CEA Comité Español de Automática
XLVII JORNADAS DE AUTOMÁTICA CÓRDOBA 2026
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Exploración activa autónoma usando *ROS2* y Aprendizaje por Refuerzo

Javier Arambarri-Calvo[a], Raul Fernandez-Fernandez[b,*], Jesús Chacón[b]

[a]*Departamento de Ingeniería de Sistemas y Automática, Escuela de Ingeniería de Bilbao, Universidad del País Vasco (EHU), 48013, Bilbao, España.*
[b]*Departamento de Arquitectura de Computadores y Automática, Universidad Complutense de Madrid (UCM), Plaza de las Ciencias 1, 28040, Madrid, España.*

To cite this article: Arambarri-Calvo, J., Fernandez-Fernandez, R., Chacon, J. 2026. Autonomous active exploration using *ROS2* and reinforcement learning. Jornadas de Automática, 47. https://doi.org/10.17979/ja-cea.2026.47.13735

Resumen

Este trabajo presenta un sistema autónomo de exploración activa y localización simultánea (*A-SLAM*) basado en Aprendizaje por Refuerzo Profundo para un robot diferencial. Para su implementación en entornos continuos se usa el algoritmo *TD3*, que permite optimizar trayectorias que maximicen la completitud del mapa en entornos desconocidos usando políticas de control paramétricas continuas. Proponemos una arquitectura completa que integra *TD3* con *ROS2*, el robot *iCreate3* y un sensor *RPLIDAR-C1*. Esta arquitectura utiliza un vector de estado que integra lecturas del *LiDAR*, una representación dual del mapa —egocéntrica local de ocupación y de exploración global— e información de la posición del robot. La información del *LiDAR* y de los mapas es procesada mediante redes neuronales convolucionales para reducir su dimensionalidad y extraer las características espaciales más relevantes. El sistema se implementó en *ROS2 Humble* y se validó en el simulador *Gazebo Classic*. Los resultados obtenidos tras 11.000 episodios entrenados alcanzaron exploraciones superiores al 95 % en la completitud del mapa.

Palabras clave: Control por aprendizaje por refuerzo, Fusión de información y sensores, Sistemas robóticos autónomos, Robots móviles autónomos, Robótica inteligente, Creación de mapas.

Autonomous active exploration using *ROS2* and Reinforcement Learning

Abstract

This paper presents an autonomous active simultaneous exploration and localization system (A-SLAM) based on Deep Reinforcement Learning for a differential robot. For implementation in continuous environments, the *TD3* algorithm is used to optimise trajectories that maximise map completeness in unknown environments using continuous parametric control policies. We propose a complete architecture that integrates *TD3* with *ROS2*, the *iCreate3* robot, and an *RPLIDAR-C1* sensor. This architecture uses a state vector that integrates *LiDAR* readings, a dual map representation —a local egocentric representation of occupancy and a global exploration representation— and robot position information. The *LiDAR* and map information is processed using convolutional neural networks to reduce its dimensionality and extract the most relevant spatial features. The system was implemented in *ROS2 Humble* and validated in the *Gazebo Classic* simulator. The results obtained after 11,000 training episodes achieved explorations exceeding 95 % in map completeness.

Keywords: Reinforcement learning control, Information and sensor fusion, Autonomous robotic systems, Autonomous Mobile Robots, Intelligent robotics, Map building.

1. Introducción

La exploración y el mapeo autónomos constituyen uno de los desafíos abiertos más relevantes en la robótica móvil inteligente (Alcalde et al., 2022). En entornos desconocidos, donde no se dispone de sistemas de localización externa ni de mapas previos, el robot debe construir un mapa del entorno mientras estima simultáneamente su propia posición, proceso conocido

*Autor para correspondencia: raufer06@ucm.es

como Localización y Mapeo Simultáneos (*Simultaneous Localization and Mapping -SLAM-*) (Thrun et al., 2005).

Sin embargo, los enfoques tradicionales de *SLAM*, comúnmente denominados *SLAM* pasivo, no incorporan una planificación explícita de la trayectoria del robot dentro del problema a optimizar y, por tanto, ignoran cómo sus acciones afectan a la precisión de la localización y a la calidad del mapa. Métodos reactivos como la exploración por fronteras emplean el mapa parcial para seleccionar objetivos de navegación (Topiwala et al., 2018; Cimurs et al., 2021; Wang et al., 2024), pero no consideran la incertidumbre futura ni el valor informativo de los posibles movimientos.

Para mitigar estas limitaciones surge el paradigma del *SLAM* activo (*A-SLAM*) (Placed and Castellanos, 2020), que integra la toma de decisiones en el proceso de estimación con el fin de reducir la incertidumbre tanto de la posición como del mapa. El reto consiste en equilibrar el progreso de la exploración con la reducción de la incertidumbre, lo que requiere evaluar de forma continua el impacto de las acciones sobre la estimación futura. Este planteamiento se ajusta de manera natural al marco del aprendizaje por refuerzo (*Reinforcement Learning –RL–*) (Sutton and Barto, 2018), dado su carácter secuencial y dependiente del estado.

En la actualidad, es habitual abordar el *A-SLAM* mediante algoritmos de aprendizaje por refuerzo profundo (*Deep Reinforcement Learning –DRL–*), seleccionados en función de la complejidad del entorno y del espacio de acciones. En este trabajo se desarrolla una primera aproximación a un sistema de exploración activa basado en *DRL* empleando un robot diferencial *iCreate3* en el simulador *Gazebo Classic* bajo *ROS2 Humble* (Macenski et al., 2022). Para la toma de decisiones se adopta el algoritmo *Twin Delayed Deep Deterministic Policy Gradient –TD3–* (Fujimoto et al., 2018), que mejora la estabilidad de métodos como *Deep Deterministic Policy Gradient –DDPG–* (Lillicrap et al., 2015) y permite operar de forma robusta en espacios de acción continuos. La Figura 1 muestra una descripción visual del enfoque propuesto. En la zona inferior izquierda se observa el escenario en el que se ha trabajado.

Este artículo se organiza de la siguiente manera. En la Sección 2 se presenta el estado del arte. En la Sección 3 se describe el método desarrollado para abordar el problema. En la Sección 4 se muestran los resultados obtenidos. Finalmente, en la Sección 5 se detallan las conclusiones alcanzadas y las líneas de trabajo futuro.

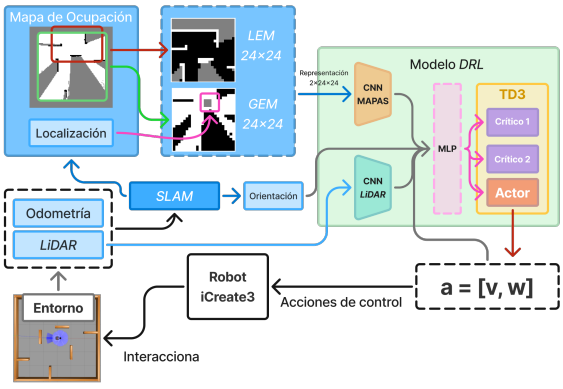

Figura 1: Modelo de *A-SLAM* propuesto basado en *DRL*.

2. Estado del arte

2.1. Evolución de los algoritmos de DRL en SLAM activo

La evolución del *A-SLAM* ha estado marcada por la transición desde métodos basados en fronteras (Topiwala et al., 2018) hacia enfoques de *DRL* capaces de razonar bajo incertidumbre. Los primeros trabajos emplearon *A3C* o *DQN* para seleccionar objetivos de exploración mediante recompensas basadas en ganancia de información o entropía (Niroui et al., 2019; Li et al., 2020; Chen et al., 2020). Posteriormente, algoritmos continuos como *TD3* mejoraron la estabilidad y permitieron políticas en espacios de acción continuos (Cimurs et al., 2021). Enfoques recientes integran *PPO* o *DDPG* con recompensas que combinan completitud del mapa, reducción de incertidumbre y cierre de bucles (Alcalde et al., 2022; Zhao and Hwang, 2024), así como representaciones multicanal del mapa del entorno que enriquecen el estado (Li et al., 2023).

2.2. Redes CNN y LSTM en algoritmos DRL y metodología

La representación del entorno mediante mapas 2D genera vectores de estado de alta dimensionalidad difíciles de manejar por algoritmos de *DRL*. Para extraer características relevantes se emplean redes neuronales convolucionales (*Convolutional Neural Network –CNN–*) (Li et al., 2023; Fernandez-Fernandez et al., 2023). Asimismo, la observabilidad parcial inherente al problema se aborda mediante redes recurrentes como la memoria larga a corto plazo (*Long Short Term Memory –LSTM–*) (Niroui et al., 2019), que incorporan la historia reciente de estados y acciones. Un ejemplo destacado es *DDPG-LSTM*, que combina memoria y estado instantáneo para obtener trayectorias más estables (Arroyo de la Torre and Barea Navarro, 2021). En cuanto al entrenamiento, herramientas como *Isaac Sim* e *Isaac Lab* permiten aprendizaje masivo y paralelizado (Llobera et al., 2026), mientras que en Li et al. (2023) se introduce el enfoque *Cumulative Curriculum Reinforcement Learning –CCRL–*, que incrementa progresivamente la complejidad del entorno para mejorar la eficiencia de muestreo y la generalización.

3. Método propuesto

3.1. Definición del problema

Se considera un Proceso de Decisión de Markov (*Markov Decision Process –MDP–*) (Sutton and Barto, 2018; Du et al., 2025), definido por la tupla

$$\mathcal{M} = (S, A, P, R, \gamma), \tag{1}$$

donde S representa el conjunto de estados del sistema, A el conjunto de acciones disponibles, $P(s' \mid s, a)$ la probabilidad de transición entre estados, R la función de recompensa inmediata asociada a cada transición y γ el factor de descuento que pondera la contribución de las recompensas futuras.

En los métodos de control por diferencia temporal (*Temporal Difference –TD–*) de naturaleza *off-policy*, el objetivo es satisfacer la ecuación de optimidad de Bellman para la función de valor de acción óptima $q^*(s, a)$ (Sutton and Barto, 2018). No obstante, en espacios de acciones continuas, la resolución de la ecuación clásica resulta computacionalmente inmanejable, por lo que se emplean arquitecturas de tipo actor-crítico.

En este esquema, el máximo de la función de valor no se busca mediante optimización iterativa, sino que se aproxima mediante una función denominada actor (π_ϕ). El actor aprende a mapear directamente los estados a la acción que se estima que maximiza la función de valor proporcionada por otra red, denominada crítico (Q_θ) (Lillicrap et al., 2015). De este modo, la regla de actualización de la función de valor se redefine utilizando la inferencia del actor para determinar la mejor acción futura:

$$y = R_{t+1} + \gamma Q(S_{t+1}, \pi(S_{t+1})) \tag{2}$$

donde $\pi(S_{t+1})$ actúa como el aproximador del operador $\max_{a'}$ sobre el espacio continuo.

3.2. Estado del entorno

En cada paso de tiempo t, el agente percibe un estado S_t que integra la siguiente información del entorno:

- **Datos del *LiDAR*:** 500 lecturas de distancia distribuidas uniformemente en un barrido de 360° alrededor del robot, denotadas como $L_t \in \mathbb{R}^{500}$. Esta nube de puntos proporciona una percepción de los obstáculos cercanos.
- **Representación dual del mapa:** se emplea la *SLAM Toolbox* (Macenski and Jambrecic, 2021) para obtener dos representaciones espaciales complementarias (Li et al., 2023), Figura 2:

  Mapa Egocéntrico Local (LEM). Se construye extrayendo una región del mapa de ocupación global, siempre centrada en la posición estimada del robot (x_t, y_t). La ventana extraída se reduce posteriormente a una resolución fija de 24×24 píxeles mediante un submuestreo por selección de máximo. Esta ventana puede contener más o menos información del mapa original según un factor de escala $s > 1$, que en nuestra aplicación es $s = 3,5$. El *LEM* se denota $M_t^{LEM} \in \mathbb{R}^{24\times24}$ y las celdas se codifican como 0 para espacio libre, 128 para desconocido y 255 para ocupado.

  Mapa de Exploración Global (GEM). Proporciona una visión global del progreso de la exploración. Se genera calculando el *minimum bounding box* de la región ya explorada en el mapa de ocupación global. Este mapa se redimensiona a una resolución fija de 24×24 píxeles mediante una operación de *Max Pooling*. En esta representación, el valor 0 denota zonas inexploradas, 255 áreas ya conocidas y se marca la posición relativa del robot con un valor de 128 en un cuadrado de 3×3 píxeles. El *GEM* se denota $M_t^{GEM} \in \mathbb{R}^{24\times24}$.

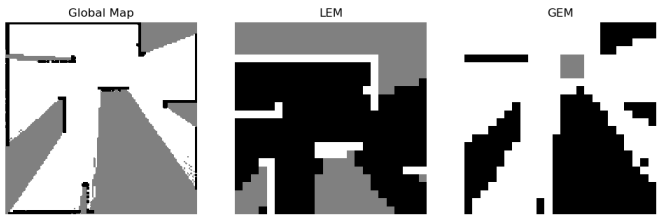

Figura 2: Mapas definidos dentro de la arquitectura propuesta. A la izquierda, el mapa original. En el centro, el *LEM*. A la derecha, el *GEM*.

- **Orientación y cinemática:** la orientación *yaw* del robot, denotada como ψ, se descompone en sus componentes trigonométricas $[\sin(\psi), \cos(\psi)]$ para evitar discontinuidades y se incorporan las acciones previas $a_{t-1} = [v_{t-1}, \omega_{t-1}]$, es decir, la velocidad lineal y angular.

El vector de estado resultante S_t en el instante t está formado por un total de 1656 elementos:

$$S_t = \left[L_t, M_t^{LEM}, M_t^{GEM}, \sin(\psi_t), \cos(\psi_t), v_{t-1}, \omega_{t-1}\right] \tag{3}$$

3.3. Espacio de Acciones

El agente controla el robot mediante un espacio de acciones continuo de dos dimensiones:

$$a_t = [v_t, \omega_t]. \tag{4}$$

Cada acción generada por la política está normalizada en el intervalo $[-1, 1]$ y corresponde a:

- **Velocidad lineal** $v_t \in [-1, 1]$, que se transforma a una velocidad física en el rango real $[0, 0{,}3]$ m/s.
- **Velocidad angular** $\omega_t \in [-1, 1]$, que concuerda con el rango físico $[-1, 1]$ rad/s.

3.4. Algoritmo TD3

El método de aprendizaje empleado es *TD3* (Fujimoto et al., 2018), un algoritmo actor–crítico *off-policy* diseñado para espacios de acción continuos que aborda el problema del sesgo por sobreestimación de los valores Q y es capaz de aprender de experiencias antiguas. Está basado en el algoritmo *DDPG* (Lillicrap et al., 2015) con tres diferencias principales que ayudan a la convergencia:

1. ***Clipped Double Q-learning:*** utiliza dos redes críticas independientes $(Q_{\theta_1}, Q_{\theta_2})$ y toma el valor mínimo entre ambas para calcular el objetivo del aprendizaje.
2. **Actualización Retrasada de la Política:** evita que el agente visite los mejores valores de Q demasiado rápido. Los parámetros del actor y de todas las redes objetivo se actualizan con una frecuencia menor que la de los críticos.
3. **Suavizado de la Política Objetivo:** se añade ruido aleatorio a las acciones seleccionadas por la red objetivo del actor, que impide que el sistema sobreajuste a picos estrechos en la función de valor Q.

Para estabilizar el entrenamiento, los críticos calculan el valor objetivo, Ecuación 2, empleando redes objetivo ($Q_{\theta_i'}$ y $\pi_{\phi'}$) actualizadas mediante *soft update*, que evitan oscilaciones bruscas y mejoran la estabilidad del aprendizaje:

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i', \qquad \phi' \leftarrow \tau\phi + (1-\tau)\phi', \tag{5}$$

donde θ_i y ϕ son los parámetros actuales de los críticos y del actor, θ_i' y ϕ' son los parámetros de sus respectivas redes objetivo y $\tau \in (0, 1)$ es el coeficiente de suavizado que controla cuánto se acercan las redes objetivo a las redes principales.

En la Tabla 1 se muestran los valores de los parámetros utilizados para entrenar el algoritmo. El factor de descuento $\gamma = 0{,}99$ controla la importancia de las recompensas futuras respecto de las inmediatas, mientras que la tasa de aprendizaje

de 0,001 fue escogida para evitar oscilaciones en la optimización de las redes. El coeficiente de suavizado $\tau = 0{,}003$ garantiza actualizaciones suaves de las redes objetivo porque los valores de sus parámetros se desplazan únicamente un 0,3 % de la diferencia existente respecto a los valores de la red principal. El tamaño del *batch* (128) y la capacidad del *replay buffer* (10^6 transiciones) fueron escogidos de manera iterativa teniendo en cuenta que un episodio son máximo 400 pasos (*timeout* = $40s$), y cada paso es $0{,}1s$, que es la frecuencia de trabajo del *LiDAR*. El ruido *gaussiano* de 0,2 y su recorte a ±0,5 se eligieron para limitar el ruido a un máximo del 50 % del rango de acción. En la práctica, el ruido se genera como $\mathcal{N}(0, 0{,}2^2)$, se trunca a ±0,5, se suma a la acción objetivo y la acción resultante se vuelve a limitar al intervalo $[-1, 1]$. Siguiendo la estrategia de actualización retrasada de la política, el actor se actualiza cada 4 actualizaciones de los críticos. Finalmente, el número de pasos de exploración inicial utilizados para poblar el *replay buffer* antes de comenzar los episodios de entrenamiento de las redes se fija en 15.000. Por tanto, esta fase inicial durará unos ($0{,}1 \cdot 15.000$) segundos.

Tabla 1: Parámetros de configuración del algoritmo *TD3*.

Parámetro	**Valor**
Factor de descuento (γ)	0,99
Tasa de aprendizaje (*Learning Rate*)	0,001
Coeficiente de suavizado objetivo (τ)	0,003
Tamaño de *batch*	128
Tamaño del *buffer* de repetición	1.000.000
Ruido *gaussiano* de la política (*Policy Noise*)	0,2
Clip de ruido de la política	0,5
Frecuencia de actualización de la política (d)	4
Tiempo de paso (*Step time*)	0,1 s
Pasos de observación inicial	15.000
Resolución *SLAM*	0,05 m
Intervalo de actualización del mapa	1 s

3.4.1. Reducción de dimensionalidad del estado

El estado definido presenta una dimensionalidad muy elevada con 1656 valores. La información del mapa y el *LiDAR* se procesa mediante dos redes convolucionales, Tabla 2, que actúan como extractores de características espaciales relevantes y reducen la dimensionalidad. Las salidas de ambas redes se concatenan con el resto de variables escalares restantes, formando un vector de características compacto que se introduce en un perceptrón multicapa (*MLP*). Este *MLP* está formado por dos capas densas de tamaño 512 con activación *ReLU* y una capa de salida *Q* de tamaño 1 con activación *tanh*.

En la primera red convolucional se reciben los 1×500 valores del *LiDAR* y se procesan mediante una convolución unidimensional que aplica 16 filtros con un *kernel* de tamaño 7 y un *stride* de 3. Tras esta operación se obtienen 16×164 valores. A continuación, una segunda convolución reduce los 16 canales a un único canal, condensando la información relevante en 1×159 valores. Finalmente, una capa de *Adaptive Max Pooling* ajusta la salida a una dimensión fija de 20 valores. Esta compresión no es arbitraria, ya que sintetizar los 360º del *LiDAR* en 20 elementos implica una resolución angular del espacio latente de características de 18º, lo que consideramos suficiente para la aplicación planteada.

La segunda red procesa los dos mapas de 24×24 píxeles cada uno, por lo que recibe 2 canales de entrada. La primera convolución aplica 16 filtros con un *kernel* 3×3 y un *stride* de 2, desplazando el filtro dos píxeles en cada dirección. Esto reduce la resolución espacial a la mitad, produciendo una salida de tamaño $16 \times 12 \times 12$. La segunda convolución reduce los canales de 16 a 2, generando una representación de tamaño $2 \times 6 \times 6$. Posteriormente, una capa de *Adaptive Max Pooling* normaliza la salida a un tamaño fijo de $2 \times 5 \times 5$, es decir, 25 valores por cada mapa antes de aplanarlos.

Tabla 2: Arquitecturas de las *CNN* para *LiDAR* y mapas.

CNN LiDAR (1D)	**CNN Mapas (2D)**
Conv1d(1,16, k=7, s=3)	Conv2d(2,16, k=3, s=2, p=1)
ReLU()	ReLU()
Conv1d(16,1, k=6, s=1, p=1)	Conv2d(16,2, k=2, s=2, p=1)
ReLU()	ReLU()
AdaptiveMaxPool1d(20)	AdaptiveMaxPool2d((5,5))
Flatten()	Flatten()

3.5. Función de recompensa

Se define una función multiobjetivo que equilibra el control cinemático, la evitación de obstáculos y el progreso de la exploración. La recompensa total en el instante t se formula:

$$R_t = \frac{r_{ctrl} + r_{obs} + r_{expl} + r_{time}}{1000} + R_{term}, \tag{6}$$

donde r_{ctrl} es el incentivo al control, r_{obs} a la esquivación de obstáculos, r_{expl} a la exploración, r_{time} al tiempo y R_{term} al evento terminal.

3.5.1. Incentivos de Movimiento y Control

Se penalizan tanto las velocidades angulares elevadas como las desviaciones respecto a una velocidad lineal objetivo $v_{target} = 0{,}3$ m/s. Las penalizaciones se definen como:

$$r_\omega = -3{,}0 \cdot \omega_t^2, \qquad r_v = -3{,}0 \cdot [10{,}0 \cdot (0{,}3 - v_t)]^2, \tag{7}$$

de modo que la contribución total de control queda dada por:

$$r_{ctrl} = r_\omega + r_v. \tag{8}$$

3.5.2. Evitación de Obstáculos y Seguridad

Se introduce una penalización reactiva inmediata para mantener al robot alejado de estructuras con las que pueda colisionar. Si la distancia mínima medida por el *LiDAR* (d_{min}) es inferior a un umbral de seguridad de 0,22 m, se aplica una penalización severa:

$$r_{obs} = \begin{cases} -20{,}0 & \text{si } d_{min} < 0{,}22 \\ 0{,}0 & \text{en otro caso} \end{cases} \tag{9}$$

3.5.3. Progreso de la Exploración

Siguiendo la metodología propuesta por Li et al. (2023), la recompensa de exploración se basa en el incremento de la tasa de completitud del mapa ρ, esto es, el porcentaje de celdas conocidas del mapa. Se define $\Delta\rho^2 = (\rho_t - \rho_{t-1})^2$ para medir la ganancia de información:

$$r_{expl} = \begin{cases} \text{mín}(5{,}0 \cdot \Delta\rho^2, 1{,}0) \cdot 100{,}0 & \text{si } \Delta\rho^2 > 0 \\ -0{,}5 & \text{en otro caso} \end{cases} \tag{10}$$

3.5.4. *Penalización por tiempo*

Todos los pasos reciben una penalización constante para minimizar el tiempo de ejecución y favorecer la eficiencia:

$$r_{time} = -1{,}0 \tag{11}$$

3.5.5. *Recompensas por Eventos Terminales*

Finalmente, se asigna un valor R_{term} basado en el estado final del episodio para reforzar los comportamientos de éxito frente a los fallos críticos:

$$R_{term} = \begin{cases} +5{,}0 & \text{si exito } (\rho \geq 95\,\%) \\ -2{,}0 & \text{si colisión o vuelco} \\ -1{,}0 & \text{si tiempo agotado } (timeout = 40s) \end{cases} \tag{12}$$

3.6. *Arquitectura ROS2*

La implementación del método propuesto se ha llevado a cabo en *ROS2* (Macenski et al., 2022), tal y como se presenta en la Figura 3. Los círculos representan los nodos y las flechas negras, azules y verdes los procesos internos, los tópicos y los servicios, respectivamente.

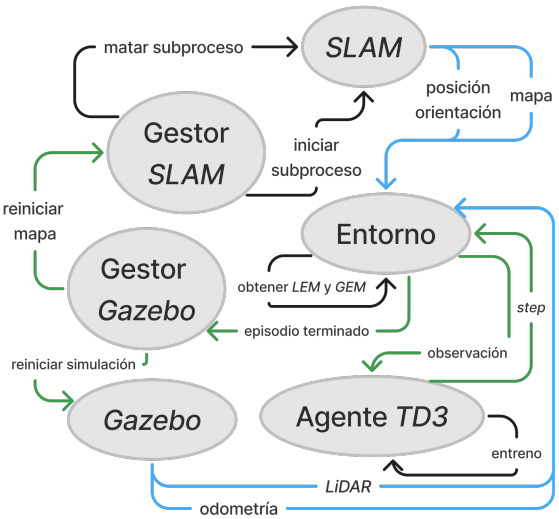

Figura 3: Esquema *ROS2* de la implementación realizada.

4. Resultados

El entrenamiento realizado consta de 11.000 episodios. Se ha evaluado el desenlace de los episodios junto al comportamiento del robot en la simulación, la pérdida del crítico y del actor y las recompensas obtenidas. Los resultados mostrados consisten en 500 episodios utilizando el actor entrenado sobre el mismo escenario, donde el 77,8 % de los episodios ha finalizado con éxito, el 21 % en colisión y el 1,2 % en *timeout*.

En la Figura 4 se presentan las métricas del entrenamiento del modelo de *DRL*. Arriba a la izquierda se observa que la terminación exitosa (señal verde oscuro), es decir, haber explorado más del 95 % del entorno, aumenta de forma sostenida a medida que avanzan los episodios hasta convertirse en el desenlace predominante. No obstante, aunque las colisiones contra paredes (señal roja) disminuyen de manera notable conforme el agente adquiere una política más estable y eficiente, todavía aparecen ocasionalmente.

Por otra parte, la evolución de las métricas internas del modelo confirma que el algoritmo está siendo capaz de explorar adecuadamente el espacio de acciones para lograr una mayor recompensa. La pérdida del crítico presenta oscilaciones crecientes propias de los métodos actor-crítico, donde en cada momento se evalúan diferentes políticas que se van optimizando y termina estabilizando en el rango [0,075; 0,15]. La pérdida del actor disminuye de forma progresiva, señal de que la política cada vez es capaz de hacer acciones con mayor Q. Además, la recompensa media cada 10 episodios aumenta con rapidez en las primeras fases y posteriormente se estabiliza, reflejando que el agente descubre estrategias efectivas y las consolida.

5. Conclusiones

Este trabajo ha demostrado la eficacia del algoritmo *TD3* de *DRL* para abordar el problema del *A-SLAM* integrando observaciones ricas en información, pero de alta dimensionalidad, siendo necesarias dos redes *CNN* y una *MLP*. Los resultados del entrenamiento evidencian un aprendizaje estable en el que la tasa de éxito, definida como una completitud del mapa superior al 95 %, se convierte progresivamente en el desenlace predominante. Este rendimiento está condicionado por la necesidad de mantener arquitecturas *CNN* sencillas, ya que deben entrenarse simultáneamente con el actor y el crítico, lo que obliga a reducir la complejidad para evitar inestabilidades y tiempos de convergencia excesivos.

El análisis del comportamiento del agente ha revelado un defecto significativo en la política aprendida: el robot tiende a ejecutar trayectorias oscilatorias, esto es, en forma de "S" (Figura 5). Aunque estas trayectorias permiten completar el mapeo, comprometen la seguridad del movimiento.

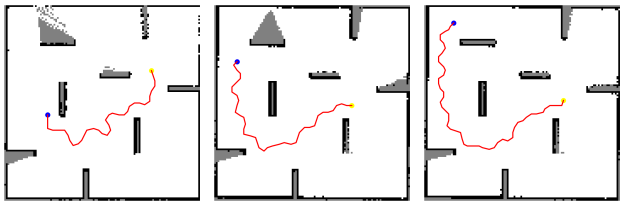

Figura 5: Trayectoria seguida por el robot en color rojo. En azul, el punto de inicio. En amarillo, el punto final.

Para corregir estas oscilaciones, se está trabajando en el ajuste de la escala de los términos de la recompensa mediante el error de Bellman cuadrático medio (Sutton and Barto, 2018; Fujimoto et al., 2018). Las trayectorias oscilantes sugieren que el término asociado a la velocidad angular tiene un peso insuficiente en la función de valor, lo que provoca que el gradiente de la política respecto a ese componente sea demasiado pequeño, incluso mucho menor que el error de aprendizaje del crítico. Asimismo, se plantea incorporar un término de explotación que favorezca el cierre activo de bucles y el refinamiento de zonas previamente exploradas, con el fin de mejorar la precisión global del mapa (Zhao and Hwang, 2024), así como revisar la función de recompensa de exploración.

Finalmente, se propone desacoplar el entrenamiento de las *CNN* del entrenamiento del *TD3*, explorando enfoques basa-

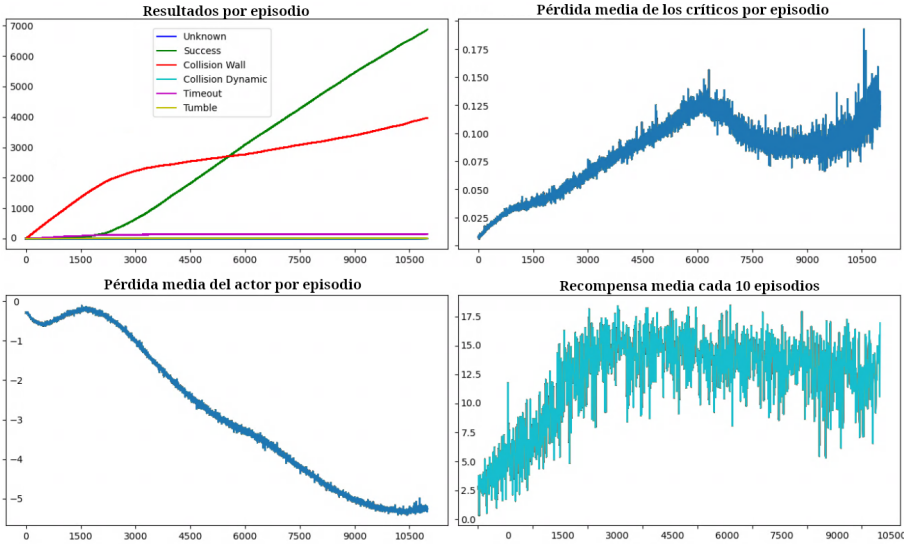

Figura 4: Métricas de los resultados del entrenamiento.

dos en *encoder–decoder* para reconstruir mapas o procesar datos del *LiDAR* de forma supervisada. Una vez preentrenadas, estas redes podrían integrarse en el *pipeline* de *DRL*, reduciendo la carga de aprendizaje simultáneo y mejorando la estabilidad del entrenamiento. Como paso final, se evaluará la política en entornos distintos al de entrenamiento y se avanzará hacia su implementación en un robot físico *iCreate3*.

Agradecimientos

The research leading to these results was supported in part by iRoboCity2030-CM, Robótica Inteligente para Ciudades Sostenibles (TEC-2024/TEC-62), funded by the Programas de Actividades I+D en Tecnologías en la Comunidad de Madrid.

Referencias

Alcalde, M., Ferreira, M., González, P., Andrade, F., Tejera, G., 2022. Daslam: Deep active slam based on deep reinforcement learning. In: 2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE). pp. 282–287.
DOI: 10.1109/LARS/SBR/WRE56824.2022.9996006

Arroyo de la Torre, V., Barea Navarro, R., 2021. Aplicación de técnicas de deep reinforcement learning mediante agente cnn-ddpg a la conducción autónoma. Universidad de Alcalá, España, trabajo de fin de grado.
URL: https://ebuah.uah.es/dspace/handle/10017/49476

Chen, F., Martin, J. D., Huang, Y., Wang, J., Englot, B. J., 2020. Autonomous exploration under uncertainty via deep reinforcement learning on graphs. CoRR abs/2007.12640.
URL: https://arxiv.org/abs/2007.12640

Cimurs, R., Suh, I. H., Lee, J. H., 2021. Goal-driven autonomous exploration through deep reinforcement learning.
URL: https://arxiv.org/abs/2103.07119

Du, S., Duan, T., Yang, X., Gong, X., 2025. Review and frontier exploration of active slam. In: 2025 4th Conference on Fully Actuated System Theory and Applications (FASTA). pp. 2850–2855.
DOI: 10.1109/FASTA65681.2025.11138631

Fernandez-Fernandez, R., Victores, J. G., Balaguer, C., 2023. Deep robot sketching: An application of deep q-learning networks for human-like sketching. Cognitive Systems Research 81, 57–63.
URL: https://www.sciencedirect.com/science/article/pii/S1389041723000372
DOI: https://doi.org/10.1016/j.cogsys.2023.05.004

Fujimoto, S., van Hoof, H., Meger, D., 2018. Addressing function approximation error in actor-critic methods. CoRR abs/1802.09477.
URL: http://arxiv.org/abs/1802.09477

Li, H., Zhang, Q., Zhao, D., 07 2020. Deep reinforcement learning based automatic exploration for navigation in unknown environment.
DOI: 10.48550/arXiv.2007.11808

Li, Z., Xin, J., Li, N., 2023. Autonomous exploration and mapping for mobile robots via cumulative curriculum reinforcement learning.
URL: https://arxiv.org/abs/2302.13025

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N. M. O., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv: Learning.
URL: https://api.semanticscholar.org/CorpusID:16326763

Llobera, M. A., Placed, J. A., Paula, M. D., Cristóforis, P. D., 2026. Massive parallel deep reinforcement learning for active slam.
URL: https://arxiv.org/abs/2603.25834

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W., 2022. Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics 7 (66), eabm6074.
URL: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074
DOI: 10.1126/scirobotics.abm6074

Macenski, S., Jambrecic, I., 2021. Slam toolbox: Slam for the dynamic world. Journal of Open Source Software 6 (61), 2783.
URL: https://doi.org/10.21105/joss.02783
DOI: 10.21105/joss.02783

Niroui, F., Zhang, K., Kashino, Z., Nejat, G., 2019. Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments. IEEE Robotics and Automation Letters 4 (2), 610–617.
DOI: 10.1109/LRA.2019.2891991

Placed, J. A., Castellanos, J. A., 2020. A deep reinforcement learning approach for active slam. Applied Sciences 10 (23).
URL: https://www.mdpi.com/2076-3417/10/23/8386
DOI: 10.3390/app10238386

Sutton, R. S., Barto, A. G., 2018. Reinforcement Learning: An Introduction, 2nd Edition. The MIT Press.
URL: http://incompleteideas.net/book/the-book-2nd.html

Thrun, S., Burgard, W., Fox, D., 2005. Probabilistic robotics. MIT Press, Cambridge, Mass.

Topiwala, A., Inani, P., Kathpal, A., 2018. Frontier based exploration for autonomous robot.
URL: https://arxiv.org/abs/1806.03581

Wang, R., Zhang, J., Lyu, M., Yan, C., Chen, Y., 2024. An improved frontier-based robot exploration strategy combined with deep reinforcement learning. Robotics and Autonomous Systems 181, 104783.
URL: https://www.sciencedirect.com/science/article/pii/S0921889024001672
DOI: https://doi.org/10.1016/j.robot.2024.104783

Zhao, S., Hwang, S.-H., 2024. Exploration- and exploitation-driven deep deterministic policy gradient for active slam in unknown indoor environments. Electronics 13 (5).
URL: https://www.mdpi.com/2079-9292/13/5/999
DOI: 10.3390/electronics13050999