252**	**73.535**
5x5	48.010	72.778	44.577	65.687
7x7	49.007	73.300	42.763	64.564
Con diagonales				
1x1	63.905	**81.294**	48.319	68.004
3x3	**64.274**	81.100	**70.251**	**81.583**
5x5	63.117	80.688	-95.201	-20.850

7x7	62.785	80.515	10.034	44.301

Tabla 4.28 mejoras porcentuales para los métodos DG y MTA en comparación con el método convencional.

Análisis del vuelo del cuadricóptero en simulación durante el seguimiento de las trayectorias generadas

A continuación, se realiza un análisis del desempeño del cuadricóptero durante el seguimiento de las trayectorias, considerando la inclusión y no inclusión de diagonales. Se exponen el método empleado, el tamaño de ventana de proximidad al obstáculo y el retardo temporal que se utilizaron para generar la trayectoria. Además, se presentan métricas de error, velocidad máxima y media, tiempo de vuelo y la distancia recorrida. Esto permite una evaluación del desempeño en la navegación autónoma del UAV durante el seguimiento de la trayectoria y ayuda a identificar la configuración óptima de método, ventana y retardo que produce los mejores resultados en esta tarea.

Trayectorias finalizadas sin colisión

La Tabla 4.29 presenta las trayectorias que se completaron exitosamente sin colisiones, considerando tanto la incorporación como la no incorporación de diagonales. En la primera columna se detallan los métodos que coinciden al generar la misma trayectoria. La segunda columna muestra el tamaño de la ventana de proximidad al obstáculo utilizando en cada caso. Las columnas de la tercera a la onceava presentan los siguientes valores de interés para evaluar el seguimiento:

- RMSE para los ejes x e y
- Porcentaje de error para los ejes x e y
- RMSE para el ángulo yaw
- Porcentaje de error para el ángulo yaw
- Retardo temporal
- Distancia en metros

- Velocidad máxima en metros por segundo
- Velocidad media en metros por segundo
- Tiempo de vuelo en segundos

Cuando no se incorporan diagonales, se observa que los métodos generan la misma trayectoria para cada ventana. Para los ejes x e y el RMSE y porcentaje de error más bajo de 0.770 metros de desviación promedio y 15.480% de error, en la trayectoria se logra con la ventana $5x5$, mientras que la ventana $3x3$ presenta los valores de error más altos de 0.968 metros y 21.191%.

En relación con el ángulo yaw, la ventana $3x3$ exhibe los valores más bajos tanto en RMSE, con $17.564°$ de desviación promedio, como en porcentaje de error, con 4.687%. Por otro lado, la ventana $7x7$ registra los valores más altos, llegando a $27.657°$ de desviación y 14.617% de error.

En cuanto a retardo temporal, las ventanas de $1x1$ y $3x3$, alcanzan retardos de hasta 20 milisegundos, equivalente a velocidad máxima de $2.9\frac{m}{s}$ y velocidad media de $1.8\frac{m}{s}$. La ventana $1x1$ termina en un tiempo de vuelo de 17.777 segundos, ligeramente más rápido que la ventana $3x3$. Para las ventanas $5x5$ y $7x7$, los valores más bajos de retardo son de 9 milisegundos, equivalente a velocidad máxima de $4.6\frac{m}{s}$ y velocidad media de $3.5\frac{m}{s}$. La ventana de $7x7$ termina en un tiempo de vuelo de 7.801 segundos, ligeramente más rápido que la ventana $5x5$.

Con la inclusión de diagonales, se simplifica la columna método, ya que solo un método logra generar una trayectoria sin colisión. El menor error en las métricas de RMSE y porcentaje de error para los ejes x e y, se obtiene con la ventana $3x3$ correspondiente al método DG, y es de 0.218 metros de desviación y 4.933% de error en la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria. El menor retardo para las únicas dos ventanas que finalizaron sin colisión $1x1$ y $3x3$, es de 40 milisegundos que equivale a una velocidad máxima de $2\frac{m}{s}$ y a una velocidad media de $0.7\frac{m}{s}$, esta velocidad hace que ambas trayectorias terminen en un tiempo de vuelo de 36 segundos

Al considerar estos resultados, se concluye que la configuración más favorable para generar la trayectoria es sin diagonales, utilizando la ventana $7x7$ y el método DG (Figura 4.7). Aunque existen ventanas con menor error en métricas, el retardo y el tiempo de vuelo son los factores determinantes para elegir la mejor configuración en términos de eficiencia y cumplimiento de la tarea de navegación. Además, debe tenerse en cuenta que el error puede disminuirse disminuyendo la velocidad en la navegación autónoma del cuadricóptero durante el seguimiento de la trayectoria aumentando el retardo temporal.

Método	Ventana	RMSE XY	%error XY	RMSE Yaw	%error Yaw	Retardo (ms)	Distancia (m)	Velocidad máxima (m/s)	Velocidad media (m/s)	Tiempo de vuelo (s)
Sin diagonales										
Convencional, MTA, DG	1x1	0.956	20.932	17.621	4.873	20	41	2.911	1.800	17.777
Convencional, MTA, DG	3x3	0.968	21.191	**17.564**	**4.867**	20	41	2.900	1.801	17.820
Convencional, MTA, DG	5x5	**0.770**	**15.480**	26.679	13.147	**9**	41	4.647	3.547	8.034
Convencional, MTA, DG	7x7	0.788	16.322	27.657	14.617	**9**	41	**4.654**	**3.572**	**7.801**
Con diagonales										
Convencional	1x1	0.278	5.654	**15.822**	0.800	40	30.5	2.093	0.783	**36.456**
DG	3x3	**0.218**	**4.933**	16.143	**0.259**	40	30.5	**2.268**	**0.791**	36.857

Tabla 4.29 Trayectorias finalizadas sin colisión.

Trayectorias finalizadas con colisión

La Tabla 4.30 presenta las trayectorias finalizadas con colisión, en este caso solo hubo colisiones con trayectorias que incorporan diagonales.

Con diagonales

Método	Ventana
MTA, DG	1x1
Convencional, MTA	3x3

Convencional, DG	5x5
MTA	5x5
Convencional	7x7
MTA	7x7
DG	7x7

Tabla 4.30 Trayectorias finalizadas con colisión.

Análisis del vuelo del cuadricóptero durante el seguimiento de la mejor trayectoria

A continuación, en la parte derecha de la Figura 4.57 se muestra el seguimiento en los ejes x e y de la trayectoria sin utilizar diagonales y utilizando la ventana $7x7$. Se puede notar que la desviación de la trayectoria ocurre en las secciones curvas, y al final, se evidencia una pérdida de control debido a la acumulación de velocidad. En la parte izquierda de la Figura 4.57, se puede observar la gráfica de la velocidad del cuadricóptero en el tiempo durante el seguimiento de la trayectoria, donde se observa que la máxima aceleración se produce en la tercera prominencia del gráfico, la cual coincide con la recta más larga de la trayectoria, donde el dron acumula más velocidad.

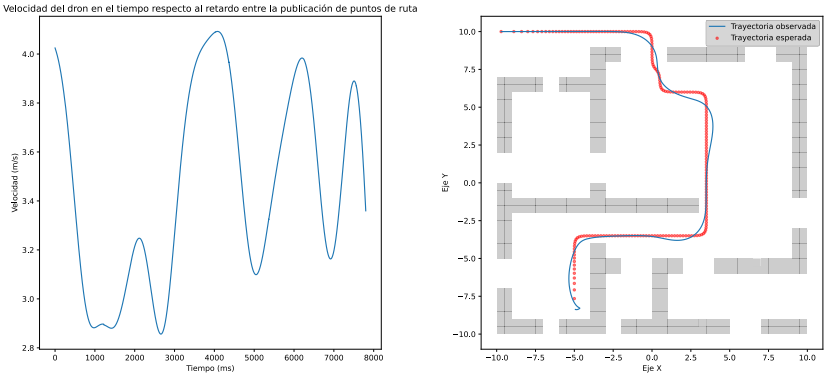


Figura 4.57 Seguimiento en los ejes x e y de la trayectoria sin utilizar diagonales y utilizando la ventana 7x7.

Evolución de la diferencia por época de la Q-table al generar la mejor trayectoria

En la Figura 4.58, se presenta el gráfico en escala logarítmica que representa la evolución de la diferencia hasta la convergencia por época en la Q-table para los métodos:

Convencional, MTA y DG. Observando el método Convencional, se nota que a medida que avanzan las épocas, la diferencia disminuye gradualmente a lo largo de 9970 épocas, equivalentes a 16,759,570 iteraciones, con una variación mínima en la diferencia. En cuanto al método MTA, se evidencia una disminución gradual de la diferencia durante un periodo de 3533 épocas, que equivale a 5,938,973 iteraciones, con una variación también mínima la diferencia.

Con relación al método DG, se observa que la diferencia disminuye gradualmente a lo largo de 2662 épocas, correspondientes a 4,474,822 iteraciones, con la particularidad de que existen picos en los que la diferencia aumenta y luego vuelve a disminuir. A través de este análisis, se llega a la conclusión de que el enfoque Convencional, siendo exploratorio, no resulta eficiente y, por lo tanto, requiere más tiempo para converger hacia una solución óptima. En contraste, el método MTA, equilibrando exploración y explotación del entorno, demuestra ser más eficiente en alcanzar una solución óptima. Por último, el método DG, al combinar exploración y explotación de manera similar a MTA, resulta más eficaz al explorar y explotar el entorno en un menor número de épocas de comparación con MTA.

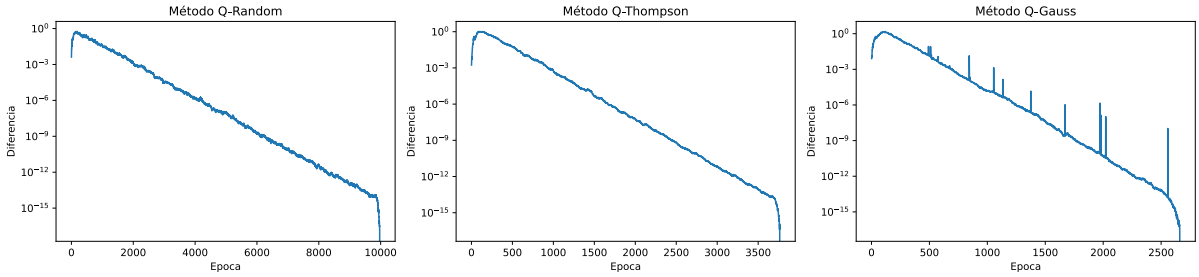


Figura 4.58 Evolución de la diferencia por época de la Q-table.

4.4.2 Comparación de tres métodos para la generación de trayectorias en un entorno inspirado en un supermercado

El objetivo de este experimento es comparar diferentes métodos de Aprendizaje Q en la generación de trayectorias en un entorno simulado inspirado en un supermercado.

Generación del entorno simulado inspirado en un supermercado

El experimento se lleva a cabo en un entorno inspirado en un supermercado (Figura 4.59). El entorno se ha construido sobre un *canvas* de $45x100$ $pixeles$. Dentro de este espacio, se han identificado 2687 $pixeles$ como espacios libres, con líneas de un $píxel$ de grosor se han representado las paredes que definen los limites de las habitaciones y corredores, también se han usado líneas más gruesas para representar obstáculos como estantes de comestibles, refrigeradores y almacenes.

Selección de los puntos de ruta de inicio y fin de las trayectorias

Para generar trayectorias, se definieron cuatro puntos de inicio y fin. La Figura 4.59 muestra los números correspondientes a cada trayectoria, representando el inicio con un círculo gris y el final con un círculo negro. Cada trayectoria se diseñó para realizar una tarea de navegación en este entorno:

1. La trayectoria uno inicia en el "Cuarto de preparación de productos" y finaliza en el "Taller de reparación de productos".
2. La trayectoria dos inicia en el "Congelador" y finaliza en el "Refrigerador".
3. La trayectoria tres inicia en "Información al cliente" y finaliza en el "Almacén".
4. La trayectoria cuatro inicia en un estante de "Comestibles" y finaliza en la "Cafetería".

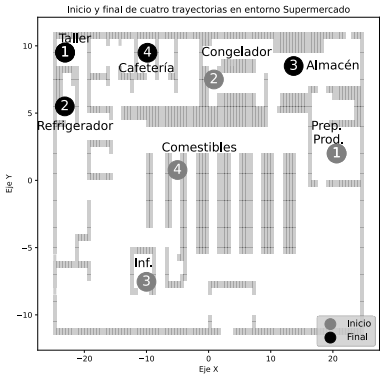


Figura 4.59 Inicio y final de cuatro trayectorias en entorno supermercado.

Análisis comparativo de los métodos en la generación de la trayectoria más extensa

A continuación, en la Figura 4.60 se presenta una comparación entre los métodos en relación con el tiempo e iteraciones necesarias para generar la trayectoria más extensa, donde la trayectoria más extensa denotada como la numero uno (Figura 4.59), se inicia en el "Cuarto de preparación de productos" y finaliza en el "Taller de reparación de productos". Cada comparación se ilustra considerando todas las ventanas de proximidad a los obstáculos, además de evaluar la influencia del uso de diagonales.

A continuación, se detallan las observaciones específicas para cada grafica mostrada en la Figura 4.60.

Comparación entre los métodos cuando no se incorporan diagonales

Comparación en términos de tiempo (Fig. 4.3 arriba-izquierda)

Cuando no se incorporan diagonales, el método DG exhibe el menor tiempo para lograr la convergencia en las ventanas $1x1$, $5x5$ y $7x7$. Por otro lado, para la ventana $3x3$, el método MTA muestra ser la opción más eficiente en términos de tiempo para alcanzar la convergencia.

Por otro lado, el método Convencional se posiciona como el enfoque más demandante en términos de tiempo para alcanzar la convergencia en las ventanas $1x1$, $3x3$ y $5x5$. En contraste, para la ventana $7x7$, es el método MTA el que requiere el mayor tiempo para lograr la convergencia.

Comparación en términos de iteraciones (Fig. 4.3 arriba-derecha)

Cuando no se incorporan diagonales, el método DG exhibe el menor número de iteraciones para lograr la convergencia en las ventanas $1x1$, $3x3$, $5x5$ y $7x7$.

Por otro lado, el método Convencional se posiciona como el enfoque más demandante en términos de tiempo para alcanzar la convergencia para todas las configuraciones de ventana.

Comparación entre los métodos cuando se incorporan diagonales

Comparación en términos de tiempo (Fig. 4.3 abajo-izquierda)

Al considerar la inclusión de diagonales, se reafirma que el método DG permanece como la elección óptima en cuanto a tiempo para lograr la convergencia en las ventanas de $1x1$, $5x5$ y $7x7$. Sin embargo, para la ventana $3x3$, el método MTA es el más eficiente en términos de tiempo.

Por otro lado, se observa que el método Convencional es el que demanda más tiempo para la convergencia en las ventanas de $1x1$ y $3x3$ cuando se incluyen diagonales. Mientras que, para las ventanas $5x5$ y $7x7$ el método MTA se posiciona como el método más lento

Comparación en términos de iteraciones (Fig. 4.3 abajo-derecha)

Al considerar la inclusión de diagonales, se reafirma que el método DG permanece como la elección óptima en cuanto a número de iteraciones requeridas para lograr la convergencia en las ventanas de $1x1$, $5x5$ y $7x7$. Sin embargo, para la ventana $3x3$, el método MTA es el más eficiente en términos iteraciones.

Por otro lado, se observa que el método Convencional es el que demanda más tiempo para la convergencia todas las ventanas.

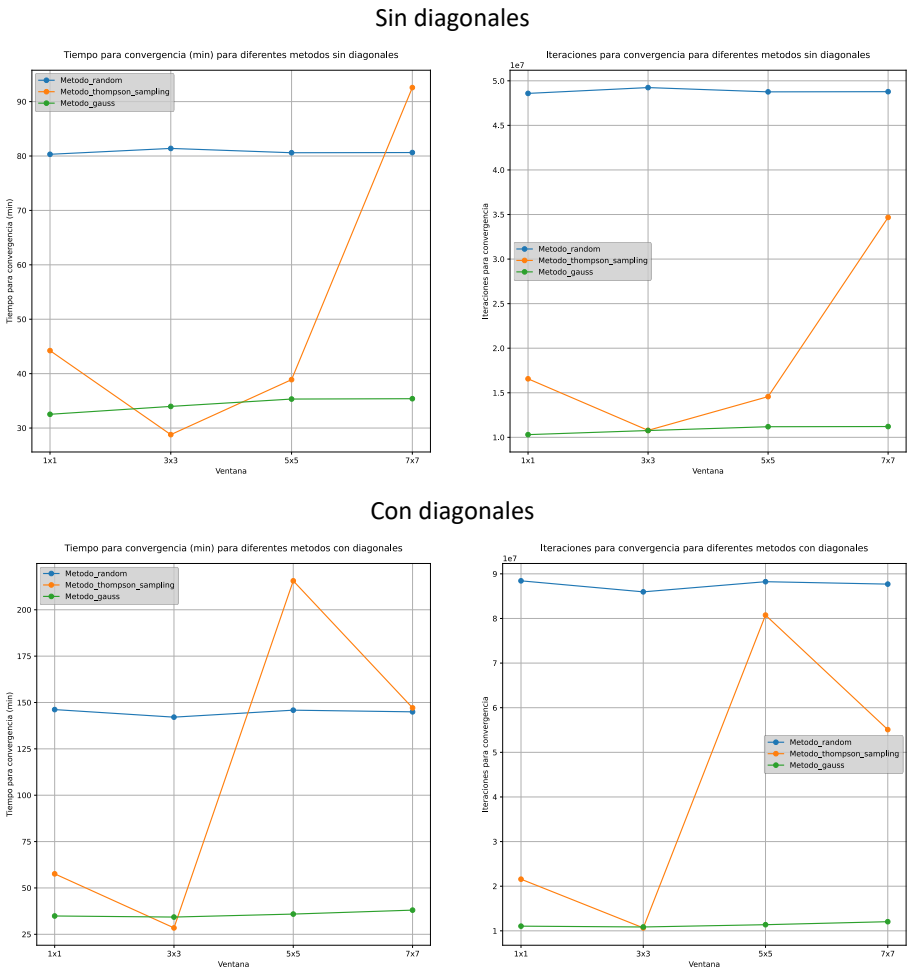


Figura 4.60 Comparación entre los métodos en términos de tiempo e iteraciones para las ventanas de proximidad al obstáculo.

Análisis del vuelo del cuadricóptero en simulación durante el seguimiento de la mejor trayectoria

A continuación, en la Tabla 4.31 se presenta la representación de la mejor trayectoria, obtenida mediante una ventana de $7x7$ sin la incorporación de diagonales. Esta trayectoria abarca una distancia de 58.5 metros. El cuadricóptero realiza un seguimiento a velocidad máxima de $2.583\,\frac{m}{s}$ y velocidad media de $1.779\,\frac{m}{s}$, completando el vuelo en 30.66 segundos. En cuanto a la métrica RMSE, se registra una desviación promedio de los ejes x e y de la trayectoria de 4.751 metros, y una desviación promedio en el ángulo yaw del cuadricóptero

de 44.225°. En relación con el porcentaje del error, se identifica un error del 89.236% en los ejes x e y, mientras que en el ángulo yaw hay una desviación de 7.884%.

Método	Ventana	RMSE XY	%error XY	RMSE Yaw	%error Yaw	Retardo (ms)	Distancia (m)	Velocidad máxima (m/s)	Velocidad media (m/s)	Tiempo de vuelo (s)
Sin diagonales										
Convencional, MTA, DG	7x7	4.751	89.236	44.225	7.884	30	58.5	2.583	1.779	30.66

Tabla 4.31 Análisis del vuelo del cuadricóptero en simulación durante el seguimiento de la mejor trayectoria.

A continuación, en la parte derecha de la Figura 4.61 se muestra el seguimiento en los ejes x e y de la trayectoria sin utilizar diagonales y utilizando la ventana $7x7$. Se puede notar que la desviación de la trayectoria ocurre en las secciones curvas, y al final, se evidencia una pérdida de control debido a la acumulación de velocidad. En la parte izquierda de la Figura 4.61, se puede observar la gráfica de la velocidad del cuadricóptero durante el seguimiento de la trayectoria, donde se observa que la máxima aceleración se produce en el inicio, mientras que, en la recta más larga de la trayectoria, el cuadricóptero se mantiene a una velocidad constante sin incrementar.

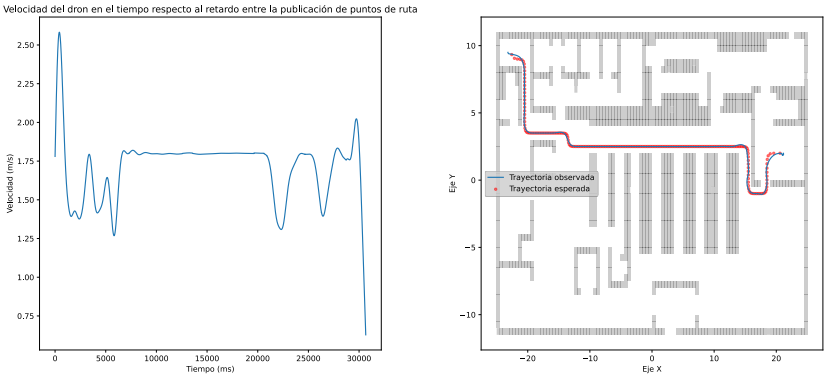


Figura 4.61 seguimiento en los ejes x e y de la trayectoria sin utilizar diagonales y utilizando la ventana $7x7$.

4.4.3 Comparación de tres métodos para la generación de trayectorias en un entorno inspirado en una oficina

El objetivo de este experimento es comparar diferentes métodos de Aprendizaje Q en la generación de trayectorias en un entorno simulado inspirado en una oficina.

Generación del entorno simulado inspirado en una oficina

El experimento se lleva a cabo en un entorno inspirado en una oficina (Figura 4.62). El entorno se ha construido sobre un *canvas* de $44x88$ $pixeles$. Dentro de este espacio, se han identificado 2938 $pixeles$ como espacios libres, con líneas de un $píxel$ de grosor se han representado las paredes que definen los limites de las habitaciones y corredores.

Selección de los puntos de ruta de inicio y fin de las trayectorias

Para generar trayectorias, se definieron cuatro puntos de inicio y fin. La Figura 4.1 muestra los números correspondientes a cada trayectoria, representando el inicio con un círculo gris y el final con un círculo negro. Cada trayectoria se diseñó para realizar una tarea de navegación en este entorno:

1. La trayectoria uno inicia en la “Sala de espera” y finaliza en la “Cafetería”.
2. La trayectoria dos inicia en la “Oficina 1” y finaliza en la “Papelería”.
3. La trayectoria tres inicia en la “Oficina 5” y finaliza en el “Almacén”.
4. La trayectoria cuatro inicia en la “Gerencia” y finaliza en la “Recepción”.

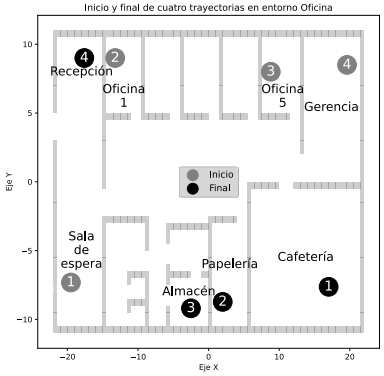


Figura 4.62 Inicio y final de cuatro trayectorias en entorno casa.

Análisis comparativo de los métodos en la generación de la trayectoria más extensa

A continuación, en la Figura 4.63 se presenta una comparación entre los métodos en relación con el tiempo e iteraciones necesarias para generar la trayectoria más extensa, donde la trayectoria más extensa denotada como la numero cuatro (Figura 4.62), se inicia en la "Gerencia" y finaliza en la "Recepción". Cada comparación se ilustra considerando todas las ventanas de proximidad a los obstáculos, además de evaluar la influencia del uso de diagonales.

A continuación, se detallan las observaciones específicas para cada grafica mostrada en la Figura 4.63.

Comparación entre los métodos cuando no se incorporan diagonales

Comparación en términos de tiempo (Fig. 4.3 arriba-izquierda)

Cuando no se incorporan diagonales, el método DG exhibe el menor tiempo para lograr la convergencia en las ventanas $1x1$, $5x5$ y $7x7$. Por otro lado, para la ventana $3x3$, el método MTA muestra ser la opción más eficiente en términos de tiempo para alcanzar la convergencia.

Por otro lado, el método Convencional se posiciona como el enfoque más demandante en términos de tiempo para alcanzar la convergencia en las ventanas $1x1$, $3x3$ y $7x7$. En contraste, para la ventana $5x5$, es el método MTA el que requiere el mayor tiempo para lograr la convergencia.

Comparación en términos de iteraciones (Fig. 4.3 arriba-derecha)

Cuando no se incorporan diagonales, el método DG exhibe el menor número de iteraciones para lograr la convergencia en las ventanas $1x1$, $3x3$, $5x5$ y $7x7$.

Por otro lado, el método Convencional se posiciona como el enfoque más demandante en términos de tiempo para alcanzar la convergencia para todas las configuraciones de ventana.

Comparación entre los métodos cuando se incorporan diagonales

Comparación en términos de tiempo (Fig. 4.3 abajo-izquierda)

Al considerar la inclusión de diagonales, se reafirma que el método DG permanece como la elección óptima en cuanto a tiempo para lograr la convergencia en las ventanas de $1x1$, $5x5$ y $7x7$. Sin embargo, para la ventana $3x3$, el método MTA es el más eficiente en términos de tiempo.

Por otro lado, se observa que el método Convencional es el que demanda más tiempo para la convergencia en las ventanas de $1x1$ y $3x3$ cuando se incluyen diagonales. Mientras que, para las ventanas $5x5$ y $7x7$ el método MTA se posiciona como el método más lento

Comparación en términos de iteraciones (Fig. 4.3 abajo-derecha)

Al considerar la inclusión de diagonales, se reafirma que el método DG permanece como la elección óptima en cuanto a número de iteraciones requeridas para lograr la convergencia en las ventanas de $1x1$, $5x5$ y $7x7$. Sin embargo, para la ventana $3x3$, el método MTA es el más eficiente en términos iteraciones.

Por otro lado, se observa que el método Convencional es el que demanda más iteraciones para la convergencia en las ventanas de $1x1$, $3x3$ y $5x5$ cuando se incluyen diagonales. Mientras que, para la ventana $7x7$ el método MTA se posiciona como el método que demanda más iteraciones.

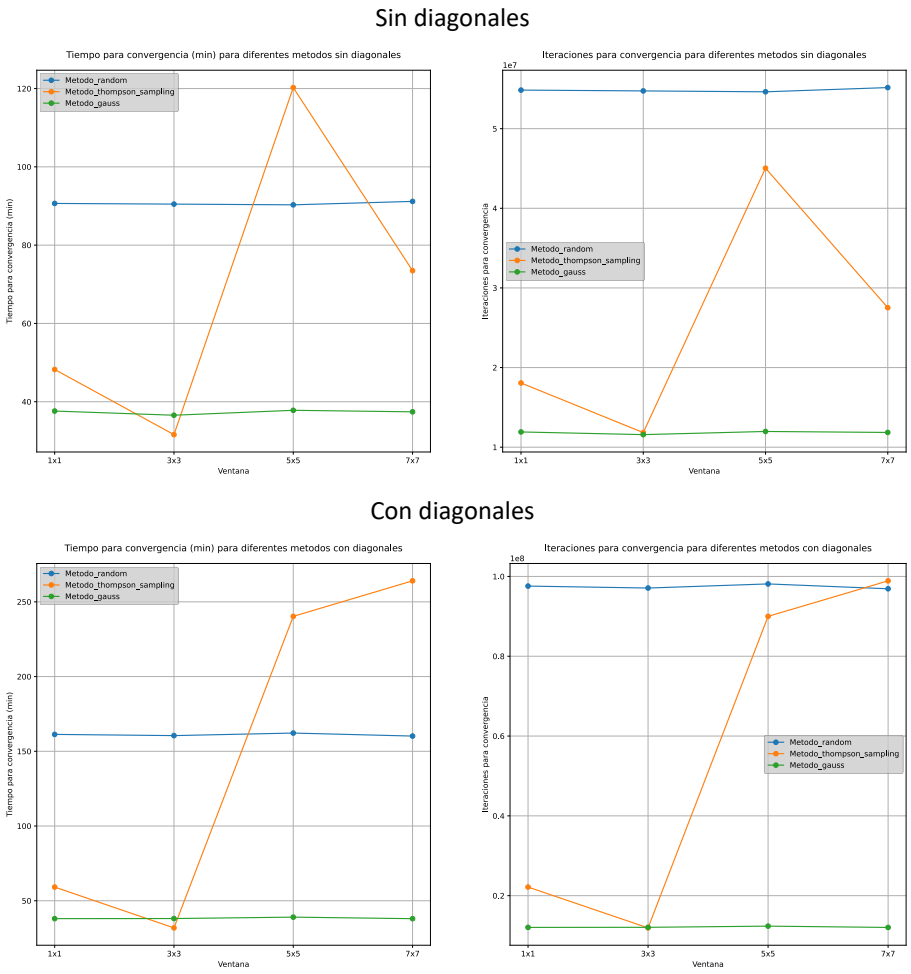


Figura 4.63 Comparación entre los métodos en términos de tiempo e iteraciones para las ventanas de proximidad al obstáculo.

Análisis del vuelo del cuadricóptero en simulación durante el seguimiento de la mejor trayectoria

A continuación, en la Tabla 4.18 se presenta la representación de la mejor trayectoria, obtenida mediante una ventana de $7x7$ sin la incorporación de diagonales. Esta trayectoria abarca una distancia de 58 metros. El cuadricóptero realiza un seguimiento a velocidad máxima de $4.693\frac{m}{s}$ y velocidad media de $3.541\frac{m}{s}$, completando el vuelo en 12.254 segundos. En cuanto a la métrica RMSE, se registra una desviación promedio de los ejes x e y de la trayectoria de 9.523 metros, y una desviación promedio en el ángulo yaw del

cuadricóptero de 56.210°. En relación con el porcentaje del error, se identifica un error del 279.37% en los ejes x e y, mientras que en el ángulo yaw hay una desviación de 13.871%.

Método	Ventana	RMSE XY	%error XY	RMSE Yaw	%error Yaw	Retardo (ms)	Distancia (m)	Velocidad máxima (m/s)	Velocidad media (m/s)	Tiempo de vuelo (s)
Sin diagonales										
Convencional, MTA, DG	7x7	9.523	279.37	56.210	13.871	12	58	4.693	3.541	12.254

Tabla 4.32 Análisis del vuelo del cuadricóptero en simulación durante el seguimiento de la mejor trayectoria.

A continuación, en la parte derecha de la Figura 4.64 se muestra el seguimiento en los ejes x e y de la trayectoria sin utilizar diagonales y utilizando la ventana $7x7$. Se puede notar que la desviación de la trayectoria ocurre en las secciones curvas, y al final, se evidencia una pérdida de control debido a la acumulación de velocidad. En la parte izquierda de la Figura 4.64, se puede observar la gráfica de la velocidad del cuadricóptero durante el seguimiento de la trayectoria, donde se observa que la máxima aceleración se produce en el final, mientras que, en la recta más larga de la trayectoria, el cuadricóptero se mantiene a una velocidad constante sin incrementar notoriamente.

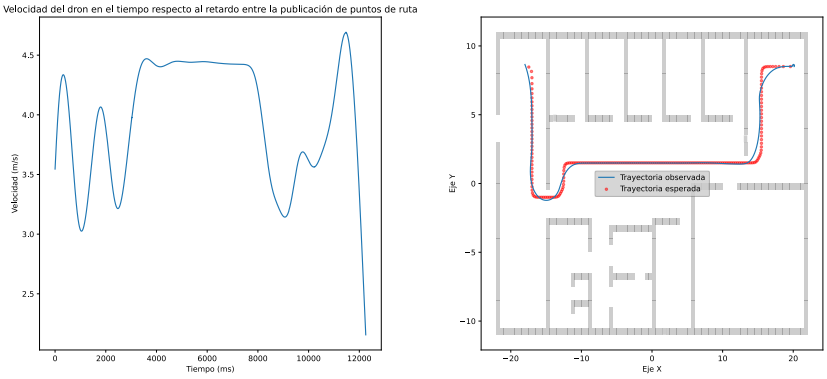


Figura 4.64 seguimiento en los ejes x e y de la trayectoria sin utilizar diagonales y utilizando la ventana $7x7$.

4.4.4 Comparación de tres métodos para la generación de trayectorias en un entorno inspirado en una farmacia

El objetivo de este experimento es comparar diferentes métodos de Aprendizaje Q en la generación de trayectorias en un entorno simulado inspirado en una oficina.

Generación del entorno simulado inspirado en una oficina

El experimento se lleva a cabo en un entorno inspirado en una oficina (Figura 4.1). El entorno se ha construido sobre un *canvas* de $44x88$ $pixeles$. Dentro de este espacio, se han identificado 2938 $pixeles$ como espacios libres, con líneas de un $píxel$ de grosor se han representado las paredes que definen los limites de las habitaciones y corredores.

Selección de los puntos de ruta de inicio y fin de las trayectorias

Para generar trayectorias, se definieron cuatro puntos de inicio y fin. La Figura 4.1 muestra los números correspondientes a cada trayectoria, representando el inicio con un círculo gris y el final con un círculo negro. Cada trayectoria se diseñó para realizar una tarea de navegación en este entorno:

1. La trayectoria uno inicia en el estacionamiento de “Drones” y finaliza en el “Cuarto de limpieza”.
2. La trayectoria dos inicia en la “Recepción de muestras” y finaliza en el “Laboratorio”.
3. La trayectoria tres inicia en el “Almacén” y finaliza en la “Farmacia”.
4. La trayectoria cuatro inicia en “Atención al cliente” y finaliza en la “Sala de espera”.

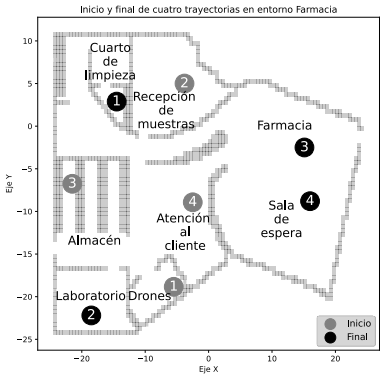


Figura 4.65 Inicio y final de cuatro trayectorias en entorno Farmacia.

Análisis comparativo de los métodos en la generación de la trayectoria más extensa

A continuación, en la Figura 4.66 se presenta una comparación entre los métodos en relación con el tiempo e iteraciones necesarias para generar la trayectoria más extensa, donde la trayectoria más extensa denotada como la numero tres (Figura 4.65), se inicia en el "Almacén" y finaliza en la "Farmacia". Cada comparación se ilustra considerando todas las ventanas de proximidad a los obstáculos, además de evaluar la influencia del uso de diagonales.

A continuación, se detallan las observaciones específicas para cada grafica mostrada en la Figura 4.66.

Comparación entre los métodos cuando no se incorporan diagonales

Comparación en términos de tiempo (Fig. 4.3 arriba-izquierda)

Cuando no se incorporan diagonales, el método DG exhibe el menor tiempo para lograr la convergencia en las ventanas $1x1$, $5x5$ y $7x7$. Por otro lado, para la ventana $3x3$, el método MTA muestra ser la opción más eficiente en términos de tiempo para alcanzar la convergencia.

Por otro lado, el método Convencional se posiciona como el enfoque más demandante en términos de tiempo para alcanzar la convergencia en las ventanas $1x1$ y $3x3$. En contraste,

para las ventanas $5x5$ y $7x7$ es el método MTA el que requiere el mayor tiempo para lograr la convergencia.

Comparación en términos de iteraciones (Fig. 4.3 arriba-derecha)

Cuando no se incorporan diagonales, el método DG exhibe el menor número de iteraciones para lograr la convergencia en las ventanas $1x1$, $5x5$ y $7x7$. Por otro lado, para la ventana $3x3$, el método MTA muestra ser la opción más eficiente en términos de iteraciones para alcanzar la convergencia.

Por otro lado, el método Convencional se posiciona como el enfoque más demandante en términos de iteraciones para alcanzar la convergencia en las ventanas $1x1$ y $3x3$. En contraste, para las ventanas $5x5$ y $7x7$ es el método MTA el que requiere el mayor número de iteraciones para lograr la convergencia.

Comparación entre los métodos cuando se incorporan diagonales

Comparación en términos de tiempo (Fig. 4.3 abajo-izquierda)

Al considerar la inclusión de diagonales, se reafirma que el método DG permanece como la elección óptima en cuanto a tiempo para lograr la convergencia en las ventanas de $1x1$, $5x5$ y $7x7$. Sin embargo, para la ventana $3x3$, el método MTA es el más eficiente en términos de tiempo.

Por otro lado, se observa que el método Convencional es el que demanda más tiempo para la convergencia en las ventanas de $1x1$ y $3x3$ cuando se incluyen diagonales. Mientras que, para las ventanas $5x5$ y $7x7$ el método MTA se posiciona como el método más lento

Comparación en términos de iteraciones (Fig. 4.3 abajo-derecha)

Al considerar la inclusión de diagonales, se reafirma que el método DG permanece como la elección óptima en cuanto a número de iteraciones requeridas para lograr la convergencia en las ventanas de $1x1$, $5x5$ y $7x7$. Sin embargo, para la ventana $3x3$, el método MTA es el más eficiente en términos iteraciones.

Por otro lado, se observa que el método Convencional es el que demanda más iteraciones para la convergencia en las ventanas de $1x1$ y $3x3$ cuando se incluyen diagonales.

Mientras que, para las ventanas $5x5$ y $7x7$ el método MTA se posiciona como el método que demanda más iteraciones.

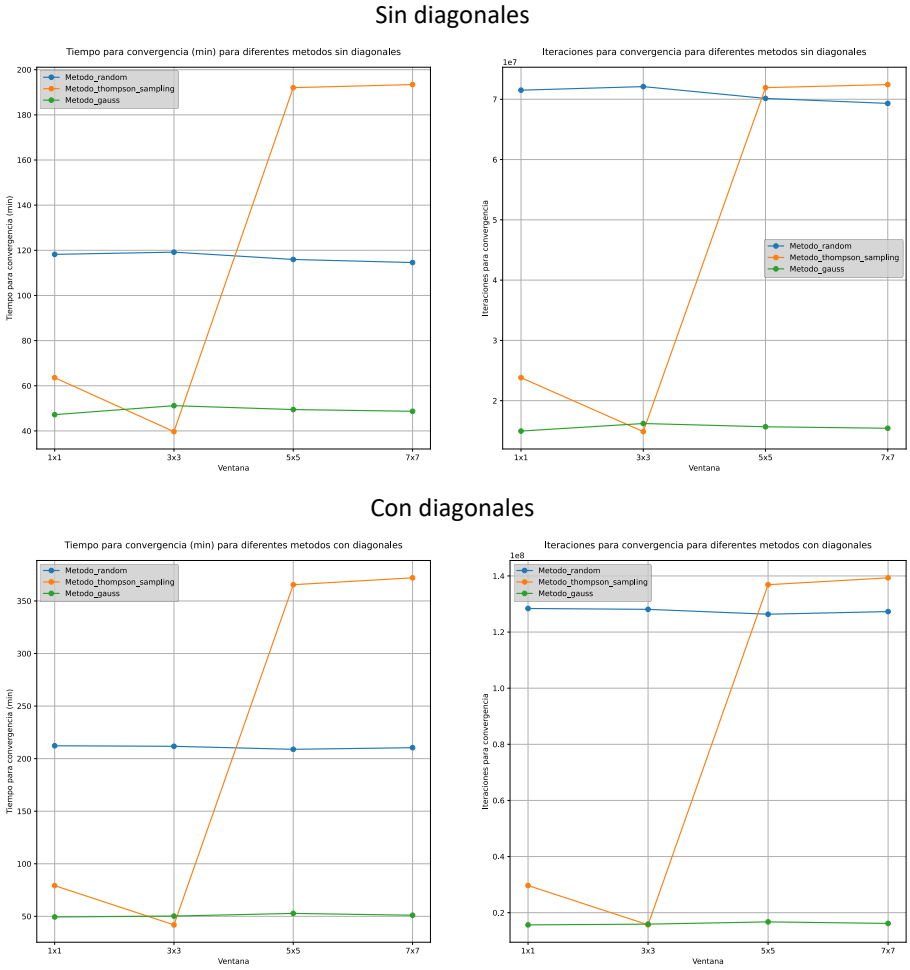


Figura 4.66 Comparación entre los métodos en términos de tiempo e iteraciones para las ventanas de proximidad al obstáculo.

Análisis del vuelo del cuadricóptero en simulación durante el seguimiento de la mejor trayectoria

A continuación, en la Tabla 4.33 se presenta la representación de la mejor trayectoria, obtenida mediante una ventana de $5x5$ incorporando diagonales. Esta trayectoria abarca una distancia de 44 metros. El cuadricóptero realiza un seguimiento a velocidad máxima de $6.611\,\frac{m}{s}$ y velocidad media de $5.304\,\frac{m}{s}$, completando el vuelo en 8.469 segundos. En cuanto

a la métrica RMSE, se registra una desviación promedio de los ejes x e y de la trayectoria de 8.429 metros, y una desviación promedio en el ángulo yaw del cuadricóptero de 53.187°. En relación con el porcentaje del error, se identifica un error del 38.555% en los ejes x e y, mientras que en el ángulo yaw hay una desviación de 23.010%.

Método	Ventana	RMSE XY	%error XY	RMSE Yaw	%error Yaw	Retardo (ms)	Distancia (m)	Velocidad máxima (m/s)	Velocidad media (m/s)	Tiempo de vuelo (s)
Con diagonales										
Convencional, MTA, DG	5x5	8.429	38.555	53.187	23.010	8	44	6.611	5.304	8.469

Tabla 4.33 Análisis del vuelo del cuadricóptero en simulación durante el seguimiento de la mejor trayectoria.

A continuación, en la parte derecha de la Figura 4.67 se muestra el seguimiento en los ejes x e y de la trayectoria incorporando diagonales y utilizando la ventana $5x5$. Se puede notar que la desviación de la trayectoria ocurre en las secciones curvas de manera suave. En la parte izquierda de la Figura 4.67, se puede observar la gráfica de la velocidad del cuadricóptero durante el seguimiento de la trayectoria, donde se observa que la máxima aceleración se produce a la mitad de la trayectoria, donde la diagonal se mantiene continua una mayor distancia.

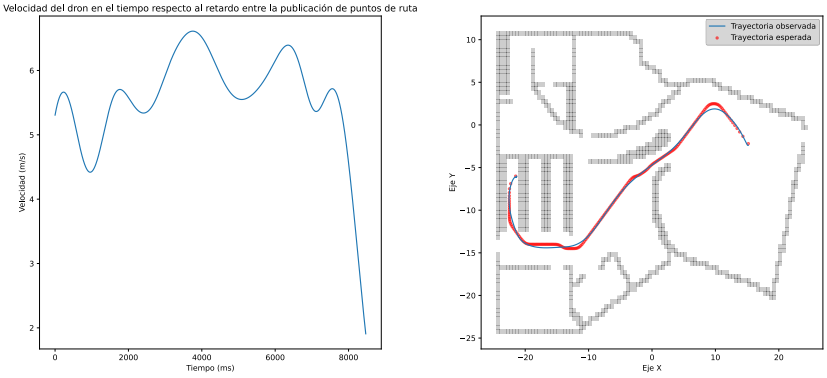


Figura 4.67 seguimiento en los ejes x e y de la trayectoria sin utilizar diagonales y utilizando la ventana $7x7$.

4.4.5 Comparación de tres métodos para la generación de trayectorias en un entorno inspirado en una bodega

El objetivo de este experimento es comparar diferentes métodos de Aprendizaje Q en la generación de trayectorias en un entorno simulado inspirado en una oficina.

Generación del entorno simulado inspirado en una oficina

El experimento se lleva a cabo en un entorno inspirado en una oficina (Figura 4.68). El entorno se ha construido sobre un *canvas* de $44x88$ $pixeles$. Dentro de este espacio, se han identificado 2938 $pixeles$ como espacios libres, con líneas de un $píxel$ de grosor se han representado las paredes que definen los limites de las habitaciones y corredores.

Selección de los puntos de ruta de inicio y fin de las trayectorias

Para generar trayectorias, se definieron cuatro puntos de inicio y fin. La Figura 4.68 muestra los números correspondientes a cada trayectoria, representando el inicio con un círculo gris y el final con un círculo negro. Cada trayectoria se diseñó para realizar una tarea de navegación en este entorno:

1. La trayectoria uno inicia en la “Zona de descarga” y finaliza en la “Estantería C-7”.
2. La trayectoria dos inicia en la “Zona de carga” y finaliza en la “Estantería C-2”.
3. La trayectoria tres inicia en la “Estantería D-4” y finaliza en la “Estantería B-2”.
4. La trayectoria cuatro inicia en la “Oficina” y finaliza en la “Entrada”.

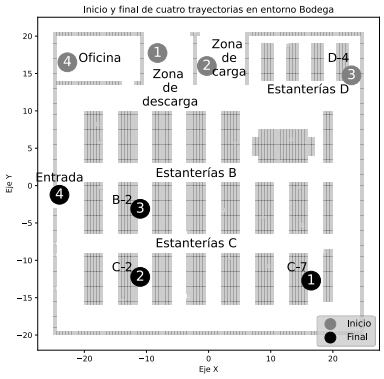


Figura 4.68 Inicio y final de cuatro trayectorias en entorno bodega.

Análisis comparativo de los métodos en la generación de la trayectoria más extensa

A continuación, en la Figura 4.69 se presenta una comparación entre los métodos en relación con el tiempo e iteraciones necesarias para generar la trayectoria más extensa, donde la trayectoria más extensa denotada como la numero tres (Figura 4.50), se inicia en la "Estantería D-4" y finaliza en la "Estantería B-2". Cada comparación se ilustra considerando todas las ventanas de proximidad a los obstáculos, además de evaluar la influencia del uso de diagonales.

A continuación, se detallan las observaciones específicas para cada grafica mostrada en la Figura 4.69.

Comparación entre los métodos cuando no se incorporan diagonales

Comparación en términos de tiempo (Fig. 4.3 arriba-izquierda)

Cuando no se incorporan diagonales, el método DG exhibe el menor tiempo para lograr la convergencia en las ventanas $1x1$, $5x5$ y $7x7$. Por otro lado, para la ventana $3x3$, el método MTA muestra ser la opción más eficiente en términos de tiempo para alcanzar la convergencia.

Por otro lado, el método Convencional se posiciona como el enfoque más demandante en términos de tiempo para alcanzar la convergencia en todas las ventanas.

Comparación en términos de iteraciones (Fig. 4.3 arriba-derecha)

Cuando no se incorporan diagonales, el método DG exhibe el menor número de iteraciones para lograr la convergencia en las ventanas $1x1$, $5x5$ y $7x7$. Por otro lado, para la ventana $3x3$, el método MTA muestra ser la opción más eficiente en términos de iteraciones para alcanzar la convergencia.

Por otro lado, el método Convencional se posiciona como el enfoque más demandante en términos de iteraciones para alcanzar la convergencia en todas las ventanas.

Comparación entre los métodos cuando se incorporan diagonales

Comparación en términos de tiempo (Fig. 4.3 abajo-izquierda)

Al considerar la inclusión de diagonales, se reafirma que el método DG permanece como la elección óptima en cuanto a tiempo para lograr la convergencia en las ventanas de $1x1$, $5x5$ y $7x7$. Sin embargo, para la ventana $3x3$, el método MTA es el más eficiente en términos de tiempo.

Por otro lado, se observa que el método Convencional es el que demanda más tiempo para la convergencia en las ventanas de $1x1$ y $3x3$ cuando se incluyen diagonales. Mientras que, para las ventanas $5x5$ y $7x7$ el método MTA se posiciona como el método más lento

Comparación en términos de iteraciones (Fig. 4.3 abajo-derecha)

Al considerar la inclusión de diagonales, se reafirma que el método DG permanece como la elección óptima en cuanto a número de iteraciones requeridas para lograr la convergencia en las ventanas de $1x1$, $5x5$ y $7x7$. Sin embargo, para la ventana $3x3$, el método MTA es el más eficiente en términos iteraciones.

Por otro lado, se observa que el método Convencional es el que demanda más iteraciones para la convergencia en las ventanas de $1x1$ y $3x3$ cuando se incluyen diagonales. Mientras que, para las ventanas $5x5$ y $7x7$ el método MTA se posiciona como el método que demanda más iteraciones.

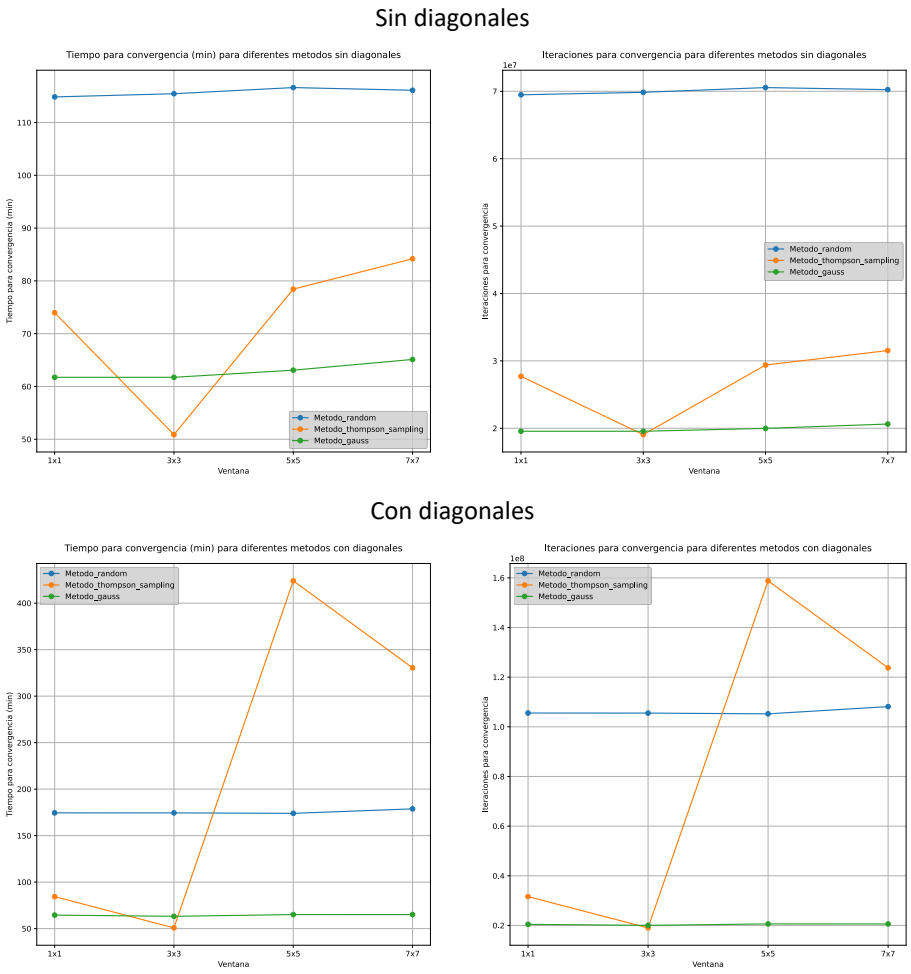


Figura 4.69 Comparación entre los métodos en términos de tiempo e iteraciones para las ventanas de proximidad al obstáculo.

A continuación, se presenta una tabla que muestra las mejoras porcentuales en tiempo y número de iteraciones para los métodos DG y MTA en comparación con el método convencional.

Cuando no se incorporan diagonales con el método DG, el mayor porcentaje de mejora, en cuanto a tiempo, se registra en la ventana $3x3$, con un 46.529%. En cuanto al número de iteraciones, la mejora es de 72.002% en la misma ventana $3x3$.

Por otro lado, cuando no se incorporan diagonales con el método MTA, en cuanto a tiempo, el mayor porcentaje de mejora es de 55.933% registrado con la ventana $3x3$, en cuanto a iteraciones la mejora es de 72.718% en la misma ventana $3x3$.

Cuando se incorporan diagonales con el método DG, en cuanto a tiempo, el mayor porcentaje de mejora es de 63.759% registrado con la ventana $3x3$, en cuanto a iteraciones la mejora es de 81.025% en la misma ventana $3x3$.

Por otro lado, cuando se incorporan diagonales con el método MTA, en cuanto a tiempo, el mayor porcentaje de mejora es de 70.896% con la ventana $3x3$, en cuanto a iteraciones la mejora es de 81.982% con la misma ventana $3x3$.

	DG		MTA	
	Tiempo	Iteraciones	Tiempo	Iteraciones
Ventana	Mejora (%)			
Sin diagonales				
1x1	46.246	71.854	35.589	60.123
3x3	**46.529**	**72.002**	**55.933**	**72.718**
5x5	45.902	71.674	32.737	58.357
7x7	43.918	70.636	27.494	55.112
Con diagonales				
1x1	63.028	80.642	51.623	70.049
3x3	**63.759**	**81.025**	**70.896**	**81.982**
5x5	62.581	80.407	-143.767	-50.916
7x7	63.608	80.945	-84.852	-14.442

Tabla 4.34 mejoras porcentuales para los métodos DG y MTA en comparación con el método convencional.

Análisis del vuelo del cuadricóptero en simulación durante el seguimiento de la mejor trayectoria

A continuación, en la Tabla 4.35 se presenta la representación de la mejor trayectoria, obtenida mediante una ventana de 7x7 sin la incorporación de diagonales. Esta trayectoria abarca una distancia de 57 metros. El cuadricóptero realiza un seguimiento a velocidad máxima de $6.780\,\frac{m}{s}$ y velocidad media de $1.779\,\frac{m}{s}$, completando el vuelo en 30.66 segundos. En cuanto a la métrica RMSE, se registra una desviación promedio de los ejes x e y de la trayectoria de 8.329 metros, y una desviación promedio en el ángulo yaw del cuadricóptero

de 78.032°. En relación con el porcentaje del error, se identifica un error del 60.600% en los ejes x e y, mientras que en el ángulo yaw hay una desviación de 25.385%.

Método	Ventana	RMSE XY	%error XY	RMSE Yaw	%error Yaw	Retardo (ms)	Distancia (m)	Velocidad máxima (m/s)	Velocidad media (m/s)	Tiempo de vuelo (s)
Sin diagonales										
Convencional, MTA, DG	7x7	8.329	60.600	78.032	25.385	9	57	6.780	5.332	9.195

Tabla 4.35 Análisis del vuelo del cuadricóptero en simulación durante el seguimiento de la mejor trayectoria.

A continuación, en la parte derecha de la Figura 4.70 se muestra el seguimiento en los ejes x e y de la trayectoria sin utilizar diagonales y utilizando la ventana $7x7$. Se pueden notar que la desviación de la trayectoria ocurre en las secciones curvas. En la parte izquierda de la Figura 4.7, se puede observar la gráfica de la velocidad del cuadricóptero durante el seguimiento de la trayectoria, donde se observa que la máxima aceleración se produce en la primera curva.

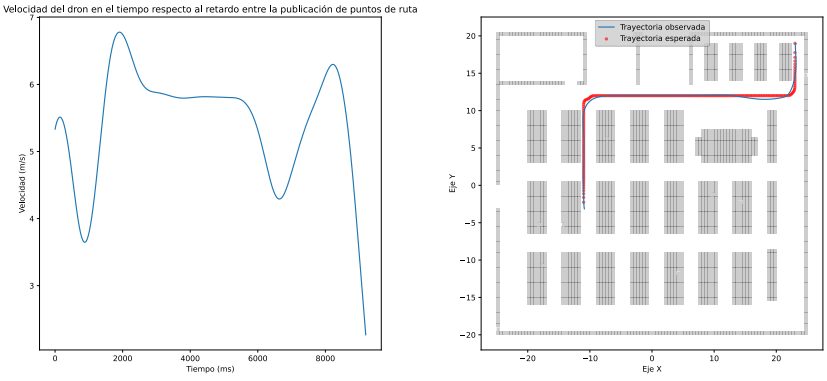


Figura 4.70 seguimiento en los ejes x e y de la trayectoria sin utilizar diagonales y utilizando la ventana $7x7$.

4.4.6 Conclusión

En este experimento comparativo entre métodos para generar trayectorias en diferentes configuraciones de ventanas de proximidad a obstáculos, considerando la incorporación y no incorporación de diagonales, se han identificado patrones y tendencias significativas:

Cuando no se incorporan diagonales, tanto el RMSE como el porcentaje de error se mantienen iguales en todas las ventanas, lo que sugiere que todos los métodos convergen hacia la misma trayectoria en la mayoría de los entornos. Se identificó una excepción en la oficina, donde el método MTA con la ventana $5x5$ mostro una leve variación en la trayectoria en comparación con el método convencional y DG (Figura 4.71). Es importante destacar que esta diferencia fue mínima y afectó solo a un estado de la trayectoria en el método MTA, específicamente en la coordenada $[15, 8.5]$.

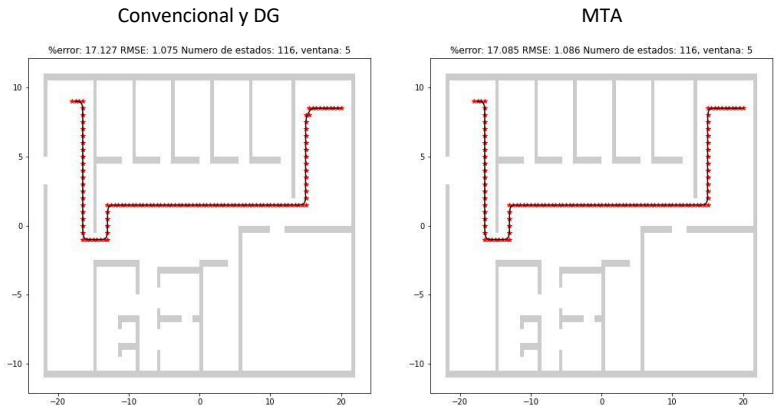


Figura 4.71 Diferencia entre trayectorias MTA, DG y convencional.

Por otro lado, al incorporan diagonales, nuevamente se observó que tanto el RMSE como el porcentaje de error se mantuvieron iguales en la mayoría de los entornos. No obstante, hubo algunas excepciones notables: en la bodega, se detectó una leve variación en la ventana $1x1$, y en la casa, aunque se registraron algunas variaciones en todas las ventanas, estas fueron mínimas. La diferencia más significativa se presentó solo en el método MTA

con la ventana 5x5, en comparación con el método convencional y DG (Figura 4.72). Esta diferencia fue destacable, afectando diez coordenadas de la trayectoria en el método MTA, específicamente en las coordenadas $[7.5, -6.5]$, $[8, -6]$,. y en el rango de -4 a -8 en el eje y, al final.

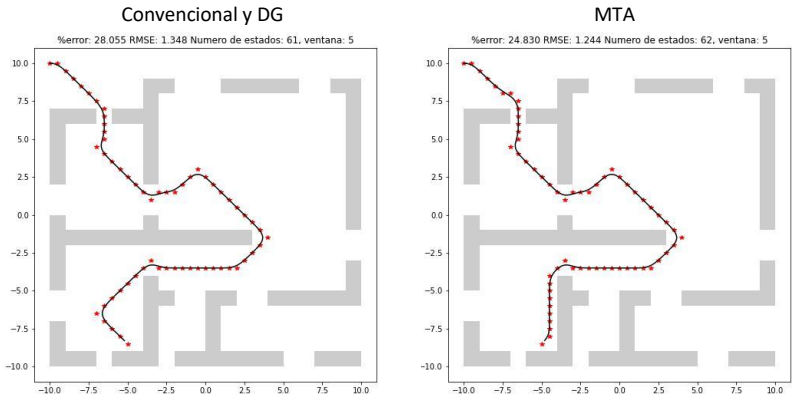


Figura 4.72 Diferencia entre trayectorias MTA, DG y convencional.

En general, se puede concluir que todos los métodos tienden a converger hacia la misma trayectoria en la mayoría de los casos, con solo una excepción significativa donde las diferencia en la trayectoria fue notable.

Además, es importante señalar que las rutas intermedias generadas con diferentes números de iteraciones antes de la convergencia introducen variabilidad en las trayectorias para los métodos. Esto se ejemplifica utilizando el método MTA con la ventana $7x7$ en la Figura 4.73, donde en la trayectoria a) se muestra una solución con 300 mil iteraciones, en la trayectoria b) se presenta una trayectoria con 600 mil iteraciones y en la trayectoria c) se presenta una trayectoria con 900 mil iteraciones y d) es la trayectoria a la cual converge el algoritmo luego de 5,938,973 iteraciones. Esta variabilidad puede proporcionar oportunidades para explorar configuraciones en el vuelo del cuadricóptero y encontrar la mejor condición para la convergencia del algoritmo.

Es importante notar que las trayectorias c) y d) son la misma trayectoria. En este caso específico, para la trayectoria c) se requirieron 900 mil iteraciones (2.403 minutos), en

contraste con las 5,938,973 iteraciones (15.857 minutos) requeridas para la trayectoria d) con la condición de convergencia empleada en este estudio. Estos resultados sugieren la importancia de considerar otras heurísticas para la convergencia como parte del análisis y optimización del algoritmo.

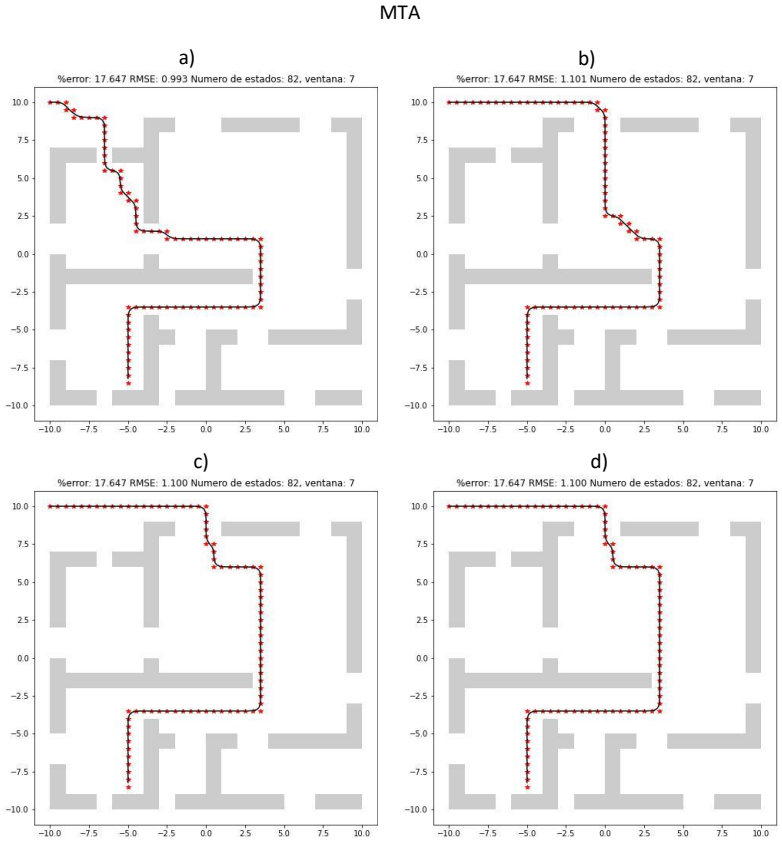


Figura 4.73 Rutas intermedias generadas con diferentes números de iteraciones.

En términos de eficiencia, el método DG se destaca como el más eficiente en cuanto a tiempo y número de iteraciones requeridos para lograr la convergencia, para las ventanas $1x1$,$5x5$ y $7x7$, tanto con la inclusión como sin la inclusión de diagonales. Sin embargo, para la ventana de $3x3$, el método MTA demostró ser la opción más rápida y eficiente en términos de tiempo e iteraciones requeridas para alcanzar la convergencia.

En cuanto al vuelo del cuadricóptero en simulación, la configuración más favorable para generar la trayectoria en términos de eficiencia y cumplimiento de la tarea de navegación

suele ser sin diagonales, utilizando la ventana $7x7$ y el método DG. No obstante, se observó una excepción en la farmacia donde la mejor trayectoria se obtuvo incorporando diagonales con ventana $5x5$ y el método DG. Es importante destacar que, aunque existen ventanas con menor error en métricas, el tiempo de vuelo es el factor determinante en la elección de la mejor configuración. Esto se debe a que es posible reducir el error disminuyendo la velocidad del cuadricóptero durante el seguimiento de la trayectoria mediante un aumento en el retardo temporal**.**

Resulta evidente la existencia de un compromiso intrínseco entre el número de coordenadas de la trayectoria (n), el retardo temporal (r) y la precisión en el vuelo (P). Este compromiso se manifiesta mediante la relación funcional, donde un incremento en n o r resulta en un aumento significativo de P, de lo contrario si se disminuye n o r resulta en un decremento significativo en P.

$$P = f(n, r) \tag{4.1}$$

Es importante destacar que las trayectorias que incorporan diagonales son menos extensas en comparación con aquellas que no las incluyen. Esto muestra una oportunidad de mejora que merece ser explorada. Como se pudo observar en el caso de la farmacia, utilizando la ventana $5x5$ con la incorporación de diagonales, se logró una de las trayectorias que pudieron ser ejecutadas a una mayor velocidad.

La mayor velocidad máxima obtenida por el cuadricóptero durante la navegación fue $6.780\,\frac{m}{s}$, la cual se dio en la bodega sin diagonales con la ventana $7x7$. Además, cabe resaltar el caso de la farmacia, donde se logró una notable velocidad máxima $6.611\,\frac{m}{s}$ al utilizar la ventana $5x5$ con la incorporación de diagonales.

Se observo que tanto el tiempo como el número de iteraciones aumentan a medida que aumenta el número de estados en el entorno, En el caso de la casa, que es el entorno con el menor número de estados (1123), el tiempo mínimo registrado para generar una trayectoria fue de 11.553 minutos $(4{,}326{,}894$ iteraciones) utilizando el método MTA, sin

incorporar diagonales, con la ventana $3x3$, mientras que el tiempo máximo, también con el método MTA, pero incorporando diagonales y con una ventana de $5x5$, alcanzó los 77.602 minutos (29,064,490 iteraciones).

Por otro lado, en el caso de la bodega, que es el entorno con el mayor número de estados (4853), el tiempo mínimo registrado para generar una trayectoria fue de 50.878 minutos (19,055,468 iteraciones) utilizando el método MTA, sin incorporar diagonales, con la ventana $3x3$, mientras que el tiempo máximo, también con el método MTA, pero incorporando diagonales y con una ventana de $5x5$, alcanzó los 423.993 minutos (158,798,967 iteraciones).

Considerando que la mejor trayectoria se obtiene con la ventana 7x7, sin incorporar diagonales y con el método GA, el tiempo necesario para generarla en el caso de la casa es de 14.127 minutos (4,474,822 iteraciones), lo que representa un porcentaje de mejora respecto al método convencional del 49% , mientras que en el caso de la bodega es de 65.118 minutos (20,626,422 iteraciones), lo que representa un porcentaje de mejora respecto al método convencional del 44%.

En conjunto, estos resultados destacan la importancia de considerar las configuraciones especificas al elegir un método de generación de trayectorias. Las similitudes y diferencias identificadas proporcionan información valiosa para la toma de decisiones informadas en función de las necesidades y restricciones del contexto.

Capítulo 5: Conclusiones y recomendaciones

En este capítulo se presentan las conclusiones del trabajo, así como también las recomendaciones para la mejora del modelo desarrollado y los posibles trabajos futuros que podrían darle continuidad a esta investigación.

5.1 CONCLUSIONES

Se han introducido dos métodos para equilibrar la exploración y la explotación en un problema modelado como un PDM discreto que aborda entornos que involucran decenas a miles de estados previamente desconocidos utilizando el algoritmo de Aprendizaje-Q: MTA y DG. Notablemente, estos métodos demostraron un aumento del 81% en eficiencia, en términos de iteraciones para alcanzar la convergencia, en comparación con el método convencional.

Se ha observado que el método MTA funciona de manera óptima cuando se utiliza una ventana $3x3$, porque la distribución de la recompensa en el entorno permite que MTA explore más de una acción con la máxima recompensa, y explore pocas acciones con menor recompensa.

Sin embargo, al ampliar la ventana a $5x5$, se observa que MTA requiere un 50.916% más de iteraciones que el método convencional para alcanzar la convergencia. Esto se debe a que la distribución de la recompensa en el entorno hace que MTA explote solo una acción con la máxima recompensa y no explore las acciones con menos recompensa.

Se ha notado que el método DG exhibe una consistencia notable en su desempeño en las ventanas $1x1$, 3x3, $5x5$ y $7x7$ siendo 0.5% mejor, en cuanto al número de iteraciones, en la ventana $3x3$, porque la distribución de la recompensa en el entorno permite que DG explore más de una acción con la máxima recompensa, y explore menos acciones con menor recompensa.

Por lo anterior, la elección de la ventana para los métodos MTA y DG esta intrínsecamente relacionada con la distribución de recompensas en el entorno, y una ventana

$3x3$ ha demostrado ser más efectiva que las ventanas $1x1$, $5x5$ y $7x7$. Cabe mencionar que MTA fue superior un 9%, en el número de iteraciones para alcanzar la convergencia, a DG con ventana $3x3$, porque DG explora arriba del 10% de las iteraciones las recompensas bajas en el entorno, mientras que MTA las explora abajo del 1% de las iteraciones. Por lo mismo DG se mantiene consistente en su desempeño, mientras que MTA solo es óptimo con ventana $3x3$.

En esta investigación, se logró desarrollar un modelo computacional que permite la simulación de la navegación autónoma 2D de un cuadricóptero en entornos estáticos de interiores utilizando el algoritmo de Aprendizaje-Q. Además, se consiguió optimizar el modelo con una configuración para generar trayectorias libres de colisiones que cumplen con precisión los objetivos de navegación. Este avance se presenta como una herramienta para la prueba y validación de trayectorias en interiores antes de su implementación en entornos reales.

El modelo computacional presentado se centra en la planificación de trayectorias en entornos previamente diseñados. Esto implica que el sistema no realiza la planificación en tiempo real, lo que puede ser una ventaja en términos de reducir la carga computacional a bordo del cuadricóptero.

Un aspecto crítico es que estas trayectorias deben generarse antes del vuelo porque, entrenar una tabla Q para obtener trayectorias efectivas puede ser un proceso que consume mucho tiempo, lo que implica que la planificación debe realizarse con anticipación. Cabe mencionar que existe la posibilidad de mejorar el tiempo para generar una trayectoria mediante la investigación de mejores heurísticas para la convergencia del algoritmo.

En cuanto a los métodos de interpolación utilizados para suavizar la trayectoria, se encontró que el mejor método para la navegación en interiores con poco espacio para maniobrar es B-spline de grado 10. Por otro lado, como se evidenció en el Experimento 4.1, Bézier es útil en entornos abiertos, pero presenta limitaciones en espacios estrechos.

También se encontró que la incorporación de ventanas $5x5$ y $7x7$, que reducen la recompensa de los estados ubicados a distancia de 1 y 1.5 metros del obstáculo, generan trayectorias más alejadas de los obstáculos que permiten un vuelo seguro y preciso.

En resumen, este trabajo resalta la viabilidad de utilizar el aprendizaje por refuerzo para generar trayectorias de vuelo autónomo de UAVs en simulación. Además, se introducen los métodos MTA y DG que son dos innovadores enfoques para acelerar la convergencia del algoritmo Aprendizaje-Q. También muestra la importancia de la planificación de trayectorias en función de los obstáculos en el entorno. En conjunto, estos hallazgos son fundamentales para el desarrollo y la optimización de sistemas de navegación autónoma en entornos interiores. Los resultados son prometedores y sugieren aplicaciones futuras tanto en simulación como en entornos reales.

5.2 RECOMENDACIONES

En esta sección, se presentan recomendaciones para utilizar de manera efectiva el modelo computacional desarrollado en esta tesis.

Es aconsejable que se inicie la exploración del modelo utilizando la configuración más éxitos obtenida en los experimentos de esta tesis, que consiste en establecer $\gamma = 0.99$, $\alpha = 0.9$, aplicar B-spline de grado 10 y emplear el método DG con la ventana $7x7$, excluyendo la inclusión de diagonales. Posteriormente, se pueden buscar mejoras explorando otras opciones y ajustes, con esta configuración como punto de partida.

En aplicaciones futuras en entornos exteriores que no estén tan restringidos en términos de espacio, se puede considerar el uso de Bézier y la inclusión de diagonales.

Dado que el proceso de entrenamiento del algoritmo de Aprendizaje-Q puede ser intensivo en términos de tiempo, se recomienda almacenar todas las tablas Q generada durante este proceso. Esto no solo permite preservar los resultados obtenidos, sino que también ofrece la posibilidad de continuar el entrenamiento en un momento posterior.

Para lograr un uso más eficiente del tiempo y de los recursos computacionales, se recomienda emplear cómputo en paralelo mediante la ejecución simultánea de múltiples procesos para entrenar varias tablas Q al mismo tiempo.

5.3 TRABAJOS FUTUROS

En esta sección, se presentan sugerencias para futuros trabajos destinados a fortalecer la robustez y funcionalidad del modelo computacional desarrollado en este estudio, así como para expandir sus capacidades a través de una mayor experimentación y exploración de nuevas áreas.

Para mejorar la eficiencia del algoritmo de Aprendizaje-Q, se pude investigar la reducción del número de estados con los que el agente interactúa para alcanzar la solución deseada.

Para mejorar la precisión en el vuelo del cuadricóptero durante el seguimiento de la trayectoria, se puede investigar una estrategia que busque disminuir gradualmente el retardo temporal a medida que las coordenadas de la trayectoria se vuelvan más cercanos, y aumentarlo cuando estas coordenadas estén más distantes. Este enfoque debe ser desarrollado con consideración del compromiso previamente identificado entre el número de coordenadas en la trayectoria (n), el retardo temporal (r) y la precisión en el vuelo (P).

En trabajos futuros, se recomienda integrar al modelo un proceso de filtrado Kalman (Miranda et al. 2022) con el propósito de refinar y suavizar abrupciones de la trayectoria generada con Aprendizaje-Q.

Para aumentar la robustez del modelo, se sugiere incorporar la dimensión de altura en la generación de trayectorias esto permitirá un vuelo más completo y realista del dron en entornos tridimensionales. Además, para evaluar la capacidad del cuadricóptero para reaccionar en entornos cambiantes y desafiantes, se sugiere incorporar obstáculos dinámicos que se muevan en tres dimensiones.

Para aumentar la robustez del modelo, se sugiere incorporar obstáculos dinámicos que se muevan en tres dimensiones. Esto permitirá evaluar la capacidad del dron para reaccionar en entornos cambiantes y desafiantes, lo que es fundamental para la navegación autónoma en situaciones dinámicas. Además, para enriquecer la experiencia de vuelo y adaptarse a una variedad de situaciones y escenarios, se recomienda incluir la dimensión de altura en la generación de trayectorias. Esto permitirá un vuelo más completo y realista del dron en entornos tridimensionales.

Probar el modelo en entornos reales representa un desafío adicional debido a factores como el viento, la luz solar y la presencia de obstáculos dinámicos. Para utilizar el modelo en entornos reales, se aconseja llevar a cabo mediciones precisas del entorno, incorporar ruido característico de ese ambiente y crear un modelo específico para el dron que se utilizará. Esto garantizará que las pruebas de simulación se asemejen de manera más precisa al entorno real.

También, es recomendable explorar más algoritmos de aprendizaje por refuerzo en el modelo. Esto puede proporcionar más variedad de resultados y posiblemente revelar enfoques más eficaces para la navegación autónoma.

Como enfoque para mejorar la precisión y confiabilidad del modelo, se recomienda explorar la utilización de algoritmos que fusionen datos de diferentes tipos de sensores para obtener la posición del cuadricóptero con mayor exactitud. Esta estrategia puede contribuir a la mejora del rendimiento. Además, se pude explorar la implementación de una cámara monocular y técnicas avanzadas de visión por computadora para ampliar aún más las capacidades del modelo.

Aunque trabajar con entornos discretos en Gazebo resultó importante, se recomienda como dirección futura llevar a cabo experimentos en Gazebo configurado como un entorno continuo. Esto permitirá una aproximación aún más cercana a entornos del mundo real.

En futuras investigaciones, se sugiere considerar la utilización de diferentes tipos de UAVs. Cada modelo de UAV posee características y comportamientos específicos que

pueden influir en el rendimiento del modelo de navegación autónoma. Explorar estos aspectos aumentará la robustez y versatilidad del modelo.

Referencias

Abaunza, H, P Castillo, and R Lozano. 2018. “Quaternion Modeling and Control Approaches.” *Handbook of Unmanned Aerial Vehicles* (September 2018). https://link.springer.com/book/10.1007/978-1-4419-1750-8.

Anwar, Aqeel, and Arijit Raychowdhury. 2020. “Autonomous Navigation via Deep Reinforcement Learning for Resource Constraint Edge Nodes Using Transfer Learning.” *IEEE Access* 8: 26549–60.

Benic, Zoran, Petar Piljek, and Denis Kotarski. 2016. “Mathematical Modelling of Unmanned Aerial Vehicles with Four Rotors.” *Interdisciplinary Description of Complex Systems* 14(1): 88–100.

Bestaoui Sebbane, Yasmina. 2018. *Intelligent Autonomy of UAVs Advanced Missions and Future Use*. Taylor & Francis Group.

De Boor, Carl. 1972. “On Calculating with B-Splines.” *Journal of Approximation Theory* 6(1): 50–65.

Cabreira, Tauã M., Lisane B. Brisolara, and R. Ferreira Paulo. 2019. 3 Drones *Survey on Coverage Path Planning with Unmanned Aerial Vehicles*.

Carino, J., H. Abaunza, and P. Castillo. 2015. “Quadrotor Quaternion Control.” *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015* (December): 825–31.

Castillo García, Pedro, Laura Elena Munoz Hernandez, and Pedro García Gil. 2017. *INDOOR NAVIGATION STRATEGIES FOR AERIAL AUTONOMOUS SYSTEMS*. Elsevier. https://www.elsevier.com/books/indoor-navigation-strategies-for-aerial-autonomous-systems/castillo-garcia/978-0-12-805189-4.

Cerón Contreras, Kevin Sebastian. 2022. “Aproximación de Un Sistema de Navegación Autónoma En UAVs Basado En Aprendizaje Por Refuerzo.” tesis de grado, Univerisdad de los Andes. http://hdl.handle.net/1992/55658.

Choi, Ji Wung, Renwick Curry, and Gabriel Elkaim. 2008. “Path Planning Based on Bézier Curve for Autonomous Ground Vehicles.” *Proceedings - Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008, WCECS 2008* (August): 158–66.

Curtis, Shiloh. 2020. “Map2gazebo.” *https://github.com/shilohc/map2gazebo*.

Dierckx, P. 1975. “An Algorithm for Smoothing, Differentiation and Integration of Experimental Data Using Spline Functions.” *Journal of Computational and Applied Mathematics* 1(3): 165–84.

Evans, M., N. Hastings, and B. Peacock. 2000. “Beta Distribution.” In *Statistical Distributions*, New York: Wiley, 34–42.

Fairchild, Carol, and Thomas L. Harman. 2016. *ROS Robotics By Example*. Livery Place 35 Livery Street Birmingham B3 2PB, UK.: Packt. https://www.packtpub.com/product/ros-robotics-by-example/9781782175193.

Furrer, Fadri, Michael Burri, Markus Achtelik, and Roland Siegwart. 2016. 625 Studies in Computational Intelligence *RotorS—A Modular Gazebo MAV Simulator Framework*.

Géron, Aurélien. 2019. *Hands-On Machine Learning with Scikit-Learn , Keras , and TensorFlow*. 2nd ed. ed. O'Reilly Media. O'Reilly Media.

Gupta, Abhishek, Alagan Anpalagan, Ling Guan, and Ahmed Shaharyar Khwaja. 2021. "Deep Learning for Object Detection and Scene Perception in Self-Driving Cars: Survey, Challenges, and Open Issues." *Array* 10(December 2020): 100057. https://doi.org/10.1016/j.array.2021.100057.

He, Zefang, and Long Zhao. 2014. "A Simple Attitude Control of Quadrotor Helicopter Based on Ziegler-Nichols Rules for Tuning Pd Parameters." *Scientific World Journal* 2014.

Hodge, Victoria J., Richard Hawkins, and Rob Alexander. 2021. "Deep Reinforcement Learning for Drone Navigation Using Sensor Data." *Neural Computing and Applications* 33(6): 2015–33. https://doi.org/10.1007/s00521-020-05097-x.

Hwangbo, J., Inkyu Sa, Roland Siegwart, and Marco Hutter. 2017. "Control of a Quadrotor with Reinforcement Learning." *IEEE Robotics and Automation Letters* 2(4): 2096–2103.

Jang, Sooyoung, and Changbeom Choi. 2022. "Prioritized Environment Configuration for Drone Control with Deep Reinforcement Learning." *Human-centric Computing and Information Sciences* 12(02): 16.

Joseph, Lentin, and Jonathan Cacace. 2021. 64 Physiological Research *Mastering ROS for Robotics Programming Third Edition Best Practices and Troubleshooting Solutions When Working with ROS*.

Krukhmalev, V., and Viacheslav Pshikhopov. 2017. Path Planning for Vehicles Operating in Uncertain 2D Environments *Genetic Algorithms Path Planning*.

Littman, Michael L., Leslie Pack Kaelbling, and Andrew W. Moore. 1996. "Reinforcement Learning: A Survey." *Journal of Arti cial Intelligence Research 4*: 237–85.

Mesa Santana, Edgar. 2019. "Sistema de Control de Un Dron Para Tareas En Interiores." tesis de licenciatura,Universidad de La Laguna. https://riull.ull.es/xmlui/handle/915/16579 (February 27, 2022).

Miranda, Victor R.F. et al. 2022. "Autonomous Navigation System for a Delivery Drone." *Journal of Control, Automation and Electrical Systems* 33(1): 141–55. https://doi.org/10.1007/s40313-021-00828-4.

Mordechai, Ben-Ari, and Mondada Francesco. 2018. Elements of Robotics *Elements of Robotics*. Springer.

Nex, F. et al. 2022. "UAV in the Advent of the Twenties: Where We Stand and What Is Next." *ISPRS Journal of Photogrammetry and Remote Sensing* 184(September 2021): 215–42. https://doi.org/10.1016/j.isprsjprs.2021.12.006.

Niu, Haoyu, Tiebiao Zhao, Dong Wang, and Yangquan Chen. 2019. "A UAV Resolution and Waveband Aware Path Planning for Onion Irrigation Treatments Inference." *2019 International Conference on Unmanned Aircraft Systems, ICUAS 2019*: 808–12.

Nüchter, Andreas. 2012. 52 Springer Tracts in Advanced Robotics *3D Robotic Mapping*.

Padhy, Ram Prasad et al. 2018. "Deep Neural Network for Autonomous UAV Navigation in Indoor

Corridor Environments." *Procedia Computer Science* 133: 643–50. https://doi.org/10.1016/j.procs.2018.07.099.

Paluszny, Marco, Hartmut Prautzsch, and Wolfgang Boehm. 2002. *Métodos de Bézier y B-Splines*.

Plaza Rey, Daniel. 2017. "Navegación Autónoma de Drones y Automatización de Rutas Aplicadas a La Limpieza de Edificios." tesis de Licenciatura, Universitat Politècnica de Catalunya. http://hdl.handle.net/2117/106479.

Ponteves, Hadelin. 2019. *AI Crash Course*. Packt.

Ramos Catari, Alain Rodrigo. 2017. "Sistemas de Navegación Autónomo Para Drones." tesis de grado, Universidad Mayor de San Andrés. http://repositorio.umsa.bo/xmlui/handle/123456789/16342.

ROS. 2022. "The Robot Operating System (ROS)." https://www.ros.org/.

Russo, Daniel J. et al. 2018. "A Tutorial on Thompson Sampling." *Foundations and Trends in Machine Learning* 11(1): 1–96.

Sammut, Claude, and Webb Geoffrey I. 2010. "Encyclopedia of Machine Learning." *Encyclopedia of Machine Learning*.

Schoenberg. 1967. *On Spline Functions*. ed. O. Sischa. New York: Academic Press.

Sciavicco, Lorenzo, and Bruno Siciliano. 2001. 17 Review Literature And Arts Of The Americas *Modelling and Control of Robot Manipulators*. Second. Springer Science & Business Media.

Sederberg, Thomas W. 2012. "Computer Aided Geometric Design." In *Computer Aided Geometric Design Course Notes*,.

Sewak, Mohit. 2019. Springer *Deep Reinforcement Learning Frontier of Artificial Intelligence*.

Song, Yunlong et al. 2020. "Flightmare: A Flexible Quadrotor Simulator." (CoRL). http://arxiv.org/abs/2009.00563.

Song, Yunlong, Mats Steinweg, Elia Kaufmann, and Davide Scaramuzza. 2021. "Autonomous Drone Racing with Deep Reinforcement Learning." *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021*: 1205–12.

Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. 2nd ed. The MIT Press.

Tello Vargas, Carlos Alfonso, and Eduardo Guillermo Herrera Victorio. 2019. "Diseño e Implementación de Un Drone de Ala Fija Para El Estudio de Índices de Vegetación Para La Agricultura de Precisión En El Fundo Altamirano - Ica." tesis de licenciatura, Universidad Ricardo Palma. http://repositorio.urp.edu.pe/handle/URP/2747.

Thompson, W. R. 1933. "On the Likelihood That One Unknown Probability Exceeds Another in View of the Evidence of Two Samples." *Biometrika* 25(3/4): 285–94.

———. 1935. "On the Theory of Apportionment." *American Journal of Mathematics* 57(2): 450–56.

Tiemann, Janis, Andrew Ramsey, and Christian Wietfeld. 2018. "Enhanced UAV Indoor Navigation

through SLAM-Augmented UWB Localization." *2018 IEEE International Conference on Communications Workshops, ICC Workshops 2018 - Proceedings*: 1–6.

Wang, Chao, Jian Wang, Yuan Shen, and Xudong Zhang. 2019. "Autonomous Navigation of UAVs in Large-Scale Complex Environments: A Deep Reinforcement Learning Approach." *IEEE Transactions on Vehicular Technology* 68(3): 2124–36.

Wang, Honglun et al. 2015. "Three-Dimensional Path Planning for Unmanned Aerial Vehicle Based on Interfered Fluid Dynamical System." *Chinese Journal of Aeronautics* 28(1): 229–39. http://dx.doi.org/10.1016/j.cja.2014.12.031.

Watkins, C.J.C.H. 1989. Robotics and Autonomous Systems "Learning from Delayed Rewards." King's College, Cabridge, UK.

Yousuf, Asad, William Lehman, Mohamad A. Mustafa, and Mir M. Hayder. 2015. "Introducing Kinematics with Robot Operating System (ROS)." *ASEE Annual Conference and Exposition, Conference Proceedings* 122nd ASEE(122nd ASEE Annual Conference and Exposition: Making Value for Society).

Zhang, Sitong, Yibing Li, and Qianhui Dong. 2022. "Autonomous Navigation of UAV in Multi-Obstacle Environments Based on a Deep Reinforcement Learning Approach." *Applied Soft Computing* 115: 108194. https://doi.org/10.1016/j.asoc.2021.108194.

Zhou, X., Yi, Z., Liu, Y., Huang, K., Huang, H. 2020. "Survey on Path and View Planning for UAVs." *Virtual Reality & Intelligent Hardware* 2: 56–69.

Apéndice

Entorno Supermercado

Análisis de la trayectoria más extensa sin y con diagonales utilizando el método convencional

Ventana	Iteraciones	Tiempo (min)	Número de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1X1	**48587363**	**80.315**	117	58.5	1.038	**4.795**
3x3	49240227	81.394	117	58.5	1.038	**4.795**
5x5	48760780	80.602	117	58.5	**1.032**	6.97
7x7	48781182	80.635	117	58.5	1.038	6.844
Con diagonales						
1X1	88432469	146.179	99	49.5	1.267	5.688
3x3	**85963827**	**142.098**	99	49.5	1.267	5.688
5x5	88238650	145.858	99	49.5	1.293	7.211
7x7	87697997	144.965	99	49.5	**1.291**	**7.126**

Análisis de la trayectoria más extensa sin y con diagonales utilizando el método MTA

Ventana	Iteraciones	Tiempo (min)	Numero de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1x1	16566424	44.232	117	58.5	1.038	**4.795**
3x3	**10782457**	**28.789**	117	58.5	1.038	**4.795**
5x5	14567028	38.894	117	58.5	**1.032**	6.97
7x7	34673199	92.577	117	58.5	1.038	6.844
Con diagonales						
1x1	21575115	57.606	99	49.5	**1.267**	**5.688**
3x3	**10660045**	**28.462**	99	49.5	**1.267**	**5.688**
5x5	80751116	215.605	99	49.5	1.293	7.211
7x7	55095601	147.105	99	49.5	1.291	7.126

Análisis de la trayectoria más extensa sin y con diagonales utilizando el método DG

Ventana	Iteraciones	Tiempo (min)	Numero de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1x1	**10303010**	**32.527**	117	58.5	1.038	**4.795**
3x3	10762055	33.976	117	58.5	1.038	**4.795**
5x5	11190497	35.328	117	58.5	**1.032**	6.97
7x7	11210899	35.393	117	58.5	1.038	6.844
Con diagonales						
1x1	11047683	34.878	99	49.5	1.267	5.688
3x3	**10864065**	**34.298**	99	49.5	1.267	5.688
5x5	11374115	35.908	99	49.5	1.293	7.211
7x7	12047381	38.034	99	49.5	**1.291**	**7.126**

Trayectorias finalizadas sin colisión

Método	Ventana	RMSE XY	%error XY	RMSE Yaw	%error Yaw	Retardo (ms)	Distancia (m)	Velocidad máxima (m/s)	Velocidad media (m/s)	Tiempo de vuelo (s)
Sin diagonales										
Convencional, MTA, DG	1x1	**3.810**	70.358	42.637	**5.916**	40	58.5	2.329	1.338	40.67
Convencional, MTA, DG	3x3	3.856	71.620	43.163	6.123	40	58.5	2.339	1.338	40.667
Convencional, MTA, DG	**5x5**	3.882	**70.147**	**41.654**	6.123	40	58.5	2.334	1.332	40.757
Convencional, MTA, DG	**7x7**	4.751	89.236	44.225	7.884	**30**	58.5	**2.583**	**1.779**	**30.66**

Trayectorias finalizadas con colisión

Con diagonales	
Método	**Ventana**

Convencional, MTA, DG	1x1
Convencional, MTA, DG	3x3
Convencional, MTA, DG	5x5
Convencional, MTA, DG	7x7

Entorno Oficina

Análisis de la trayectoria más extensa sin y con diagonales utilizando el método Convencional

Ventana	Iteraciones	Tiempo (min)	Número de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1X1	54840576	90.651	116	58	1.082	**15.629**
3x3	54738566	90.483	116	58	1.082	**15.629**
5x5	**54626355**	**90.297**	116	58	**1.075**	17.127
7x7	55156807	91.174	116	58	**1.075**	17.216
Con diagonales						
1X1	97582766	161.304	92	46	1.41	**11.662**
3x3	97093118	160.495	92	46	1.41	**11.662**
5x5	98123419	162.198	92	46	1.355	14.553
7x7	**96909500**	**160.191**	92	46	**1.331**	15.437

Análisis de la trayectoria más extensa sin y con diagonales utilizando el método MTA

Ventana	Iteraciones	Tiempo (min)	Número de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1X1	18065971	48.236	116	58	**1.082**	**15.629**
3x3	**11833160**	**31.595**	116	58	**1.082**	**15.629**
5x5	45037415	120.25	116	58	1.086	17.085
7x7	27522298	73.485	116	58	1.075	17.216
Con diagonales						
1X1	22156572	59.158	92	46	1.41	**11.662**
3x3	**11924969**	**31.84**	92	46	1.41	**11.662**

5x5	89993222	240.282	92	46	1.355	14.553
7x7	98898695	264.06	92	46	**1.331**	15.437

Análisis de la trayectoria más extensa sin y con diagonales utilizando el método DG

Ventana	Iteraciones	Tiempo (min)	Número de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1X1	11914768	37.615	116	58	1.082	15.629
3x3	**11578135**	**36.552**	116	58	1.082	15.629
5x5	11975974	37.808	116	58	1.075	17.127
7x7	11853562	37.422	116	58	1.075	17.216
Con diagonales						
1X1	12047381	38.034	92	46	1.41	**11.662**
3x3	12067783	38.098	92	46	1.41	**11.662**
5x5	12363612	39.032	92	46	1.355	14.553
7x7	**12037180**	**38.001**	92	46	**1.331**	15.437

Trayectorias finalizadas sin colisión

Método	Ventana	RMSE XY	%error XY	RMSE Yaw	%error Yaw	Retardo (ms)	Distancia (m)	Velocidad máxima (m/s)	Velocidad media (m/s)	Tiempo de vuelo (s)
Sin diagonales										
Convencional, MTA, DG	1x1	5.283	127.36	58.554	**5.926**	30	58	2.869	1.777	30.369
Convencional, MTA, DG	**3x3**	**5.271**	**127**	58.497	6.099	30	58	2.864	1.777	30.4
Convencional, DG	**5x5**	8.499	236.75	**55.903**	10.736	**15**	58	3.947	**3.055**	**15.217**
MTA	5x5	8.464	235.11	56.067	10.609	**15**	58	**4.052**	3.047	15.257
Convencional, MTA, DG	7x7	9.523	279.37	56.210	13.871	12	58	4.693	3.541	12.254
Con diagonales										

Convencional, MTA, DG	5x5	3.420	63.375	32.907	7.361	40	46	3.031	1.052	40.599

Trayectorias finalizadas con colisión

Con diagonales

Método	Ventana
Convencional, MTA, DG	1x1
Convencional, MTA, DG	3x3
Convencional, MTA, DG	7x7

Entorno Farmacia

Análisis de la trayectoria más extensa sin y con diagonales utilizando el método Convencional

Ventana	Iteraciones	Tiempo (min)	Número de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1X1	54840576	90.651	116	58	1.082	**15.629**
3x3	54738566	90.483	116	58	1.082	**15.629**
5x5	**54626355**	**90.297**	116	58	**1.075**	17.127
7x7	55156807	91.174	116	58	**1.075**	17.216
Con diagonales						
1X1	97582766	161.304	92	46	1.41	**11.662**
3x3	97093118	160.495	92	46	1.41	**11.662**
5x5	98123419	162.198	92	46	1.355	14.553
7x7	**96909500**	**160.191**	92	46	**1.331**	15.437

Análisis de la trayectoria más extensa sin y con diagonales utilizando el método MTA

Ventana	Iteraciones	Tiempo (min)	Número de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1X1	18065971	48.236	116	58	**1.082**	**15.629**

3x3	**11833160**	**31.595**	116	58	**1.082**	**15.629**
5x5	45037415	120.25	116	58	1.086	17.085
7x7	27522298	73.485	116	58	1.075	17.216
Con diagonales						
1X1	22156572	59.158	92	46	1.41	**11.662**
3x3	**11924969**	**31.84**	92	46	1.41	**11.662**
5x5	89993222	240.282	92	46	1.355	14.553
7x7	98898695	264.06	92	46	**1.331**	15.437

Análisis de la trayectoria más extensa sin y con diagonales utilizando el método DG

Ventana	Iteraciones	Tiempo (min)	Número de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1X1	11914768	37.615	116	58	1.082	15.629
3x3	**11578135**	**36.552**	116	58	1.082	15.629
5x5	11975974	37.808	116	58	1.075	17.127
7x7	11853562	37.422	116	58	1.075	17.216
Con diagonales						
1X1	12047381	38.034	92	46	1.41	**11.662**
3x3	12067783	38.098	92	46	1.41	**11.662**
5x5	12363612	39.032	92	46	1.355	14.553
7x7	**12037180**	**38.001**	92	46	**1.331**	15.437

Trayectorias finalizadas sin colisión

Método	Ventana	RMSE XY	%error XY	RMSE Yaw	%error Yaw	Retardo (ms)	Distancia (m)	Velocidad máxima (m/s)	Velocidad media (m/s)	Tiempo de vuelo (s)
Sin diagonales										
Convencional, MTA, DG	1x1	5.083	20.987	51.145	29.064	20	56	3.501	2.371	20.356
Convencional, MTA, DG	**3x3**	**5.051**	**20.572**	52.262	**24.222**	20	56	3.515	2.375	20.412

Convencional, MTA, DG	5x5	6.647	31.102	**40.310**	33.273	**12**	56	**4.890**	**3.550**	**12.348**
Convencional, MTA, DG	7x7	6.639	31.591	40.671	29.519	**12**	56	4.889	3.534	12.391
Con diagonales										
Convencional, MTA, DG	**5x5**	8.429	38.555	53.187	23.010	**8**	44	**6.611**	**5.304**	**8.469**
Convencional, MTA, DG	7x7	**6.295**	**27.913**	**53.159**	**18.520**	15	44	4.680	3.000	15.324

Trayectorias finalizadas con colisión

Con diagonales

Método	Ventana
Convencional, MTA, DG	1x1
Convencional, MTA, DG	3x3

Entorno Bodega

Análisis de la trayectoria más extensa sin y con diagonales utilizando el método Convencional

Ventana	Iteraciones	Tiempo (min)	Número de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1X1	**69479011**	**114.849**	114	57	1.119	3.938
3x3	69846247	115.456	114	57	1.119	3.938
5x5	70550116	116.619	114	57	1.121	3.368
7x7	70244086	116.113	114	57	**1.112**	**3.124**
Con diagonales						
1X1	105549747	174.474	87	43.5	1.253	**1.396**
3x3	105529345	174.44	87	43.5	1.253	**1.396**
5x5	**105223315**	**173.934**	87	43.5	**1.242**	1.855
7x7	108140801	178.757	87	43.5	1.267	1.739

Análisis de la trayectoria más extensa sin y con diagonales utilizando el método MTA

Ventana	Iteraciones	Tiempo (min)	Número de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1X1	27705916	73.975	114	57	1.119	3.938
3x3	**19055468**	**50.878**	114	57	1.119	3.938
5x5	29378880	78.442	114	57	1.121	3.368
7x7	31531291	84.189	114	57	**1.112**	**3.124**
Con diagonales						
1X1	31612899	84.406	87	43.5	1.253	1.396
3x3	**19014664**	**50.769**	87	43.5	1.253	**1.396**
5x5	158798967	423.993	87	43.5	**1.242**	1.855
7x7	123758532	330.435	87	43.5	1.267	1.739

Análisis de la trayectoria más extensa sin y con diagonales utilizando el método DG

Ventana	Iteraciones	Tiempo (min)	Número de estados	Distancia (m)	RMSE	%error
Sin diagonales						
1X1	**19555317**	**61.736**	114	57	1.119	3.938
3x3	**19555317**	**61.736**	114	57	1.119	3.938
5x5	19983759	63.089	114	57	1.121	3.368
7x7	20626422	65.118	114	57	**1.112**	**3.124**
Con diagonales						
1X1	20432603	64.506	87	43.5	1.253	1.402
3x3	**20024563**	**63.218**	87	43.5	1.253	**1.396**
5x5	20616221	65.085	87	43.5	**1.242**	1.855
7x7	20606020	65.053	87	43.5	1.267	1.739

Trayectorias finalizadas sin colisión

Método	Ventana	RMSE XY	%error XY	RMSE Yaw	%error Yaw	Retardo (ms)	Distancia (m)	Velocidad máxima (m/s)	Velocidad media (m/s)	Tiempo de vuelo (s)
Sin diagonales										
Convencional, MTA, DG	**1x1**	3.958	27.449	49.781	**10.201**	30	57	2.831	1.746	30.29
Convencional, MTA, DG	3x3	**3.934**	**26.927**	**49.712**	10.314	30	57	2.836	1.746	30.37
Convencional, MTA, DG	5x5	7.939	60.207	77.827	22.727	10	57	5.655	4.888	10.1
Convencional, MTA, DG	**7x7**	8.329	60.600	78.032	25.385	**9**	57	**6.780**	**5.332**	**9.195**
Con diagonales										
Convencional, MTA, DG	5x5	3.810	29.630	50.609	8.181	40	43.5	**2.026**	**1.005**	**40.433**
Convencional, MTA, DG	7x7	**3.229**	**24.666**	**49.246**	**6.174**	50	43.5	1.892	0.810	50.345

Trayectorias finalizadas con colisión

Con diagonales

Método	Ventana
Convencional, MTA	1x1
DG	1x1
Convencional, MTA, DG	3x3

Curriculum Vitae

RAYDESEL ARIEL SÁNCHEZ MONTES

Campo Bello Etapa I, Juventud Norte, Chihuahua | (614) 526-97-23 |
raydeselariels@gmail.com | GitHub @Raydesel

Skilled in **Machine Learning** with experience in **academic research**. Proficient in programming with **Python** and **C++** and experienced in both **Linux** and **Windows** systems.
Positive attitude, strong work ethic, a drive for results while focusing on high code quality and performance.

Research project | Autonomous University of Chihuahua | 2021 - 2023
Thesis: Autonomous Navigation of an Unmanned Aerial Vehicle Using Reinforcement Learning
Autonomous indoor drone navigation system developed using Gazebo-Ros simulator and Python/C++ programming languages. Valuable experience in robotics and software development for industrial applications.

Research project | Autonomous University of Chihuahua | 2019 - 2020
Thesis: Emotion recognition in speech using MFCC and multi-band spectral entropy signatures.
Development of a speech emotion recognition system using Python. Processing and normalization of databases with speech signals for emotional classification.

Education

Master of Computer Engineering | Autonomous University of Chihuahua | Chihuahua | 2021 - 2023
Self-taught with 5+ Certificates of Completion | Udemy & Other Plataforms | Online | 2021 - Present
Computer Systems Engineer in Hardware | Autonomous University of Chihuahua | Chihuahua | 2016 - 2020