Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Seguimiento de líneas con técnicas de aprendizaje por refuerzo en robótica móvil

Autor: Eduardo López Delmás
Tutor: Javier de Lope Asiain

Madrid, Junio - 2025

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática

Título: Seguimiento de líneas con técnicas de aprendizaje por refuerzo en robótica móvil

Junio - 2025

Autor: Eduardo López Delmás
Tutor: Javier de Lope Asiain
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

Este Trabajo de Fin de Grado aborda el diseño e implementación de un sistema de control autónomo para un robot móvil usando técnicas de aprendizaje por refuerzo. El objetivo principal es lograr que el robot sea capaz de seguir una línea dibujada en el suelo empleando únicamente la percepción visual. Para lograr este objetivo, se implementa en Python un controlador que combina el algoritmo Q-learning con técnicas de visión por computadora utilizando la librería OpenCV.

La interacción física del agente con el entorno se simula a través de CoppeliaSim, que replica de forma precisa el comportamiento de un robot Pioneer P3DX, equipado con una cámara para capturar su campo de visión frontal. Las imágenes son procesadas para determinar la posición del robot con respecto a la línea, la cual se representa por medio de un número predeterminado de estados discretos. El entrenamiento consiste en un proceso de prueba y error, en el que el agente aprende a ejecutar las acciones óptimas para maximizar la recompensa acumulada, que se define en base a la precisión del alineamiento entre el robot y la línea.

Durante el desarrollo del proyecto se evaluan diversas configuraciones de *Q-table* para determinar su impacto en la precisión, la estabilidad y la velocidad de convergencia. Asimismo, se analizan en profundidad los efectos de los hiperparámetros: tasa de aprendizaje, factor de descuento y tasa de exploración. Los resultados demuestran que el agente es capaz de aprender políticas de control eficientes y reproducibles con una correcta configuración y un buen diseño del espacio de estados. No obstante, se identifican ciertas limitaciones en relación con la capacidad de generalización y la sensibilidad del sistema de percepción a la discretización fina del espacio.

El trabajo desarrollado supone una contribución académica y técnica al campo de estudio de la inteligencia artificial aplicada a robots móviles, sentando una sólida base para futuros avances, incluyendo la implementación en un robot real y la integración de redes neuronales profundas en la arquitectura del sistema.

Palabras Clave:

Aprendizaje por refuerzo, Q-learning, Robótica móvil, Visión por computadora

Abstract

This Final Degree Project presents the design and implementation of an autonomous control system for a mobile robot using reinforcement learning techniques. The primary objective is to enable the robot to follow a line drawn on the ground using only visual perception. To achieve this, the Q-learning algorithm is employed in combination with computer vision techniques implemented through the OpenCV library and the Python programming language.

The simulation environment used is CoppeliaSim, which accurately replicates the physical behavior of the Pioneer P3DX robot, equipped with a front-facing camera for image acquisition. These images are processed to extract the robot's position relative to the line, and the information is discretized into states that serve as input to the agent's learning policy. The system is trained via a trial and error process, where the agent learns to make optimal decisions by maximizing cumulative rewards, defined based on its alignment with the line.

Throughout the project, various Q-table configurations are evaluated to assess their impact on control precision, behavioral stability, and convergence speed. Additionally, the effects of key hyperparameters, such as learning rate, discount factor, and exploration strategy, are analyzed in depth. The results show that, with a correct configuration and well-designed state space, the agent can learn efficient and reproducible control policies. However, several limitations are also identified, particularly regarding generalization and sensitivity to small perception disturbances.

This work represents both an academic and technical contribution to the study of artificial intelligence applied to mobile robotics, laying a solid foundation for future research, including a real world hardware implementation and the integration of deep neural networks for more complex visual environments.

Keywords:

Reinforcement Learning, Q-learning, Mobile robotics, Computer vision

Índice general

Índice de Figuras

Índice de Tablas

Índice de Listings

Capítulo 1

Introducción

En este capítulo se proporciona el contexto y los objetivos del presente proyecto, así como la estructura que seguirá el documento. En la sección 1.1 se explica la motivación detrás de la realización del trabajo, subrayando la relevancia de la robótica móvil y el aprendizaje por refuerzo en el contexto tecnológico actual. En el apartado 1.2 se describen los objetivos concretos del trabajo, exponiendo brevemente el enfoque propuesto para la resolución del problema, además de las herramientas que se emplearán. Finalmente, en la sección 1.3 se presenta la estructura del documento, aportando una breve descripción de los contenidos de cada uno de los apartados.

1.1. Motivación del proyecto

En los últimos años, la robótica se ha convertido en uno de los campos más atractivos y prometedores entre las áreas de aplicación de la inteligencia artificial. Esta tendencia se ve reflejada por la creciente inversión por parte de las principales empresas tecnológicas, entre ellas Tesla, Meta y Apple, en el desarrollo de robots autónomos, particularmente de robots humanoides [1]. La capacidad de percibir, tomar decisiones y actuar de forma autónoma en entornos dinámicos sitúa a los robots móviles ante un amplio espectro de aplicaciones en diversos sectores, como logística, salud, monitorización medioambiental y transporte autónomo [2].

Las técnicas tradicionales de control y navegación de robots a menudo dependen de un modelado muy preciso del entorno, así como de la especificación de unos patrones de comportamiento predefinidos. Esto se presenta como un inconveniente importante cuando el robot ha de enfrentarse a entornos dinámicos y repletos de incertidumbre, limitando su habilidad para adaptarse y mejorar a lo largo del tiempo. Es por ello que la unión con el aprendizaje automático ha desempeñado un papel fundamental en el rápido avance de la robótica en las últimas décadas. Una de las técnicas más empleadas para el entrenamiento de sistemas robóticos es el aprendizaje por refuerzo, el cual plantea un enfoque potente para mejorar la toma de decisiones, en el que el agente aprende a comportarse de forma óptima mediante un proceso de prueba y error, caracterizado por una interacción directa con su entorno [3].

Entre los diversos algoritmos de aprendizaje por refuerzo, el Q-learning destaca por su simplicidad y efectividad demostrada en espacios de acción discretos. Este algoritmo permite al agente aprender una política óptima sin la necesidad de contar con

ningún conocimiento previo de su entorno, convirtiéndolo en una gran opción para afrontar un amplio habanico de problemas del mundo real. Uno de los más relevantes a día de hoy, estrechamente relacionado con los robots móviles y el transportes autónomo, es la navegación, donde se deben tomar decisiones continuamente en entornos complejos y variables en base a información recogida principalmente por cámaras y sensores.

Este proyecto se centra específicamente en la aplicación del Q-learning al problema de seguimiento de líneas en robótica móvil, un desafío que requiere de la combinación en tiempo real del procesamiento de imágenes mediante técnicas de visión por computadora con la toma de decisiones automatizada. Se implementará un agente funcional y se evaluará su proceso de aprendizaje, robustez y posibles limitaciones, contribuyendo de esta forma a la creciente exploración en el campo del aprendizaje por refuerzo en sistemas robóticos y aportando una base para futuros trabajos en los que se exploren entornos de mayor complejidad.

1.2. Objetivos

El trabajo trata del diseño e implementación de un controlador para un robot móvil, cuya función es seguir una línea dibujada sobre el suelo. Para lograr este objetivo se emplearán técnicas de aprendizaje por refuerzo, en particular el algoritmo Q-learning, en combinación con técnicas de visión por computadora implementadas usando la librería OpenCV con el lenguaje Python.

Los datos de entrada del controlador se recogerán en forma de imágenes captadas por una cámara frontal a bordo del robot para detectar la línea. El desarrollo y las pruebas del sistema se realizarán en el entorno de simulación de Coppelia Robotics.

Los principales objetivos de este Trabajo de Fin de Grado son:

- Desarrollo de un controlador para robots móviles que implemente un modelo de aprendizaje por refuerzo
- Estudio y selección de técnicas de aprendizaje por refuerzo para la resolución del problema planteado
- Captación y procesamiento de imágenes mediante OpenCV en Python
- Evaluación del rendimiento del agente en base a métricas de navegación
- Establecer una base para futuras extensiones en entornos más complejos o implementaciones reales
- Documentación del trabajo realizado

1.3. Estructura del documento

El presente documento está divido en siete capítulos:

1. **Capítulo 1:** Introducción al proyecto, incluyendo motivación, objetivos y estructura del documento.
2. **Capítulo 2:** Presentación de trabajos relacionados y estado del arte de las tecnologías y metodologías empleadas en este proyecto.
3. **Capítulo 3:** Explicación detallada de los fundamentos teóricos necesarios para el desarrollo del trabajo.
4. **Capítulo 4:** Descripción del diseño e implementación del agente de aprendizaje por refuerzo.
5. **Capítulo 5:** Exposición y análisis de los resultados obtenidos.
6. **Capítulo 6:** Presentación de conclusiones finales y potenciales futuros trabajos.
7. **Capítulo 7:** Reseña sobre el impacto del trabajo realizado.

Capítulo 2

Trabajo relacionado y Estado del Arte

En este capítulo se estudia, en primer lugar, el rol del aprendizaje por refuerzo en el campo de la robótica, describiendo sus principales desafíos, avances más recientes y aplicaciones industriales. En segundo lugar, la exploración se centra en el Q-learning, detallando sus principios fundamentales, limitaciones y como ha evolucionado mediante la integración con diversas tecnologías.

2.1. Aprendizaje por refuerzo en robótica

El aprendizaje por refuerzo se ha establecido como una metodología clave en el mundo de la robótica, gracias a su facilidad para lidiar con entornos dinámicos, cuya formalización resulta compleja. Un estudio de revisión realizado por Kober et al. (2013) [2] analiza la integración del aprendizaje por refuerzo en sistemas robóticos, identificando desafíos clave como la necesidad del rendimiento en tiempo real y las consideraciones de seguridad durante el aprendizaje. Los autores enfatizan la importancia de la conjunción entre el aprendizaje y el conocimiento previo para acelerar la convergencia y mejorar la efectividad. Su trabajo estableció fundamentos sólidos para avanzar desde simulaciones controladas hacia aplicaciones en entornos reales.

Uno de los mayores avances que ha experimentado el campo del aprendizaje por refuerzo en robótica se atribuye a la introducción del Deep Reinforcement Learning, que combina técnicas tradicionales de aprendizaje por refuerzo con redes neuronales profundas para afrontar problemas complejos. En un estudio reciente se analizan algunos de los factores que han contribuído al despliegue exitoso de sistemas que utilizan este enfoque, como el uso de arquitecturas modulares, aleatorización del dominio y flujos de control híbrido [4]. No obstante, se remarca que, aunque este enfoque muestre un gran potencial, la generalización y la robustez plantean un desafío que sigue siendo prevalente, especialmente al transferir políticas establecidas en simulaciones a robots reales, además de al presentar al agente con entornos desconocidos sin un proceso de reentrenamiento previo.

De forma similar, en un estudio llevado a cabo por Han et al. (2023) se comparan en profundidad diferentes familias de algoritmos de aprendizaje por refuerzo, entre ellas los métodos basados en valores, que estiman el valor esperado de cada acción

para seleccionar la mejor opción, los métodos basados en políticas, que ajustan directamente la probabilidad de tomar cada acción sin estimar valores intermedios y los métodos actor-crítico, en los que el actor aprende la política y el crítico evalúa la acción mediante una función de valor [5]. La conclusión apunta a que, aunque los métodos de Deep Reinforcement Learning hayan superado a los métodos tradicionales en flexibilidad y diversidad de tareas, su eficiencia sigue siendo un aspecto cuestionable, ya que precisan de millones de interacciones en entrenamientos simulados antes de ser viables para introducirse en robots físicos.

Asimismo, la combinación de aprendizaje por refuerzo con modelos fundacionales, redes neuronales de gran escala preentrenadas con grandes volúmenes de datos y reutilizables en múltiples tareas, ha abierto nuevas posibilidades. Actualmente se está explorando como aprovechar este planteamiento para generalización *zero-shot*, la cual consiste en resolver problemas que el modelo no ha tratado durante el entrenamiento, procesamiento de lenguaje natural y toma de decisiones en entornos complejos [6]. Se sugiere que la unión entre el proceso de aprendizaje estructurado del aprendizaje por refuerzo, junto a la capacidad de generalización de los modelos fundacionales, podrían conducir a robots dotados de mayor autonomía y adaptabilidad, capaces de aprender en entornos reales con datos limitados y supervisión escasa.

Un ejemplo notorio del mundo real se encuentra en la empresa especializada en robótica Boston Dynamics, que ha comenzado recientemente a integrar algoritmos de aprendizaje por refuerzo en los sistemas de locomoción de sus robots. Por ejemplo, el aprendizaje por refuerzo se ha empleado para potenciar la adaptabilidad de algunos de sus robots como Spot, permitiéndole recuperarse de incidentes y mantener la estabilidad al recorrer terrenos desiguales e impredecibles [7]. Aunque los algoritmos específicos empleados no son accesibles al público, estas aplicaciones subrayan la creciente relevancia a nivel industrial del aprendizaje por refuerzo como una herramienta para mejorar la autonomía, resiliencia e inteligencia de robots, más allá del umbral que alcanzan los sistemas de control tradicionales.

En conclusión, estos estudios y trabajos muestran como el aprendizaje por refuerzo y el aprendizaje por refuerzo profundo, a pesar de estar ya transformando el campo de la robótica, permanecen lejos de poder considerarse problemas resueltos. Su aplicación masiva en entornos reales aún requiere sobrepasar ciertos desafíos, entre ellos la ineficiencia, la transferencia de simulación a realidad y la seguridad, convertiéndolo en un interesante campo de investigación.

2.2. Q-learning en robótica

El Q-learning es un algoritmo de aprendizaje por refuerzo introducido por Chistopher J. C. Watkins en 1989 como parte de su tesis en la Universidad de Cambdridge [8]. El algoritmo se basa en un proceso interactivo de prueba y error del agente con su entorno, el cual conduce al aprendizaje de una política óptima sin la necesidad de contar con un modelo que describa dicho entorno. Su principal fortaleza radica en su capacidad para estimar el valor de los pares de estado-acción, también llamados *Q-values*, y ajustar mediante un proceso iterativo la política óptima basándose en estas estimaciones. Desde su introducción, el Q-learning se ha convertido en uno de los algoritmos de aprendizaje por refuerzo más extendidos, especialmente en sistemas

robóticos que trabajan con conjuntos de estados y acciones discretos.

La principal limitación de este algoritmo reside en la dificultad para escalar a problemas de alta dimensionalidad y estados continuos. El enfoque tabular empleado en el Q-learning clásico se presenta como un incoveniente cuando crece exponencialmente el número de estados y acciones, careciendo, asimismo, de la capacidad de generalización requerida para trabajar directamente con datos no procesados, como por ejemplo imágenes no preprocesadas. Por ello, el Q-learning a menudo se combina con otras metodologías para lograr soluciones más escalables para problemas complejos.

Un ejemplo de este enfoque híbrido se presenta en un trabajo de Gajewski et al. (2024) [9], donde se introduce una adaptación de la arquitectura Decision Transformer para entornos de aprendizaje por refuerzo multiobjetivo. Su trabajo demuestra como los principios del Q-learning pueden integrarse con modelos basados en transformadores para tratar tareas complejas, realzando la versatilidad del Q-learning en aplicaciones robóticas modernas.

Un avance muy significativo fue la introducción de las Deep Q-Networks o DQN, las cuales combinan Q-learning con redes neuronales profundas para manejar espacios de estados de alta dimensionalidad [10], como es el caso del tratamiento de imágenes sin preprocesar. Inicialmente, este enfoque se aplicó al área de los videojuegos, no obstante, las DQN se han podido transferir desde entonces de forma exitosa a sistemas robóticos, permitiendo a los agentes aprender políticas de control procesando datos visuales de forma directa. Un ejemplo de este planteamiento es la utilización de arquitecturas basadas en DQN en robots móviles para navegar y seguir objetivos utilizando únicamente la información capturada por medio de cámaras, reduciendo significativamente la necesidad de ingeniería de características desarrollada de forma manual [11].

Como hemos podido observar, los avances y las posibles aplicaciones que surgen al combinar el Q-learning con otras técnicas de aprendizaje autónomo resultan prometedoras. Sin embargo, en este trabajo se explorará el Q-learning clásico, para así comprender de manera extensa sus fundamentos y limitaciones, sentando así las bases para poder explorar arquitecturas más complejas en un futuro.

Capítulo 3

Fundamentos Teóricos

En este capítulo se presentan los fundamentos teóricos necesarios para comprender la implementación del agente de seguimiento de líneas desarrollado en este trabajo. Comienza con una descripción en profundidad del aprendizaje por refuerzo, un paradigma que permite al agente aprender mediante la interacción directa con su entorno. A continuación, se realiza una explicación detallada del Q-learning, el algoritmo empleado en este proyecto, exponiendo sus principios operacionales, puntos fuertes y limitaciones. Finalmente, se cubren los principios más básicos de la visión por computadora, esencial para poder capturar las características del entorno.

3.1. Aprendizaje por refuerzo

3.1.1. Introducción al aprendizaje por refuerzo

El aprendizaje por refuerzo es un subconjunto del aprendizaje automático orientado a lograr que un agente aprenda a tomar decisiones óptimas en un entorno determinado a través de un proceso en el que realiza acciones y recibe retroalimentación en forma de recompensas [3]. A través de la interacción con su entorno, el agente debe descubrir cuáles son las acciones que aportan una mayor recompensa, explorando diferentes estrategias y aprendiendo de las consecuencias de sus acciones.

A diferencia del aprendizaje supervisado, donde los modelos se entrenan con conjuntos de datos que contienen la salida correcta para cada entrada, los agentes de aprendizaje por refuerzo deben enfrentarse a problemas de toma de decisiones secuencial, en los que la acción efectuada no solo afecta a la recompensa inmediata, sino también a los estados subsiguientes e, indirectamente, a las futuras recompensas [12]. Asimismo, en contraste con el aprendizaje no supervisado, caracterizado por la búsqueda de patrones ocultos en datos no etiquetados, el aprendizaje por refuerzo se centra principalmente en maximizar la recompensa cumulativa a lo largo del tiempo. Este paradigma de aprendizaje mediante prueba y error encuentra su inspiración en la psicología conductual, particularmente en la teoría del condicionamiento operante de Skinner, que sostiene que el comportamiento se moldea a través de recompensas y castigos [13].

En el contexto de la inteligencia artificial moderna, el aprendizaje por refuerzo se ha convertido en un enfoque fundamental para tareas en las que la supervisión directa

resulta inviable o poco práctica, y cuya resolución éxitosa precisa de aprender de la experiencia. Sus aplicaciones abarcan campos como la robótica, en la que se centra este trabajo, los videojuegos y la conducción autónoma.

3.1.2. Elementos del aprendizaje por refuerzo

Un sistema estándar de aprendizaje por refuerzo está formado por varios componentes interrelacionados. Por un lado, el agente es el encargado de aprender y tomar decisiones y se programa con el objetivo de descubrir el comportamiento idóneo. Por otro lado, el entorno abarca todo aquello que no forma parte del agente y que responde a sus acciones. La interacción entre el agente y el entorno da lugar a un circuito cerrado de percepción y acción.

El estado representa la situación actual o la configuración del entorno según es percibida por el agente, encapsulando toda la información necesaria para la toma de decisiones. Las acciones son las elecciones disponibles para el agente y tienen la capacidad de alterar el estado actual del entorno. En ocasiones, el espacio de acciones puede ser discreto y estar definido de forma clara, como en un juego de mesa, donde comprende el conjunto de movimientos legales disponibles en un momento determinado [14]. En otros casos, como en la conducción autónoma, el espacio de acciones está compuesto por una cantidad de posibilidades difícilmente cuantificable, dada la enorme complejidad de su entorno. Tras cada una de las acciones, el agente recibe una retroalimentación que debe traducirse en una señal de recompensa: una cuantificación escalar de comentarios positivos (o negativos) [14].

Dos conceptos cruciales adicionales son la política y la función de valor. La política equivale a una relación entre los estados y la probabilidad de seleccionar cada una de las acciones disponibles, determinando de esta forma el comportamiento del agente. La política puede variar desde una simple tabla de consulta hasta complejos procesos de búsqueda que involucren una alta carga computacional [3]. La función de valor, por otra parte, estima el retorno total que se puede obtener partiendo de un estado específico y aplicando una política determinada. En otras palabras, la función de valor indica la recompensa a largo plazo de seleccionar una acción. En consecuencia, aunque tomar una decisión concreta aporte una recompensa baja, puede aportar un alto valor si conduce a estados futuros que puedan otorgar una recompensa elevada [3].

Formalmente, el retorno G_t en un punto en el tiempo t está definido por la recompensa total descontada de ese punto en adelante:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \ ,$$

donde γ es el factor de descuento [0, 1], que determina la importancia de las recompensas futuras en relación a las inmediatas [15].

Todos estos elementos están interconectados en un bucle de retroalimentación, en el que el agente observa el estado, realiza una acción de acuerdo a su política, recibe una recompensa, transiciona al nuevo estado y actualiza su política en función de la experiencia acumulada.

3.1.3. Proceso de aprendizaje

El proceso del aprendizaje por refuerzo se define formalmente bajo el marco de un proceso de decisión de Markov, que queda determinado por una tupla $< S, A, P, R, \gamma >$ [16], donde:

- **S** es un conjunto finito de estados.
- **A** es un conjunto finito de acciones.
- **P** es la matriz de transición $P(s'|s, a)$, la cual recoge la probabilidad de transicionar al estado s' tras realizar la acción a desde el estado s.
- **R** es la función de recompensa.
- γ es el factor de descuento.

El mecanismo de aprendizaje está compuesto por dos piezas fundamentales: exploración y explotación. Por un lado, la exploración consiste en realizar acciones nuevas o que se han ejecutado con poca frecuencia para descubrir su efecto y potencial recompensa. Por otro lado, la explotación radica en seleccionar aquellas acciones que otorgan una mayor recompensa, según la experiencia previa del agente. Lograr un buen equilibrio entre estas dos estrategias es un elemento crítico en el contexto del aprendizaje por refuerzo. Un uso excesivo de la explotación puede resultar en una convergencia prematura a políticas subóptimas, mientras que una exploración desproporcionada puede conducir a un malgasto de tiempo y recursos [3].

La estrategia *epsilon-greedy* es uno de los enfoques más extendidos para equilibrar el uso entre la explotación y la exploración. Una vez definido el valor de la variable $\epsilon \in [0, 1]$, el agente selecciona una acción aleatoria con una probabilidad de ϵ y la que se estima que conduce a una mayor recompensa con una probabilidad de $1 - \epsilon$. Habitualmente, el valor de ϵ se reduce a lo largo del entrenamiento, favoreciendo de esta manera la explotación de las mejores acciones a medida que el agente se familiariza con su entorno y desarrolla una política óptima [12].

3.1.4. Categorías de algoritmos de aprendizaje por refuerzo

3.1.4.1. Política

Los algoritmos de aprendizaje por refuerzo se pueden clasificar en función de la estrategia empleada para optimizar la política. Siguiendo este enfoque, existen principalmente tres categorías: algoritmos basados en el valor, algoritmos basados en la política y algoritmos actor-crítico.

Los algoritmos basados en el valor se centran en aprender una función de valor, la cual, como hemos visto anteriormente, estima el retorno total de realizar una acción determinada en un estado concreto. Uno de los algoritmos más extendidos de esta categoría es el Q-learning, en el que se basa el agente desarrollado en este trabajo, el cual aproxima de forma directa una función de acción-valor óptima [17]. Estos métodos optimizan la política de manera implícita, al seleccionar la acción con el mayor retorno estimado.

Alternativamente, los algoritmos basados en la política aprenden una política parametrizada sin la necesidad de hacer uso de una función de valor. Estos algoritmos se

centran en optimizar la política en función del gradiente de retorno estimado relativo a los parámetros de la política. Algunos de los ejemplos más destacables son *REINFORCE* [18] y métodos del gradiente de políticas. Este enfoque es particularmente efectivo en espacios de acciones continuos, ya que definir funciones de valor discretas resulta poco práctico.

Por último, los algoritmos actor-crítico combinan las ventajas de ambos enfoques por medio de una estructura diferenciada en dos bloques: un actor que propone la acción a ejecutar y un crítico encargado de evaluar la elección. Para ello, el crítico estima la función de valor, mientras que el actor actualiza la política en base a la evaluación expedida por el crítico [19]. Esta arquitectura tiene como resultado un proceso de aprendizaje más estable y eficiente, especialmente en entornos complejos y de alta dimensionalidad.

Cada una de estas categorías cuenta con ventajas y desventajas en relación a su eficiencia muestral, que mide la cantidad de datos que necesita un modelo para converger [20], su coste computacional, estabilidad y escalabilidad, siendo cada una de ellas consideraciones críticas para aplicaciones reales exitosas.

3.1.4.2. Modelo

Además de esta clasificación funcional, los algoritmos de aprendizaje por refuerzo se pueden distinguir en función del uso o ausencia de un modelo del entorno. Esto conduce a dos amplios paradigmas: aprendizaje basado en modelos y aprendizaje libre de modelos, conocidos comúnmente por sus términos en inglés *model-based* y *model-free*.

Los algoritmos *model-free* no utilizan ningún modelo que represente la dinámica del entorno, sino que optimizan la política y/o la función de valor basándose únicamente en las transiciones y sus respectivas recompensas. Por este motivo, la implementación de agentes *model-free* es habitualmente más sencilla y robusta, especialmente en contextos en los que el modelado del entorno resulta complejo o impreciso. La mayor parte de algoritmos estándar, como Q-learning, SARSA y REINFORCE, además de numerosos métodos de aprendizaje profundo, cuentan con una arquitectura libre de modelo [3].

Por otra parte, los algoritmos *model-based* aprenden o asumen un modelo explícito del entorno, representado típicamente por una función de transición $P(s'|s,a)$ y una función de recompensa $R(s,a)$. Gracias a este modelo, el agente puede simular secuencias de acciones hipotéticas para estimar su retorno y optimizar su política sin necesidad de una interacción real con el entorno [21]. Esto puede resultar en una mejora considerable de la eficiencia muestral, puesto que el agente puede aprender tanto de experiencia real como de experiencia simulada.

Estas dos formas de clasificar los algoritmos de aprendizaje por refuerzo no son mutuamente excluyentes, sino complementarias en ejes independientes. Esto se traduce en que cualquiera de los métodos de la primera clasificación puede ser definido como *model-free* o *model-based*. Por ejemplo, el Q-learning opera normalmente sin modelo, sin embargo, puede ser adaptado a variantes en las que se emplea un modelo de transición para simular futuros resultados [22].

Comprender ambas dimensiones de esta clasificación es indispensable para diseñar buenos algoritmos de aprendizaje por refuerzo, especialmente en su aplicación en la robótica, ya que la compensación entre eficiencia muestral, coste computacional y estabilidad de la política resulta fundamental.

3.2. Q-learning

3.2.1. Introducción al Q-learning

El Q-learning es uno de los algoritmos más utilizados dentro del encuadre del aprendizaje por refuerzo y sirve como base para muchos enfoques de mayor complejidad. Fue introducido originariamente por Christopher J.C.H. Watkins en su tesis doctoral de 1989 [8], con el objetivo de desarrollar un agente *model-free* capaz de aprender la política de actuación óptima en procesos de decisión de Markov.

La principal fortaleza del Q-learrning radica en su gran capacidad para aprender directamente de la experiencia, sin la necesidad de contar con un modelo del entorno. Esto convierte al Q-learning en una opción especialmente llamativa para aplicaciones en las que el modelo se desconoce o es difícil de construir, como es el caso en los entornos dinámicos y complicados del mundo real a los que se enfrenta la robótica. Por medio de la definición y actualización progresiva de una función de acción-valor, denominada *Q-function*, el algoritmo aprende a estimar la recompensa total que el agente puede obtener al seleccionar una acción en un estado específico.

Formalmente, la *Q-function*, definida por $Q(s, a)$, recibe un par de estado-acción y retorna un número que representa la recompensa acumulada estimada al partir del estado s, realizar la acción a y continuar seleccionando las acciones óptimas. A medida que el agente explora su entorno y actualiza los llamados *Q-values*, esta función converge hacia la función de acción-valor óptima $Q^*(s, a)$, de la cual se deriva la política óptima $\pi^*(s) = argmax_a Q^*(s, a)$ [3].

Su formulación simple y elegante ha permitido al Q-learning establecerse como uno de los principales algoritmos de aprendizaje por refuerzo para cursos introductorios e investigación, además de sentar una sólida base para métodos más avanzados, como las Deep Q-Networks [10] y Double Q-learning [23].

3.2.2. Funcionamiento del algoritmo

El Q-learning es un algorimto iterativo, que depende de la interacción continua del agente con su entorno. En cada paso en el tiempo t, el agente observa el estado actual s, selecciona una acción a en base a una política (comúnmente una política ϵ-*greedy*), ejecuta la acción, recibe una recompensa numérica r_{t+1} y transiciona al nuevo estado s_{t+1}.

El componente central del proceso de aprendizaje descrito es la regla *Q-update*, que ajusta el *Q-value* del par de estado-acción anterior en función de la recompensa recibida, así como del valor máximo estimado para el próximo estado. La regla de actualización está definida por [3]:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \, max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] ,$$

donde:

- $\alpha \in (0, 1]$ es la tasa de aprendizaje, encargada de regular la magnitud del cambio que se realiza en los *Q-values* en cada actualización. Al igual que en cualquier otro modelo de aprendizaje automático, seleccionar una tasa de aprendizaje adecuada es fundamental para garantizar una convergencia estable.
- $\gamma \in [0, 1)$ es el factor de descuento, el cual refleja la importancia de futuras recompensas en contraposición a las inmediatas. Un valor cercano al 1 aumenta la significación de ganancias a largo plazo, promoviendo la capacidad de previsión del agente, pudiendo resultar, no obstante, en un proceso de aprendizaje más lento e inestable.
- $max_a Q(s_{t+1}, a)$ representa el valor máximo de Q en el siguiente estado, considerando todas las acciones que el agente podría tomar.

Este mecanismo permite al agente refinar sus estimaciones de los pares de acción-valor a medida que gana experiencia mediante un proceso iterativo. En particular, ha sido probado que, bajo las condiciones de una exploración suficiente y una tasa de aprendizaje menguante, el Q-learning converge con toda probabilidad hacia la función óptima de acción-valor Q^*, incluso sin emplear una política ϵ-*greedy* [17]. Esta garantía teórica aporta al Q-learning una notable robustez para diversas tareas de aprendizaje por refuerzo.

En implementaciones prácticas, especialmente en entornos con espacios de acciones discretos y finitos, la función Q se almacena en la llamada *Q-table*, una matriz bidimensional cuyas celdas contienen el valor estimado para un cierto par de acción-valor. Habitualmente, la tabla se inicializa con valores nulos o aleatorios, actualizándose tras cada una de las acciones que realiza el agente. A pesar de su naturaleza simple y efectividad provada, esta implementación no escala adecuadamente a espacios de estados grandes o continuos, aspecto que se discutirá en el siguiente apartado.

3.2.3. Limitaciones del Q-learning

Pese a ser un algoritmo fundamental en el contexto del aprendizaje por refuerzo, dada su sencillez y garantía de convergencia teórica, el Q-learning cuenta con varias limitaciones. Comprender estas restricciones es crucial para evaluar su aplicabilidad en problemas del mundo real y potenciar el desarrollo de variantes más avanzadas.

Una de las limitaciones más destacadas del Q-learning clásico es su carente escalabilidad, ya que requiere almacenar el valor estimado para cada par de estado-acción en una *Q-table*. En entornos definidos por espacios de estados o acciones muy amplios o continuos el número de entradas de la tabla crece de manera exponencial, un fenómeno conocido como maldición de la dimensión [3]. Por ello, el Q-learning no es viable desde un punto de vista computacional para problemas de alta dimensionalidad, por ejemplo, en el marco de la navegación visual o manipulación robótica.

Otra de las principales limitaciones del Q-learning tradicional se da por la falta de capacidad del algoritmo para generalizar entre estados o acciones similares. Al almacenar los valores en una tabla de consulta discreta, todos los estados y acciones se procesan de forma independiente, ignorando cualquier posible estructura o similaridad subyacente. En consecuencia, el conocimiento adquirido en base a una región determinada del espacio de estados no puede ser transferido a otras, lo que limita la eficiencia de aprendizaje del algoritmo en ambientes complejos o estructurados [10].

Asimismo, el Q-learning emplea implícitamente un máximo sobre valores estimados para estimar el valor Q máximo, lo que puede conducir a un sesgo positivo. Consideremos un estado en el que todas las posibles acciones retornan un valor real de 0, pero cuyos valores estimados se desconocen y se distribuyen, por tanto, por encima y por debajo del cero. El máximo de los valores reales será siempre cero, sin embargo, el máximo de los valores estimados será positivo, lo que en este contexto se conoce como sesgo de maximización [3].

En el siguiente ejemplo se presenta un proceso de decisión de Markov simple, en el que los episodios comienzan en el estado A, seguidos de una elección de dos alternativas: izquierda o derecha [3]. Ambas transiciones otorgan una recompensa inmediata de cero, no obstante, el camino derecho conduce directamente a un estado terminal, mientras que el izquierdo ofrece una variedad de alternativas con un retorno estimado de -0,1. Podemos observar como, a pesar de que seleccionar la opción izquierda es en realidad siempre un error, el Q-learning favorece inicialmente esta elección por el sesgo de maximización, convergiendo ocasionalmente hacia la política óptima.

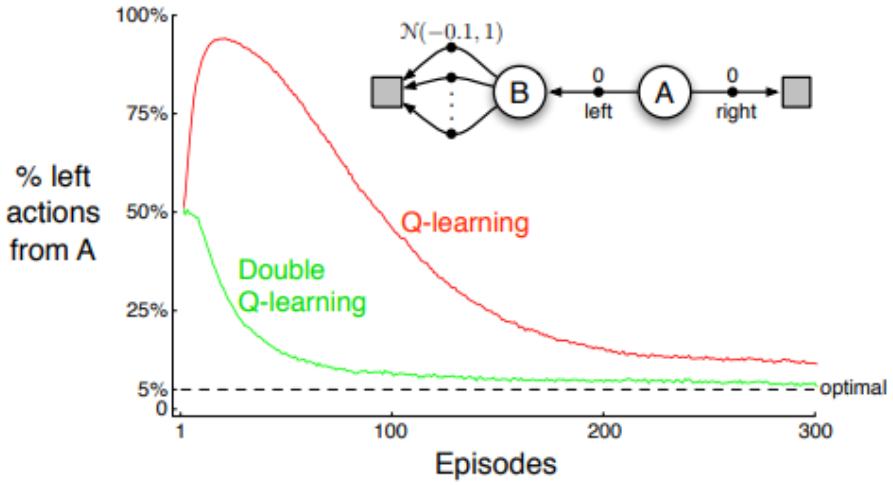

Figura 3.1: Sesgo de maximización en Q-learning y Double Q-learning [3].

Así como el Double Q-learning se presenta en este caso como una alternativa efectiva para solucionar el problema del sesgo de maximización, existen numerosas ampliaciones y evoluciones del Q-learning capaces de aprovechar sus ventajas, a la vez que contrarrestan las limitaciones expuestas.

3.2.4. Extensiones

En los últimos años, se han propuesto numerosos avances y extensiones del Q-learning clásico, con el fin de explotar sus virtudes y minimizar sus restricciones. Estas mejoras potencian significativamente la capacidad del algoritmo para lidiar con problemas reales, particularmente en robótica y tareas basadas en visión por computadora.

Una de las extensiones más notorias son las Deep Q-Networks, introducidas por Mnih et al. en 2015 [10], las cuales reemplazan la tabla Q por una red neuronal profunda encargada de aproximar la función Q. Esta actualización posibilita la operación del algoritmo con entradas complejas como imágenes sin preprocesar, capturadas por una cámara de un robot. Además, las DQNs emplean dos técnicas clave para mejorar la estabilidad del aprendizaje: repetición de la experiencia, que almacena transiciones pasadas y las muestrea de forma aleatoria para el entrenamiento, y redes objetivo, responsables de estabilizar la actualización de los valores Q durante el entrenamiento mediante el uso de un conjunto fijo de pesos [10].

Otro avance importante fue la introducción del Double Q-learning por parte de van Hasselt en 2010 [23], con el objetivo principal de minimizar el impacto del sesgo de maximización. El Double Q-learning resuelve esta limitación desacoplando la selección de las acciones de su evaluación incorporando dos estimadores independientes. En consecuencia, se obtiene una mayor precisión a la hora de estimar los valores Q, así como una mejora en el rendimiento de la política [Ver Figura 3.1].

Más allá de estos avances algorítmicos han surgido métodos híbridos que combinan el Q-learning con elementos del razonamiento basado en modelos. Por ejemplo, Dyna-Q integra el aprendizaje con la planificación simulada usando un modelo del entorno [22], fusionando los enfoques *model-based* y *model-free* para mejorar la eficiencia del algoritmo.

Estos avances resaltan la flexibilidad y adaptabilidad del Q-learning como algoritmo fundamental en el contexto del aprendizaje por refuerzo. A pesar de que las modificaciones presentadas abordan, en su mayoría, una limitación específica, en la práctica se pueden combinar en sistemas únicos para obtener agentes más robustos y escalables.

3.3. Procesamiento de imágenes

3.3.1. Introducción a la visión por computadora

La visión por computadora es un campo de la inteligencia artificial que se centra en capacitar a máquinas para interpretar y comprender información visual, imitando las habilidades perceptuales de los humanos. Su objetivo principal es extraer información relevante de imágenes digitales y vídeos para impulsar la toma de decisiones y actuación automatizada. La visión por computadora es hoy en día una pieza esencial en diversos campos, como la conducción autónoma, la robótica móvil y la interpretación de imágenes médicas.

En este trabajo, la visión por computadora se emplea como la herramienta de percepción del robot, que le permite detectar y seguir una línea dibujada en el suelo. Los

datos de entrada derivados de las imágenes forman la base para el proceso de aprendizaje del agente. Para lograr esto, se deben aplicar una serie de transformaciones para poder extraer características relevantes y reducir la complejidad del procesamiento.

3.3.2. Adquisición y representación de imágenes

El primer paso en cualquier sistema de visión por computadora es la adquisición de imágenes, que consiste en capturar datos visuales del entorno usando sensores, particularmente cámaras. En este proyecto, las imágenes se capturan a través de una cámara virtual montada en el robot dentro del entorno de simulación ofrecido por Coppelia Robotics.

En el medio digital, las imágenes se representan en forma de matrices bidimensionales, en el caso de la escala de grises o matrices tridimensionales para encarnar imágenes en escala de color, habitualmente en formato RGB. En esta última configuración, cada uno de los píxeles de la imagen contiene valores de intensidad para los canales rojo, verde y azul, los cuales forman en conjunción cualquier color del espectro visible, como se muestra en la siguiente figura:

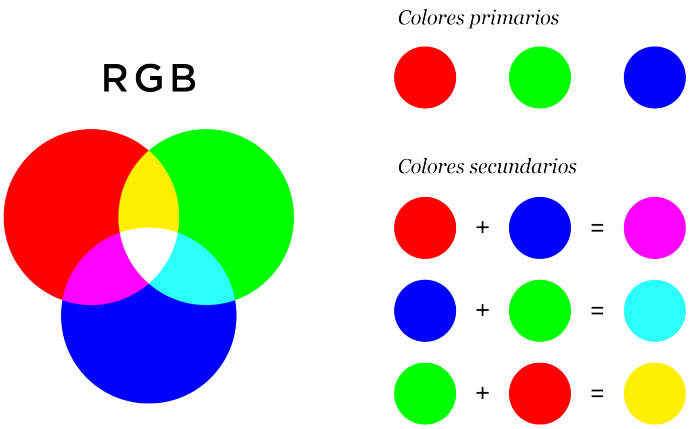

Figura 3.2: Composición de colores en formato RGB [24]

La elección del formato de representación de la imagen impacta directamente la eficiencia del procesamiento. Para aplicaciones robóticas en tiempo real, es habitual convertir imágenes a escala de grises o binario (blanco y negro) para simplificar el tratamiento y centrar el análisis en propiedades estructurales.

3.3.3. Procesamiento de imágenes

El procesamiento de imágenes abarca una serie de técnicas orientadas a la transformación y análisis de imágenes digitales, con el fin de facilitar la extracción de información relevante. Estas operaciones son esenciales, entre otros factores, para reducir el ruido, resaltar los bordes y segmentar regiones de interés en las imágenes antes de ser ingestadas por el algoritmo de aprendizaje.

Algunas de las técnicas más relevantes son:

- **Conversión a escala de grises**, que elimina la información de color y mantiene únicamente la luminancia de los píxeles.
- **Filtrado de imágenes**, tal como el desenfoque y la detección de bordes, los cuales ayudan a detectar contornos y transiciones de intensidad.
- **Binarización**, que consiste en transformar los colores de una imagen a únicamente blanco y negro en base a umbrales de intensidad predefinidos, permitiendo isolar elementos principales del fondo de la imagen.

Estas transformaciones conducen a una identificación más efectiva de patrones visuales, por ejemplo, líneas y trayectorias, por lo que resulta crucial para un problema de seguimiento de líneas.

3.3.4. Detección de líneas y seguimiento

La detección de líneas es un componente crítico en sistemas robóticos móviles destinados al seguimiento de trayectorias, donde un robot debe detectar un camino predefinido y ajustar sus movimientos continuamente para permanecer sobre él. Este problema requiere del preprocesamiento de la imagen para facilitar la distinición de la línea, habitualmente por medio de conversión a escala de grises, filtrado y binarización, seguidos de la aplicación de un algoritmo para detectar la posición exacta de la línea con respecto al robot.

En entornos simulados, como el de este trabajo, la línea a menudo se diseña para que tenga un contraste significativo con el suelo, además de contar con imágenes con menor ruido y distorsión, por lo que el preprocesamiento resulta más sencillo. Una vez se ha binarizado la imagen, la posición de la línea se puede determinar analizando una región específica de la imagen, normalmente la sección inferior, y calculando el centroide de los píxeles blancos. La diferencia entre el centro real de la imagen y la posición del centroide se interpreta como el error que el agente procesará para impulsar su toma de decisiones, tratando de reducirlo al máximo.

Capítulo 4

Desarrollo

En este capítulo se expone la implementación técnica del sistema desarrollado durante este trabajo, detallando las herramientas y los componentes clave que permiten al robot aprender y ejecutar con éxito la tarea de seguimiento de líneas a través del aprendizaje por refuerzo.

4.1. Entorno de desarrollo

4.1.1. CoppeliaSim

CoppeliaSim es una plataforma de simulación diseñada para desarrollar y probar sistemas robóticos en entornos virtuales. Su motor físico integrado, junto a la amplia compatibilidad con diversos tipos de sensores y actuadores, lo convierten en una gran opción para la investigación y aprendizaje en el campo de la robótica.

El entorno de simulación empleado en este trabajo se ha diseñado para abordar un problema clásico de seguimiento de líneas. El entorno está compuesto por un circuito cerrado, representado por medio de una línea negra sobre un suelo de tonos claros, y un robot Pioneer P3DX, un robot compacto y ligero que monta dos ruedas y dos motores de tracción diferencial, es decir, que tienen la capacidad de girar a diferentes velocidades sobre un mismo eje. Además, el circuito incorpora un sensor en el punto de inicio para detectar si el robot completa una vuelta.

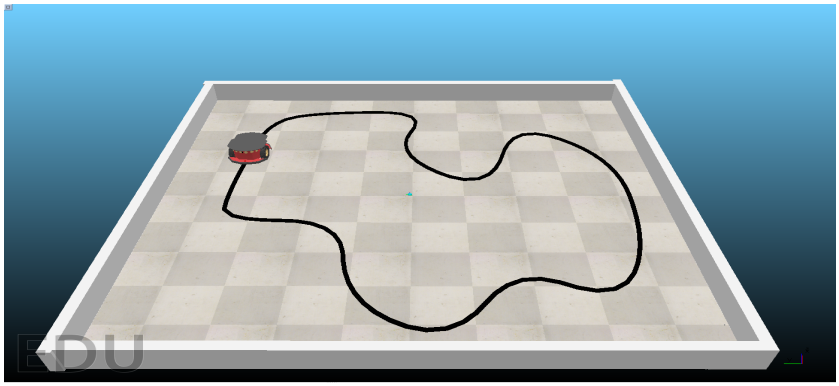

Figura 4.1: Circuito básico para el entrenamiento del agente en CoppeliaSim

La comunicación entre el algoritmo de control y el simulador se establece por medio de ZMQ Remote API, que permite al código Python:

- Iniciar, detener y monitorizar el estado de la simulación.
- Acceder a datos de sensores, como las imágenes capturadas por la cámara y la señal del sensor posicionado en el circuito.
- Controlar la velocidad de giro de los motores del robot.

Esta configuración permite la ejecución externalizada de todo el proceso de aprendizaje en Python, mientras que el comportamiento físico y la percepción del robot se simulan en tiempo real dentro de CoppeliaSim. La versión utilizada en el desarrollo de este trabajo es *CoppeliaSim Edu v4.9.0* [25], apta para estudiantes y profesores.

4.1.2. OpenCV

OpenCV (Open Source Computer Vision Library) es una librería ampliamente extendida para visión por computadora y procesamiento de imágenes. La librería está implementada en C++ principalmente, no obstante, ofrece enlaces de alto nivel para poder ser utilizada extensamente desde Python. En este proyecto, OpenCV tiene un rol fundamental dentro el sistema de percepción del robot, ya que, mediante la extracción de información relevante de las imágenes capturadas, permite determinar el estado del agente de aprendizaje por refuerzo.

La cámara montada en el robot P3DX captura continuamente su campo de visión frontal. De cada uno de los fotogramas procesados se extrae la región de interés, conocida por sus siglas en inglés como ROI, localizada en este caso en la sección inferior de la imagen. A continuación, se aplica un proceso de binarización y se calcula el centroide, con el fin de determinar la posición del robot con respecto al centro de la línea. Este proceso se explica en profundidad en la sección 4.2.3

La versión utilizada es *OpenCV - v4.10.0* [26], el último lanzamiento disponible en el momento del inicio de este trabajo.

4.1.3. Python

Python es el lenguaje de programación empleado para implementar el sistema de control y el agente de aprendizaje por refuerzo. Su elección para la implementación de este trabajo se fundamenta en su simplicidad y compatibilidad con CoppeliaSim y OpenCV.

Las principales librerías utilizadas son las siguientes:

- **ZMQ RemoteAPIClient** - para la comunicación con CoppeliaSim.
- **cv2 (OpenCV)** - para el procesamiento de imágenes.
- **numpy** - para almacenar y mantener la Q-table.
- **matplotlib** - para la visualización de resultados

Se emplea la versión *Python 3.9.10* [27].

4.2. Implementación

4.2.1. Arquitectura general del sistema

El sistema de aprendizaje implementado en este trabajo sigue un diseño modular, que facilita la isolación de las diferentes funcionalidades y permite desarrollar, probar y actualizar cada componente de forma independiente.

Los tres módulos principales son:

- **Percepción:** Módulo encargado de capturar las imágenes a través de la cámara del robot y procesarlas para obtener información relevante, en este caso, la posición del robot con respecto a la línea.
- **Control:** Módulo encargado de seleccionar y ejecutar las acciones (cambios en la velocidad de giro de los motores) en base al estado actual, el cual viene determinado por el módulo de percepción.
- **Aprendizaje:** Módulo encargado del aprendizaje mediante la aplicación del algoritmo Q-learning, actualizando iterativamente la función de valor almacenada en la *Q-table*, que relaciona los estados con las acciones.

La interacción entre los módulos es continua, llevada a cabo en un bucle donde se ejecutan secuencialmente los procesos de percepción, control y aprendizaje.

4.2.2. Comunicación con CoppeliaSim

La comunicación con el entorno de simulación de CoppeliaSim se establece por medio de ZMQ Remote API, que permite la interacción asíncrona entre un script externo de Python y el motor de simulación. La lógica para el control de la simulación y la interacción con el robot se encuentra encapsulada en las clases Coppelia y P3DX, una adaptación de las clases proporcionadas por el tutor de este trabajo como base para el desarrollo del modelo de aprendizaje por refuerzo [28].

Las funciones de estas dos clases permiten:

- Establecer una conexión con el entorno de simulación.
- Iniciar, detener y controlar el estado de una simulación.
- Acceder a los componentes de la escena, entre ellos la cámara y los motores del robot, además del sensor localizado en el punto de incio del circuito.

4.2.3. Adquisición y procesamiento de imágenes

El robot utiliza una cámara de a bordo para observar el suelo delante de él, donde una línea negra define el camino a seguir. Para extraer datos significativos de las imágenes se aplica un flujo de procesamiento usando OpenCV:

- **Región de interés:** Se selecciona únicamente la región inferior de la imagen, ya que permite detectar la posición inmediata del robot con respecto a la línea.
- **Conversión a escala de grises:** Convierte la imagen RGB a escala de grises, con el fin de simplificar el procesamiento restante.

- **Desenfoque gaussiano:** Suaviza la imagen para reducir ruido que podría interferir en la detección de la línea.
- **Segmentación binaria:** Transforma la imagen a una representación con dos valores únicos: blanco para la línea y negro para el resto de la imagen.
- **Cálculo de momentos:** Computa el centroide de la región blanca, produciendo las coordenadas de la línea detectada.

Si no se detecta una línea, el sistema retorna un estado único para notificar la pérdida de la línea. De lo contrario, retorna la posición del centroide para computar la desviación del robot.

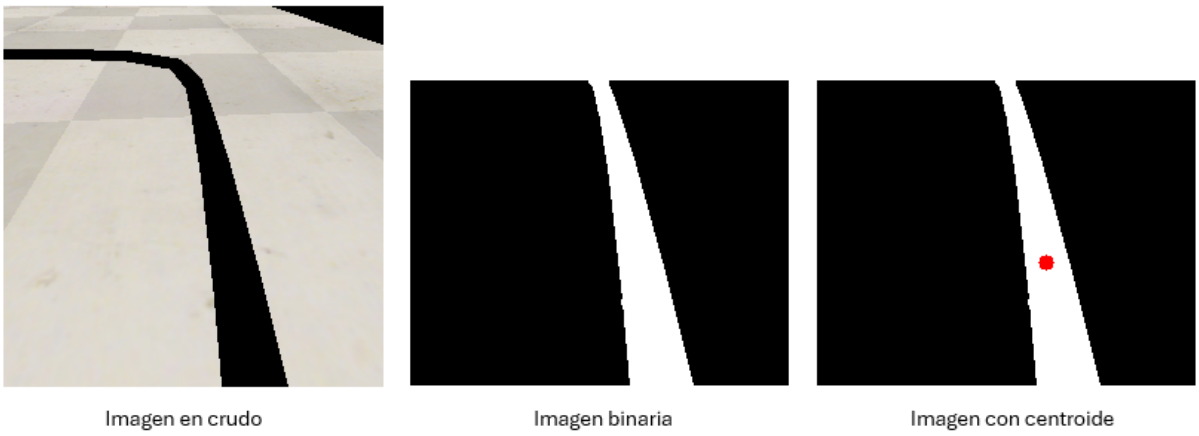

Figura 4.2: Procesamiento de una imagen capturada con la cámara del robot en CoppeliaSim

4.2.4. Representación del estado

La posición relativa del robot con respecto a la línea se codifica en forma de un único valor continuo, el cual representa el desplazamiento horizontal normalizado del centroide de la línea detectada con relación al centro de la imagen. Este valor se escala al intervalo [-1, 1], donde:

- **0** indica un alineamiento perfecto entre el robot y la línea
- **-1** representa la desviación máxima hacia la izquierda
- **1** representa la desviación máxima hacia la derecha

Para ajustar la representación del estado al Q-learning, el desplazamiento continuo se transforma en un número discreto de estados, en función de las dimensiones preestablecidas de la Q-table. Cada estado corresponde a un rango del desplazamiento horizontal normalizado, recogiendo la información sobre la magnitud y la dirección del desplazamiento.

Listing 4.1: Transformación del valor continuo de desplazamiento horizontal normalizado a una representación discreta en base al número de estados

```
1     state = int((offset_norm + 1) / (2 / NUM_STATES))
```

Esta estrategia de discretización dinámica permite al sistema adaptarse sin problemas a distintos números de estados, que se definen por medio de la constante *NUM_STATES* antes de inicializar el modelo.

4.2.5. Selección y aplicación de acciones

El robot puede realizar un número limitado de acciones discretas, que se define por medio de la constante *NUM_ACTIONS*. En el escenario más sencillo de tres posibles acciones, estas corresponden a:

- giro a la izquierda
- continuar recto
- giro a la derecha

Cuando el número de acciones aumenta, que debe ser impar para preservar la simetría entre las acciones de giro a izquierda y derecha, cada dirección de desvío se subdivide en múltiples niveles de intensidad. La acción central corresponde siempre a un avance recto, mientras que el resto de acciones se distribuyen de forma simétrica para representar diversos ángulos de giro en ambas direcciones.

Para mantener el ajuste dinámico de los módulos en base al número de estados y acciones predefinido, se aplica una interpolación lineal para convertir los índices de las acciones en comandos para los motores. En primer lugar, se transforma el índice de la acción en un valor en el rango [-1, 1]:

Listing 4.2: Cálculo de la intensidad de giro en proporción al índice de la acción

```
1     proportion = (action / (NUM_ACTIONS - 1)) * 2 - 1
```

Este valor determina la dirección y la intensidad del giro a realizar. Partiendo de una velocidad base y un rango de giro predeterminados, se computa y se modifica la velocidad de los motores de ambas ruedas de forma independiente:

Listing 4.3: Cálculo y modificación de la velocidad de las ruedas

```
1     left_speed = base_speed - turn_range * proportion
2     right_speed = base_speed + turn_range * proportion
3
4     robot.set_speed(left_speed, right_speed)
```

Este método garantiza la robustez del sistema ante cambios en el número de acciones y elimina la necesidad de ajustar manualmente las velocidades de giro de los motores para cada acción.

4.2.6. Recompensas

La función de recompensa está diseñada para que el agente minimice al máximo la desviación lateral con respecto al centro de la línea. Está definida como:

Listing 4.4: Cálculo de la recompensa en base a la desviación del robot con respecto a la línea

```
    if state == -1:
        reward = -10
    else:
        reward = 1 - 2 * abs(offset)
```

Esta función retorna un valor de recompensa, comprendido en el rango [-1, 1], que desciende gradualmente en proporción al aumento de la desviación lateral del robot. Si el robot pierde visión de la línea se aplica una penalización mayor con un valor de -10 y, además, se avanza al siguiente episodio, devolviendo al robot al punto de inicio. Este diseño impulsa al agente no solo a seguir la línea, sino a hacerlo de forma precisa y, asimismo, castiga rápidamente estados fallidos para prevenir un impacto negativo en el aprendizaje.

4.2.7. Algoritmo de aprendizaje: Q-learning

El núcleo de aprendizaje por refuerzo del sistema implementa el algoritmo clásico Q-learning. Una matriz bidimensional Q(NUM_STATES, NUM_ACTIONS) se inicializa con todos los valores a 0 y se actualiza iterativamente utilizando la siguiente función:

Listing 4.5: Actualización de la Q-table

```
    Q[state, action] += ALPHA * (reward + GAMMA * max(Q[new_state]) - Q[state, action])
```

donde:

- **ALPHA** es la tasa de aprendizaje
- **GAMMA** es el factor de descuento
- **reward** es el último valor de recompensa recibido
- **max(Q[new_state])** es la estimación del retorno máximo a partir del siguiente estado

Además, se aplica una política ϵ-greedy para la selección de las acciones, balanceando exploración y explotación:

Listing 4.6: Aplicación de política epsilon-greedy para la selección de acciones

```
    if np.random.rand() < EPSILON:
        action = random.randint(0, NUM_ACTIONS -1)
    else:
        action = np.argmax(Q[state])
```

Los hiperparámetros ALPHA, GAMMA y EPSILON se pueden modificar para ajustar el rendimiento del agente, impactando métricas fundamentales como velocidad de convergencia, estabilidad y precisión final, aspectos que se estudiarán en profundidad en el siguiente capítulo.

4.2.8. Entrenamiento

El proceso de entrenamiento está estructurado en episodios. Al comienzo de cada uno de ellos, la posición y orientación del robot se resetean a su estado inicial y, a continuación, se ejecuta la siguiente secuencia en bucle, empleando los módulos descritos anteriormente:

1. Captura de la imagen y cómputo del estado actual.
2. Selección y ejecución de una acción en base al estado actual.
3. Captura de la siguiente imagen y cómputo del nuevo estado.
4. Cálculo del valor de recompensa en función del nuevo estado alcanzado.
5. Actualización de la Q-table.
6. Repetición hasta:
    - Pérdida de la línea
    - Finalización del recorrido

El entrenamiento continúa hasta que se alcanza un número máximo de episodios, almacenando el valor de recompensa total de cada episodio, ya que se emplea posteriormente como una métrica para valorar la convergencia del agente. Además, el sistema incluye un mecanismo para registrar el desplazamiento medio del agente con respecto a la línea en cada episodio, puesto que se utiliza como métrica principal para medir la precisión de navegación.

Capítulo 5

Resultados

En este capítulo se presentan los resultados obtenidos durante el proceso de entrenamiento y evaluación del agente de aprendizaje por refuerzo. En los experimentos realizados se modifican sistemáticamente las dimensiones de la *Q-table*, así como los valores de los hiperparámetros, para analizar su impacto en el aprendizaje, la convergencia, la precisión y la robustez del modelo.

Cada episodio se define como un ensayo individual, el cual comienza con el robot posicionado en el punto inicial del circuito y finaliza cuando completa una vuelta o pierde de vista la línea. Además, el rendimiento del agente se mide principalmente a través del desplazamiento horizontal normalizado medio por episodio. Este valor, comprendido en el rango [0, 1], cuantifica el nivel de precisión con el que el agente sigue la línea, correspondiendo un valor de 0 a un alineamiento perfecto. Se emplea esta métrica de rendimiento en lugar de la recompensa acumulada, ya que ofrece un valor claramente interpretable e independiente de factores de confusión como la velocidad del robot, la cual altera la cantidad total de recompensa que se puede acumular en cada episodio, dificultando la comparación del rendimiento entre las diferentes configuraciones.

Las siguientes secciones detallan, en primera instancia, el impacto de las dimensiones de la *Q-table*, seguido por un análisis del efecto de los hiperparámetros, un resumen y comparación de las configuraciones óptimas y, finalmente, una discusión de las limitaciones observadas.

5.1. Evaluación del tamaño de la Q-table

A continuación, se muestra un análisis comparativo de diferentes tamaños de *Q-table*, manteniendo un número impar en estados y acciones con el fin de conservar la simetría entre los espacios de la izquierda y la derecha. El objetivo es evaluar el impacto de la resolución del espacio de estado-acción en el proceso de aprendizaje, exponiendo las ventajas y desventajas que ofrecen cada una de las configuraciones.

Aunque esta sección se centre en el análisis del impacto de las dimensiones de la *Q-table* en el proceso de aprendizaje, se aplican los hiperparámetros más adecuados para cada configuración para ofrecer una comparación rigurosa, puesto que mantenerlos en valores estáticos conduciría a la comparación de políticas con distintos

grados de madurez. Asimismo, los resultados pueden variar ligeramente entre entrenamientos, dada la naturaleza exploratoria del proceso de aprendizaje. Por ello, para cada configuración se presentan los mejores resultados obtenidos tras múltiples entrenamientos.

5.1.1. 3x3

La disposición de tres estados y tres acciones exhibe una convergencia casi inmediata, logrando alcanzar una política estable tras solamente dos episodios. No obstante, la política resultante carece de precisión, con un desplazamiento medio final de aproximadamente 0,38, produciendo, además, trayectorias visiblemente oscilatorias. Al existir únicamente nueve pares de estado-acción en la *Q-table*, el espacio de búsqueda se abarca rápidamente sin necesidad de aplicar una estrategia de exploración. Se mantuvieron los valores estándar 0,2 y 0,8 para la tasa de aprendizaje y el factor de descuento respectivamente, que mostraron tener una influencia muy limitada a esta escala.

La imprecisión y falta de suavidad en el control del robot evidenciaron la necesidad de una mayor resolución en la percepción y el espacio de acciones.

Figura 5.1: Desplazamiento horizontal normalizado medio por episodio para *Q-table* 3x3

5.1.2. 5x5

Esta configuración supuso una mejora considerable en comparación a la anterior, alcanzando un buen equilibrio entre precisión y velocidad de convergencia. El agente obtuvo una política de control estable tras quince episodios, acompañada de una notable reducción en el desplazamiento medio, logrando un valor aproximado de 0,23. Se mantuvieron los valores estándar de hiperparámetros utilizados en la configuración 3x3, los cuales resultaron igualmente efectivos en este caso. Asimismo, la ausencia de exploración volvió a ofrecer el mejor rendimiento, dado el aún reducido tamaño de la Q-table.

La disposición 5x5 demostró un comportamiento consistente a lo largo del circuito, ofreciendo un aumento sustancial de la nitidez del control sin introducir un coste de entrenamiento excesivo. Comparado con la tabla 3x3, se mejoró la precisión en un 40 %, preservando, asimismo, la simplicidad en el ajuste de los parámetros.

Figura 5.2: Desplazamiento horizontal normalizado medio por episodio para *Q-table* 5x5

5.1.3. 7x7

La introducción de siete estados y siete acciones proporcionó un mayor refinamiento del control del robot, alcanzando un desplazamiento medio de 0,16. Sin embargo, se puede apreciar un aumento considerable del tiempo de convergencia, precisándose 50 episodios hasta comenzar a estabilizarse la política y un total de 65 hasta alcanzar

la precisión máxima. A diferencia de las configuraciones anteriores, esta mostró una mayor sensibilidad al ajuste de hiperparámetros. En consecuencia, fue necesario introducir exploración para evitar la convergencia a políticas subóptimas, además de decaimiento en la tasa de aprendizaje, para mitigar ruido y lograr una mayor estabilidad en episodios avanzados [Ver Apartado 5.2.2].

En comparación a la tabla 5x5, esta disposición ofrece una mejora de un 30 % en la precisión del control, a costa de un proceso de entrenamiento más extenso y un aumento en la sensibilidad y complejidad del ajuste de hiperparámetros.

Figura 5.3: Desplazamiento horizontal normalizado medio por episodio para *Q-table* 7x7

5.1.4. 9x9

La configuración 9x9 comenzó a revelar las limitaciones de una resolución excesiva de la *Q-table*. A pesar de apreciarse cierta convergencia a partir de 90 o 100 episodios, la política resultante carece de consistencia. Mientras algunas series de episodios alcanzaron un desplazamiento medio muy reducido, con un valor aproximado de 0,12, otras mostraban una notoria disminución de la precisión o terminaban incluso con la pérdida total de la línea.

Pese a la introducción del decaimiento en las tasas de aprendizaje y exploración, el aumento del factor de descuento para facilitar la consistencia de la política a largo plazo y el aumento del tiempo de entrenamiento hasta un máximo de 300 episodios, no se pudo obtener una política más estable. El agente permaneció vulnerable a

actualizaciones ruidosas y degradación de la política, particularmente en curvas muy pronunciadas.

Esto se debe principalmente a la sensibilidad asociada a la discretización fina del espacio de estados. Cuando aumenta el número de estados, pequeñas variaciones en la percepción visual, por ejemplo, desplazamientos de apenas unos píxeles del centroide, pueden resultar en una transición a un estado diferente que esté asociado a una acción distinta o insuficientemente entrenada. Esta fragmentación provoca que la política establecida no sea capaz de generalizar de forma correcta entre estados muy similares, lo que resulta en una toma de decisiones inconsistente.

Figura 5.4: Desplazamiento horizontal normalizado medio por episodio para *Q-table* 9x9

5.1.5. 7x5

Esta configuración asimétrica se introdujo con el fin de explorar un posible equilibrio entre las dos mejores políticas alcanzadas, sin embargo, no proporcionó ninguna ventaja medible. El tiempo de entrenamiento fue comparable al de la *Q-table* de 7x7, mientras que la precisión máxima fue análoga a la configuración 5x5.

Este resultado puso de manifiesto un principio de diseño fundamental: una mayor granularidad en la percepción no puede aprovecharse si el espacio de acciones carece de la expresividad correspondiente. El agente es capaz de detectar desviaciones más sutiles, pero es incapaz de responder con el grado de control adecuado.

5.2. Ajuste de hiperparámetros

Ajustar los hiperparámetros de forma correcta es primordial para lograr un aprendizaje estable y eficiente, especialmente para *Q-tables* de mayor resolución. En esta sección se resume el impacto de la exploración, la tasa de aprendizaje y el factor de descuento en el proceso de aprendizaje.

5.2.1. Estrategia de exploración

Para tablas de baja resolución (3x3 y 5x5) la exploración es prescindible e incluso contraproducente. El reducido número de pares de estado-acción conduce a una cobertura automática de todas las entradas de la *Q-table*, por lo que la selección de acciones *greedy* ($\epsilon = 0$) logra una convergencia rápida y estable. Introducir exploración de manera innecesaria provoca un comportamiento errático del agente en configuraciones de baja resolución, ya que frecuentemente se seleccionan acciones inoportunas.

Por el contrario, las tablas de mayor resolución (7x7 y 9x9) requieren de la introducción de una estrategia de exploración para evitar la convergencia a políticas subóptimas. En estas configuraciones, omitir la exploración por completo puede causar que el agente adopte un comportamiento deficiente, por ejemplo, manteniendo constantemente la línea en uno de los laterales de su campo de visión. Una tasa de exploración positiva desencadena desviaciones ocasionales, asegurando que cada par de estado-acción se visite y evalúe en algún momento.

En el caso de la tabla 7x7, un valor constante de $\epsilon = 0,01$ fue suficiente para guiar al agente hacia la política óptima. Sin embargo, la configuración 9x9 precisó de una estrategia más avanzada: *epsilon-decay*. Comenzar el entrenamiento con una tasa de exploración elevada (p.e. $\epsilon = 0,2$), la cual se reduce progresivamente a la par que madura la política, permite abarcar la mayoria de los 81 estados durante los episodios iniciales para evitar una convergencia subóptima, mientras que favorece la explotación de la política aprendida a medida que avanza el entrenamiento, evitando un comportamiento errático y la sobrescritura de valores ya establecidos.

5.2.2. Tasa de aprendizaje y factor de descuento

La tasa de aprendizaje mostró tener un grado de influencia limitado en bajas resoluciones de la *Q-table*, cuyo reducido número de entradas se abarca rápidamente. En estos casos, la política se estabiliza velozmente y la simplicidad del entorno minimiza el riesgo de sobrescritura de valores óptimos. Por el contrario, a medida que crece el espacio de estados y acciones (7x7 y 9x9), la tasa de aprendizaje se convierte en una pieza fundamental para garantizar la estabilidad del aprendizaje y la retención de la política. Al permanecer α con un valor constante en estas configuraciones, el agente tiende a olvidar patrones de comportamiento previamente aprendidos. Este fenómeno ocurre principalmente en curvas cerradas del circuito, donde pequeñas variaciones en la trayectoria de entrada o en la percepción del estado pueden conducir a actualizaciones no deseadas en estados similares.

Para abordar este problema, se introdujo *alpha-decay*, reduciendo progresivamente la tasa de aprendizaje de 0,4 a 0,05. Esta estrategia otorga dos beneficios principales: acelerar el proceso de aprendizaje en los episodios iniciales y estabilizar los *Q-values*

en fases avanzadas del entrenamiento, reduciendo la degradación de políticas efectivas.

Aunque el impacto del factor de descuento resultase menos significativo que el de la tasa de aprendizaje, su correcto ajuste contribuyó también a la construcción de una política óptima. Las configuraciones de mayor tamaño se vieron beneficiadas por un aumento del factor de descuento, puesto que permite coordinar secuencias de acciones más largas. Esta característica es particularmente útil para afrontar curvas cerradas, en las que el agente mostró un leve incremento en la consistencia y fluidez a la hora de recorrerlas, al priorizar el control continuo sobre las correcciones cortoplacistas.

En resumen, la tasa de aprendizaje α y el factor de descuento γ tienen roles complementarios, donde:

- α gobierna el ritmo y la estabilidad del aprendizaje
- γ moldea la profundidad temporal de la política aprendida

5.3. Configuraciones óptimas y trade-offs

Cuadro 5.1: Tabla comparativa de configuraciones de *Q-table* e hiperparámetros

Q-table	ε	α	γ	**Despl. medio**	**Episodios conv.**
3x3	0	0.2	0.8	0.38	2
5x5	0	0.2	0.8	0.23	15
7x7	0.01	$0.4 \longrightarrow 0.05$	0.9	**0.16**	65
9x9	$0.2 \longrightarrow 0.01$	$0.4 \longrightarrow 0.05$	0.95	$\sim$0.12 (inestable)	120+

- El punto de partida de 3x3 converge casi inmediatamente, a costa de una precisión deficiente y una trayectoria visiblemente oscilatoria.
- La configuración 5x5 ofrece un buen balance entre velocidad de convergencia, precisión y estabilidad, demostrando una gran consistencia a lo largo del circuito y una baja sensibilidad al ajuste de hiperparámetros.
- La tabla de 7x7 alcanza la mayor precisión estable dentro de un tiempo de entrenamiento razonable, requiriendo un ajuste minucioso de los hiperparámetros para garantizar su buen funcionamiento.
- La configuración 9x9 define la cota superior para la resolución de la *Q-table* en este problema de seguimiento de líneas, mostrando problemas de convergencia y estabilidad.

En última instancia, la selección de la configuración adecuada es inherentemente dependiente del problema y el valor relativo de cada métrica de rendimiento debe considerarse en el contexto específico de aplicación. En este caso, la configuración 7x7 ofrece el mejor compromiso, al lograr la mayor precisión en un tiempo de convergencia aceptable.

5.4. Limitaciones

Esta implementación del Q-learning ha mostrado ser efectiva dentro del entorno controlado empleado durante el proceso de entrenamiento y validación, sin embargo, existen diversas limitaciones que restringen su robustez y capacidad de generalización. A continuación, se presentan observaciones basadas tanto en un análisis teórico, como en pruebas empíricas, en las que se ha empleado una *Q-table* 5x5 preentrenada para medir las limitaciones del agente desarrollado.

5.4.1. Discretización

Una de las principales limitaciones del enfoque tabular utilizado en el Q-learning radica en su dependencia frente a la discretización del espacio de estados y acciones. A medida que crece la resolución de la *Q-table*, cambios pequeños en la posición detectada de la línea pueden conducir a estados distintos e independientes. Esta fragmentación a menudo causa un comportamiento inconsistente del agente ante situaciones prácticamente idénticas.

Este problema es menos significativo en resoluciones menores como la de 5x5, ya que se realiza una distinción más acentuada de los estados existentes. No obstante, la limitación subyacente persiste: la discretización reduce inevitablemente la capacidad del agente para generalizar entre estados muy similares.

5.4.2. Profundidad temporal

Otra de las limitaciones más destacadas del algoritmo implementado reside en su naturaleza puramente reactiva, donde cada decisión se toma únicamente en base al estado percibido en cada momento específico, careciendo de conocimiento alguno acerca de observaciones pasadas o predicciones futuras. Esta falta de profunidad temporal limita al agente a emplear políticas que responden exclusivamente de manera inmediata a la información local.

En el contexto del seguimiento de líneas, este tipo de política es suficiente para lograr una navegación sumamente precisa y estable, no obstante, puede limitar la capacidad del agente para enfrentar escenarios ambiguos, como al recuperarse de una desviación o perder momentáneamente de vista la línea.

5.4.3. Generalización

Para explorar la robustez y la capacidad de generalización de la política aprendida, se realizaron diversos experimentos utilizando una *Q-table* 5x5 preentrenada, dada su simplicidad y comportamiento estable. El agente demostró un alto nivel de tolerancia frente a algunas modificaciones de su entorno:

- Manejó cambios en el grosor de la línea sin perjudicar la precisión de navegación.
- Se adaptó sin problemas a cambios en la velocidad de circulación, siempre y cuando no se superase un umbral más allá del cual el agente no tuviese tiempo suficiente para procesar la información del entorno y ejecutar las acciones.

Sin embargo, hubo otros escenarios en los que el agente no logró un funcionamiento correcto:

- Mantuvo un buen rendimiento ante ciertos cambios en el color de la línea, en tanto que se conservase un buen contraste con el suelo. Sin embargo, colores claros ocasionaron graves problemas para detectar la línea, lo que señala la sensibilidad del sistema de percepción ante la magnitud del contraste entre la línea y el suelo.
- La introducción de intersecciones, puntos en los que dos líneas se cruzan de manera ortogonal, provocó un comportamiento errático del agente, que fue incapaz de determinar la dirección a seguir de forma correcta y predecible, lo que exhibe la incapacidad del Q-learning clásico para identificar patrones complejos.

5.4.4. Procesamiento en tiempo real

Por último, una consideración práctica adicional es la limitación impuesta por el modelo de ejecución en tiempo real. Aunque el Q-learning es muy ligero desde un punto de vista computacional, la capacidad de respuesta del agente está restringida por la frecuencia del bucle de control y la tasa de refresco de la cámara. A velocidades elevadas, el agente requiere de una toma de decisiones muy rápida para garantizar un control estable, por lo que, si el retardo del programa de control resulta demasiado elevado, incluso una política óptima fallará inevitablemente.

Por ello, la velocidad máxima efectiva se ve afectada no sólo por la precisión de la política, sino por la latencia computacional del sistema, la cual debe considerarse a la hora de desplegar agentes de aprendizaje por refuerzo en robots físicos.

Capítulo 6

Conclusiones y futuros trabajos

6.1. Conclusiones

En este Trabajo de Fin de Grado se ha explorado la implementación de un controlador de un robot móvil para el seguimiento de líneas mediante técnicas de aprendizaje por refuerzo, particularmente con ayuda del algortimo Q-learning. Además, se han integrado módulos de visión por computadora empleando OpenCV para procesar la información del entorno y discretizar el espacio de estados, permitiendo al agente interactuar con su entorno a través de un proceso continuo de prueba y error.

Por medio de la experimentación en el entorno de simulación CoppeliaSim, se evaluaron diferentes configuraciones de *Q-table*, acompañado por un análisis sistemático del impacto de los hiperparámetros: tasa de aprendizaje, factor de descuento y tasa de exploración. Los resultados demostraron la capacidad del agente para aprender políticas estables y efectivas utilizando configuraciones relativamente sencillas. Asimismo, la función de recompensa seleccionada logró guiar al agente de forma exitosa hacia un seguimiento preciso de la línea.

A pesar de la consecución de los objetivos principales de este trabajo, se identificaron varias limitaciones, entre ellas una gran sensibilidad a la discretización del espacio y una escasa capacidad de generalización en entornos de mayor complejidad. Estos descubrimientos indican que, aunque el Q-learning tabular supone un punto de partida válido y computacionalmente eficiente, puede resultar insuficiente para abordar tareas robóticas más complicadas.

6.2. Futuros trabajos

Los futuros trabajos deberían enfocarse en mitigar las limitaciones identificadas y extender la aplicabilidad del sistema a escenarios más complejos y realistas. Un avance fundamental sería la implementación del agente en un robot móvil físico, lo que introduciría consideraciones adicionales relativas al hardware, como la autonomía, la posible imprecisión de los motores y la variabilidad de los sensores. Estos factores introducen ruido, latencia e incertidumbres que a menudo están ausentes o simplificados en la simulación. La validación del sistema en un entorno real porporcionaría información valiosa acerca de su robustez y adaptabilidad más allá de un entorno simulado.

Además, con el fin de mejorar la capacidad de generalización y la escalabilidad del modelo, resulta esencial evolucionar hacia un planteamiento basado en la aproximación de funciones, eliminando la necesidad de mantener la representación de la función de valor por medio de una *Q-table* discreta. Un enfoque prometedor sería el uso de Deep Q-Networks, que permiten al agente procesar las imágenes de entrada en crudo mediante la incorporación de redes neuronales convolucionales. Esta funcionalidad eliminaría la necesidad de discretizar el espacio de estado-acción de forma manual, facilitando la adaptación a espacios dinámicos y continuos.

Por último, un futuro trabajo podría explorar la automatización del ajuste de hiperparámetros mediante el uso de algoritmos de optimización. En la implementación actual, la tasa de aprendizaje, factor de descuento y tasa de exploración deben ser seleccionadas manualmente a través de un proceso iterativo de experimentación. Al crecer la complejidad del modelo, especialmente al adoptar técnicas más avanzadas como las DQN, la interacción entre los hiperparámetros incrementa su nivel de complejidad, por lo que la elección de valores subóptimos puede limitar la convergencia y estabilidad.

Para prevenir este problema, técnicas tales como optimización bayesiana y algoritmos géneticos podrían emplearse para realizar una búsqueda sistemática del espacio de hiperparámetros e identificar las configuraciones que maximizan el aprendizaje. En el contexto del aprendizaje por refuerzo, donde el entrenamiento tiene a menudo un alto coste computacional asociado, además de una gran sensibilidad al ajuste de los parámetros, la optimización automatizada podría mejorar de forma significativa la eficiencia muestral, el tiempo de convergencia y la efectividad de la política.

Capítulo 7

Impacto

7.1. Análisis del impacto

El trabajo desarrollado supone una contribución académica y técnica al campo de la robótica autónoma y el aprendizaje por refuerzo. Al implementar con éxito un controlador de un robot móvil para el seguimiento de líneas empleando Q-learning, el proyecto exhibe el potencial de los algoritmos clásicos de aprendizaje por fuerzo para realizar tareas de navegación en entornos simplificados. La arquitectura modular y la documentación extensa facilitan su reutilización y adaptación para experimentación futura o fines educacionales, particularmente en formaciones en inteligencia artificial y robótica.

Desde un punto de vista educativo, este proyecto ha tenido un impacto considerable en el desarrollo del autor. El trabajo realizado ha fomentado una comprensión profunda de la teoría del aprendizaje por refuerzo, visión por computadora e integración software-hardware mediante la simulación robótica. Asimismo, el carácter práctico del proyecto ha favorecido la consolidación de conocimientos de programación en Python, el uso de librerías como OpenCV y el diseño de experimentos en entornos controlados. Estas experiencias forman una sólida base para el desarrollo de futuros trabajos de investigación, así como de aplicaciones industriales de sistemas autónomos.

En última instancia, existen algunas consideraciones éticas y de seguridad que se deben contemplar cuidadosamente antes de desplegar un sistema de estas características en un entorno real. Los agentes autónomos deben estar equipados con robustos procesos para enfrentar situaciones inesperadas y fallos potenciales, puesto que la incapacidad para reaccionar ante estos imprevistos puede provocar accidentes, lesiones y daños materiales. Por este motivo, todo sistema robótico autónomo expuesto al entorno humano debe incorporar mecanismos de tolerancia a fallos, protocolos de emergencia y supervisión activa para garantizar la seguridad.

7.2. Objetivos de Desarrollo Sostenible

Este Trabajo de Fin de Grado se alinea con los siguientes Objetivos de Desarrollo Sostenible de la Asamblea General de las Naciones Unidas:

- **ODS 9 - Industria, Innovación e Infraestructura:** Este trabajo contribuye a la innovación tecnológica en el campo de la robótica con el desarrollo de un enfoque ligero y reproducible para el diseño de controladores para navegación autónoma utilizando aprendizaje por refuerzo. Esta implementación aporta una base para aplicaciones más complejas en robótica industrial, logística y transporte autónomo.
- **ODS 11- Ciudades y Comunidades Sostenibles:** Los robots móviles autónomos tienen el potencial para ser integrados en futuras ciudades inteligentes, realizando tareas como transporte de pasajeros, monitorización medioambiental y recolección de residuos.
- **ODS 12 - Producción y Consumo Responsables:** El desarrollo se ha realizado en su totalidad empleando un entorno simulado, reduciendo de forma significativa el consumo de recursos materiales y la cantidad de residuos electrónicos generados.

Bibliografía

[1] Europa Press PortalTIC. *Meta y Apple ven en los robots humanoides el futuro de la inteligencia artificial*. 2025. URL: https://www.europapress.es/portaltic/sector/noticia-meta-apple-ven-robots-humanoides-futuro-inteligencia-artificial-20250217142937.html (visitado 01-04-2025).

[2] Jens Kober, J. Bagnell y Jan Peters. «Reinforcement Learning in Robotics: A Survey». En: *The International Journal of Robotics Research* 32 (sep. de 2013), págs. 1238-1274. DOI: 10.1177/0278364913495721.

[3] Richard S. Sutton y Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

[4] Chen Tang et al. *Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes*. 2024. arXiv: 2408.03539 [cs.RO]. URL: https://arxiv.org/abs/2408.03539.

[5] Dong Han et al. «A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation». En: *Sensors* 23.7 (2023). ISSN: 1424-8220. DOI: 10.3390/s23073762. URL: https://www.mdpi.com/1424-8220/23/7/3762.

[6] Angelo Moroncelli et al. *Integrating Reinforcement Learning with Foundation Models for Autonomous Robotics: Methods and Perspectives*. 2024. arXiv: 2410.16411 [cs.RO]. URL: https://arxiv.org/abs/2410.16411.

[7] Boston Dynamics. *Starting on the Right Foot with Reinforcement Learning*. 2024. URL: https://bostondynamics.com/blog/starting-on-the-right-foot-with-reinforcement-learning/ (visitado 03-04-2025).

[8] Christopher Watkins. «Learning From Delayed Rewards». En: (ene. de 1989). URL: https://www.researchgate.net/publication/33784417_Learning_From_Delayed_Rewards.

[9] Paul Gajewski et al. *Solving Multi-Goal Robotic Tasks with Decision Transformer*. 2024. arXiv: 2410.06347 [cs.RO]. URL: https://arxiv.org/abs/2410.06347.

[10] Volodymyr Mnih et al. «Human-level control through deep reinforcement learning». En: *Nature* 518 (feb. de 2015). ISSN: 1476-4687. DOI: 10.1038/nature14236. URL: https://doi.org/10.1038/nature14236.

[11] Murad Bashabsheh. «Autonomous Robotic Systems with Artificial Intelligence Technology Using a Deep Q Network-Based Approach for Goal-Oriented 2D Arm Control». En: 5 (oct. de 2024), págs. 1872-1887. DOI: 10.18196/jrc.v5i6.23850. URL: https://www.researchgate.net/publication/389500139_Autonomous_Robotic_Systems_with_Artificial_Intelligence_Technology_Using_a_Deep_Q_Network-Based_Approach_for_Goal-Oriented_2D_Arm_Control.

[12] Leslie Pack Kaelbling, Michael L. Littman y Andrew W. Moore. «Reinforcement Learning: A Survey». En: 4 (mayo de 1996), págs. 237-285. DOI: https://doi.org/10.1613/jair.301.

[13] B. F. Skinner. *La conducta de los organismos*. Editorial Fontanella, 1975.

[14] Dave Bergmann. *Qué es el aprendizaje por refuerzo a partir de la retroalimentación humana (RHLF)?* 2023. URL: https://www.ibm.com/es-es/think/topics/rlhf (visitado 21-04-2025).

[15] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

[16] Luis Esteve Elfau. «Procesos de decisión de Markov». En: *Fundación Universitat Oberta de Catalunya* (2021). URL: https://openaccess.uoc.edu/bitstream/10609/151792/7/ProcesosDeDecisionDeMarkov.pdf.

[17] Christopher J. C. H. Watkins y Peter Dayan. «Q-learning». En: *Machine Learning* 8 (mayo de 1992), págs. 279-292. DOI: https://doi.org/10.1007/BF00992698.

[18] Ronald J. Williams. «Simple statistical gradient-following algorithms for connectionist reinforcement learning». En: *Machine Learning* 8 (mayo de 1992), págs. 229-256. DOI: https://doi.org/10.1007/BF00992696.

[19] Vijay Konda y John Tsitsiklis. «Actor-Critic Algorithms». En: *Advances in Neural Information Processing Systems*. Ed. por S. Solla, T. Leen y K. Müller. Vol. 12. MIT Press, 1999. URL: https://proceedings.neurips.cc/paper_files/paper/1999/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[20] Speechmatics - Equipo editorial. *Eficiencia de la muestra y lo que podría significar para la industria*. 2023. URL: https://www.speechmatics.com/company/articles-and-news/sample-efficiency-industry (visitado 21-04-2025).

[21] Thomas M. Moerland et al. *Model-based Reinforcement Learning: A Survey*. 2022. arXiv: 2006.16712 [cs.LG]. URL: https://arxiv.org/abs/2006.16712.

[22] Richard S. Sutton. «Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming». En: *Machine Learning Proceedings 1990*. Ed. por Bruce Porter y Raymond Mooney. San Francisco (CA): Morgan Kaufmann, 1990, págs. 216-224. ISBN: 978-1-55860-141-3. DOI: https://doi.org/10.1016/B978-1-55860-141-3.50030-4. URL: https://www.sciencedirect.com/science/article/pii/B9781558601413500304.

[23] Hado Hasselt. «Double Q-learning». En: *Advances in Neural Information Processing Systems*. Ed. por J. Lafferty et al. Vol. 23. Curran Associates, Inc., 2010. URL: https://proceedings.neurips.cc/paper_files/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf.

[24] Jaime P. Llasera. *RGB y CMYK: Qué son y cuándo usar cada modo de color*. 2020. URL: https://heyjaime.com/blog/rgb-y-cmyk/ (visitado 23-04-2025).

[25] Coppelia Robotics. *CoppeliaSim*. Versión CoppeliaSim Edu 4.9.0. 2025. URL: https://www.coppeliarobotics.com.

[26] OpenCV team. *OpenCV*. Versión 4.10.0. 2024. URL: https://opencv.org/releases.

[27] Python Software Foundation. *Python*. Versión 3.9.10. 2022. URL: https://www.python.org/downloads/release/python-3910.

[28] Javier de Lope. *robotica.py*. 2024. URL: https://github.com/jdlope/robotica/blob/master/coppelia/zmqRemoteApi/python/robotica.py.

Apéndice A

Anexo

A.1. Código

Los archivos desarrollados durante este Trabajo de Fin de Grado se encuentran en el siguiente repositorio de GitHub:

- https://github.com/eduld30/Q-learning_TFG