La investigación reportada en esta tesis es parte de los programas de investigación del CICESE (Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California).
La investigación fue financiada por el SECIHTI (Secretaría de Ciencia, Humanidades, Tecnología e Innovación).

Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias
en Ciencias de la Computación

Captura y seguimiento de un evasor móvil utilizando Aprendizaje por Refuerzo Profundo

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

Enrique Manuel Companioni Valle

Ensenada, Baja California, México

2025

Tesis defendida por

Enrique Manuel Companioni Valle

y aprobada por el siguiente Comité

Dr. Ubaldo Ruiz López
Director de tesis

Dr. Carlos Alberto Brizuela Rodríguez

Dr. Jonatán Peña Ramírez

Dr. Pedro Gilberto López Mariscal
Coordinador del Posgrado en Ciencias de la Computación

Dra. Ana Denise Re Araujo
Directora de Estudios de Posgrado

Resumen de la tesis que presenta Enrique Manuel Companioni Valle como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Captura y seguimiento de un evasor móvil utilizando Aprendizaje por Refuerzo Profundo

Resumen aprobado por:

Dr. Ubaldo Ruiz López
Director de tesis

El problema de captura dentro del dominio de persecución-evasión en entornos desconocidos con obstáculos constituye uno de los desafíos más relevantes en el campo de la robótica móvil. Tradicionalmente, esta clase de problemas se ha abordado mediante enfoques de control, representaciones basadas en grafos, aproximaciones diferenciales e incluso algoritmos genéticos con el objetivo de introducir cierto grado de inteligencia. No obstante, estos métodos suelen ser efectivos únicamente bajo condiciones específicas y presentan dificultades para adaptarse o escalar a escenarios más complejos. En este contexto, el presente trabajo propone la utilización de aprendizaje por refuerzo profundo, empleando como datos de entrada un conjunto de variables que representan el estado de un robot móvil. El objetivo es aprender una política de movimiento capaz de navegar en un entorno bidimensional con obstáculos, frente a la presencia de un agente evasor, con la finalidad de ejecutar exitosamente la tarea de captura. Para ello, se implementó un simulador que representa un entorno con obstáculos, un perseguidor y un evasor, así como un sistema de recompensas que guía al perseguidor para completar la tarea de seguimiento y captura. Se empleó el algoritmo Proximal Policy Optimization (PPO) junto con aprendizaje curricular como métodos de entrenamiento, evaluando diferentes arquitecturas de redes neuronales y configuraciones del estado del perseguidor. Los experimentos se llevaron a cabo en entornos de dificultad creciente, denominados niveles, que determinan la complejidad del problema en función de factores como la cantidad de obstáculos, el rango de captura y las configuraciones iniciales de los agentes. Como resultado, se obtuvieron dos modelos con buen desempeño en los niveles más difíciles: una red MLP de cuatro capas densas mostró mejor desempeño en el nivel 3, alcanzando una tasa de éxito de captura de 0.94, mientras que una arquitectura CNN se desempeñó mejor en el nivel 4, logrando una tasa de éxito de 0.66.

Palabras clave: problemas de persecución-evasión, aprendizaje por refuerzo profundo, PPO, políticas de captura, aprendizaje curricular

Abstract of the thesis presented by Enrique Manuel Companioni Valle as a partial requirement to obtain the Master of Science degree in Computer Science.

Capture and tracking of a mobile evader using Deep Reinforcement Learning

Abstract approved by:

Dr. Ubaldo Ruiz López
Thesis Director

The capture problem within the pursuit-evasion domain in unknown environments with obstacles represents one of the most significant challenges in the field of mobile robotics. Traditionally, this type of problem has been addressed through control-based approaches, graph representations, differential methods, and even genetic algorithms in an attempt to introduce some degree of intelligence. However, these methods are usually effective only under specific conditions and face difficulties when adapting or scaling to more complex scenarios. In this context, the present work proposes the use of deep reinforcement learning, employing as input a set of variables that represent the state of a mobile robot. The objective is to learn a motion policy capable of navigating in a two-dimensional environment with obstacles, in the presence of an evader agent, in order to successfully accomplish the capture task. To this end, a simulator was implemented to represent an environment with obstacles, a pursuer, and an evader, as well as a reward system to guide the pursuer in completing the tracking and capture task. The Proximal Policy Optimization (PPO) algorithm was employed along with curriculum learning as training methods, testing different neural network architectures and state configurations of the pursuer. The experiments were performed in environments of increasing difficulty, referred to as levels, which define the complexity of the problem according to factors such as the number of obstacles, the capture range, and the initial configurations of the agents. As a result, two models with strong performance were obtained in the most difficult levels: a four-layer dense MLP performed best at level 3, achieving a capture success rate of 0.94, while a CNN architecture performed best at level 4, achieving a capture success rate of 0.66.

Keywords: pursuit-evasion problems, deep reinforcement learning, PPO, capture policies, curriculum learning

Dedicatoria

A mi esposa y a mis padres, por su apoyo, sacrificio y el amor que me han brindado durante esta etapa.

Agradecimientos

Al Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California en conjunto con el Departamento de Ciencias de la Computación del CICESE por la oportunidad y las herramientas para la elaboración de este trabajo.

A la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) por brindarme el apoyo económico para realizar mis estudios de maestría con el No. de becario 22265124.

A mi asesor, Dr Ubaldo Ruiz López, por la orientación, el apoyo y la atención recibida en la realización de este trabajo.

A mi comité de tesis, Dr. Carlos Alberto Brizuela Rodríguez y Dr. Jonatán Peña Ramírez, por las observaciones y comentarios.

Tabla de contenido

Página

Lista de figuras

Figura Página

Figura Página

Lista de tablas

Tabla Página

Capítulo 1. Introducción

En este capítulo se presenta un panorama general del trabajo desarrollado. Se expone la motivación que dio origen a esta investigación, se plantea la problemática que se busca abordar, y se definen los objetivos generales y específicos que guiaron el desarrollo de la propuesta.

1.1. Motivación

En el campo de la robótica móvil, la teoría de juegos ha surgido como una herramienta fundamental para mejorar la eficacia y eficiencia de los agentes en la realización de tareas en entornos desconocidos y dinámicos (Isaacs (1999)). Muchos de los problemas en este campo pueden ser modelados mediante formulaciones de teoría de juegos, especialmente aquellos que involucran múltiples agentes —ya sea en escenarios de competencia o cooperación— con el fin de alcanzar objetivos específicos.

Ejemplos representativos incluyen tareas de búsqueda y rescate, donde robots autónomos deben localizar personas u objetos en movimiento dentro de entornos con obstáculos (Boston-Dynamics (2023)); la vigilancia y patrullaje, en la que uno o varios agentes deben seguir de cerca a un intruso o adversario que intenta evadir la detección (Boston-Dynamics (2023)); y la seguridad en entornos estratégicos, donde sistemas móviles deben interceptar amenazas antes de que alcancen zonas críticas. De manera similar, en la logística industrial, los robots móviles enfrentan el desafío de navegar hacia objetivos dinámicos evitando colisiones con otros robots o con obstáculos en movimiento (Roach (2022); Boston-Dynamics (2024)).

La complejidad de estos escenarios, caracterizados por entornos inciertos, presencia de obstáculos y adversarios con comportamientos dinámicos, hace que el seguimiento y captura no sea únicamente un reto académico, sino un problema crítico cuya resolución efectiva puede generar un impacto directo en la seguridad, la eficiencia operativa y la capacidad de respuesta de sistemas autónomos en escenarios reales.

Desde un punto de vista teórico, los juegos de persecución-evasión pueden definirse de diferentes formas en dependencia de la naturaleza del problema que se desea resolver (Isaacs (1999)). Por ejemplo, en la tarea de seguimiento de objetivos, uno o varios robots intentan mantener en vista a otro robot (o agente adversario), mientras este intenta evadirlos. El reto consiste en que los perseguidores mantengan el rastreo visual del evasor el mayor tiempo posible, evitando tanto la colisión con los obstáculos del

entorno como la pérdida de contacto visual. Por otro lado, el evasor busca escapar de la región de detección lo más pronto posible.

Los estudios tradicionales para la solución de juegos de PE en robótica móvil suelen basarse en herramientas matemáticas como ecuaciones diferenciales, teoría de grafos, y teoría de juegos, muchas veces en conjunción con otras áreas interdisciplinarias. Sin embargo, estas metodologías presentan dos limitaciones fundamentales. Primero, la variabilidad en las formas geométricas de los obstáculos en entornos reales pueden comprometer la generalización de algoritmos diseñados para escenarios específicos. Segundo, estos enfoques enfrentan dificultades para integrar los datos provenientes de diferentes sensores con los algoritmos, lo que restringe su aplicabilidad práctica.

Frente a estas limitaciones, surge la necesidad de explorar enfoques alternativos que integren técnicas de inteligencia artificial y aprendizaje automático. En este sentido, el aprendizaje por refuerzo profundo (*Deep Reinforcement Learning*, DRL) se ha consolidado como una herramienta prometedora (Tang et al. (2025); Gronauer & Diepold (2022); Matsuo et al. (2022); François-Lavet et al. (2018)). Su auge se debe tanto a su capacidad para resolver problemas de alta dimensionalidad como a su habilidad de aprender directamente de la interacción con el entorno, lo cual lo convierte en un enfoque idóneo para enfrentar los desafíos inherentes al seguimiento y captura en juegos de persecución-evasión.

1.2. Antecedentes

En esta sección se presenta el trabajo previo vinculado con la motivación para el desarrollo de este proyecto de tesis. El contenido se organiza en torno a los principales enfoques propuestos en la literatura para abordar problemas de persecución–evasión, destacando sus aportes, limitaciones y la manera en que sirven de base para la investigación realizada.

A partir de la literatura consultada, se desarrolló una taxonomía parcial, presentada en la Figura 1, que agrupa los parámetros relacionados con los problemas de persecución-evasión (PE). Los dos parámetros más relevantes que permiten clasificar este tipo de problemas son: (1) la forma en que se modela y caracteriza el objetivo; y (2) la representación del entorno. De acuerdo con Chung et al. (2011), la forma en que se representa el entorno define los enfoques principales para resolver los juegos de PE: el enfoque diferencial y el enfoque combinatorio.

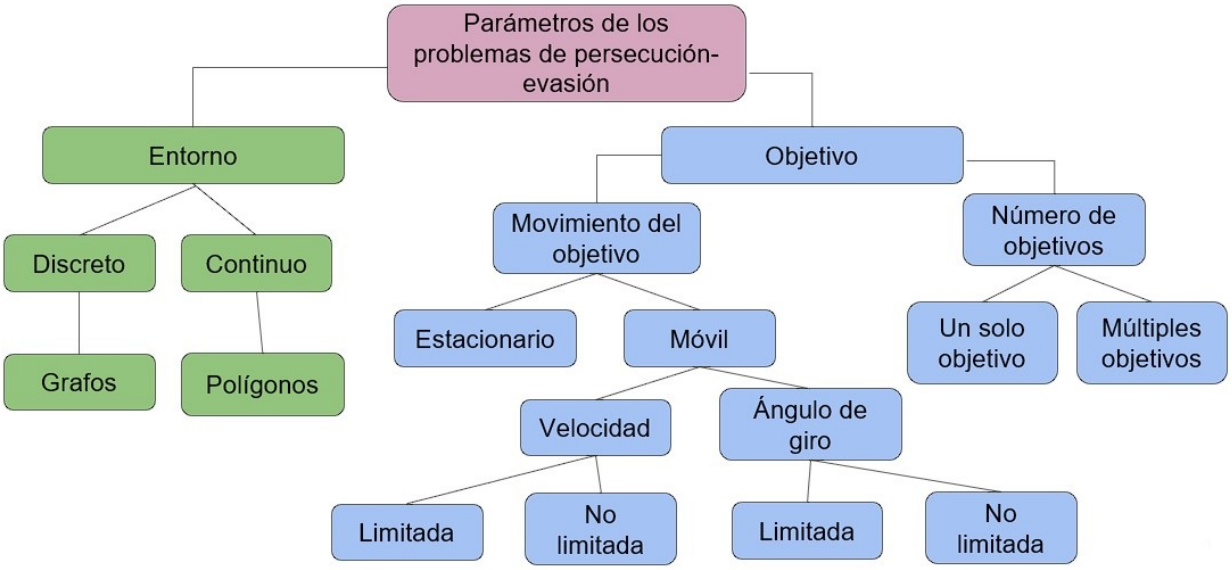

Figura 1. Taxonomía parcial de los parámetros de los problemas de persecución-evasión, adaptado de Robin & Lacroix (2016)

1.2.1. Persecución-evasión en grafos

En entornos complejos, los juegos de PE suelen abordarse mediente el enfoque combinatorio, en el cual el entorno se representa como un grafo. Este enfoque se basa en realizar estrategias de búsqueda sobre dicho grafo, de manera que no exista una conexión directa entre un nodo visitado por el perseguidor y uno no visitado donde el evasor podría encontrarse. Una de las limitaciones principales de este enfoque es que no permite modelar adecuadamente las restricciones cinemáticas ni las capacidades sensoriales de los agentes.

Isler et al. (2005) modelan el entorno mediante polígonos y lo segmentan utilizando triangulación. El evasor es tratado como un agente con visión global del ambiente y velocidad infinita, mientras que el perseguidor tiene una velocidad limitada y visión local limitada a la región triangular donde se encuentra. Se propone una estrategia de visita aleatoria sobre el árbol generado por la triangulación, donde la elección de nodos hijos se realiza de forma probabilística. Este enfoque permite al perseguidor, con cierta probabilidad, detectar al evasor incluso en entornos donde este puede cambiar de región y volverse invisible para el perseguidor. Se argumenta que, en estos entornos, las estrategias determinísticas permiten al evasor escapar indefinidamente, mientras que la aleatoriedad introduce una ventaja para el perseguidor.

Kehagias et al. (2009) formulan el problema de PE como una búsqueda sobre un grafo que represente un entorno en interiores (por ejemplo, el interior de una vivienda u oficina). El mapa del entorno se descompone en celdas, cada una representa un nodo de un grafo finito. Las aristas del grafo corresponden a la conectividad entre las celdas o habitaciones. El objetivo es lograr una cobertura completa del grafo, garantizando que no exista un camino directo entre los nodos no visitados y los ya explorados. Además, se considera el rango de los sensores de un robot, con esto se establecen nodos visitados y nodos visibles donde se puede descubrir al evasor. Se propone una variante del algoritmo IGNS (Iterative Greedy Node Search), que mejora la eficiencia computacional en la búsqueda y permite la extención al uso de múltiples perseguidores. Los resultados, evaluados en cinco entornos comunes en la literatura, muestran que si bien el algoritmo no garantiza solución en todos los casos, su desempeño es adecuado para entornos complejos, aunque no necesariamente óptimo.

Por otro lado, en Kolling & Carpin (2010), los grafos no representan directamente la geometría del entorno, sino el costo asociado a su cobertura. Se propone un algoritmo distribuido para la detección de intrusos en entornos desconocidos mediante múltiples robots. El algoritmo se basa en el avance coordinado de una línea de robots y en la formación de particiones para explorar el entorno. En este trabajo, los llamados "grafos de supervisión" representan tanto la topología del entorno como el costo en términos de recursos (robots) necesarios para cubrir toda una región del mismo.

1.2.2. Persecución-evasión con enfoque diferencial

En el enfoque diferencial, los agentes involucrados en el juego de persecución–evasión son modelados como sistemas dinámicos cuya evolución temporal está determinada por ecuaciones diferenciales. El objetivo central es resolver la ecuación de Hamilton–Jacobi–Bellman (HJB) (Liberzon (2011)), para obtener la función de valor óptima que define la política de control óptima para cada estado del sistema. Una ventaja clave de este enfoque es que permite representar de manera explícita restricciones físicas, como cotas en la velocidad o aceleración, a través de ecuaciones diferenciales. Sin embargo, presenta limitaciones para escenarios donde la geometría del entorno juega un papel importante, especialmente cuando aumenta la complejidad de las ecuaciones diferenciales involucradas en la modelación de la dinámica de los agentes.

Bhattacharya & Hutchinson (2010) analizan el caso de seguimiento alrededor de una esquina para agentes con velocidad limitada. En este trabajo se obtienen estrategias en equilibrio de Nash utilizando la teoría

de juegos diferenciales (Isaacs (1999)). Es decir, si uno de los jugadores se desvía de su estrategia óptima el otro se ve beneficiado. Se concluye que, cuando la línea de visibilidad entre los jugadores es tangente a un vértice del obstáculo, la estrategia óptima para ambos agentes consiste en desplazarse en direcciones opuestas a través de un par de rectas paralelas.

La planificación de movimiento, considerando tanto restricciones mecánicas como sensoriales en robots móviles ha sido estudiada ampliamente, particularmente para sistemas no holonómicos [1] Li & Canny (1990); Murray & Sastry (1993); Laumond et al. (1998); Balkcom & Mason (2002)). Otros estudios han abordado la interacción de perseguidores no holonómicos y evasores menos restringidos, como se muestra en Murrieta-Cid et al. (2011), donde el perseguidor es modelado como un robot de manejo diferencial con restricciones en la velocidad de giro y un evasor con movimiento omnidireccional únicamente limitado en velocidad.

Zou & Bhattacharya (2016) presentan una estrategia basada en control óptimo para resolver un juego de PE en escenarios donde existen restricciones de estado, sin necesidad de emplear herramientas de teoría de juegos diferenciales. El entorno es dividido en cinco particiones donde las condiciones para ganar el juego son diferentes en cada una de ellas. Con su propuesta proporciona una solución óptima al problema de seguimiento durante un horizonte finito en un entorno con una sola esquina.

En Parras et al. (2016) se presenta una formulación del problema de persecución–evasión para un vehículo aéreo no tripulado (UAV), en la que el perseguido busca evadir ataques de interferencia de otro UAV. El planteamiento, limitado a entornos sin obstáculos, utiliza como criterio de optimización la fórmula de capacidad de Shannon para todas las estaciones de comunicación involucradas. Mediante la resolución de dieciséis ecuaciones de trayectoria regresivas, derivadas de la dinámica de movimiento del UAV junto con la ecuación principal de Isaacs (1999), se determina la trayectoria óptima para ambos agentes.

Otros estudios han propuesto representar el problema de persecución–evasión mediante entornos que simulan geometrías de objetos del mundo real. Por ejemplo, en Fang et al. (2020), el problema se modela como un "juego de pesca" en un entorno 3D sin obstáculos, donde el evasor (pez) busca escapar de una red de captura, en forma de cono, definida por uno o varios perseguidores (pescadores). Se analizan dos escenarios: uno con un evasor y dos perseguidores, y otro con un evasor y múltiples perseguidores. En ambos casos, la interacción de los agentes es planteada como un problema de optimización mín–máx alrededor del vértice del cono para determinar las condiciones de éxito para cada equipo, junto con

[1] Un sistema no holonómico en robótica es aquel en el cual sus variables de movimiento pueden ser dependientes entre ellas y al integrarlas no se obtiene directamente la posición. Esta dependencia resulta en restricciones de movimiento en el sistema, por lo tanto, no se puede mover instantáneamente en todas las direcciones. Formalmente, las propiedades de un sistema no holonómico se pueden determinar usando el teorema de Frobenius (Laumond et al. (1998)

algoritmos de control para los perseguidores y el evasor.

Lee et al. (2020) analizan el juego de persecución–evasión en el contexto de defensa de perímetros, donde un invasor aéreo y un defensor terrestre compiten para alcanzar una posición definida en un hemisferio libre de obstáculos. El modelo utilizado representa la dinámica en términos de diferencias angulares (azimut y altitud) y la distancia euclidiana entre ambos agentes. El costo para cada jugador está dado por la diferencia de tiempos para alcanzar un punto específico del perímetro, y se presenta una división de la frontera del espacio de estados en regiones de éxito para cada agente.

La inclusión de obstáculos en escenarios de persecución–evasión introduce desafíos adicionales para representar entornos realistas en robótica móvil (LaValle et al. (1997); Jung & Sukhatme (2002); Bandyopadhyay et al. (2006)). En Lozano et al. (2022a) se presenta un modelo donde tanto perseguidor como evasor son sistemas no holonómicos representados por discos en un entorno con múltiples obstáculos, considerando además limitaciones en el campo de visión del perseguidor.

Por último, otros enfoques han abordado explícitamente la interacción entre los agentes y el entorno. Por ejemplo, en Lozano et al. (2022b) se analiza la colisión entre jugadores y obstáculos como un evento estratégico en el que el evasor podría obtener ventaja al cambiar la posición del perseguidor, perdiéndose así la línea de visión. Proponen una estrategia de movimiento libre de colisiones basada en una formulación matemática general de minimización-maximización para representar la interacción. Asimismo, para adaptar su propuesta al mundo real, incorporan una cámara con campo de visión limitado.

1.2.3. Solución del juego de persecución-evasión utilizando DRL

El Aprendizaje por Refuerzo (Reinforcement Learning, RL) se basa en la interacción entre un agente y su entorno, típicamente modelada mediante Procesos de Decisión de Markov (MDPs) o su variante parcialmente observable (POMDPs) como se puede observar en Sutton et al. (1998); Arulkumaran et al. (2017); Li (2017); Károly et al. (2020). El Aprendizaje por Refuerzo Profundo (Deep Reinforcement Learning, DRL) combina RL con redes neuronales profundas para aproximar políticas y funciones de valor en dominios de alta dimensionalidad. Existen tres principales enfoques algorítmicos: métodos basados en funciones de valor, métodos basados en políticas, y enfoques híbridos basados en la arquitectura Actor-Critic. Además, la interacción puede caracterizarse como libre del modelo del entorno (aprendizaje directo de la experiencia) o basados en el modelo del entorno (aprendizaje teniendo en cuenta un modelo

del entorno dado), siendo este último menos realista en muchas aplicaciones de robótica.

Abordar el problema de PE mediante técnicas de DRL constituye una línea de investigación emergente que ha mostrado avances significativos en los últimos años (Tai et al. (2016); Pierson & Gashler (2017)). Estos enfoques aprovechan la capacidad de las redes neuronales profundas para procesar representaciones complejas del entorno y aprender políticas de decisión directamente a partir de la experiencia, sin necesidad de un modelo explícito de la dinámica del sistema.

Egorov (2016) introduce una arquitectura Multi-Agent Deep Q-Network (MADQN) para entrenar un perseguidor mientras el evasor actúa siguiendo una estrategia heurística. Por su parte, Zhong et al. (2019) proponen un mecanismo de dueling asimétrico para el seguimiento visual activo (AD-VAT).

Un desafío específico del seguimiento visual es el cambio en la relación de aspecto (ARC) del objetivo en las imágenes. Zhang et al. (2018) abordan este problema con UAVs mediante un esquema coarse-to-fine. Este esquema utiliza como base una configuración de red perceptora como el modelo preentrenado VGG-M. La salida del modelo preentrenado tiene dos salidas de bloques de redes neuronales, uno para predecir las acciones del UAV y otro para refinar las acciones determinadas por el primer bloque.

Inspirado en la estrategia de caza cooperativa de los delfines, Ma et al. (2020) presentan una arquitectura multi-agente basada en Actor–Critic off-policy. Esta arquitectura se prueba en tres simulaciones que van incrementando en dificultad. Además, se diseña una función de recompensa que mitiga los problemas asociados con la formación circular de los agentes perseguidores.

En Qi et al. (2020), los obstáculos se modelan como objetos circulares. Se implementa un esquema de aprendizaje curricular (curriculum learning) con un enfoque DQN, en el cual el perseguidor primero aprende sin movimiento del evasor y luego ambos aprenden simultáneamente. El desempeño mejora para evasores con estrategias entrenadas, aunque decae con comportamientos aleatorios.

Jeong et al. (2021) proponen un algoritmo llamado Active Tracking Target Network (ATTN), un enfoque de DRL que permite resolver tres tareas simultáneamente: navegación, exploración y mantenimiento de visibilidad del objetivo. Comparan su rendimiento con el algoritmo Anytime Reduced Value Iteration (ARVI), obteniendo mejores resultados en términos de resiliencia al perder temporalmente al objetivo. La desventaja del algoritmo planteado es que no tiene en cuenta las características mecánicas del agente.

Tiritiris et al. (2021) desarrollan un entorno de simulación para el seguimiento activo, considerando configuraciones de control discretas y continuas. Se demuestra que es posible entrenar agentes directamente con imágenes RGB como entrada, usando robots reales (e-puck) en diferentes configuraciones. Además,

introducen una función de recompensa basada en diferencias temporales, incrementando la precisión del seguimiento bajo distintas condiciones experimentales.

En Serra-Gómez et al. (2023), el objetivo es que un dron recopile información semántica para clasificar múltiples objetivos en movimiento. Utilizan una arquitectura basada en atención entrenada mediante RL, cuya salida es la pose del dron para maximizar la adquisición de datos. Aunque el estudio propone una metodología innovadora, no detalla el algoritmo de percepción empleado, limitándose a mencionar el uso de un clasificador CNN. Su función de recompensa se basa en la entropía de confianza de clasificación, progreso en la clasificación de objetivos y penalizaciones por tiempo. El logro principal es que determina la política para el tiempo óptimo de clasificación de todos los objetivos, bajo las condiciones planteadas en el trabajo.

Gao et al. (2023) utilizan el algoritmo Soft Actor–Critic junto con sensores y visión panorámica para capturar un adversario simulado en un entorno tipo RoboMaster University AI (RMUA). Su contribución destaca por lograr una estabilización rápida de la función de recompensa usando la técnica Two Time Scale Update Rule (TTUR). Comparan su propuesta con el algoritmo Dueling DQN, demostrando este último que aunque converge más rápidamente, su rendimiento final es menos estable.

A pesar de estos avances, la investigación en DRL aplicada a PE presenta limitaciones relevantes. Una de ellas es la ausencia de criterios estandarizados para comparar distintos enfoques, lo cual dificulta una evaluación objetiva entre propuestas. Asimismo, preguntas clásicas como la cantidad óptima de perseguidores o la idoneidad de ciertos entornos para modelos específicos, rara vez son abordadas de forma sistemática. Otro aspecto identificado en la literatura es que la mayoría de las soluciones abordan solo un escenario específico, con configuraciones iniciales fijas de los agentes, lo que limita su capacidad de generalización. Además, muchos trabajos omiten restricciones físicas realistas, como las no holonómicas, el consumo energético o la percepción parcial del entorno, factores clave para una aplicación robusta en robótica móvil.

1.3. Planteamiento del problema

La meta en problemas de PE es encontrar una política de movimiento óptima[2], capaz de reaccionar tanto al comportamiento del evasor como a la geometría de los obstáculos.

[2]Una política de movimiento óptima es la mejor secuencia de acciones, dentro de un conjunto posible, que permite realizar una tarea de forma eficiente.

Representar y resolver este problema en entornos reales implica considerar tres factores claves: la geometría de los agentes y del ambiente, las limitaciones de sensado y comunicación entre los agentes, y las restricciones físicas y mecánicas de los agentes.

El uso de DRL en este tipo de escenarios permite integrar información sensorial diversa, proveniente de múltiples sensores o incluso de distintos agentes, lo que mejora la percepción del entorno y facilita una representación más precisa de obstáculos y objetivos. Esta capacidad resulta especialmente útil para generar políticas de control que permitan a los agentes operar de forma eficiente en entornos complejos.

Adicionalmente, el DRL permite aplicar técnicas de aprendizaje por transferencia (*transfer learning*), facilitando la adaptación de políticas aprendidas en un entorno específico a nuevas configuraciones sin requerir un reentrenamiento completo, lo cual resulta fundamental para entornos dinámicos o cambiantes.

A partir de estas consideraciones, se formula la siguiente pregunta de investigación:

¿Cómo generar políticas de movimiento para un robot móvil que busca mantener seguimiento de un evasor que se desplaza en un entorno con obstáculos utilizando técnicas de aprendizaje por refuerzo profundo?

1.4. Objetivos

En esta sección se presentan los objetivos generales y específicos definidos para guiar el desarrollo de este trabajo, orientados al diseño, implementación y evaluación de un algoritmo de Aprendizaje por Refuerzo Profundo aplicado a un escenario de persecución–evasión en entornos con obstáculos.

1.4.1. Objetivo general

Diseñar, implementar y evaluar un algoritmo de Aprendizaje por Refuerzo Profundo que utilice como datos de entrada las lecturas de los sensores de un robot móvil, mediante simulación, y genere como resultado una política de movimiento óptima para un juego de persecución-evasión en un entorno con obstáculos.

1.4.2. Objetivos específicos

- Obtener un conjunto de datos provenientes de un sensor de rango de un robot móvil (perseguidor) y de un sistema central de localización, en un entorno simulado que incluya la presencia de un agente evasor.
- Diseñar, implementar y entrenar una red neuronal que utilice los datos sensoriales como conjunto de entrenamiento para aprender una política de movimiento.
- Evaluar el desempeño del modelo entrenado en entornos con diferentes configuraciones geométricas y niveles de complejidad.

1.5. Contribuciones

Las principales contribuciones de este proyecto de tesis se pueden resumir en los siguientes puntos:

- Desarrollo de una librería que permite la simulación flexible y reproducible de escenarios de persecución-evasión en un entorno bidimensional, integrando agentes móviles y obstáculos estáticos, así como parámetros configurables de dificultad.
- Propuesta y validación de una metodología de navegación para un agente perseguidor en entornos 2D con obstáculos, empleando técnicas de aprendizaje por refuerzo profundo para la generación de políticas de movimiento.
- Formulación de una estrategia metodológica orientada a abordar el problema de la generalización en robótica móvil, utilizando aprendizaje por refuerzo profundo y un enfoque curricular que mejora la capacidad de adaptación del agente a distintos niveles de complejidad ambiental.
- Implementación de mecanismos de inteligencia para un agente perseguidor capaz de localizar, seguir y capturar a un agente evasor.
- Diseño de un sistema de recompensas que promueve la navegación libre de colisiones y, al mismo tiempo, guía al perseguidor hacia el objetivo final de captura.

Capítulo 2. Marco teórico

Este capítulo presenta una revisión de los conceptos teóricos, metodologías y herramientas tecnológicas que sustentan y respaldan el enfoque adoptado en esta investigación.

2.1. Clases de problemas en Aprendizaje de Máquina

El campo del *Aprendizaje de Máquina* abarca una amplia variedad de problemas que pueden ser clasificados según la naturaleza de los datos disponibles y el tipo de tarea que se desea resolver,(Mitchell (1997)). En esta sección se describen las principales clases de problemas en aprendizaje automático: *aprendizaje supervisado*, *aprendizaje no supervisado*, y *aprendizaje por refuerzo*.

2.1.1. Aprendizaje supervisado

El aprendizaje supervisado se refiere a un conjunto de técnicas en las que se dispone de un conjunto de datos etiquetado, es decir, cada entrada está asociada a una salida deseada (conocida como **valor verdadero**). El objetivo principal es inferir una función que modele la relación entre las entradas y sus correspondientes salidas, permitiendo así predecir etiquetas para nuevos datos no vistos.

$$\hat{y} = f(x; \theta). \tag{1}$$

Donde x son las características o entrada, $\hat{y}$ es la etiqueta esperada o salida, θ son los parámetros del modelo o función, y $f(x; \theta)$ es la función que aproxima la relación entre la entrada y salida, parametrizada por θ.

2.1.2. Aprendizaje no supervisado

En el aprendizaje no supervisado, el conjunto de datos carece de etiquetas explícitas. El objetivo es descubrir patrones o estructuras subyacentes en los datos que permitan agruparlos o reducir su dimen-

sionalidad. Este proceso facilita la identificación de similitudes y relaciones entre los datos. Ejemplos comunes incluyen algoritmos de agrupamiento y reducción de dimensionalidad como PCA o t-SNE.

$$\hat{f}(x;\theta) = z. \tag{2}$$

Donde x son los datos de entrada sin etiquetar, θ son los parámetros del modelo o función, $\hat{f}(x;\theta)$ es la representación aprendida de x, y z es la estructura subyacente aprendida.

2.1.3. Aprendizaje por refuerzo

El aprendizaje por refuerzo (Reinforcement Learning, RL en inglés) se distingue por su enfoque en la toma de decisiones secuenciales (Matsuo et al. (2022); Sutton et al. (1998)). En este paradigma, un agente interactúa con un entorno a lo largo del tiempo, tomando decisiones (acciones) basadas en observaciones (estados) y recibiendo retroalimentación en forma de recompensas o penalidades. El objetivo del agente es aprender una política que maximice la recompensa acumulada a largo plazo.

$$\hat{f}(s;\theta) = \pi(s;\theta) = a. \tag{3}$$

Donde s es el estado del agente o entrada, θ son los parámetros del modelo o función, a es la acción predicha por el modelo, y $\hat{f}(s;\theta)$ es representado usualmente como $\pi(s;\theta)$ donde π es la política que describe el comportamiento del agente.

Una representación típica en RL involucra los siguientes elementos fundamentales, (Matsuo et al. (2022); Sutton et al. (1998)):

- **Agente:** La entidad que toma decisiones.
- **Entorno:** Todo lo externo al agente con lo que este interactúa.
- **Estado:** Una representación de la situación actual del entorno.
- **Acción:** Una decisión tomada por el agente que afecta al entorno.
- **Recompensa:** Una señal que el agente recibe como consecuencia de sus acciones.

El proceso de aprendizaje se basa en la interacción entre estos componentes y se describe generalmente mediante los siguientes pasos:

1. **Inicialización del agente:** Configurar los parámetros iniciales y preparar al agente para la interacción.
2. **Observación del entorno:** Capturar el estado actual para ser procesado por el agente.
3. **Selección de una acción:** Elegir una acción óptima basada en la estrategia actual del agente.
4. **Ejecución de la acción:** Interactuar con el entorno mediante la acción seleccionada.
5. **Recepción de la recompensa y nuevo estado:** Obtener retroalimentación del entorno y observar el nuevo estado alcanzado.
6. **Actualización de la política:** Utilizar la recompensa y el nuevo estado para ajustar la política o estrategia del agente.
7. **Iteración:** Repetir los pasos del 2 al 6 hasta alcanzar una política óptima o satisfacer un criterio de convergencia.

En la Figura 1 se visualiza el ciclo de aprendizaje descrito anteriormente.

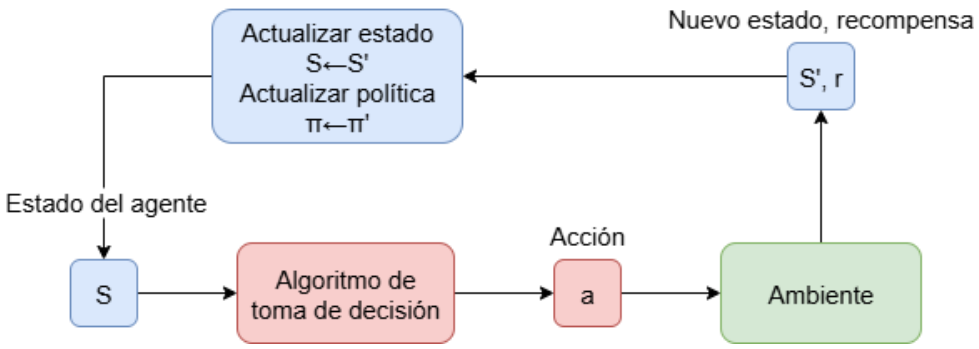

Figura 2. Ciclo de aprendizaje utilizando aprendizaje por refuerzo, adaptado de Sutton et al. (1998)

2.2. Elementos del aprendizaje por refuerzo

A continuación, se describen los principales componentes involucrados en el aprendizaje por refuerzo, junto con los conceptos fundamentales y estrategias que deben considerarse al analizar o diseñar algoritmos dentro de este paradigma.

2.2.1. Proceso de Decisión de Markov

El Proceso de Decisión de Markov (Markov Decision Process, MDP en inglés) es el marco matemático común para representar el proceso de toma de decisiones secuenciales de un agente, (Xiang & Foo (2021)). El MDP permite darle importancia no solo a los estados y recompensa inmediatas, sino a los estados y recompensas en el futuro como consecuencia de aplicar ciertas acciones. Como consecuencia, surge una relación de compromiso entre obtener una recompensa inmediata beneficiosa y el valor de la recompensa futura a partir de una acción que maximizó la recompensa inmediata.

Un MDP se define formalmente como una tupla $(\mathcal{S}, \mathcal{A}, P, R)$ donde:

- $\mathcal{S}$ es el conjunto de estados posibles del entorno.
- $\mathcal{A}$ es el conjunto de acciones disponibles para el agente.
- $P(s'|s, a)$ es la función de transición de estados.
- $R(s, a)$ es la función de recompensa.

El agente comienza en un estado inicial s_0 y basado en una estrategia o política selecciona una acción a_0. Luego de ejecutar la acción, recibe una recompensa r_1 y transiciona al siguiente estado s_1, utilizando la función $R(s, a)$ y $P(s'|s, a)$, respectivamente. Por último, repite este ciclo de manera iterativa hasta aprender una política óptima. La secuencia de interacción entre agente y entorno se representa como:

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, \ldots \tag{4}$$

2.2.2. Procesos de Decisión de Markov parcialmente observables

En muchos escenarios reales, el agente no puede observar completamente el estado del entorno. En estos casos, se utiliza el marco de los Procesos de Decisión de Markov Parcialmente Observables (Partially Observable Markov Decision Process, POMDP en inglés)(Xiang & Foo (2021)), que extiende el MDP incorporando incertidumbre sobre el estado actual.

Un escenario representativo de este tipo de incertidumbre es la robótica móvil, donde el agente (el robot) no tiene acceso directo al estado real del entorno. En su lugar, el robot dispone únicamente de lecturas provenientes de sensores (por ejemplo, cámaras, LIDAR, entre otros), las cuales actúan como observaciones parciales del estado verdadero. Esta representación parcial introduce ambigüedad en la percepción del entorno, lo que a su vez puede generar incertidumbre en la selección de acciones. A pesar de esta limitación, es posible aprender políticas subóptimas que permitan al agente completar satisfactoriamente la tarea.

Un POMDP se define como una tupla $(\mathcal{S}, \mathcal{A}, \Omega, P, R, O)$ donde:

- $\mathcal{S}$ es el conjunto de estados del entorno.
- $\mathcal{A}$ es el conjunto de acciones disponibles para el agente.
- $\Omega(s)$ es el conjunto de observaciones proveniente de los sensores.
- $P(s'|s, a)$ es la función de transición de estados.
- $R(s, a)$ es la función de recompensa.
- $O(o|s, a)$ es la probabilidad de observar o tras tomar una acción a y llegar al estado s.

2.2.3. Función de valor Q

La función Q, también llamada función de valor estado-acción, denotada como $Q(s, a)$, evalúa qué tan favorable es ejecutar una acción a en el estado s y seguir una política dada posteriormente. Los algoritmos de RL tradicionales se enfocan en elaborar una tabla de valores Q mediante una búsqueda exhaustiva para cada par (s, a). Sin embargo, este enfoque solo es práctico para problemas con baja dimensionalidad, es decir pocos estados y acciones.

Para extender a problemas con alta dimensionalidad debemos aproximar la función Q utilizando parametrización. Una de las soluciones para este desafío es utilizar redes neuronales donde sus pesos θ sirven como parámetros para aproximar la función Q. Comúnmente, se conoce a estas redes como *redes Q*.

Para entender como funciona la función Q primero se debe introducir el concepto de recompensa acumulada para el agente (Sutton et al. (1998)). La recompensa acumulada desde el tiempo t, denotada como R_t, se define como:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n r_{t+n}. \tag{5}$$

Donde $\gamma \in [0, 1)$ es el factor de descuento que determina la importancia relativa de las recompensas futuras. Generalmente, se escoge un valor cercano a 1 para considerar recompensas a largo plazo, sin darles un peso excesivo.

Como puede observarse en la ecuación 5, a medida que n aumenta, el impacto de las recompensas futuras se atenúa exponencialmente debido al efecto del factor de descuento. Esta ponderación es fundamental, ya que decisiones con efectos inciertos en el futuro deben influir menos en la toma de decisiones actuales del agente.

Finalmente, el valor de Q puede interpretarse como el valor esperado de la recompensa acumulada al ejecutar la acción a_t en el estado s_t y seguir una política dada:

$$Q(s_t, a_t) = \mathbb{E}[R_t \mid s_t, a_t]. \tag{6}$$

2.2.4. Política óptima

El objetivo fundamental del agente en el aprendizaje por refuerzo es aprender una política óptima, denotada como π^*, que maximice la recompensa acumulada esperada a lo largo del tiempo. Dicha política óptima debe seleccionar, para cada estado s, la acción a que maximice la función de valor estado-acción $Q(s, a)$:

$$\pi^*(s) = \arg\max_a Q(s, a). \tag{7}$$

Es decir, la política óptima prescribe la acción que proporciona la mayor recompensa esperada considerando tanto la recompensa inmediata como las futuras, bajo el supuesto de que se seguirá actuando de manera óptima en adelante.

Nota: $\pi(s)$ se refiere a una política arbitraria o aún no optimizada, mientras que $\pi^*(s)$ representa la política óptima que maximiza el desempeño del agente.

2.2.5. Reproducción de experiencias

El búfer de reproducción de experiencias, representa la memoria donde se almacenan las transiciones de la forma $\langle s, a, r, s' \rangle$ durante la interacción del agente con el entorno. Estas transiciones se reutilizan para entrenar el modelo, lo cual rompe la correlación temporal entre muestras consecutivas y mejora la estabilidad del aprendizaje.

Sin esta técnica, entrenar directamente sobre secuencias correlacionadas puede inducir sobreajuste y dificultar la convergencia del aprendizaje. Para mitigar este problema, se emplea una estrategia de muestreo aleatorio uniforme sobre el búfer.

2.2.5.1. Reproducción de experiencias priorizadas

Esta técnica propone priorizar las experiencias almacenadas en función de su error de predicción, dando mayor probabilidad de muestreo a aquellas transiciones que se consideran más informativas. Esta técnica mejora la eficiencia del entrenamiento al centrarse en ejemplos con mayor potencial de aprendizaje (Saglam et al. (2023)).

En la práctica, el entrenamiento suele iniciar con una estrategia de muestreo uniforme similar a la empleada en la *reproducción de experiencias*, con el fin de garantizar una exploración adecuada del espacio de observaciones. A medida que avanza el entrenamiento, se incrementa progresivamente el grado de priorización, favoreciendo la explotación de transiciones que aportan mayor valor para acelerar la convergencia hacia una política óptima.

2.2.6. Diseño de la función de recompensa

La función de recompensa proporciona retroalimentación al agente sobre el valor de sus acciones. Es un componente crucial en el diseño de sistemas de RL, ya que define el objetivo que se desea optimizar. Sin embargo, diseñarla correctamente puede resultar complejo y conlleva diversos desafíos.

En tareas de persecución-evasión en entornos con obstáculos si la función de recompensa penaliza o bonifica únicamente la distancia entre el perseguidor y el evasor, el agente puede quedar atrapado entre los obstáculos o colisionar con ellos sin aprender a evitarlos. Esta situación es debido a la falta de retroalimentación específica en esos estados.

Además, una asignación inadecuada de recompensas y penalidades puede inducir comportamientos no deseados. Por ejemplo, si tenemos un problema que finaliza al colisionar el agente y una función de recompensa donde se penaliza con -1 cada paso y se otorga $+1$ al alcanzar el objetivo. El agente podría preferir finalizar el episodio rápidamente colisionando, debido a que es mayor la penalidad por permanecer en el juego que alcanzar el objetivo. En este caso el agente estaría aprendiendo a maximizar la recompensa minimizando la penalidad para cada episodio que juega.

Por ello, la función de recompensa debe balancear objetivos múltiples. En muchos casos, es necesario incluir componentes adicionales que favorezcan la exploración y se alineen los comportamientos del agente con la intención del diseñador.

2.2.7. Aprendizaje curricular

El *aprendizaje curricular (curriculum learning)*, como lo define Narvekar et al. (2020), consiste en descomponer tareas complejas en una secuencia de sub-tareas de creciente dificultad. Como se ilustra en la Figura 2, esta estrategia permite que el agente comience su entrenamiento resolviendo tareas simples y, progresivamente, utilice el conocimiento adquirido para abordar tareas cada vez más complejas.

Este enfoque ha demostrado mejorar significativamente la eficiencia del aprendizaje, facilitar la generalización y acelerar la convergencia del agente hacia una política efectiva, al reducir la probabilidad de estancarse en soluciones subóptimas y favorecer un proceso de entrenamiento más estable. El diseño de un *curriculum* puede implicar modificaciones progresivas en diversos aspectos del entorno, como la complejidad de las tareas, la distribución de los estados iniciales o la forma de la función de recompensa.

Estas adaptaciones se implementan con el objetivo de guiar el proceso de entrenamiento, permitiendo que el agente adquiera gradualmente las habilidades necesarias para resolver tareas más complejas (Wang et al. (2021); Ma et al. (2022)).

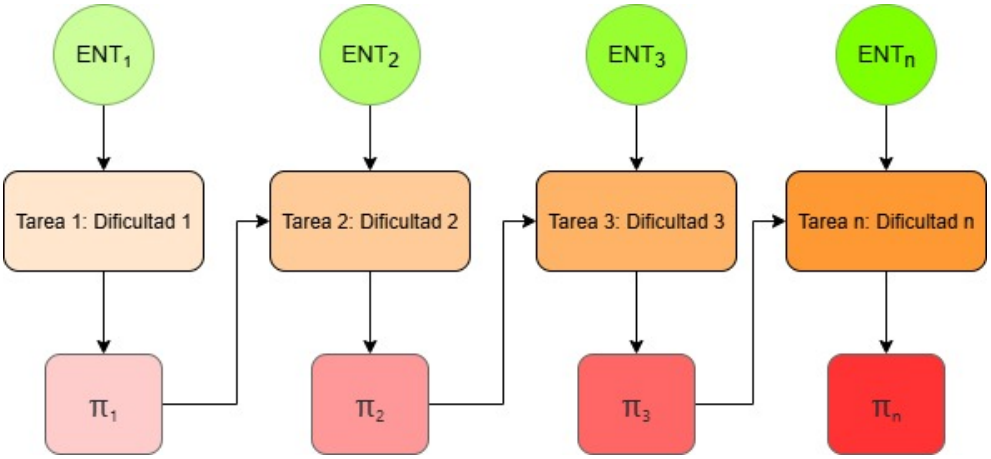

Figura 3. Representación del progreso de aprendizaje utilizando aprendizaje curricular.

2.3. Enfoques de solución de problemas utilizando DRL

Los algoritmos de Aprendizaje por Refuerzo Profundo (*Deep Reinforcement Learning*, DRL en inglés) pueden clasificarse en distintas categorías según el tipo de función que aprenden o el enfoque utilizado para optimizar el comportamiento del agente. Entre los principales enfoques se encuentran el aprendizaje basado en valores, el aprendizaje basado en políticas y los métodos híbridos.

Cada uno de estos enfoques presenta ventajas y limitaciones, así como diferentes niveles de aplicabilidad dependiendo de la naturaleza del problema a resolver. Mientras que algunos resultan más adecuados para entornos discretos con dinámicas relativamente simples, otros permiten abordar espacios continuos de gran dimensionalidad o problemas que requieren una exploración más sofisticada.

En esta sección se describen los distintos enfoques, sus fundamentos teóricos y su aplicabilidad a diferentes tipos de problemas.

2.3.1. Aprendizaje basado en valores

El objetivo principal en el aprendizaje basado en valores es estimar funciones que asignan un valor esperado a los estados o a las combinaciones estado-acción, bajo una determinada política. A partir de estas estimaciones, el agente puede derivar una política que seleccione las acciones que conducen a los estados más prometedores, priorizando aquellas que maximizan la recompensa acumulada a largo plazo. A continuación, se describen en detalle los métodos más representativos de este enfoque.

2.3.1.1. Deep Q Network (DQN)

Uno de los enfoques más representativos para la resolución de problemas mediante Aprendizaje por Refuerzo Profundo (DRL) es el algoritmo *Deep Q-Network* (DQN), una extensión del algoritmo Q-learning clásico que utiliza redes neuronales profundas para aproximar la función de valor de estado-acción $Q(s, a)$ (Mnih et al. (2015)).

La regla de actualización de Q-learning, según Sutton et al. (1998), es la siguiente:

$$Q^{nueva}(s, a) = Q(s, a) + \alpha * (r + \gamma \max_{a'} Q(s', a') - Q(s, a)). \tag{8}$$

En esta ecuación, el término $r + \gamma \max_{a'} Q(s', a')$ representa el *valor objetivo*, es decir, una estimación del retorno futuro esperado al tomar la acción a en el estado s y seguir una política determinada a partir del siguiente estado s'. El valor actual $Q(s, a)$ se actualiza moviéndose hacia el valor objetivo, ponderado por una tasa de aprendizaje α.

En DQN, la función $Q(s, a)$ se aproxima mediante una red neuronal parametrizada por θ, que toma como entrada el estado s y devuelve una estimación de los valores Q para todas las acciones posibles. Para entrenar esta red, se utiliza una función de pérdida basada en el error cuadrático medio entre la predicción actual y el valor objetivo:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} (y_i - Q(s_i, a_i; \theta))^2, \quad i = 1, 2, \ldots, N. \tag{9}$$

donde θ representa los parámetros de la red Q que se están entrenando, i representa el índice de la muestra tomada del buffer de experiencia, N es el tamaño de la muestra, y el valor objetivo y_i se define como:

$$y_i = r_i + \gamma \max_{a'} Q(s'_i, a'_i; \theta). \tag{10}$$

En la Figura 3 se muestra un esquema del funcionamiento general del algoritmo DQN.

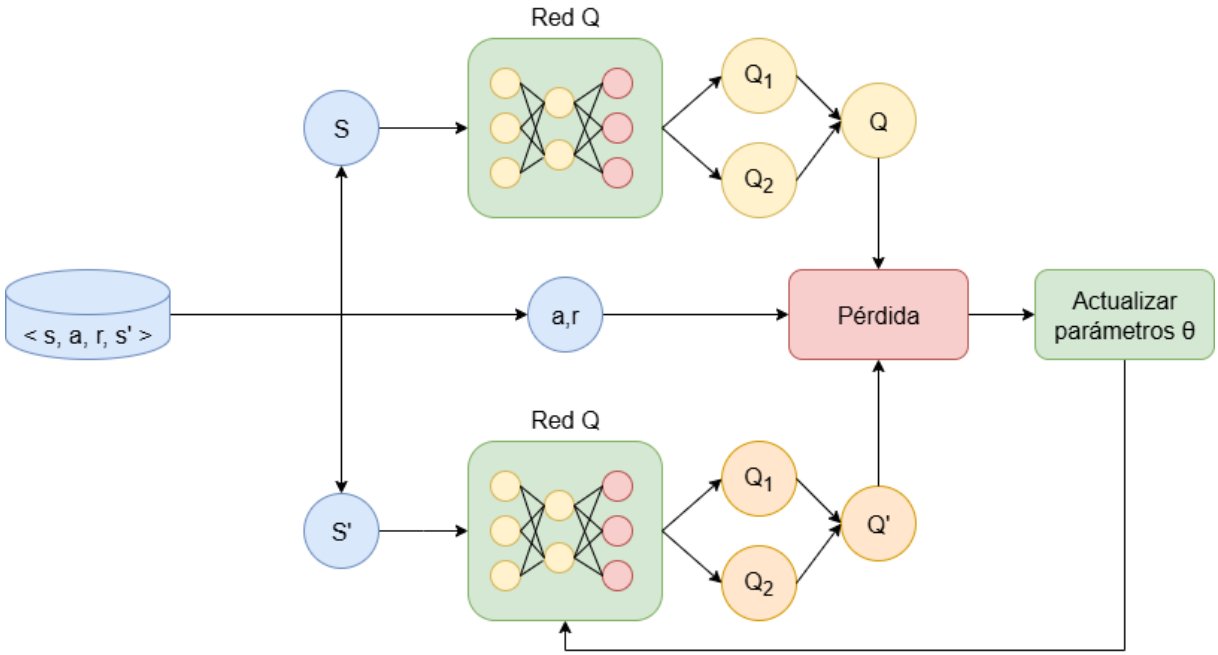

Figura 4. Esquema general del algoritmo DQN, adaptado de Mnih et al. (2015). La pérdida utilizada se plantea en la ecuación 9.

En la configuración anterior, tanto la estimación del valor actual como del valor objetivo se obtienen a partir de la misma red neuronal, utilizando los mismos parámetros θ. Este enfoque, aunque funcional, puede generar inestabilidad en el entrenamiento debido a que se actualizan ambos términos simultáneamente.

Para mitigar este problema, se introduce una segunda red denominada *red objetivo* (Q'), que se utiliza exclusivamente para calcular el valor objetivo. Esta red mantiene sus pesos congelados durante un número determinado de pasos de entrenamiento, tras los cuales se actualiza con los pesos actuales de la red Q ($\theta' \leftarrow \theta$). En la Figura 4 se muestra un esquema del funcionamiento general del algoritmo DQN utilizando una red objetivo.

La función de pérdida se modifica de la siguiente forma:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(r_i + \gamma * max_{a_i'}Q(s_i', a_i'; \theta') - Q(s_i, a_i; \theta))^2. \tag{11}$$

donde θ' son los parámetros de la red objetivo, i representa el índice de la muestra tomada del buffer de experiencia, y N es el tamaño de la muestra.

Esta técnica introduce una fuente de estabilidad al proceso de entrenamiento, al desacoplar temporalmente el valor objetivo de los cambios inmediatos en los pesos de la red Q. Como resultado, se mejora la convergencia y se reducen las oscilaciones no deseadas en la función de valor estimada.

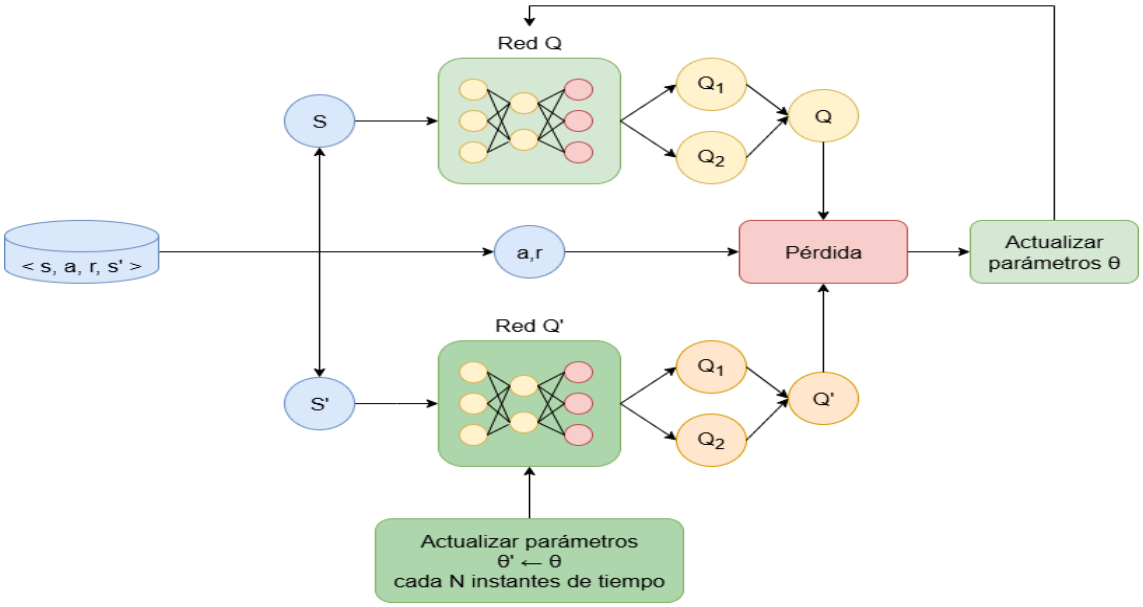

Figura 5. Esquema general del algoritmo DQN con Red Objetivo, adaptado de Mnih et al. (2015).

Algorithm 1 Pseudocódigo Algoritmo Deep Q-Network (DQN), adaptado de Mnih et al. (2015)

Inicializar la red Q con pesos aleatorios θ
Inicializar la red target Q' con pesos $\theta' = \theta$
Inicializar el buffer de experiencia $\mathcal{D}$
para episodio $= 1$ hasta M **hacer**
    Inicializar el estado s
    para t $= 1$ hasta fin del episodio **hacer**
        Con probabilidad ϵ seleccionar una acción a aleatoria
        De lo contrario, seleccionar $a = \arg\max_a Q(s, a; \theta)$
        Ejecutar la acción a, observar recompensa r y siguiente estado s'
        Almacenar transición (s, a, r, s') en $\mathcal{D}$
        Muestrear minibatch de transiciones aleatorias (s_j, a_j, r_j, s'_j) de $\mathcal{D}$
        Calcular $y_j = r_j + \gamma \cdot \max_{a'} Q(s'_j, a'; \theta')$
        Actualizar θ minimizando la pérdida:

$$\mathcal{L} = \frac{1}{N} \sum_j (y_j - Q(s_j, a_j; \theta))^2.$$

        $s \leftarrow s'$
        si cada C pasos **entonces**
            Actualizar red target: $\theta' \leftarrow \theta$
        fin si
    fin para
fin para

2.3.1.2. Double Deep Q Network (DDQN)

El algoritmo DQN ha demostrado ser efectivo en múltiples tareas; sin embargo, tiende a sobreestimar los valores de la función Q, lo cual afecta negativamente su rendimiento en entornos complejos. Este problema ocurre principalmente debido al uso del operador *max* en la ecuación de actualización de Q, donde la misma red Q es utilizada tanto para seleccionar como para evaluar las acciones. Esta doble utilización puede amplificar el efecto del ruido en las predicciones de la red.

Es común que las redes neuronales produzcan estimaciones ruidosas, especialmente en las etapas iniciales del entrenamiento. En tareas con espacios de estados y acciones grandes, como las que se presentan en robótica móvil, este ruido puede conducir al aprendizaje de políticas subóptimas o incluso a la divergencia del proceso de entrenamiento.

Por ejemplo, supongamos que para un estado dado s existen cinco acciones posibles: a_1, a_2, a_3, a_4, a_5, donde la acción óptima es a_5. A causa del ruido en la estimación de la red Q, esta podría predecir un mayor valor para la acción a_2, haciendo que el operador *max* seleccione una acción subóptima. En aplicaciones como la robótica móvil, donde una mala decisión puede resultar en trayectorias ineficientes o colisiones, este tipo de sobreestimación es crítico.

El algoritmo **Double DQN** propone una solución a este problema mediante el uso de dos redes Q independientes,(Van Hasselt et al. (2016)). Una red se utiliza para seleccionar la acción (mediante el operador *argmax*) y la otra para evaluarla. Esta separación reduce el sesgo en la estimación del valor objetivo.

La ecuación de actualización para Double DQN se define como:

$$y_i^{DoubleDQN} = r_i + \gamma * Q(s_i, argmaxQ(s_i', a_i'; \theta); \theta'). \tag{12}$$

Aquí, se tiene dos funciones Q con diferentes pesos. La primera con pesos θ es usada para la selección de la acción. La otra, con pesos θ' para la evaluación de la acción. En la Figura 5 se muestra un esquema del funcionamiento general del algoritmo DDQN.

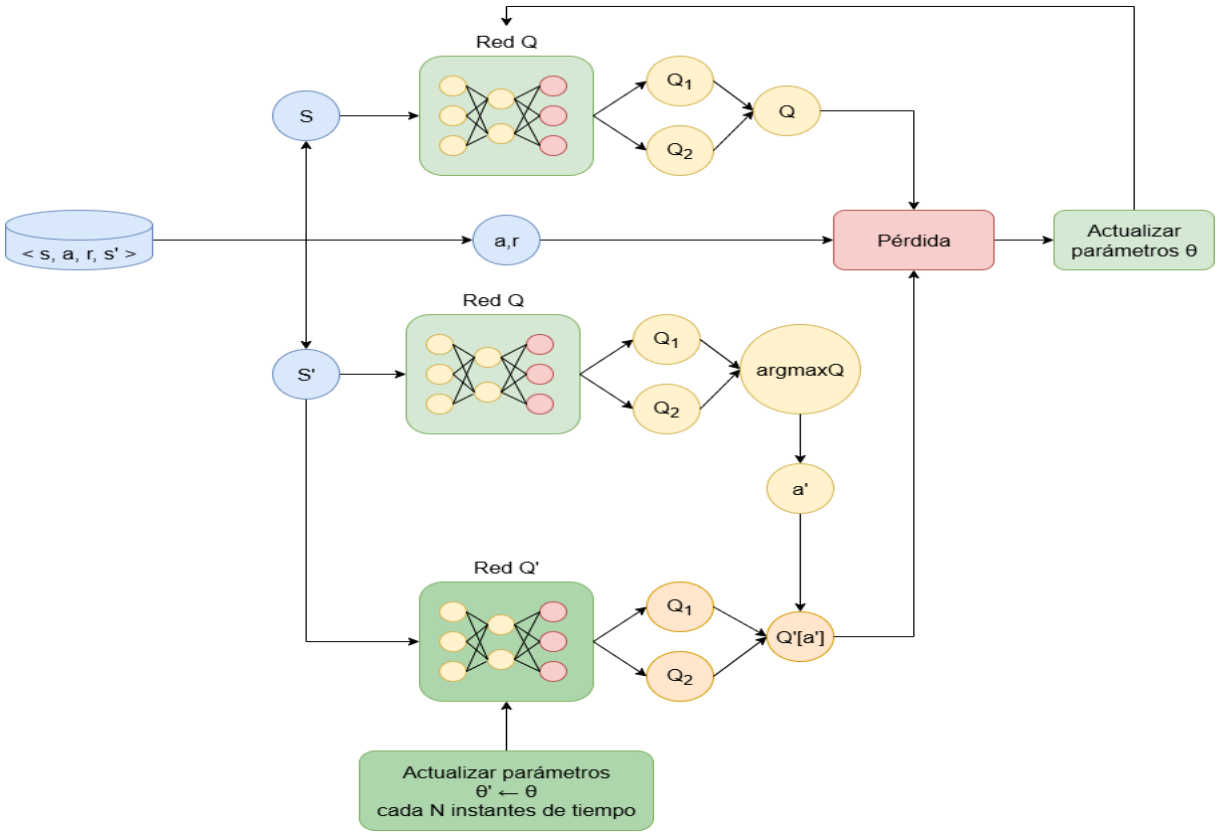

Figura 6. Esquema general del algoritmo DDQN, adaptado de Van Hasselt et al. (2016). La pérdida utilizada se plantea en la ecuación 11 y se utiliza la ecuación de actualización 12.

Algorithm 2 Pseudocódigo Algoritmo Double Deep Q-Network (DDQN), adaptado de Van Hasselt et al. (2016)

Inicializar la red Q con pesos θ
Inicializar la red target Q' con pesos $\theta^- = \theta$
Inicializar el buffer de experiencia $\mathcal{D}$
para episodio $= 1$ hasta M **hacer**
    Inicializar el estado inicial s
    para t $= 1$ hasta fin del episodio **hacer**
        Seleccionar una acción a usando una política ϵ-greedy basada en $Q(s, a; \theta)$
        Ejecutar acción a, observar recompensa r y nuevo estado s'
        Almacenar transición (s, a, r, s') en $\mathcal{D}$
        Muestrear minibatch aleatorio de transiciones (s_j, a_j, r_j, s'_j) desde $\mathcal{D}$
        para cada transición en el minibatch **hacer**
            $a^* \leftarrow \arg\max_{a'} Q(s'_j, a'; \theta)$ ▷ Selección con red Q
            $y_j \leftarrow r_j + \gamma \cdot Q(s'_j, a^*; \theta^-)$ ▷ Evaluación con red target (Q')
        fin para
        Actualizar θ minimizando la pérdida:

$$\mathcal{L} = \frac{1}{N} \sum_j \left(y_j - Q(s_j, a_j; \theta)\right)^2 .$$

        Cada C pasos, actualizar la red target: $\theta^- \leftarrow \theta$
        $s \leftarrow s'$
    fin para
fin para

2.3.1.3. Dueling Deep Q Network (D3QN)

La idea detrás de la arquitectura *Dueling* es descomponer la función de valor-acción $Q(s, a)$ en dos componentes separados (Wang et al. (2016)). El primero, la función de valor $V(s)$, que estima el valor de estar en un estado dado. El segundo, la función de ventaja $A(s, a)$, que estima la utilidad relativa de ejecutar una acción a en un estado s comparado con otras acciones posibles en el mismo estado. Esta descomposición se define formalmente como:

$$Q(s, a) = V(s) + A(a). \tag{13}$$

Sin embargo, esta descomposición no es única, ya que se puede sumar o restar constantes arbitrarias sin cambiar el resultado de Q. Para hacer el aprendizaje estable, se propone una forma específica de combinar estas dos funciones dentro de la red, restando el promedio (o el máximo) de las ventajas para normalizar la función $A(s, a)$. La versión más comúnmente utilizada es:

$$Q(s, a) = V(s) + \left(A(s, a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a') \right). \tag{14}$$

Esta formulación garantiza que la estimación de la función Q no sea afectada por la arbitrariedad de $A(s, a)$ y evita la sobreestimación de valores.

La arquitectura *Dueling* representa una modificación estructural respecto a las arquitecturas utilizadas en los algoritmos anteriores, como DQN y DDQN. Es importante destacar que esta arquitectura puede ser utilizada en conjunto tanto con el esquema de aprendizaje DQN como con el enfoque DDQN, ya que lo que se modifica es la topología de la red y no el algoritmo de actualización en sí. En la Figura 6 se muestra la topología general de la arquitectura Dueling.

La motivación para utilizar esta arquitectura surge en entornos donde algunas acciones no tienen un efecto significativo en el resultado. En esos casos, aprender una función $Q(s, a)$ directamente puede ser ineficiente, mientras que separar el valor del estado del impacto relativo de cada acción permite una representación más robusta.

La arquitectura de una *Dueling DQN* es similar a una DQN tradicional hasta las últimas capas. Luego de las capas convolucionales o densas compartidas, la red se bifurca en dos flujos separados: uno para

estimar $V(s)$ y otro para estimar $A(s, a)$. Finalmente, ambos se combinan como en la ecuación anterior para producir la salida final de la red Q.

Esta arquitectura ha mostrado mejoras significativas en la precisión y estabilidad del entrenamiento, especialmente en tareas donde no todas las acciones son igualmente relevantes.

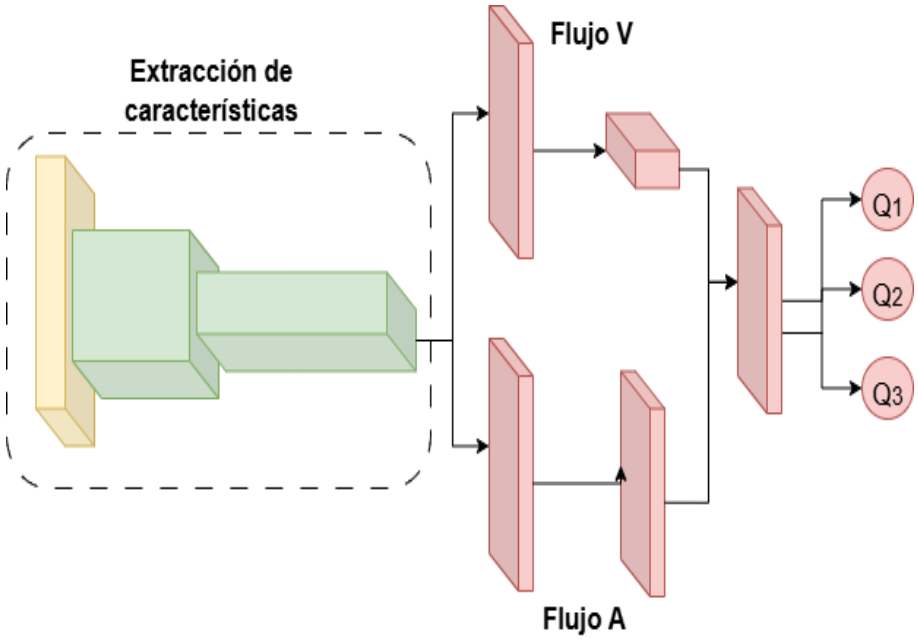

Figura 7. Arquitectura Dueling, adaptado de Wang et al. (2016).

2.3.2. Aprendizaje basado en políticas

A diferencia del enfoque basado en valores, el aprendizaje basado en políticas busca optimizar directamente la política que guía el comportamiento del agente, sin necesidad de estimar funciones de valor intermedias. Este enfoque resulta especialmente útil en entornos con espacios de acción continuos o donde la política óptima no puede derivarse fácilmente a partir de una función Q.

2.3.2.1. Algoritmo Reinforce

El algoritmo **REINFORCE** es uno de los métodos fundamentales dentro del aprendizaje por refuerzo basado en políticas (AlMahamid & Grolinger (2021)). Su objetivo principal es aprender directamente

una política estocástica $\pi_\theta(a|s)$, parametrizada mediante una red neuronal, optimizando los parámetros θ para maximizar la recompensa acumulada esperada.

A diferencia de los métodos basados en valores, REINFORCE no intenta estimar explícitamente ninguna función de valor (V o Q), sino que actualiza los parámetros de la política directamente, utilizando muestras obtenidas a partir de trayectorias completas generadas por la interacción del agente con el entorno.

Actualización de la política. La actualización de los parámetros se basa en el gradiente de la esperanza de la recompensa total respecto de θ:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) \cdot R \right]. \tag{15}$$

donde R es el retorno total acumulado a partir del estado s tras ejecutar la acción a.

En la práctica, este gradiente se estima mediante muestreo a lo largo de un episodio completo. Dado un conjunto de pares (s_t, a_t) obtenidos durante una trayectoria, la política se actualiza utilizando la siguiente regla:

$$\theta \leftarrow \theta + \alpha \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot R_t. \tag{16}$$

donde $R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$ representa el retorno descontado desde el paso t, α es la tasa de aprendizaje, y γ es el factor de descuento que controla la importancia de las recompensas futuras.

Ventajas:

- Es un método completamente *Monte Carlo*, por lo que no requiere estimaciones de valor ni modelado del entorno.
- Es adecuado para entornos con espacios de acción continuos o de alta dimensionalidad.
- Introduce aleatoriedad en la selección de las acciones. Esto provoca una estrategia de exploración sin la necesidad de introducir un algoritmo adicional.

Desventajas:

- Alta varianza en las estimaciones del gradiente, lo cual puede dificultar la estabilidad y velocidad del aprendizaje.
- Requiere completar episodios para actualizar la política, lo que puede ser costoso en entornos largos o complejos.
- Poca eficiencia en el uso de muestras, ya que cada interacción se utiliza solo una vez para actualizar la política.

Con el objetivo de reducir la varianza del gradiente y estabilizar el proceso de entrenamiento, se han propuesto extensiones como la incorporación de la función de valor $V(s)$. Esta incorporación actúa como una referencia para evaluar si una acción fue mejor o peor de lo esperado, y da lugar a algoritmos más sofisticados como los métodos *Actor-Critic*, que combinan el aprendizaje de políticas con el aprendizaje de valores.

Algorithm 3 Algoritmo REINFORCE, adaptado de AlMahamid & Grolinger (2021)

Inicializar la red con pesos θ
para cada episodio **hacer**
    Inicializar el estado inicial s_0
    Inicializar la trayectoria $\mathcal{T} \leftarrow []$
    para cada paso $t = 0, 1, \ldots$ hasta que el episodio termine **hacer**
        Seleccionar acción $a_t \sim \pi_\theta(a_t|s_t)$
        Ejecutar a_t, observar recompensa r_t y nuevo estado s_{t+1}
        Almacenar transición (s_t, a_t, r_t) en $\mathcal{T}$
        Actualizar $s_t \leftarrow s_{t+1}$
    fin para
    para cada transición (s_t, a_t, r_t) en $\mathcal{T}$ **hacer**
        Calcular el retorno descontado: $R_t \leftarrow \sum_{k=t}^{T} \gamma^{k-t} r_k$
        Calcular gradiente: $\nabla_\theta \log \pi_\theta(a_t|s_t)$
        Actualizar los parámetros: $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot R_t$
    fin para
fin para

2.3.3. Aprendizaje basado en métodos híbridos

El enfoque *basado en métodos híbridos*, busca combinar las ventajas de los métodos basados en valores y los métodos basados en políticas. Mientras que los métodos de valor son más estables pero pueden

carecer de flexibilidad en espacios de acción continuos, los métodos basados en políticas ofrecen mayor capacidad de representación pero suelen presentar una alta varianza en sus actualizaciones.

Al integrar ambos enfoques, se proporciona una señal de evaluación que guía el entrenamiento de la red que estima la política. Este marco híbrido permite actualizar la política de forma más eficiente y estable, reduciendo la varianza de los gradientes y acelerando la convergencia. En esta sección se presentan los principales algoritmos dentro de este paradigma.

2.3.3.1. Método Advantage Actor-Critic (A2C)

El término **Advantage** o ventaja hace referencia al uso de la función de ventaja $A(s, a)$. Esta función mide qué tan mejor o peor resulta una acción específica respecto al valor promedio esperado en ese estado $V(s)$. De esta manera, la ventaja indica si la acción tomada fue más valiosa $A > 0$ o menos valiosa $A < 0$ que el comportamiento promedio de la política en dicho estado.

La arquitectura **Actor-Crítico** se compone de dos módulos principales (Kumar et al. (2021)):

- **Actor**: es la red responsable de aprender la política $\pi(a|s)$, es decir, la distribución de probabilidad sobre las acciones dadas las observaciones del entorno.
- **Crítico**: es la red encargada de estimar la función de valor $V(s)$, la cual evalúa la calidad del estado y sirve como retroalimentación para la calidad de las predicciones del **Actor**.

Actualización de la política (red Actor). El actor ajusta los parámetros de la política $\pi(a|s;\theta)$, ponderado por la función de ventaja $A(s, a)$. El objetivo es aumentar la probabilidad de ejecutar acciones que resultaron ser mejores de lo esperado. La actualización se realiza según la siguiente expresión:

$$PolicyLoss(L_p) = log(\pi(a|s)) * A(s, a) * \beta H(\pi). \tag{17}$$

Donde $\pi(a|s)$ es la distribución de probabilidad de las acciones dado un estado s. $H(\pi)$ es el término de entropía que incentiva la exploración. El coeficiente β regula la importancia de la entropía. Valores altos de entropía implican que las acciones tienen probabilidades similares (usualmente en la etapa de

exploración), mientras que entropía baja sugiere convergencia hacia una acción dominante (etapa de explotación de la política). $A(s,a)$ es la ventaja estimada y se calcula como:

$$A(s,a) = R - V(s). \tag{18}$$

Donde R es el retorno descontado de la recompensa para cada estado. $V(s)$ la estimación del valor para el estado s.

Actualización del valor (red Crítico). La red critic aprende a estimar la función de valor $V(s;\phi)$ mediante un proceso de regresión. El objetivo es minimizar el error entre el valor estimado y el valor de retorno observado. Para ello, se utiliza la pérdida cuadrática como función objetivo:

$$\mathcal{L}_{\text{value}} = (R - V(s))^2. \tag{19}$$

Donde R es el retorno descontado y se calcula:

$$R = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^n V(s_{t+n}). \tag{20}$$

Finalmente, utilizando redes neuronales como aproximación para estas funciones:

$$\mathcal{L}_{\text{critic}} = \left(r + \gamma V(s';\phi) - V(s;\phi)\right)^2. \tag{21}$$

El parámetro ϕ representa los pesos de la red critic.

Entrenamiento conjunto. Ambas redes se entrenan simultáneamente. El actor utiliza el valor estimado por la red crítico para mejorar la política, mientras que la red crítico se entrena para proporcionar evaluaciones más precisas de los estados. Es notable destacar que ambas redes neuronales pueden compartir las capas iniciales que extraen las características de las variables de estado, utilizando luego capas separadas para realizar las predicciones específicas de cada red; esta técnica se denomina arquitectura compartida. Alternativamente, las redes pueden no compartir capas en absoluto y ser completamente independientes.

La Figura 7 muestra un esquema general del algoritmo A2C.

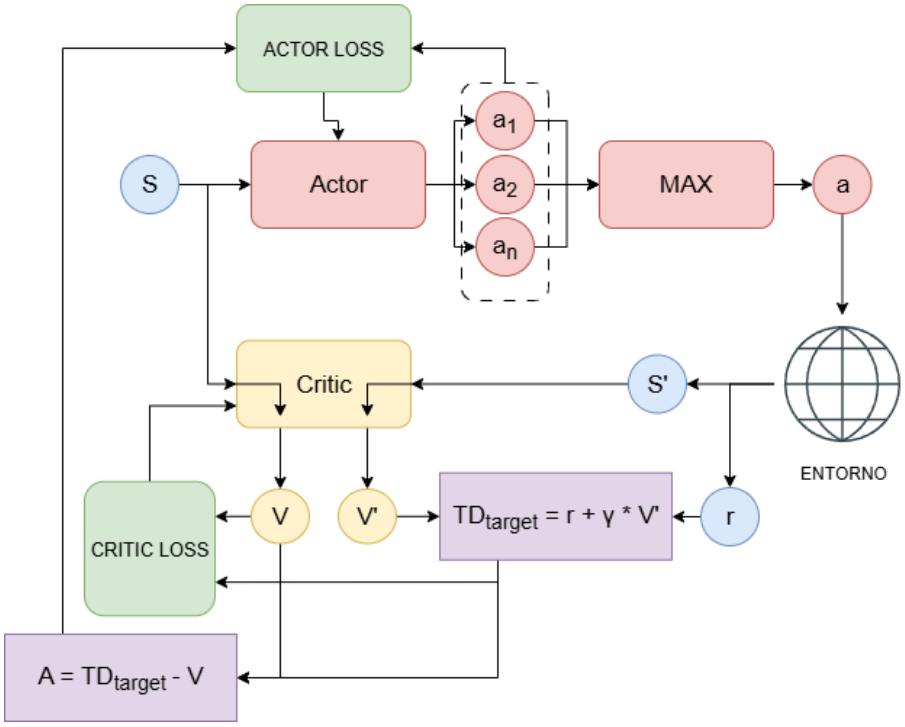

Figura 8. Esquema general del algoritmo A2C, adaptado de Kumar et al. (2021).

2.3.3.2. Trusted Region Policy Optimization (TRPO)

El algoritmo TRPO busca mejorar iterativamente una política imponiendo una restricción sobre cuánto puede cambiar dicha política en cada actualización (Schulman et al. (2015)). Esta restricción se impone utilizando la divergencia de Kullback-Leibler (KL), de modo que la diferencia entre la nueva política y la anterior esté acotada por un valor umbral δ. Esta condición es conocida como *trust region constraint*.

La divergencia KL mide cuán diferentes son dos distribuciones de probabilidad. Dado que las políticas pueden representarse como distribuciones de probabilidad sobre las acciones, esta divergencia cuantifica qué tanto difiere la nueva política π_θ de la anterior $\pi_{\theta_{\text{old}}}$. Controlar este cambio garantiza la estabilidad del entrenamiento, evitando actualizaciones abruptas que puedan degradar el desempeño del agente.

En otras palabras, se busca que la nueva política permanezca cercana a la política anterior, mientras se incorporan de manera gradual mejoras en el comportamiento del agente. Esto se puede formalizar como la maximización de una función objetivo (ecuación 22), sujeta a la restricción de que la divergencia KL

promedio no supere un umbral δ (ecuación 23):

$$\text{maximizar}_\theta \quad L(\theta) = \mathbb{E}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} A_t\right] \tag{22}$$

$$\text{sujeto a} \quad \mathbb{E}_t\Big[D_{\text{KL}}\big(\pi_{\theta_{\text{old}}}(\cdot \mid s_t) \,||\, \pi_\theta(\cdot \mid s_t)\big)\Big] \leq \delta \tag{23}$$

Este es un problema de optimización restringida, que no puede resolverse directamente mediante métodos estándar de gradiente debido a la complejidad de KL. Para su solución, se buscan aproximaciones a las ecuaciones 22–23 y se combinan con el uso de gradientes conjugados, permitiendo implementar el algoritmo en redes neuronales.

Aunque TRPO ofrece resultados sólidos y estables, su implementación presenta desafíos prácticos. Uno de los principales inconvenientes es su complejidad computacional, ya que requiere el uso de gradientes conjugados para optimizar la política, lo cual complica el entrenamiento y limita su aplicabilidad en escenarios de gran escala o en tiempo real.

Debido a estas limitaciones, se desarrollaron algoritmos alternativos más simples y eficientes, como *Proximal Policy Optimization* (PPO), el cual se describe en la siguiente sección.

2.3.3.3. Proximal Policy Optimization (PPO)

El algoritmo *Proximal Policy Optimization* (PPO) fue introducido como una mejora directa del algoritmo TRPO. Su principal ventaja radica en ofrecer un rendimiento similar, e incluso superior en muchos casos, con una menor complejidad de implementación (Schulman et al. (2017)). PPO ha ganado gran popularidad y, actualmente, es considerado el algoritmo base en una amplia gama de aplicaciones de aprendizaje por refuerzo (Chikhaoui et al. (2022); Taheri et al. (2024)).

PPO elimina la necesidad de resolver el problema de optimización restringida de TRPO (mediante gradientes conjugados) y lo reemplaza por una penalización que limita implícitamente el cambio entre la política nueva y la anterior. Esta modificación permite que el algoritmo sea más eficiente computacionalmente y más sencillo de implementar.

La función objetivo inicial propuesta por PPO, basada en el enfoque de *Conservative Policy Iteration* (CPI), se define como:

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] = \hat{\mathbb{E}}_t \left[r_t(\theta) \hat{A}_t \right]. \tag{24}$$

Donde t representa el índice temporal de un paso en un episodio de interacción con el entorno, $r_t(\theta)$ es la razón de las probabilidades de la política π_θ y la política anterior $\pi_{\theta_{old}}$. $\hat{A}_t$ es una estimación de la ventaja en el tiempo t. Generalmente, se utiliza GAE (Generalized Advantage Estimation, en inglés) como método de estimación, y es calculado como se muestra a continuación.

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + ... + (\gamma\lambda)^{T-t+1}\delta_{T-1}. \tag{25}$$

Donde $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$. El término λ es un parámetro de suavizado.

Para evitar actualizaciones excesivas que desestabilicen el entrenamiento, PPO introduce una función de pérdida con recorte (clipping), que restringe $r_t(\theta)$ al intervalo $[1-\epsilon, 1+\epsilon]$:

$$L^{CLIP} = \hat{\mathbb{E}}_t \left[\text{mín} \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right]. \tag{26}$$

Esta función mantiene el cambio de política dentro de una región acotada sin requerir restricciones explícitas. El operador mín asegura que las actualizaciones de la política no sobrepasen el umbral, penalizando los cambios demasiado grandes tanto si $\hat{A}_t > 0$ como si $\hat{A}_t < 0$. Como se muestra en la Figura 8, si $\hat{A}_t > 0$, se favorece aumentar la probabilidad de la acción, pero no más allá de $1+\epsilon$. En cambio si $\hat{A}_t < 0$, se reduce su probabilidad, pero no por debajo de $1-\epsilon$.

PPO utiliza la arquitectura Actor-Crítico (como A2C), por lo tanto, se incorporan tres funciones de pérdida: la pérdida de la política L^{CLIP}, una penalización por error de valor L^{VP} y un término de entropía para fomentar la exploración $S[\pi_\theta](s)$. La función final se expresa como:

$$L_t^{CLIP+VF+S} = \hat{\mathbb{E}}_t \left[L_t^{CLIP} - c_1 L_t^{VF} + c_2 S[\pi_\theta](s_t) \right]. \tag{27}$$

Donde c_1 y c_2 son coeficientes de ponderación para los términos de valor y entropía, respectivamente.

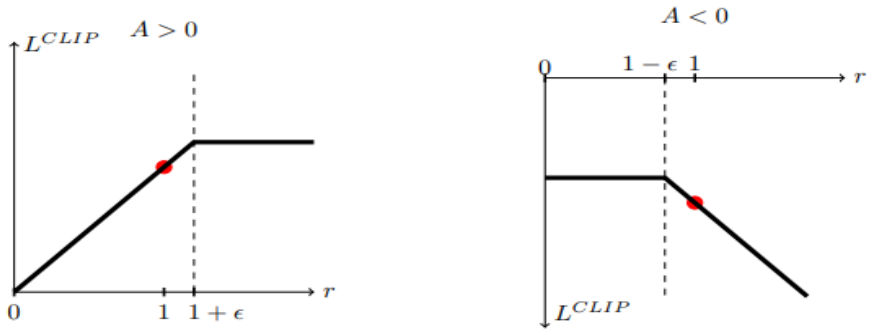

Figura 9. Las figuras muestran la pérdida L_{clip} como función de la razón de probabilidades r, para ventajas positivas a la izquierda y negativas a la derecha. El punto rojo representa el punto de partida para la optimización (cuando $r = 1$). Tomado de Schulman et al. (2017)

L_t^{VF} es la pérdida cuadrática entre el valor estimado $V(s_t)$ y el retorno objetivo y se calcula como se muestra en la ecuación 26. $S[\pi_\theta](s_t)$ es el bonus de entropía que se otorga para incentivar la exploración. En la Figura 9 se muestra un esquema general del algoritmo PPO.

$$L_t^{VF} = (V_t(s;\phi) - R_t)^2. \tag{28}$$

Este enfoque permite que PPO obtenga políticas más robustas y generalizables en una amplia variedad de entornos, convirtiéndose así en uno de los algoritmos más efectivos y utilizados en la actualidad.

Algorithm 4 Proximal Policy Optimization (PPO), adaptado de Schulman et al. (2017)

Inicializar los parámetros de la política θ y del valor crítico ϕ
para cada iteración **hacer**
    para actor $= 1$ hasta N **hacer**
        Ejecutar la política π_θ en el entorno durante T pasos
        Almacenar transiciones: $(s_t, a_t, r_t, \pi_\theta(a_t|s_t))$
    fin para
    Calcular las ventajas estimadas $\hat{A}_t$ (por ejemplo, con GAE)
    Calcular los retornos acumulados R_t para entrenar la red crítica
    para época $= 1$ hasta K **hacer**
        Dividir los datos en mini-lotes
        para cada mini-lote **hacer**
            Calcular la razón de probabilidad: $r_t(\theta)$
            Calcular la pérdida de política con clipping: L_t^{CLIP}
            Calcular la pérdida del valor: L_t^{VF}
            Calcular la pérdida de entropía: L_t^{ENT}
            Calcular la pérdida total: L_t
            Actualizar los parámetros θ y ϕ utilizando L_t
        fin para
    fin para
fin para

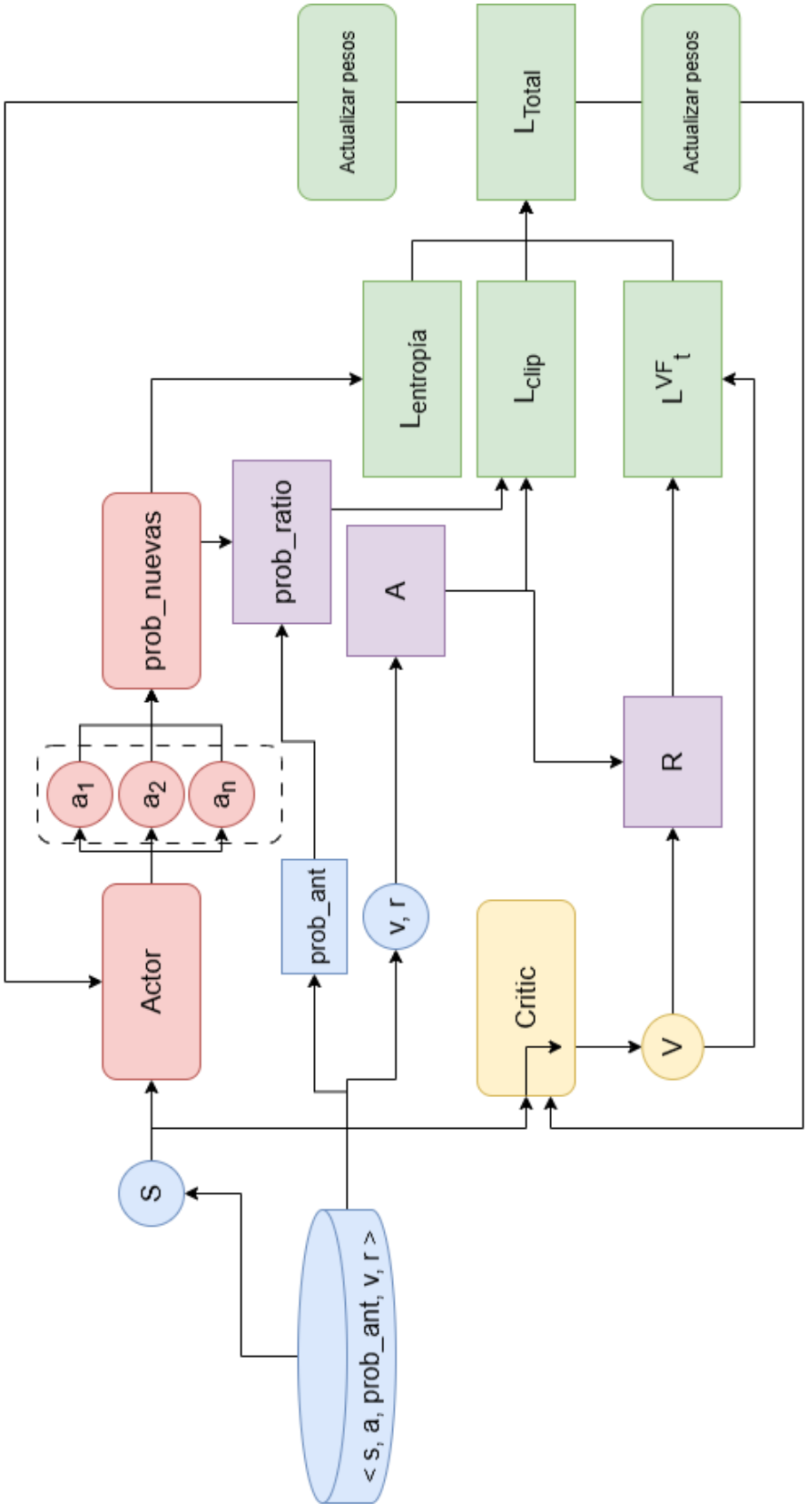

Figura 10. Esquema general del algoritmo PPO, adaptado de Schulman et al. (2017).

Capítulo 3. Captura y seguimiento de un evasor móvil utilizando DRL

En este capítulo se presenta la propuesta metodológica desarrollada para abordar el problema de localización y seguimiento de un evasor móvil utilizando técnicas de DRL. La solución se construye sobre los fundamentos teóricos expuestos en capítulos anteriores.

El diseño de la solución considera diversos componentes clave: la definición del entorno de entrenamiento, la representación del estado del agente, el procedimiento de aprendizaje utilizando curriculum learning, las diferentes funciones de recompensa utilizadas, y las propuestas de los experimentos que evaluarán el impacto de diferentes configuraciones sobre el desempeño del agente.

3.1. Entorno y reglas de simulación

El entorno de simulación fue desarrollado utilizando la biblioteca *Pygame*, con el objetivo de representar un espacio bidimensional discreto para la interacción entre un agente perseguidor, un agente evasor y un conjunto de obstáculos. Todos los elementos del entorno se modelan como entidades cuadradas, dado que el espacio se encuentra dividido en cuadrículas, lo cual facilita la discretización del estado y la implementación de reglas de movimiento.

La dimensión del entorno es de 600×600 píxeles, y tanto el perseguidor como el evasor pueden desplazarse en ocho direcciones (norte, sur, este, oeste y diagonales). Adicionalmente, el agente perseguidor tiene la capacidad de elegir entre dos velocidades de desplazamiento: 1 píxel o 2 píxeles por acción.

Los obstáculos, por su parte, se generan de manera aleatoria al inicio de cada episodio, y pueden adoptar diferentes configuraciones geométricas dentro de la cuadrícula, actuando como elementos estáticos que restringen el movimiento de los agentes.

3.2. Estados

El estado del entorno representa toda la información observable por el agente en cada paso de tiempo y constituye la base sobre la cual se toman las decisiones. Para este trabajo, se diseñó una representación

de estado que combina información relacional entre agentes con características estructurales del entorno local.

El vector de observación está compuesto por cuatro componentes. Entre ellos, la posición relativa del evasor, la posición absoluta del perseguidor y la distancia euclidiana al evasor se encuentran **normalizados con respecto al tamaño de las cuadrículas del entorno**, con el fin de garantizar una representación homogénea y estable durante el entrenamiento. A continuación, se presenta una descripción detallada de las variables de estado:

- **Posición relativa del evasor:** representada como $(\Delta x, \Delta y)$, indica la dirección desde el agente perseguidor hacia el evasor. Esta magnitud resulta fundamental para guiar el comportamiento de persecución.
- **Posición absoluta del perseguidor:** las coordenadas (x, y) indican la ubicación actual del perseguidor dentro del mapa global. Proporciona al agente un sentido de ubicación contextual.
- **Distancia euclidiana al evasor:** esta magnitud escalar cuantifica la cercanía del perseguidor respecto al evasor. Ofrece una información útil para inferir si el objetivo se encuentra cerca o lejos, facilitando la modulación del comportamiento estratégico.
- **Mapa local de obstáculos:** una matriz de tamaño $n \times n$, centrada en la posición actual del perseguidor. Cada celda representa el contenido del entorno inmediato según los siguientes valores:
  - `-1.0`: celda ocupada por un obstáculo.
  - `0.0`: celda libre (navegable).
  - `1.0`: celda donde se encuentra el evasor.

  Una representación del mapa local de obstáculos se puede visualizar en la Figura 11.

La combinación de estas características permite construir una representación del entorno que integra información global y local, así como elementos dinámicos (posición del evasor) y estáticos (obstáculos). Esta codificación busca fomentar la emergencia de comportamientos complejos como navegación eficiente, evasión de colisiones y persecución estratégica.

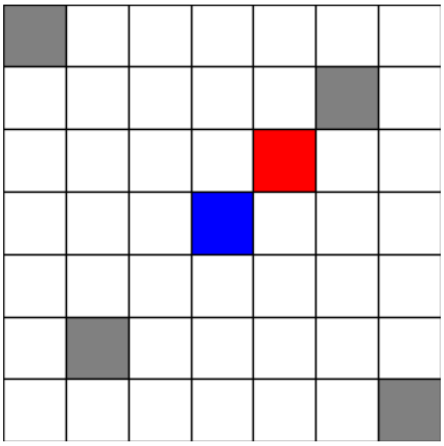

Figura 11. Representación de un mapa local 7×7 de obstáculos. En el centro se representa al agente perseguidor en azul, el agente evasor en rojo, los obstáculos en color gris, y los espacios vacíos en blanco.

Cabe destacar que esta estructura de estado se mantiene constante a lo largo de los distintos niveles del *aprendizaje curricular*, lo cual favorece la transferencia del conocimiento adquirido en entornos más simples hacia escenarios progresivamente más desafiantes.

3.3. Algoritmo de aprendizaje por refuerzo utilizado

Para el desarrollo de la propuesta de solución, se utilizó el algoritmo Proximal Policy Optimization (PPO), debido a su robustez y eficiencia para problemas de aprendizaje por refuerzo con espacios de acción continuos o discretos. Esto permite que, en el futuro, la metodología pueda extenderse fácilmente a acciones continuas, aprovechando arquitecturas preentrenadas y ajustándolas a nuevas configuraciones de acción. PPO también ha sido probado frente a otros algoritmos (Mock & Muknahallipatna (2024); Del Rio et al. (2024)), mostrando resultados excelentes, y constituye la base de muchas aplicaciones avanzadas en robótica móvil (Zhuang et al. (2023); Taheri et al. (2024)). La implementación se realizó utilizando la librería Stable Baselines3, que ofrece herramientas confiables y optimizadas para entrenar agentes con PPO, facilitando además la experimentación reproducible.

3.4. Procedimiento utilizando aprendizaje curricular

El proceso de entrenamiento del agente perseguidor se llevó a cabo mediante una estrategia de curriculum learning, diseñada para facilitar la adquisición progresiva de habilidades en un entorno de persecución-evasión de dificultad creciente. Esta técnica consiste en estructurar el aprendizaje por niveles, donde cada nivel introduce mayores desafíos, permitiendo que el agente construya conocimiento de manera incremental.

En este trabajo, el entorno fue parametrizado mediante una variable discreta de dificultad, cuyos valores van del 0 al 5. Esta variable determina la configuración del entorno en cada fase curricular, permitiendo un incremento progresivo en la complejidad del aprendizaje. Dado que, en las primeras etapas, el perseguidor parte de una configuración favorable en la que el evasor se encuentra dentro del rango de captura, se consideró más equitativo para el proceso de entrenamiento establecer como criterio de finalización el alcanzar un número fijo de pasos, en lugar de basarse en la cantidad de episodios ejecutados. De esta manera, se evita que configuraciones iniciales más ventajosas reduzcan la duración del entrenamiento en un nivel dado.

Para este propósito, se definió un total de 60,000 pasos por cada nivel de dificultad. Adicionalmente, se estableció un límite máximo de 500 pasos por episodio, tras los cuales se considera que el perseguidor no alcanzó su objetivo. En este caso, el episodio se da por finalizado y se reinicia el entorno, registrándose como un intento fallido por parte del perseguidor. Esta estrategia permite evitar episodios excesivamente largos y favorece una exploración más eficiente del espacio de estados durante el entrenamiento. Además, cada nivel curricular modifica dinámicamente los siguientes aspectos:

- **Rango de captura:** Inicialmente se parte de un rango de captura grande, de 20 cuadrículas. Este rango disminuye progresivamente con la dificultad hasta alcanzar un valor mínimo de 1 cuadrícula, condición en la cual se logra la captura del evasor.
- **Número de obstáculos:** Aumenta con la dificultad, comenzando con entornos vacíos y alcanzando hasta 400 obstáculos estáticos distribuidos aleatoriamente en el mapa, además de bordes sólidos que delimitan el entorno.
- **Inteligencia del evasor:** En los primeros niveles el evasor toma decisiones de forma aleatoria. Conforme avanza la dificultad, se establece un umbral de probabilidad que determina si el evasor elige una acción aleatoria o una que maximice su distancia respecto al perseguidor. En el nivel

máximo, el evasor siempre toma decisiones que incrementan dicha distancia.

- **Distancia inicial entre agentes:** Incrementa progresivamente con la dificultad, dificultando la localización y el seguimiento del evasor en las etapas iniciales de cada episodio.
- **Número de pasos de seguimiento:** Para que un episodio sea considerado exitoso, el perseguidor debe mantener la tarea de seguimiento durante un número mínimo de pasos. Esta condición resulta especialmente útil en los niveles iniciales, donde los agentes aparecen directamente en el rango de captura; con ello se evita que el episodio termine sin que el perseguidor acumule experiencia práctica en la persecución.

El modelo obtenido en un nivel se empleó como punto de partida para el siguiente, lo que permitió al agente transferir el conocimiento adquirido previamente y adaptarse de manera más eficiente a la nueva configuración del entorno. De esta forma, se buscó favorecer un proceso de aprendizaje progresivo que facilitara la exploración de estados más complejos a medida que aumentaba la dificultad.

En las Figuras 12–15 se presentan las distintas configuraciones resultantes de las modificaciones aplicadas al entorno. En ellas se observa en color azul la posición actual del perseguidor, en rojo la del evasor, mientras que la cuadrícula azul claro indica la ubicación previa del perseguidor y la cuadrícula roja clara representa la posición anterior del evasor. Finalmente, el círculo verde alrededor del perseguidor corresponde al radio de captura.

Figura 12. Entorno para la dificultad 0.

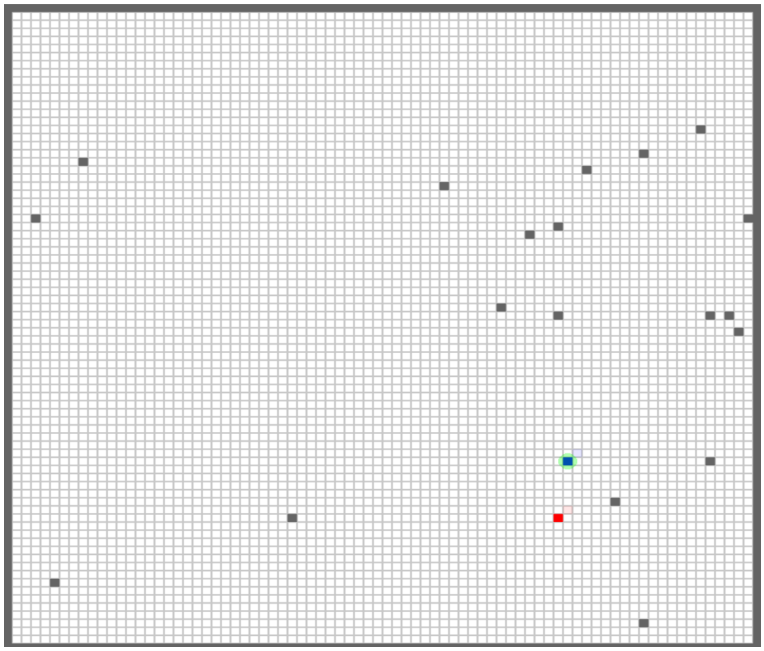

Figura 13. Entorno para la Dificultad 2

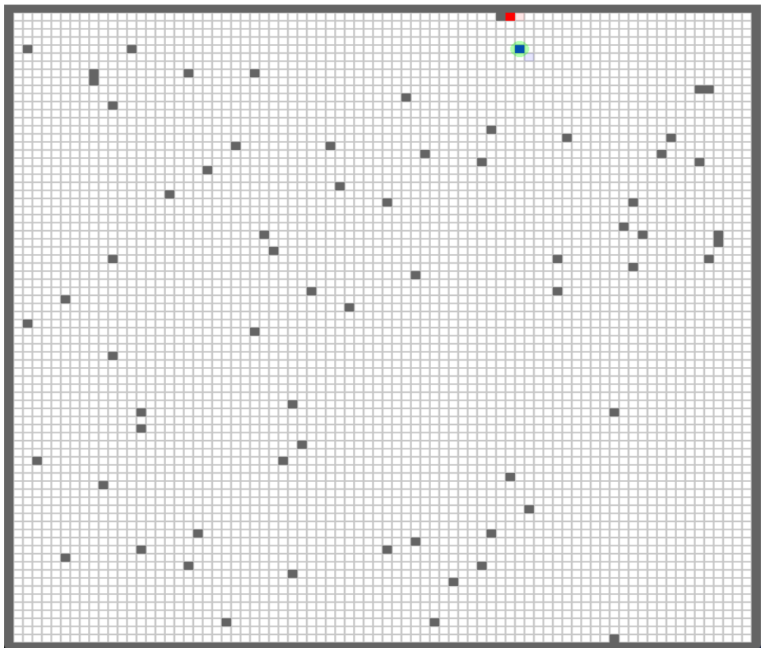

Figura 14. Entorno para la Dificultad 3

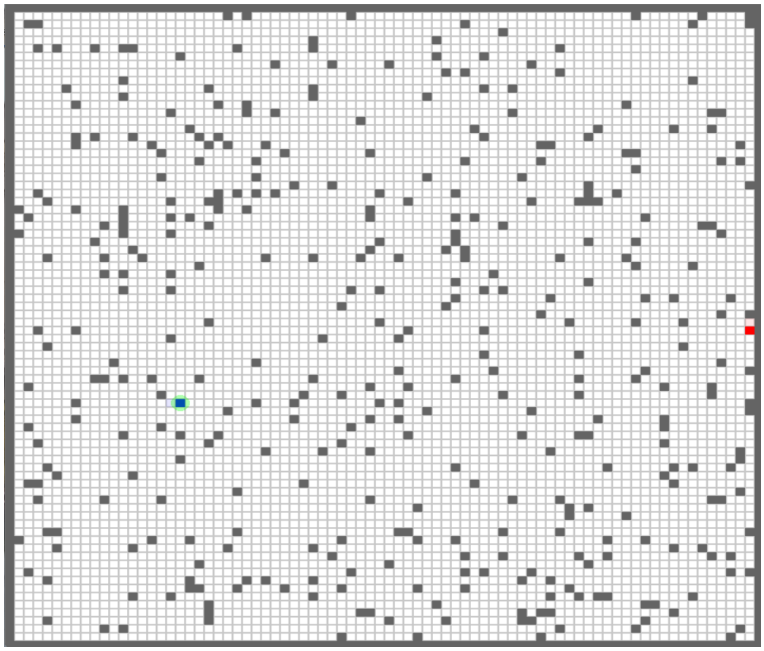

Figura 15. Entorno para la Dificultad 5

3.5. Función de recompensa

Para lograr que el agente perseguidor tenga un comportamiento de seguimiento, se diseñó una función de recompensa compuesta que incorpora múltiples recompensas y penalidades. Esta función premia llegar a la zona de visibilidad o seguimiento del agente evasor, así como el progreso constante hacia él. Por otro lado, penaliza las acciones ineficaces que alejan al agente de la zona de seguimiento y las que resultan en una colisión con los obstáculos.

La recompensa más alta que puede alcanzar el perseguidor es de 50 puntos (R_{obj}) por llegar a la zona de seguimiento. Este valor significativamente alto se utiliza para reforzar fuertemente el valor de alcanzar el objetivo final.

La penalización más severa que recibe el perseguidor ocurre en dos situaciones: cuando colisiona con un obstáculo o cuando el episodio finaliza sin que el evasor se encuentre dentro de la zona de captura. En ambos casos, la penalización asignada es de -50 unidades, denominada R_{min}.

Mientrás no se alcanza el objetivo final el agente perseguidor recibirá las penalidades y bonificaciones descritas en las subsecciones siguientes.

3.5.1. Recompensa por cambio en la distancia

La diferencia entre la distancia en el paso $t-1$ entre el perseguidor y el evasor, denotada como d_{t-1}, y la distancia en el paso actual d_t, se utiliza para calcular una recompensa que premia o penaliza al agente en función de su progreso hacia el objetivo. Esta señal de recompensa se normaliza con respecto al tamaño de las cuadrículas del entorno, representado por $grid\_size$.

$$R_{prox} = C * \frac{(d_{t-1} - d_t)}{grid\_size}. \tag{29}$$

Donde C es una constante de escala que determina la magnitud de la recompensa otorgada por unidad de progreso. Un valor positivo de R_{prox} indica que el agente se está acercado al evasor, mientras que un valor negativo representa un alejamiento.

Es importante destacar que esta recompensa debe tener una magnitud inferior a la recompensa otorgada por alcanzar el objetivo final, con el fin de evitar que el agente aprenda a oscilar alrededor del evasor para acumular recompensas por cercanía, en lugar de entrar a la zona de seguimiento directamente.

3.5.2. Penalización por tiempo

Con el fin de incentivar que el perseguidor complete su objetivo en el menor número de pasos posible, se introduce una penalización constante por cada paso de tiempo en el que no se alcanza al evasor. Esta penalización desalienta trayectorias ineficientes y favorece aquellas que se aproximan a la trayectoria óptima hacia el objetivo.

$$R_{\text{tiempo}} = -0.05. \tag{30}$$

3.5.3. Bonificación por cercanía

Cuando la distancia normalizada entre el perseguidor y el evasor, denotada como d_{norm}, es menor a 0.1 (valor determinado experimentalmente mediante prueba y error), se otorga una bonificación adicional de

recompensa de hasta 0.5. Esta bonificación aumenta de manera inversamente proporcional a la distancia, incentivando al agente a mantenerse lo más próximo posible al evasor.

$$R_{\text{cercanía}} = \begin{cases} 0.5 \times (1 - d_{\text{norm}}), & \text{si } d_{\text{norm}} < 0.1 \\ 0, & \text{en otro caso} \end{cases}. \tag{31}$$

3.5.4. Penalización por inacción

Cuando el agente no logra reducir la distancia con respecto al evasor entre dos pasos consecutivos, se interpreta como un comportamiento ineficaz o de estancamiento. En estos casos, se le aplica una penalización constante de -0.1 para desalentar este tipo de comportamientos.

$$R_{\text{inacción}} = -0.1. \tag{32}$$

La recompensa total en cada paso de tiempo se calcula como la suma de todos los componentes definidos anteriormente:

$$R_{total} = R_{obj} + R_{\text{mín}} + R_{prox} + R_{tiempo} + R_{\text{cercanía}} + R_{\text{inacción}}. \tag{33}$$

3.6. Criterios de evaluación

Para evaluar el desempeño del agente durante el proceso de entrenamiento y validación, se emplean un conjunto de criterios cuantitativos que permiten medir su eficacia, eficiencia y estabilidad. Estos criterios proporcionan una visión detallada del comportamiento aprendido, identificando fortalezas y posibles debilidades de la política entrenada. A través de estas evaluaciones, es posible comparar diferentes configuraciones, arquitecturas y niveles de dificultad del entorno, así como realizar un seguimiento objetivo del progreso del agente a lo largo del aprendizaje curricular.

3.6.1. Criterios de desempeño de los agentes

Estos criterios evalúan directamente la efectividad de los agentes en el cumplimiento de sus objetivos, ya sea capturar o evadir. Se utilizan para cuantificar el rendimiento tanto del perseguidor como del evasor en términos de éxito y eficiencia temporal.

Razón de captura: Representa el porcentaje de episodios en los que el perseguidor logra capturar al evasor dentro del límite de tiempo establecido y se define como:

$$R_{Cap} = \frac{Capturas}{Episodios\ Totales} \times 100. \tag{34}$$

Tiempo de captura: Es el número de pasos que el perseguidor requiere para alcanzar al evasor en los episodios exitosos. Esta métrica evalúa la eficiencia del comportamiento de persecución.

Razón de escape: Porcentaje de episodios en los que el evasor logra evitar ser capturado por el perseguidor hasta que se alcanza el límite de tiempo del episodio.

$$R_{Esc} = \frac{Escapes}{Episodios\ Totales} \times 100. \tag{35}$$

Tiempo de supervivencia: Número de pasos que el evasor permanece sin ser capturado. Esta métrica refleja la capacidad del evasor para mantener la evasión a lo largo del tiempo.

3.6.2. Criterios de aprendizaje

Este conjunto de criterios permiten evaluar el proceso de entrenamiento de los agentes durante la etapa de entrenamiento. Además, permiten identificar comportamientos anómalos, signos de convergencia o posibles problemas de estabilidad.

Uno de los principales criterios considerados es la **recompensa promedio por episodio**, la cual actúa como medida directa de la calidad de la política aprendida. Un aumento sostenido en esta métrica suele indicar mejoras en el comportamiento del agente.

La **pérdida de la política** constituye una métrica central en el análisis del entrenamiento, ya que refleja directamente el grado de ajuste de la estrategia aprendida por el agente. Variaciones pronunciadas o la presencia de picos inusuales en esta métrica pueden ser indicativos de inestabilidad durante el proceso de optimización, lo cual repercute negativamente en la calidad de la política resultante. Por el contrario, una tendencia estable y decreciente suele asociarse con un proceso de aprendizaje más consistente y con una política mejor adaptada a la tarea planteada.

3.6.3. Generalización y robustez

Con el fin de evaluar la capacidad de generalización y la robustez de la política aprendida, se ejecutaron 100 episodios utilizando 5 semillas aleatorias diferentes. De esta manera, se garantiza qué tanto las posiciones iniciales de los agentes como la distribución de los obstáculos difieran de aquellas empleadas durante el entrenamiento.

Esta metodología se diseñó con el objetivo de que el escenario de evaluación no coincidiera con el escenario de entrenamiento. En caso contrario, se incurriría en una práctica análoga a emplear el mismo conjunto de datos de entrenamiento en la etapa de prueba dentro de un enfoque de aprendizaje supervisado, lo que se conoce como fuga de información. Tal situación provocaría una sobreestimación del rendimiento real de la política. Por lo tanto, el uso de semillas aleatorias distintas en las fases de entrenamiento y evaluación permite garantizar configuraciones diferenciadas y, en consecuencia, una evaluación más rigurosa y realista del desempeño de la política aprendida.

3.7. Propuestas de experimentos

Para llevar a cabo los experimentos, se evaluaron distintas configuraciones tanto en las representaciones del estado observadas por el agente como en las arquitecturas de red utilizadas. El objetivo fue analizar cómo estos factores impactan la capacidad del agente para aprender una política efectiva.

3.7.1. Arquitecturas

Se ensayaron diversas arquitecturas, buscando explorar desde modelos simples hasta configuraciones más complejas que permitan capturar características espaciales y temporales del entorno:

1. Una red MLP (perceptrón multicapa), diseñada para capturar la política con una arquitectura de baja complejidad. Esta arquitectura consta de cuatro capas neuronales con 256, 128, 64, y 32 neuronas, respectivamente.

2. Una red CNN, orientada a extraer características espaciales relevantes del estado. Esta arquitectura inicia con dos capas convolucionales. La primera capa cuenta con 32 filtros de 3×3 y un desplazamiento de 1 unidad de posición. La segunda capa contiene 64 filtros de 3×3 y un desplazamiento de 1 unidad de posición. Finalmente, la salida convolucional se aplanará y se conectará a una MLP como la diseñada en 1 para la decisión final.

3.7.2. Configuración de los estados

Se diseñaron diferentes variantes en la representación del estado, modificando progresivamente la cantidad de información proporcionada al agente, así como el tamaño de la matriz de observación. Esta estrategia permitió evaluar cómo la dimensionalidad del estado afecta el desempeño del agente.

La *variante 1* incluye la posición relativa entre el perseguidor y el evasor, la posición absoluta del perseguidor, la distancia euclidiana entre el perseguidor y el evasor, y una matriz de observación local alrededor del agente. Se evaluaron tamaños de 3×3, 7×7 y 9x9 celdas.

En la *variante 2* se omite la posición absoluta del agente, y se emplea la dimensión de la matriz de observación que previamente demostró un mejor desempeño. La *variante 3* elimina la distancia normalizada entre perseguidor y evasor, dejando solo la posición relativa y la matriz de observación.

Finalmente, en la *variante 4* se elimina la posición relativa entre los agentes, conservándose el resto de las variables. Esta serie de configuraciones permite comparar el impacto relativo de cada tipo de información en el aprendizaje y desempeño del agente.

Capítulo 4. Resultados

En este capítulo, se presentan los resultados obtenidos a partir de los experimentos realizados sobre el entorno de persecución-evasión planteado. El objetivo principal es analizar el desempeño del agente en función de distintas configuraciones del espacio de observación, la representación del estado y la arquitectura de la red neuronal empleada.

4.1. Experimento 1: perceptrón multicapa con selección de variables de estado

El primer experimento tuvo como propósito evaluar el impacto tanto del **tamaño de la matriz de observación** como de la **representación del estado** en el desempeño del agente. La motivación detrás del incremento de la matriz de observación es proporcionarle al agente una visión más amplia del entorno, lo que potencialmente le permitiría tomar decisiones más informadas y, en consecuencia, obtener una mayor recompensa promedio o una tasa de éxito alta.

En cuanto a la arquitectura utilizada, como se mencionó en la subsección 3.7.1, se empleó un perceptrón multicapa (MLP) compuesto por cuatro capas densas de 256, 128, 64 y 32 neuronas, respectivamente.

De manera complementaria al análisis del tamaño de la matriz de observación, se investigó la influencia de las **variables asociadas a la posición y a la distancia entre los agentes** dentro de la representación del estado. Para este fin, se llevaron a cabo entrenamientos en los que se eliminaron progresivamente algunas de estas variables, con el objetivo de determinar si era posible mantener un desempeño competitivo con menos información. Este enfoque no solo permite identificar qué variables son realmente críticas para la toma de decisiones, sino que también ofrece la posibilidad de **reducir la complejidad computacional** del modelo al disminuir la cantidad de datos procesados en cada paso.

4.1.1. Variación del tamaño de la matriz de observación

Inicialmente, se emplearon todas las variables de estado junto con una matriz de observación de tamaño 3×3. Durante los primeros episodios, correspondientes al nivel 0, el evasor aparece directamente dentro de la zona de captura, lo que explica que la recompensa inicie con valores elevados. Sin embargo, a partir

del nivel 2, el perseguidor debe comenzar a localizar al evasor dentro del entorno, ya que este deja de aparecer en la zona de captura y tiende a alejarse progresivamente del perseguidor.

Los resultados de la recompensa promedio obtenida durante el entrenamiento se presentan en la Figura 16. Los picos irregulares que se observan corresponden a los cambios entre niveles, indicados por líneas discontinuas verticales, donde el perseguidor tiene que adaptar la política a la nueva complejidad.

A partir del nivel 3, el desempeño del agente perseguidor en términos de recompensa promedio comienza a disminuir; no obstante, los valores se mantienen en rangos positivos. Esto indica que, si bien el agente no consigue siempre cumplir con el objetivo final de mantener al evasor dentro de la zona de captura y capturarlo, sí logra localizarlo y acercarse a él de manera consistente.

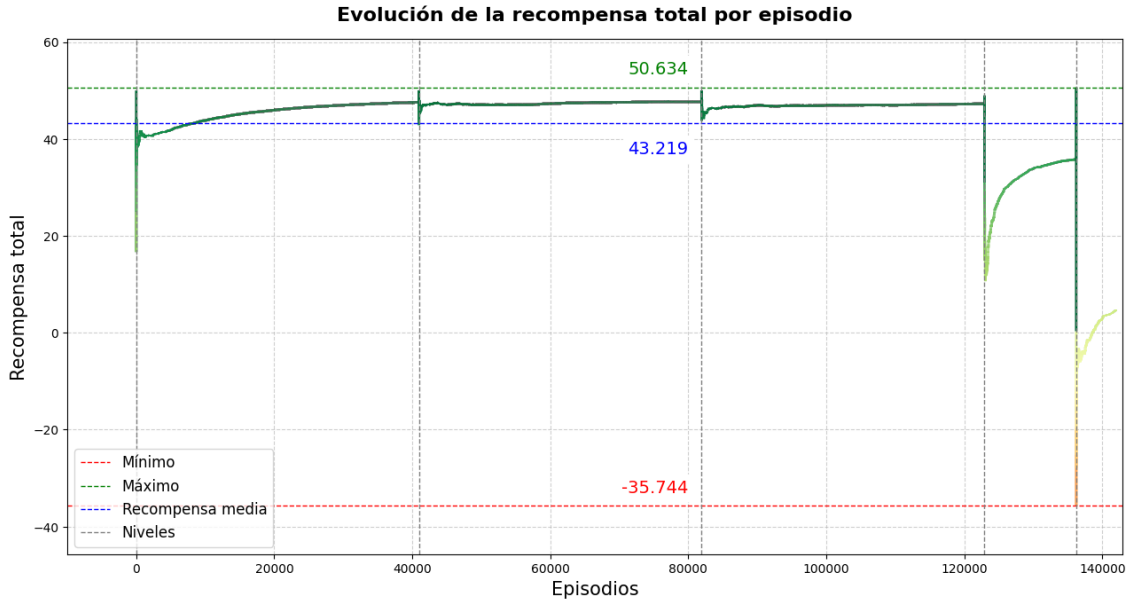

Figura 16. Evolución de la recompensa total por episodio para un tamaño de la matriz de observación de 3×3.

En la Figura 17 se muestran las pérdidas para la red Actor, la red Crítica, y la pérdida total obtenida de aplicar la ecuación 27. Como se puede apreciar, para el nivel 0 y 1, donde el evasor aparece en la zona de captura, la pérdida es mínima, esto es debido a que la evaluación del estado es muy buena inicialmente. Sin embargo, aún la red Actor no ha aprendido una política buena y esto justifica que su pérdida inicial sea más alta que para niveles posteriores. A partir del nivel 2, donde ya no se está en una posición inicial ventajosa para el perseguidor, las pérdidas comienzan a incrementarse, porque se termina en estados no deseados, sin embargo la política del Actor comienza a disminuir porque en promedio se está ejecutando las acciones que favorecen la recompensa del agente aunque no logre estar en estados favorables como

los logrados en los niveles iniciales.

En la Figura 17 se presentan las pérdidas asociadas a la red *Actor*, la red *Crítica* y la pérdida total resultante de aplicar la fórmula 27. Como se observa, durante los niveles 0 y 1, donde el evasor aparece directamente en la zona de captura, la pérdida es menor que en niveles posteriores. Esto se debe a que la evaluación del estado es altamente precisa en estas condiciones iniciales. No obstante, la red *Actor* aún no ha aprendido una política sólida, lo cual explica que su pérdida inicial sea relativamente mayor en comparación con niveles posteriores.

A partir del nivel 2, cuando el perseguidor deja de contar con una posición inicial ventajosa, las pérdidas comienzan a incrementarse, reflejando la dificultad añadida del entorno y la mayor probabilidad de terminar en estados no deseados. Sin embargo, se observa una tendencia decreciente en la pérdida de la red *Actor*, lo que indica que la política aprendida progresa gradualmente hacia la selección de acciones que, en promedio, favorecen la acumulación de recompensa, incluso si no se alcanzan los estados tan favorables como los logrados en los niveles iniciales.

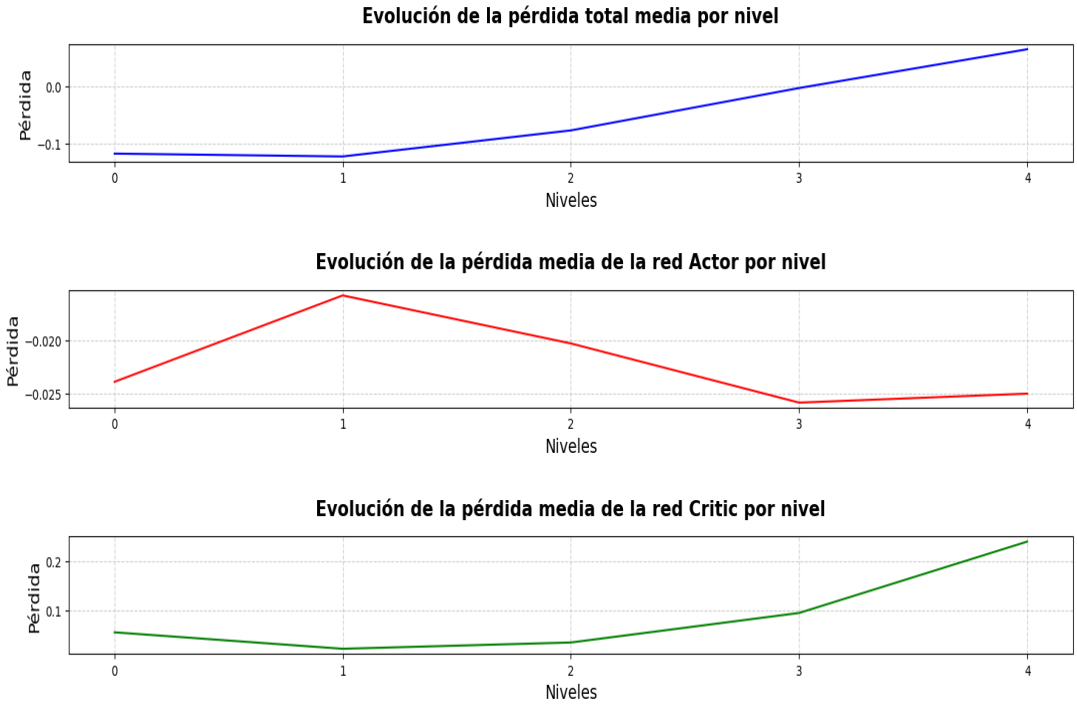

Figura 17. Evolución de la pérdida total media, pérdida media de la red Actor, y la pérdida media de la red Crítica por nivel para un tamaño de la matriz de observación de 3×3.

Posteriormente, se incrementó el tamaño de la matriz de observación a 7×7 y 9×9. Tal como se aprecia

en las Figuras 18–21, el comportamiento del agente en ambos casos mantiene una tendencia similar a la observada con el tamaño 3×3.

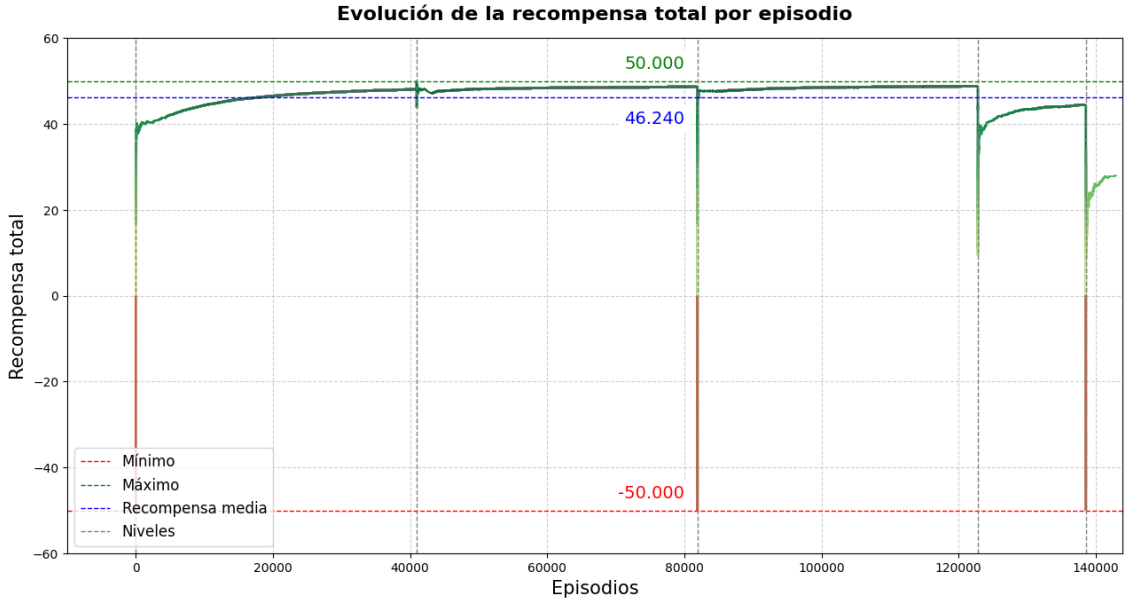

Figura 18. Evolución de la recompensa total por episodio para un tamaño de la matriz de observación de 7×7.

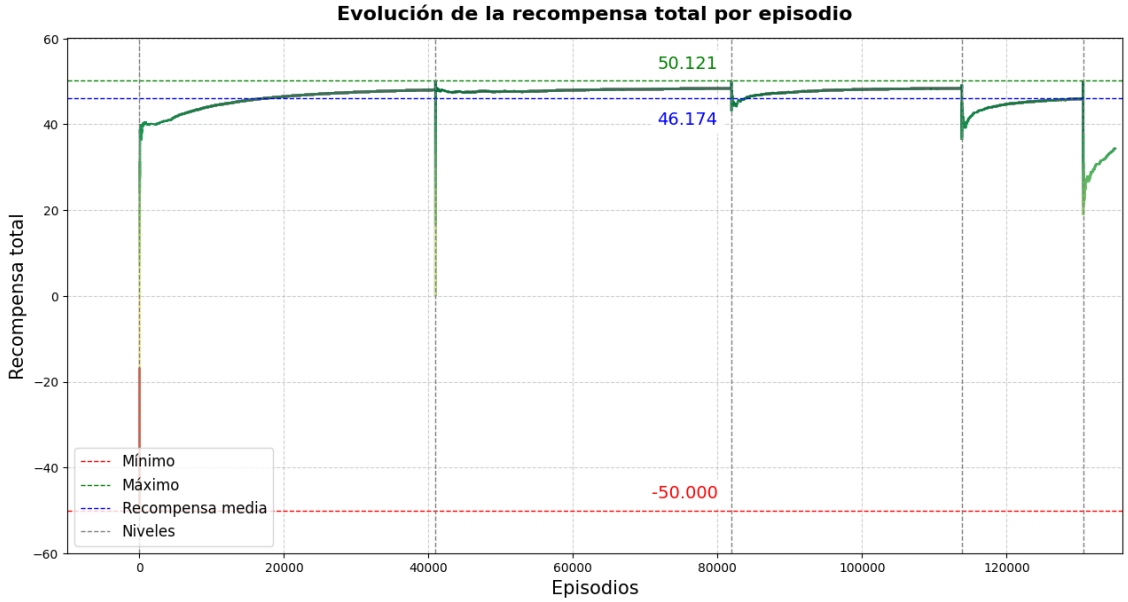

Figura 19. Evolución de la recompensa total por episodio para un tamaño de la matriz de observación de 9×9.

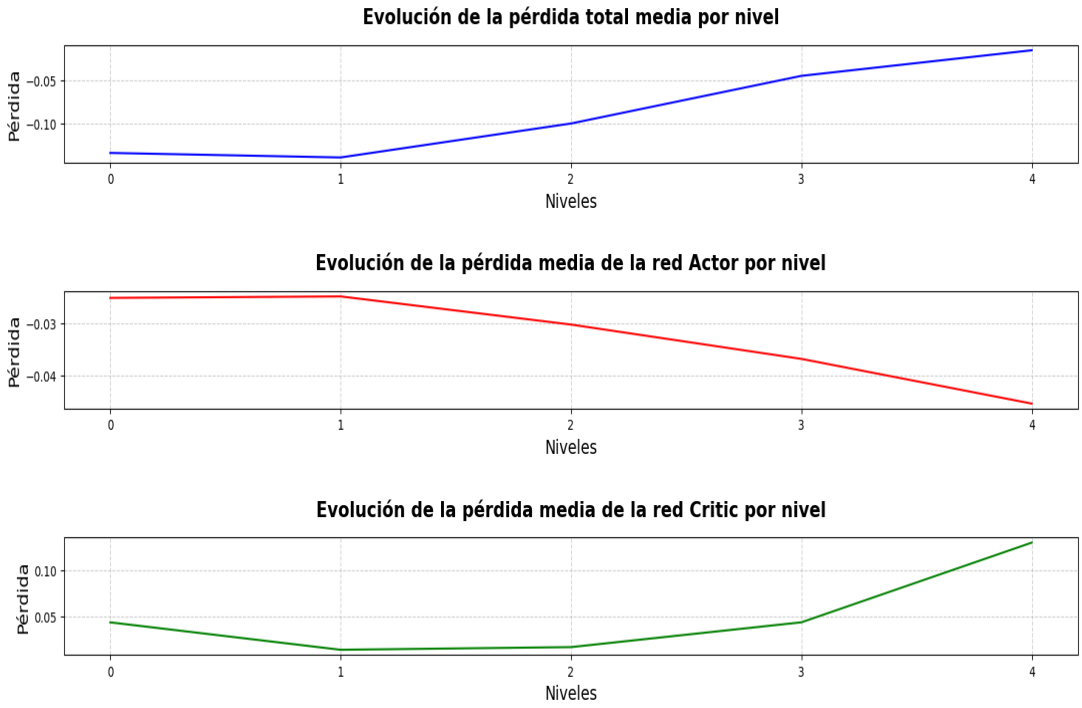

Figura 20. Evolución de la pérdida total media, pérdida media de la red Actor, y la pérdida media de la red Crítica por nivel para un tamaño de la matriz de observación de 7×7.

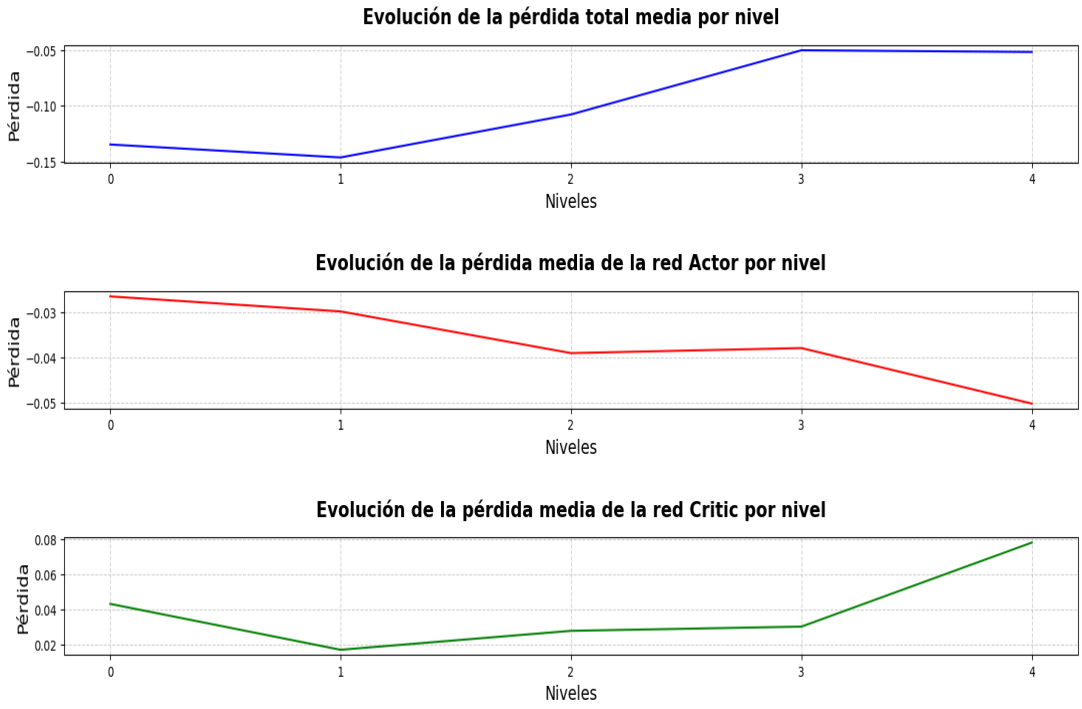

Figura 21. Evolución de la pérdida total media, pérdida media de la red Actor, y la pérdida media de la red Crítica por nivel para un tamaño de la matriz de observación de 9×9.

4.1.2. Discusión de la étapa de entrenamiento

Como se muestra en la Figura 22, donde se comparan los resultados de la recompensa promedio para los tres tamaños de matriz de observación, inicialmente no se aprecia una mejora significativa: tanto la recompensa máxima como la promedio son aproximadamente iguales hasta el nivel 2.

A partir del nivel 3, cuando la complejidad del entorno aumenta y las configuraciones iniciales colocan al perseguidor en posiciones menos ventajosas, la recompensa promedio comienza a disminuir. Esto se debe a que el perseguidor no localiza rápidamente al evasor y la navegación se complica debido a la mayor cantidad de obstáculos.

Asimismo, en los niveles más complejos se observa que matrices de mayor dimensión tienden a incrementar la recompensa promedio, lo que sugiere que una visión más amplia del entorno resulta beneficiosa en escenarios más desafiantes.

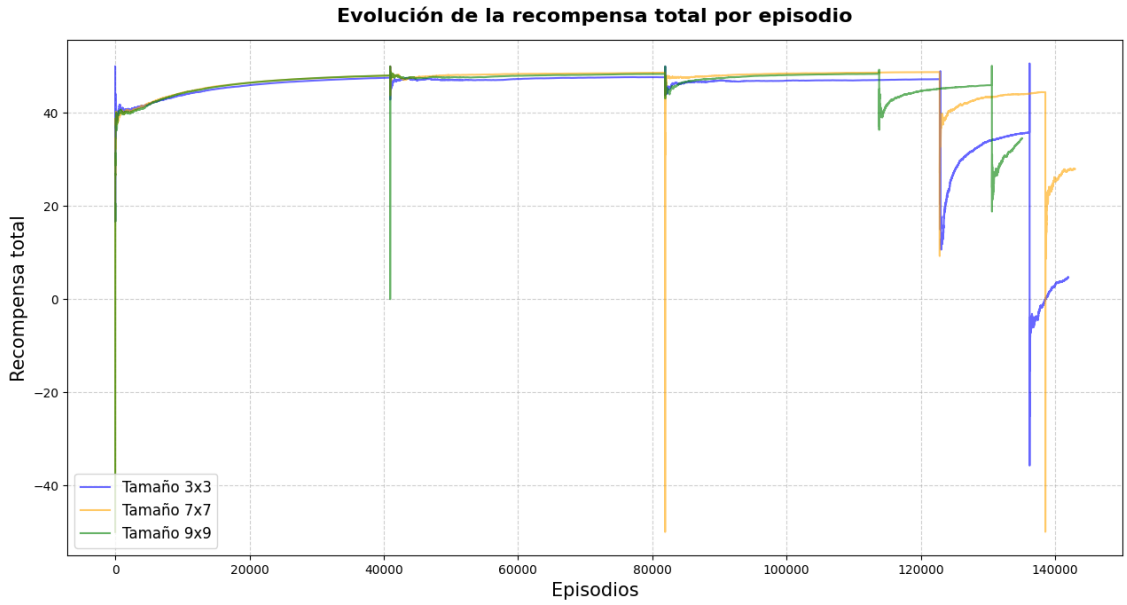

Figura 22. Comparación de la evolución de la recompensa total por episodio para los tres tamaños de matriz de observación utilizados.

4.1.3. Resultados de la evaluación

Tal como ocurre en cualquier paradigma de aprendizaje de máquinas, es posible obtener excelentes resultados durante la etapa de entrenamiento que, sin embargo, se deterioran en la fase de evaluación. Adicionalmente, una de las limitaciones inherentes al aprendizaje por refuerzo es que no se conoce de antemano cuánto tiempo ni qué cantidad de datos se requieren para que un agente logre aprender una política efectiva.

Por lo tanto, cada política obtenida a partir de los experimentos fue evaluada con el objetivo de medir la tasa de éxito y la recompensa promedio alcanzada bajo diferentes configuraciones del entorno y del estado del agente. Los resultados se resumen en la Tabla 1, donde las celdas en verde indican el mejor desempeño para cada nivel.

Tabla 1. Comparación de las diferentes políticas aprendidas para las dificultades 3 y 4.

Experimento	Dificultad	Tasa de éxito	Recompensa promedio
Experimento 1: tamaño 3x3	3	0.62	117.33
Experimento 1: tamaño 3x3	4	0.43	68.59
Experimento 1: tamaño 7x7	3	0.95	182.35
Experimento 1: tamaño 7x7	4	0.41	57.10
Experimento 1: tamaño 9x9	3	0.85	146.73
Experimento 1: tamaño 9x9	4	0.20	17.27

A pesar de que durante el entrenamiento se observó un mejor rendimiento en la recompensa promedio al incrementar la dimensión de la matriz de observación, este comportamiento no se replicó en la fase de evaluación. Una posible explicación es que, si bien el agente dispone de una mayor cantidad de información para la navegación, no logra refinar su política de manera suficiente como para alcanzar consistentemente el objetivo final.

Un aspecto crítico identificado en la evaluación es que el agente perseguidor logra localizar al evasor, pero no se aproxima lo suficiente como para situarlo dentro de la zona de captura. Este resultado confirma lo observado durante el entrenamiento en los últimos niveles, donde, aunque el agente mejoraba su capacidad de exploración y localización, no consolidaba una estrategia efectiva para lograr recompensas altas, y por consiguiente completar la tarea.

Con base en estos hallazgos, se seleccionó el tamaño de matriz de observación 7×7 para los análisis posteriores, al representar un compromiso adecuado entre la cantidad de información disponible y la

capacidad del agente para procesarla de manera eficiente.

4.1.4. Eliminación de variables del estado

Con el objetivo de analizar el impacto de la representación del estado en el desempeño del agente, se llevaron a cabo experimentos en los que se eliminaron variables que describen el entorno. La motivación principal de este análisis fue doble: por un lado, evaluar si una menor cantidad de información podría evitar introducir ruido al modelo y, por otro, comprobar si es posible reducir el costo computacional del entrenamiento sin comprometer de manera significativa el desempeño del agente.

Los resultados del entrenamiento al eliminar progresivamente las variables de posición y distancia en la representación del estado se muestran en las Figuras 23–25. Como se observa, hasta el nivel 2, el comportamiento es aproximadamente similar para las tres variaciones en cuanto a recompensa promedio, dado que la complejidad del entorno permite al perseguidor aparecer inicialmente dentro del rango de captura o muy cerca de él.

Para los niveles posteriores al nivel 2, en las dos primeras variaciones la recompensa promedio disminuye, aunque se mantiene en valores positivos, lo que indica que persiste el problema identificado en la sección 4.1.3. En cambio, cuando se elimina la variable de posición relativa, la recompensa promedio cae a valores negativos, evidenciando que el agente pierde efectividad en todos los episodios evaluados.

Como se observa, los resultados indican que es posible prescindir de dos variables específicas de la representación del estado: la posición absoluta del agente y la distancia euclidiana normalizada entre los agentes. La ausencia de estas variables no afectó de forma significativa el desempeño del agente, lo que sugiere que no aportaban información crítica para el aprendizaje de la política.

En contraste, la eliminación de la variable correspondiente a la posición relativa entre los agentes tuvo un impacto considerable en el rendimiento. Tal como se aprecia en la Figura 25, una vez que el agente pierde la ventaja de la posición inicial, se ve incapaz de localizar al agente evasor, lo que provoca una caída de la recompensa promedio a valores negativos. Esto evidencia que la posición relativa constituye un elemento fundamental para la toma de decisiones del agente perseguidor.

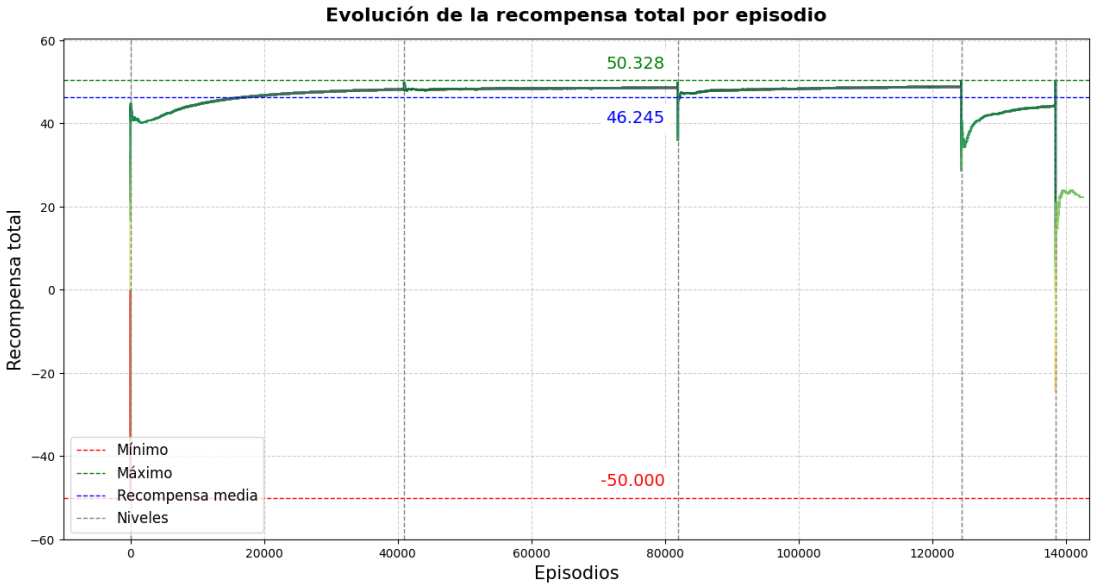

Figura 23. Evolución de la recompensa total por episodio sin la posición absoluta para un tamaño de la matriz de observación de 7×7.

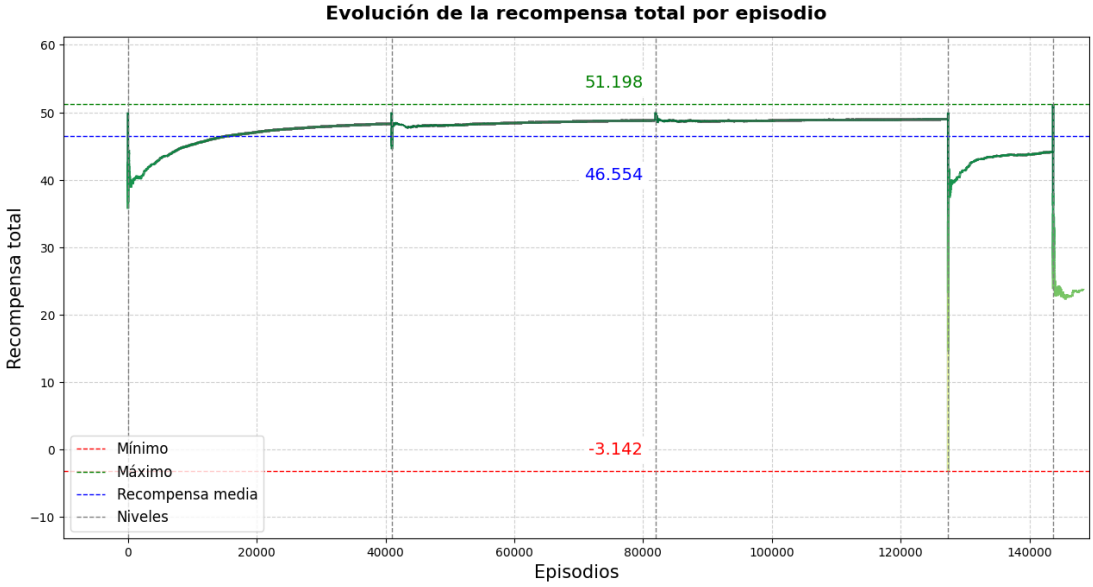

Figura 24. Evolución de la recompensa total por episodio sin la posición absoluta y sin la distancia euclidiana para un tamaño de la matriz de observación de 7×7.

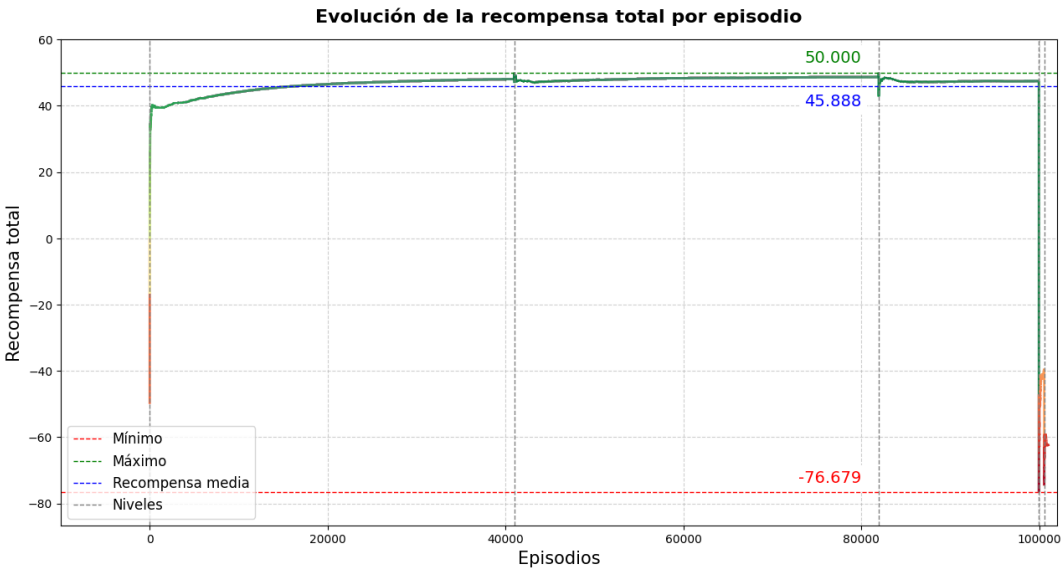

Figura 25. Evolución de la recompensa total por episodio sin la posición relativa para un tamaño de la matriz de observación de 7×7.

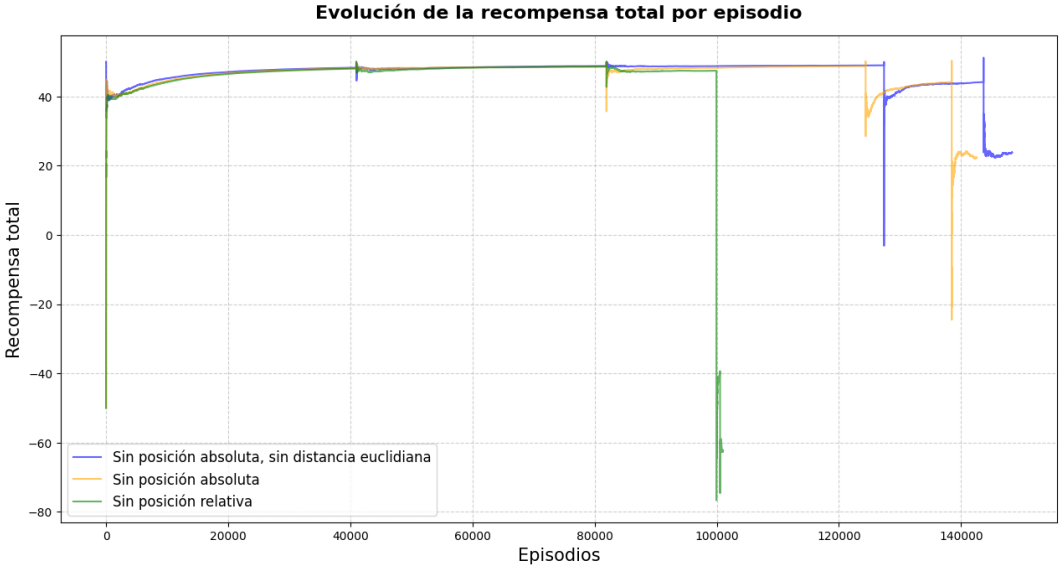

Figura 26. Comparación de la evolución de la recompensa total por episodio para las eliminaciones de las variables de posición y distancia.

En la Figura 26 se presenta la comparación de la recompensa promedio por episodios para cada una de las variantes con eliminación de variables. Se aprecia que, luego de la ventaja posicional inicial, la variante correspondiente a la eliminación de la posición absoluta y de la distancia euclidiana muestra

un comportamiento más estable, con menores caídas abruptas en la recompensa promedio por episodio. Esto sugiere que la inclusión de la posición absoluta puede introducir cierto grado de ruido en la toma de decisiones de la política.

4.1.5. Resultados y discusión de la evaluación

Las políticas obtenidas para las tres configuraciones con eliminación de variables de estado fueron evaluadas bajo las mismas condiciones. Los resultados se resumen en la Tabla 2, donde las celdas en verde señalan el mejor desempeño para cada nivel. En el caso del nivel 4, no se observa una diferencia significativa entre dos de las variantes; por lo tanto, ambas se destacan.

Como se aprecia, en el caso de la eliminación de la posición absoluta, así como en la eliminación conjunta de la posición absoluta y la distancia euclidiana, no se observa una diferencia significativa en el rendimiento, tanto en términos de tasa de éxito como de recompensa promedio.

Por otro lado, en el caso de la eliminación de la posición relativa, se reafirma la conclusión obtenida durante la etapa de entrenamiento: esta variable constituye el factor más importante para garantizar un buen desempeño del agente perseguidor.

Tabla 2. Comparación de las diferentes políticas aprendidas en las dificultades 3 y 4 para las diferentes eliminaciones de variable de estado.

Experimento	Dificultad	Tasa de éxito	Recompensa promedio
Experimento 1: tamaño 7x7, sin posición absoluta	3	0.93	175.82
Experimento 1: tamaño 7x7, sin posición absoluta	4	0.52	71.52
Experimento 1: tamaño 7x7, sin posición absoluta, sin distancia euclidiana	3	0.94	179.93
Experimento 1: tamaño 7x7, sin posición absoluta, sin distancia euclidiana	4	0.52	70.07
Experimento 1: tamaño 7x7, sin posición relativa	3	0.01	1.07
Experimento 1: tamaño 7x7, sin posición relativa	4	0.00	-1.86

4.2. Experimento 2: perceptrón multicapa con bloque extractor de características convolucional

En el segundo experimento se empleó la misma arquitectura base utilizada en el primer escenario, es decir, una red neuronal de tipo MLP, pero incorporando un bloque extractor de características con dos flujos paralelos. El primer flujo está compuesto por una serie de capas convolucionales, cuyo objetivo

es procesar directamente la matriz de observación como una matriz bidimensional, preservando así la estructura espacial de la información en lugar de aplanarla en un vector. El segundo flujo, en cambio, procesa el vector de distancia relativa entre los agentes con capas densas.

La salida de ambos flujos es posteriormente concatenada y pasada a través de capas densas totalmente conectadas como la utilizada en el primer experimento. Esta estrategia busca evaluar si la explotación de la estructura espacial contenida en la matriz de observación contribuye a un aprendizaje más eficiente y a la obtención de políticas más robustas. En particular, se busca determinar si el aprovechamiento de dichas características espaciales contribuye a mejorar el desempeño en el nivel de dificultad 4, donde el mejor modelo del primer experimento presentó un rendimiento limitado.

Además, para este experimento se realizó un ajuste de los parámetros de entrenamiento. En particular, se aplicó una reducción lineal de la tasa de aprendizaje tanto en función del progreso entre niveles como dentro de cada nivel. El objetivo de esta estrategia fue favorecer cambios más significativos en los pesos de la red durante las fases iniciales, donde la exploración resulta más relevante, y posteriormente reducir gradualmente dichos cambios con el fin de consolidar y explotar el conocimiento adquirido.

Adicionalmente, se llevaron a cabo dos ciclos de entrenamiento por cada nivel, con el propósito de otorgar mayor tiempo de aprendizaje y, en consecuencia, ampliar la fase de exploración. Con ello se buscó obtener políticas más robustas y mejor adaptadas en cada iteración.

Por último, se realizaron ajustes en la función de recompensa: la bonificación por capturar al evasor se fijó en 100 unidades, y se añadió una penalización de -2.0 unidades para evitar que el perseguidor quedara atrapado en un ciclo entre celdas sin progresar hacia el evasor.

4.2.1. Discusión del entrenamiento

En la Figura 27 se observa un comportamiento similar al registrado durante la etapa de entrenamiento del primer experimento. En los primeros niveles, el perseguidor logra capturar al evasor, debido a que este último presenta un comportamiento más aleatorio en sus decisiones, a la cercanía inicial entre los agentes, al rango de captura reducido y a la menor complejidad del entorno en términos de obstáculos. Sin embargo, a partir del segundo nivel la recompensa promedio comienza a disminuir, aunque se mantiene en valores positivos. Esto indica que persiste el problema observado en el primer experimento, el perseguidor localiza al evasor sin capturarlo.

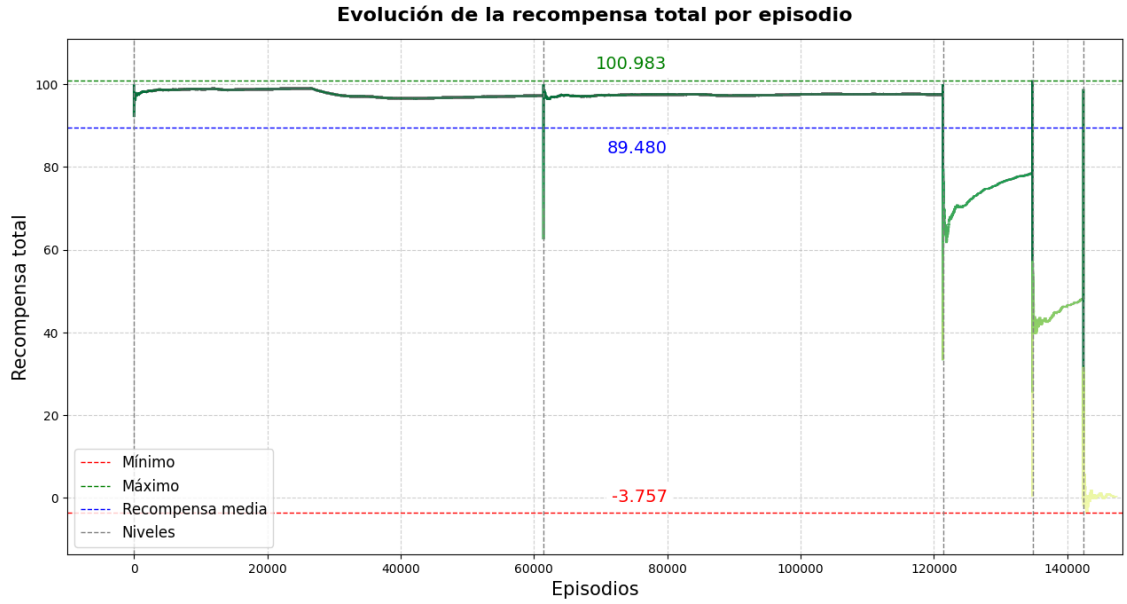

Figura 27. Evolución de la recompensa total por episodio para el segundo experimento para un tamaño de la matriz de observación de 7×7.

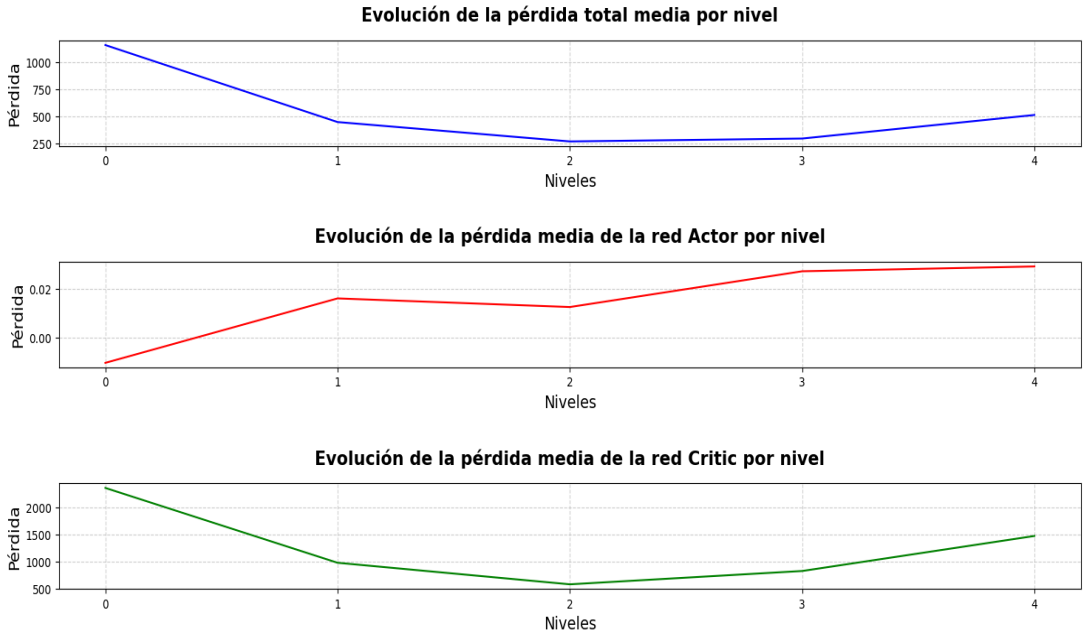

Figura 28. Evolución de la pérdida total media, pérdida media de la red Actor, y la pérdida media de la red Critic por nivel para el segundo experimento para un tamaño de la matriz de observación de 7×7.

En cuanto a las pérdidas por nivel, la Figura 28 muestra que en el nivel 2 se alcanza la menor pérdida tanto en la función total como en la red Crítica. En contraste, la pérdida asociada a la red Actor tiende

a incrementarse conforme avanzan los niveles. A partir de los niveles 3 y 4, las tres pérdidas comienzan a aumentar, lo que sugiere que el perseguidor transita con mayor frecuencia por estados poco favorables para maximizar la recompensa esperada.

4.2.2. Resultados y discusión de la evaluación

Las políticas obtenidas para el primer y segundo ciclo de aprendizaje fueron evaluadas y comparadas con el mejor modelo del primer experimento. Tal como se muestra en la Tabla 3, el segundo experimento presenta una mayor tasa de éxito en el nivel 4, tanto en el primer ciclo como en el segundo, en comparación con el primer experimento. No obstante, este incremento en la tasa de éxito no se ve reflejado en un mejor desempeño en términos de la recompensa promedio. Esto sugiere que la distribución de recompensas en los 100 episodios evaluados para las cinco semillas está sesgada hacia valores negativos, los cuales contrarrestan los valores positivos obtenidos en los episodios exitosos.

Los resultados se resumen en la Tabla 3, donde las celdas en verde indican el mejor desempeño para cada nivel. En el caso del nivel 4, el mejor desempeño se divide entre la mayor recompensa promedio obtenida en la variante del experimento 1 y la mayor tasa de éxito alcanzada en el segundo ciclo del experimento 2.

Tabla 3. Comparación de las diferentes políticas aprendidas en las dificultades 3 y 4 para el mejor modelo del experimento 1 y el modelo resultante del experimento 2.

Experimento	Dificultad	Tasa de éxito	Recompensa promedio
Experimento 1: tamaño 7x7, sin posición absoluta, sin distancia euclidiana	3	0.94	179.93
Experimento 1: tamaño 7x7, sin posición absoluta, sin distancia euclidiana	4	0.52	70.07
Experimento 2: primer ciclo	3	0.41	32.37
Experimento 2: primer ciclo	4	0.63	0.92
Experimento 2: segundo ciclo	3	0.37	51.45
Experimento 2: segundo ciclo	4	0.66	-8.39

4.3. Resumen de los resultados de los experimentos

A partir de los experimentos realizados, se pueden extraer varios hallazgos clave sobre el aprendizaje de políticas de captura en entornos de persecución-evasión. En primer lugar, la inclusión de la posición relativa entre los agentes resultó fundamental para el desempeño de la política, mientras que variables como la posición absoluta o la distancia euclidiana normalizada no aportaron mejoras significativas e

incluso pudieron introducir ruido en el aprendizaje.

En segundo lugar, el tamaño de la matriz de observación mostró que matrices de mayor dimensión proporcionan más información del entorno y pueden ser beneficiosas en escenarios complejos. Sin embargo, no se observó una correlación directa entre tamaño y desempeño final. En el primer experimento, la matriz 7×7 ofreció un equilibrio óptimo entre tasa de éxito y recompensa promedio.

En tercer lugar, para entornos más difíciles (nivel 4), una red MLP no logró un desempeño satisfactorio. La introducción de una arquitectura CNN, capaz de procesar la matriz de observación y explotar su estructura espacial, permitió mejorar la tasa de éxito, aunque con una disminución en la recompensa promedio.

Finalmente, a pesar de localizar al evasor y navegar sin colisiones, el perseguidor no siempre logró acercarse lo suficiente para efectuar la captura. Esto evidencia la necesidad de ajustes finos en los parámetros del algoritmo, así como un mayor número de episodios de entrenamiento y de situaciones diversas que pongan a prueba la política, permitiendo que el agente aprenda de errores específicos.

Capítulo 5. Conclusiones y trabajo futuro

En este proyecto se utilizó el aprendizaje por refuerzo profundo como paradigma de aprendizaje para entrenar una política que sirviera como solución a un problema de persecución–evasión, específicamente en los casos de captura y seguimiento.

A partir de los experimentos realizados se obtuvieron varios hallazgos relevantes. En primer lugar, se observó que la inclusión de variables como la posición relativa entre los agentes es esencial para el desempeño de la política, mientras que otras variables —como la posición absoluta o la distancia euclidiana normalizada— no aportaron mejoras significativas, e incluso pudieron introducir ruido en el proceso de aprendizaje.

En segundo lugar, no se evidenció una conclusión clara de que un mayor tamaño de la matriz de observación, pese a proveer más información del entorno en términos de navegación y localización, guíe efectivamente al agente hacia el objetivo final de captura, que es la fuente principal de recompensa. Estos hallazgos se observaron en el primer experimento, donde durante la etapa de entrenamiento un tamaño mayor favoreció la obtención de recompensas promedio más altas. Sin embargo, en la etapa de evaluación se manifestó un comportamiento distinto: el mejor desempeño correspondió al tamaño 7×7, el cual ofreció un equilibrio de desempeño entre tasa de éxito y recompensa total alcanzada.

Asimismo, en entornos más complejos —específicamente en el nivel 4— el mejor modelo basado en una red MLP no logró un desempeño satisfactorio. Para abordar esta limitación, se propuso una mejora en la arquitectura mediante la incorporación de capas convolucionales capaces de procesar la matriz de observación y aprovechar su estructura espacial. Los resultados mostraron que esta nueva arquitectura permitió incrementar la tasa de éxito en el nivel 4 en comparación con la MLP, aunque a costa de una disminución en la recompensa promedio alcanzada.

Adicionalmente, se identificaron limitaciones en la política aprendida: en los escenarios de evaluación, el agente perseguidor logró localizar al evasor, pero no siempre consiguió reducir la distancia necesaria para efectuar la captura. Este comportamiento evidencia una de las principales dificultades del aprendizaje por refuerzo: la dependencia de técnicas de ajustes de parámetros para mejorar la política. A ello se suma la incertidumbre respecto al tiempo de entrenamiento o la cantidad de datos requeridos para alcanzar una política verdaderamente efectiva.

En conjunto, los resultados de este trabajo demuestran que el aprendizaje por refuerzo profundo constituye una herramienta prometedora para abordar problemas de persecución–evasión. No obstante, también

se confirma que el diseño del estado, la definición de un sistema de recompensas, la configuración del entorno y el ajuste de los parámetros del algoritmo de aprendizaje son factores determinantes en la calidad de la política aprendida.

5.1. Trabajo futuro

Dado que los experimentos desarrollados en este proyecto se llevaron a cabo en un entorno bidimensional, una primera línea de exploración consistiría en extender la metodología propuesta hacia un escenario tridimensional. Esto permitiría evaluar la escalabilidad del enfoque y acercarse más a aplicaciones reales en robótica y simulación.

Por otra parte, el presente trabajo se centró en diseñar una política de movimiento para el perseguidor con el objetivo de seguir y capturar al evasor. Una dirección interesante sería desarrollar una política para el evasor orientada a evitar la captura, lo que transformaría el problema en un escenario de optimización mínimos-máximos (min-max). Este planteamiento abriría un campo de investigación más amplio y desafiante, con aplicaciones en estrategias competitivas y entornos multiagente.

Además, los agentes utilizados estuvieron limitados a un conjunto discreto de acciones y a un entorno representado mediante celdas. Una posible mejora sería el desarrollo de agentes con acciones continuas, capaces de representar configuraciones de movimiento más cercanas a robots reales.

Asimismo, se propone incorporar el *aprendizaje por imitación* como técnica complementaria. Este enfoque permitiría guiar el entrenamiento en situaciones en las que los agentes, por sí solos, no logran desarrollar una política efectiva. Una utilización de esta técnica es en casos donde el agente queda atrapado en ciclos de movimiento que resultan ventajosos en términos de recompensa inmediata, pero que impiden alcanzar el objetivo final.

En cuanto a las arquitecturas de red, además de las utilizadas en este trabajo, sería valioso explorar arquitecturas más avanzadas como las redes neuronales recurrentes (RNN), que permiten modelar dependencias temporales en la toma de decisiones.

Al extender la simulación hacia un entorno tridimensional también se abriría la posibilidad de incorporar un mayor número de sensores, lo cual facilitaría el uso de técnicas de fusión sensorial y, eventualmente, sistemas de recomendación para la toma de decisiones de los agentes.

Finalmente, dado que en este trabajo se emplearon métodos de aprendizaje *online*, es decir, entrenando en tiempo real mientras el agente interactúa con el entorno, una extensión sería explorar métodos de aprendizaje *offline*. En este caso, la política se entrenaría a partir de un conjunto de datos recopilados de experiencias previas de los agentes, lo cual resultaría particularmente útil para aplicaciones realistas.

Literatura citada

AlMahamid, F. & Grolinger, K. (2021). Reinforcement learning algorithms: An overview and classification. In *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 1–7. IEEE. https://doi.org/10.1109/CCECE53047.2021.9569056.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6), 26–38. https://doi.org/10.1109/MSP.2017.2743240.

Balkcom, D. J. & Mason, M. T. (2002). Time optimal trajectories for bounded velocity differential drive vehicles. *The International Journal of Robotics Research*, 21(3), 199–217. https://doi.org/10.1177/027836402320556403.

Bandyopadhyay, T., Li, Y., Ang, M., & Hsu, D. (2006). A greedy strategy for tracking a locally predictable target among obstacles. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2342–2347. IEEE. https://doi.org/10.1109/ROBOT.2006.1642052.

Bhattacharya, S. & Hutchinson, S. (2010). On the existence of Nash equilibrium for a two-player pursuit—evasion game with visibility constraints. *The International Journal of Robotics Research*, 29(7), 831–839. https://doi.org/10.1177/0278364909354628.

Boston-Dynamics (2023). Spot to the Rescue. https://bostondynamics.com/blog/spot-to-the-rescue/.

Boston-Dynamics (2024). A dynamic approach to industrial sensing. https://bostondynamics.com/blog/a-dynamic-approach-to-industrial-sensing.

Chikhaoui, K., Ghazzai, H., & Massoud, Y. (2022). PPO-based reinforcement learning for UAV navigation in urban environments. In *2022 IEEE 65th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 1–4. IEEE. https://doi.org/10.1109/MWSCAS54063.2022.9859287.

Chung, T. H., Hollinger, G. A., & Isler, V. (2011). Search and pursuit-evasion in mobile robotics: A survey. *Autonomous robots*, 31, 299–316. https://doi.org/10.1007/s10514-011-9241-4.

Del Rio, A., Jimenez, D., & Serrano, J. (2024). Comparative analysis of A3C and PPO algorithms in reinforcement learning: A survey on general environments. *IEEE Access*. https://doi.org/10.1109/ACCESS.2024.3472473.

Egorov, M. (2016). Multi-agent deep reinforcement learning. https://arxiv.org/pdf/1609.07084.pdf.

Fang, X., Cheng, C., & Xie, L. (2020). 3-d multi-player pursuit-evasion game with a faster evader. In *2020 39th Chinese Control Conference (CCC)*, 118–123. IEEE. https://doi.org/10.23919/CCC50068.2020.9188950.

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., Pineau, J., et al. (2018). An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4), 219–354. https://doi.org/10.1561/2200000071.

Gao, Y., Cheng, L., He, Y., & Wang, D. (2023). An active pursuit strategy for autonomous mobile robots based on deep reinforcement learning. In *2022 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, 326–331. IEEE. https://doi.org/10.1109/CBS55922.2023.10115383.

Gronauer, S. & Diepold, K. (2022). Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2), 895–943. https://doi.org/10.1007/s10462-021-09996-w.

Isaacs, R. (1999). *Differential games: a mathematical theory with applications to warfare and pursuit, control and optimization*. Courier Corporation.

Isler, V., Kannan, S., & Khanna, S. (2005). Randomized pursuit-evasion in a polygonal environment. *IEEE Transactions on Robotics*, 21(5), 875–884. https://doi.org/10.1109/TRO.2005.851373.

Jeong, H., Hassani, H., Morari, M., Lee, D. D., & Pappas, G. J. (2021). Deep Reinforcement Learning for Active Target Tracking. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 1825–1831. https://doi.org/10.1109/ICRA48506.2021.9561258.

Jung, B. & Sukhatme, G. S. (2002). Tracking targets using multiple robots: The effect of environment occlusion. *Autonomous robots*, 13, 191–205. https://doi.org/10.1023/A:1020598107671.

Károly, A. I., Galambos, P., Kuti, J., & Rudas, I. J. (2020). Deep learning in robotics: Survey on model structures and training strategies. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(1), 266–279. https://doi.org/10.1109/TSMC.2020.3018325.

Kehagias, A., Hollinger, G., & Singh, S. (2009). A graph search algorithm for indoor pursuit/evasion. *Mathematical and Computer Modelling*, 50(9-10), 1305–1317. https://doi.org/10.1016/j.mcm.2009.06.011.

Kolling, A. & Carpin, S. (2010). Multi-robot pursuit-evasion without maps. In *2010 IEEE International Conference on Robotics and Automation*, 3045–3051. IEEE. https://doi.org/10.1109/ROBOT.2010.5509318.

Kumar, A. S., Zhao, L., & Fernando, X. (2021). Multi-agent deep reinforcement learning-empowered channel allocation in vehicular networks. *IEEE Transactions on Vehicular Technology*, 71(2), 1726–1736. https://doi.org/10.1109/TVT.2021.3134272.

Laumond, J.-P. et al. (1998). *Robot motion planning and control*, volume 229. Springer.

LaValle, S. M., González-Banos, H. H., Becker, C., & Latombe, J.-C. (1997). Motion strategies for maintaining visibility of a moving target. In *Proceedings of International Conference on Robotics and Automation*, volume 1, 731–736. IEEE. https://doi.org/10.1109/ROBOT.1997.620122.

Lee, E. S., Shishika, D., & Kumar, V. (2020). Perimeter-defense game between aerial defender and ground intruder. In *2020 59th IEEE conference on decision and control (CDC)*, 1530–1536. IEEE. https://doi.org/10.1109/CDC42340.2020.9304213.

Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*. https://doi.org/10.48550/arXiv.1701.07274.

Li, Z. & Canny, J. (1990). Motion of two rigid bodies with rolling constraint. *IEEE Transactions on Robotics and Automation*, 6(1), 62–72. https://doi.org/10.1109/70.88118.

Liberzon, D. (2011). *Calculus of variations and optimal control theory: a concise introduction*. Princeton university press.

Lozano, E., Becerra, I., Ruiz, U., Bravo, L., & Murrieta-Cid, R. (2022a). A visibility-based pursuit-evasion game between two nonholonomic robots in environments with obstacles. *Autonomous Robots*, 46(2), 349–371. https://doi.org/10.1007/s10514-021-10026-5.

Lozano, E., Ruiz, U., Becerra, I., & Murrieta-Cid, R. (2022b). Surveillance and Collision-Free Tracking of an Aggressive Evader With an Actuated Sensor Pursuer. *IEEE Robotics and Automation Letters*, 7(3), 6854–6861. https://doi.org/10.1109/LRA.2022.3178799.

Ma, H., Dong, D., Ding, S. X., & Chen, C. (2022). Curriculum-based deep reinforcement learning for quantum control. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11), 8852–8865. https://doi.org/10.1109/TNNLS.2022.3153502.

Ma, J., Lu, H., Xiao, J., Zeng, Z., & Zheng, Z. (2020). Multi-robot target encirclement control with collision avoidance via deep reinforcement learning. *Journal of Intelligent & Robotic Systems*, 99, 371–386. https://doi.org/10.1007/s10846-019-01106-x.

Matsuo, Y., LeCun, Y., Sahani, M., Precup, D., Silver, D., Sugiyama, M., Uchibe, E., & Morimoto, J. (2022). Deep learning, reinforcement learning, and world models. *Neural Networks*, 152, 267–275. https://doi.org/10.1016/j.neunet.2022.03.037.

Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540), 529–533. https://doi.org/10.1038/nature14236.

Mock, J. W. & Muknahallipatna, S. S. (2024). Sim-to-real: A performance comparison of PPO, TD3, and SAC reinforcement learning algorithms for quadruped walking gait generation. *Journal of Intelligent Learning Systems and Applications*, 16(2), 23–43. https://doi.org/10.4236/jilsa.2024.162003.

Murray, R. M. & Sastry, S. S. (1993). Nonholonomic motion planning: Steering using sinusoids. *IEEE transactions on Automatic Control*, 38(5), 700–716. https://doi.org/10.1109/9.277235.

Murrieta-Cid, R., Ruiz, U., Marroquin, J. L., Laumond, J.-P., & Hutchinson, S. (2011). Tracking an omnidirectional evader with a differential drive robot. *Autonomous Robots*, 31, 345–366. https://doi.org/10.1007/s10514-011-9246-z.

Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., & Stone, P. (2020). Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181), 1–50. https://doi.org/10.5555/3455716.3455897.

Parras, J., del Val, J., Zazo, S., Zazo, J., & Macua, S. V. (2016). A new approach for solving anti-jamming games in stochastic scenarios as pursuit-evasion games. In *2016 IEEE Statistical Signal Processing Workshop (SSP)*, 1–5. IEEE. https://doi.org/10.1109/SSP.2016.7551804.

Pierson, H. A. & Gashler, M. S. (2017). Deep learning in robotics: a review of recent research. *Advanced Robotics*, 31(16), 821–835. https://doi.org/10.1080/01691864.2017.1365009.

Qi, Q., Zhang, X., & Guo, X. (2020). A deep reinforcement learning approach for the pursuit evasion game in the presence of obstacles. *IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 68–73. https://doi.org/10.1109/RCAR49640.2020.9303044.

Roach, J. (2022). The quest to deploy autonomous robots within amazon fulfillment centers. https://www.amazon.science/latest-news/the-quest-to-deploy-autonomous-robots-within-amazon-fulfillment-centers.

Robin, C. & Lacroix, S. (2016). Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, 40(4), 729–760. https://doi.org/10.1007/s10514-015-9491-7.

Ruiz, U. (2023). Time-optimal Escape of an Omnidirectional Agent from the Field of View of a Differential Drive Robot. *International Journal of Control, Automation and Systems*, 21(1), 292–305. https://doi.org/10.1007/s12555-021-0686-8.

Saglam, B., Mutlu, F. B., Cicek, D. C., & Kozat, S. S. (2023). Actor prioritized experience replay. *Journal of Artificial Intelligence Research*, 78, 639–672. https://doi.org/10.1613/jair.1.14819.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, 1889–1897. PMLR. https://doi.org/10.48550/arXiv.1502.05477.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*. https://doi.org/10.48550/arXiv.1707.06347.

Serra-Gómez, Á., Montijano, E., Böhmer, W., & Alonso-Mora, J. (2023). Active Classification of Moving Targets with Learned Control Policies. *IEEE Robotics and Automation Letters*. https://doi.org/10.1109/LRA.2023.3271508.

Sutton, R. S., Barto, A. G., et al. (1998). *Introduction to reinforcement learning*. MIT press Cambridge.

Taheri, H., Hosseini, S. R., & Nekoui, M. A. (2024). Deep reinforcement learning with enhanced PPO for safe mobile robot navigation. *arXiv preprint arXiv:2405.16266*. https://doi.org/10.48550/arXiv.2405.16266.

Tai, L., Zhang, J., Liu, M., Boedecker, J., & Burgard, W. (2016). A survey of deep network solutions for learning control in robotics: From reinforcement to imitation. *arXiv preprint arXiv:1612.07139*. https://doi.org/10.48550/arXiv.1612.07139.

Tang, C., Abbatematteo, B., Hu, J., Chandra, R., Martín-Martín, R., & Stone, P. (2025). Deep reinforcement learning for robotics: A survey of real-world successes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 28694–28698. https://doi.org/10.1609/aaai.v39i27.35095.

Tiritiris, P., Passalis, N., & Tefas, A. (2021). Temporal Difference Rewards for End-to-end Vision-based Active Robot Tracking using Deep Reinforcement Learning. In *2021 International Conference on Emerging Techniques in Computational Intelligence (ICETCI)*, 21–25. https://doi.org/10.1109/ICETCI51973.2021.9574071.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30. https://doi.org/10.1609/aaai.v30i1.10295.

Wang, X., Chen, Y., & Zhu, W. (2021). A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9), 4555–4576. https://doi.org/10.1109/TPAMI.2021.3069908.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, 1995–2003. PMLR. https://doi.org/10.48550/arXiv.1511.06581.

Xiang, X. & Foo, S. (2021). Recent advances in deep reinforcement learning applications for solving partially observable markov decision processes (pomdp) problems: Part 1—fundamentals and applications in games, robotics and natural language processing. *Machine Learning and Knowledge Extraction*, 3(3), 554–581. https://doi.org/10.3390/make3030029.

Zhang, W., Song, K., Rong, X., & Li, Y. (2018). Coarse-to-fine UAV target tracking with deep reinforcement learning. *IEEE Transactions on Automation Science and Engineering*, 16(4), 1522–1530. https://doi.org/10.1109/TASE.2018.2877499.

Zhong, F., Sun, P., Luo, W., Yan, T., & Wang, Y. (2019). Ad-vat+: An asymmetric dueling mechanism for learning and understanding visual active tracking. *IEEE transactions on pattern analysis and machine intelligence*, 43(5), 1467–1482. https://doi.org/10.1109/TPAMI.2019.2952590.

Zhou, D., Sun, G., Zhang, Z., & Wu, L. (2023). On Deep Recurrent Reinforcement Learning for Active Visual Tracking of Space Noncooperative Objects. *IEEE Robotics and Automation Letters*, 8(8), 4418–4425. https://doi.org/10.1109/LRA.2023.3282792.

Zhuang, Z., Fu, Z., Wang, J., Atkeson, C., Schwertfeger, S., Finn, C., & Zhao, H. (2023). Robot parkour learning. *arXiv preprint arXiv:2309.05665*. https://doi.org/10.48550/arXiv.2309.05665.

Zou, R. & Bhattacharya, S. (2016). Visibility-Based Finite-Horizon Target Tracking Game. *IEEE Robotics and Automation Letters*, 1(1), 399–406. https://doi.org/10.1109/LRA.2016.2521429.