



UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA



DA-SLAM: Deep Active SLAM based on Deep Reinforcement Learning

TESIS PRESENTADA A LA FACULTAD DE INGENIERÍA DE LA
UNIVERSIDAD DE LA REPÚBLICA POR

Martín Alcalde, Matías Ferreira, Pablo González

EN CUMPLIMIENTO PARCIAL DE LOS REQUERIMIENTOS
PARA LA OBTENCIÓN DEL TÍTULO DE
INGENIERO EN COMPUTACIÓN.

TUTORES

Federico Andrade Universidad de la República
Gonzalo Tejera..... Universidad de la República

TRIBUNAL

Martín Llofriu Universidad de la República
Jorge Visca Universidad de la República
Mathías Etcheverry Universidad de la República

Montevideo
jueves 23 febrero, 2023

Agradecimientos

A Federico Andrade por darnos la oportunidad de trabajar con él, por haberse involucrado realmente durante todo el proceso y por estar siempre disponible para ayudar y brindar soluciones.

A Gonzalo Tejera por guiarnos y aportar ideas.

A nuestras familias por el apoyo y sustento.

A la FING por enseñarnos a pensar.

Tabla de contenidos

Agradecimientos	I
1. Resumen	1
2. Introducción	3
3. Marco teórico	7
3.1. Localización y Mapeo Simultáneos	7
3.1.1. Clasificaciones de SLAM	8
3.1.2. Técnicas básicas de SLAM	8
3.1.3. Conceptos varios relacionados a SLAM	10
3.2. Aprendizaje automático	14
3.2.1. Proceso de decisión de Markov	14
3.2.2. Aprendizaje por refuerzo	14
3.2.3. Aprendizaje por refuerzo profundo	14
3.2.4. Política	14
3.2.5. Implementaciones de RL	15
3.3. Implementaciones de Exploración	18
3.3.1. Exploración basado en fronteras	18
3.3.2. Exploración con cierres de ciclos activo	18
4. Trabajos relacionados	19
4.1. A Deep Reinforcement Learning approach for Active SLAM	19
4.2. On reward shaping for mobile robot navigation: a RL and SLAM based approach	22
4.3. Entropy-based exploration for mobile robot navigation: a learning- based approach	28
4.4. Curiosity-driven RL agent for mapping unknown indoor environments	32
4.5. Resumen	36

Tabla de contenidos

5. Declaración del problema y Solución propuesta	37
5.1. Declaración del problema	37
5.2. Solución propuesta	38
5.2.1. Función de recompensa	38
5.2.2. Espacio de observación	38
5.2.3. Espacio de acción	40
5.2.4. Espacio de estado	40
5.2.5. Módulo de toma de decisiones	40
5.2.6. Arquitectura	41
5.2.7. Histórico de agentes	42
6. Experimentos y Resultados	47
6.1. Experimentos	47
6.1.1. Entornos	47
6.1.2. Entrenamiento	48
6.1.3. Evaluación	49
Evaluación	49
6.2. Resultados	52
6.2.1. Resultados de validación	52
6.2.2. Resultados de evaluación	53
7. Conclusiones y Trabajos futuros	57
7.1. Conclusiones	57
7.2. Trabajos futuros	58
Referencias	59
Glosario	63
Índice de tablas	64
Índice de figuras	66

Capítulo 1

Resumen

El mapeo y la localización son unos de los desafíos fundamentales de los robots móviles, junto a la planificación de trayectorias forman la base de los problemas de navegación en robots móviles autónomos.

En esta Tesis se aborda el problema de SLAM (Localización y Mapeo Simultáneos, por sus siglas en inglés), el cual es el problema de construir un mapa de un entorno previamente desconocido de forma incremental y al mismo tiempo localizar el robot dentro de ese mapa. Uno no puede desacoplar ambas tareas y resolverlas de forma independiente.

En los últimos años, RL (Aprendizaje por Refuerzo, por sus siglas en inglés) se ha utilizado para abordar y resolver varias tareas de robótica diferentes, como estabilización, manipulación, locomoción y navegación. En el contexto de la navegación, los agentes basados en RL no suelen depender de ningún mapa o SLAM y, aunque tienen mucho éxito, no aprovechan la información importante almacenada en los mapas.

En cuanto al problema de SLAM activo con RL, se plantea la necesidad de contar con funciones de recompensa que permitan: reducir la incertidumbre del mapa y motivar la exploración del mapa

Este trabajo presenta mejoras sobre los algoritmos existentes en el estado del arte, para la planificación y exploración en entornos complejos desconocidos utilizando DRL (Aprendizaje por Refuerzo Profundo, por sus siglas en inglés). Este trabajo presenta un enfoque novedoso, el cual introduce dos funciones de recompensa que toman en consideración: (i) la información del mapa, el cual es construido en tiempo real por el robot utilizando un algoritmo de SLAM, y (ii) la incertidumbre de la pose del robot, lo que lleva al cerrado activo de ciclos para incentivar la exploración y mejorar la generación de los mapas, respectivamente. Los resultados obtenidos muestran que la función de recompensa basada en la completitud del mapa obtiene trayectorias más cortas, respecto a las presentadas por la literatura; mientras que la

Capítulo 1. Resumen

función de recompensa basada en incertidumbre de la pose obtiene mapas más fieles al entorno real. Además, ambos agentes probaron su capacidad tanto de realizar SLAM activo en entornos complejos como de generalización a mapas desconocidos.

Capítulo 2

Introducción

El mapeo y la localización es uno de los desafíos fundamentales de los robots móviles y, junto a la planificación de trayectorias forman la base de los problemas de navegación en robots móviles autónomos.

En la figura 2.1 se presentan las tres áreas básicas de la robótica: Mapeo, Localización y Planificación; las subáreas resultantes de sus intersecciones: SLAM, Exploración y Localización activa; y la subárea resultante de la intersección de las tres áreas: SPLAM (Localización, Mapeo y Planificación Simultáneos, por sus siglas en inglés).

El mapeo [1] es el problema de integrar la información recopilada con los sensores del robot en una representación dada. Se puede describir con la pregunta “¿Cómo se ve el mundo?”. Los aspectos centrales en el mapeo son la representación del entorno y la interpretación de los datos del sensor. En contraste con esto, la localización es el problema de estimar la pose del robot en relación con un mapa. En otras palabras, el robot tiene que responder la pregunta: “¿Dónde estoy?”. Por lo general, se distingue entre el seguimiento de la pose, donde se conoce la pose inicial del vehículo, y la localización global, en la que no se da conocimiento a priori sobre la posición inicial. Finalmente, el problema de planificación de trayectoria o control de movimiento implica la cuestión de cómo guiar eficientemente un vehículo a una ubicación deseada o a lo largo de una trayectoria. Expresado como una simple pregunta, este problema se puede describir como “¿Cómo puedo llegar a una ubicación determinada?” [1].

De las subáreas resultantes, en esta Tesis se aborda el problema de SLAM, el cual es el problema de construir un mapa de un entorno previamente desconocido de forma incremental y al mismo tiempo localizar el robot dentro de ese mapa [2]. Uno no puede desacoplar ambas tareas y resolverlas de forma independiente. Por lo tanto, SLAM a menudo se conoce como un problema del huevo o la gallina: se necesita un buen mapa para la localización, mientras que se necesita una estimación de pose precisa para construir un mapa [1].

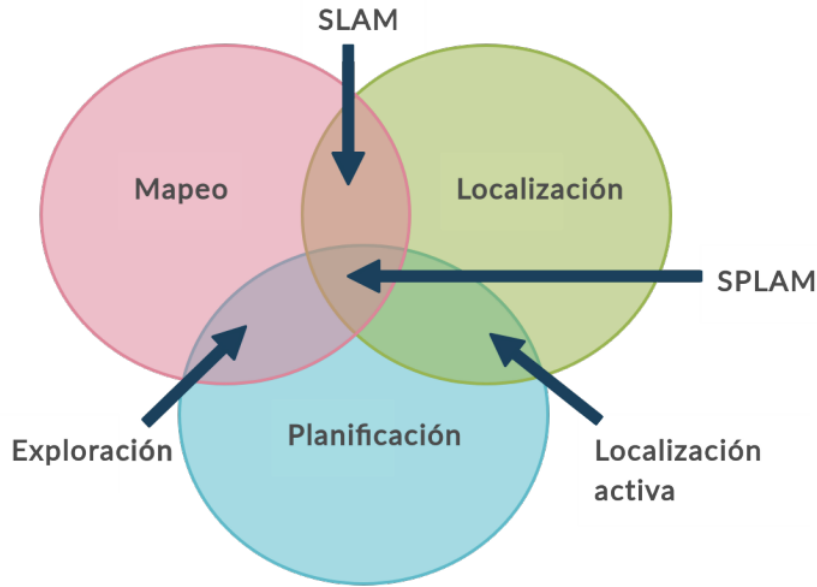


Figura 2.1: Áreas básicas de la navegación en robots móviles

Por otro lado, en los últimos años, se ha utilizado el RL [3] para estudiar y llevar a cabo distintas tareas de robótica como, por ejemplo, estabilización, manipulación, locomoción y navegación. En el contexto de la navegación, los agentes basados en RL típicamente no dependen de ningún mapa y, si bien son bastante exitosos, típicamente no explotan las características del entorno.

Algunas de las clasificaciones que pueden hacer en SLAM son: SLAM pasivo o SLAM activo, este último tiene como objetivo encontrar la secuencia de salidas de control, es decir acciones, para explorar completamente y construir los mapas de los entornos. Dicho proceso secuencial de toma de decisiones puede ser formulado como un problema de RL, donde el agente RL debe aprender las acciones que lo llevan a maximizar la recompensa acumulada. En cuanto al problema de SLAM activo con RL, existe la necesidad de tener funciones de recompensa que permitan reducir la incertidumbre del mapa y motivar la exploración del robot sobre el mismo.

En los últimos años, los conceptos básicos de DRL han demostrado tener un gran potencial en las tareas de planificación, mapeo e incluso exploración. En este sentido, uno de los principales inconvenientes de los métodos de SLAM activo tradicionales radica en transferir los cálculos intensivos a la fase de entrenamiento de la DNN (Red Neural Profunda, por sus siglas en inglés.), requiriendo en la fase de evaluación sólo la propagación hacia adelante (costo computacional bajo) [2].

Para los robots móviles, es relevante aprender a navegar en entornos complejos

e inicialmente desconocidos, ya que, de este modo, puede aprender la mayoría de las habilidades necesarias para navegar en cualquier otro tipo de entorno. Tener la información del mapa le permite al agente, entre otras cosas, conocer la incertidumbre de la pose en todo momento de forma muy precisa [2]. Motivar un valor de incertidumbre bajo tiene como ventaja que el agente prioriza no perderse en la generación del mapa, lo que resulta en un mapeo más robusto y, por lo tanto, en mapas más precisos.

El objetivo de este trabajo es presentar mejoras a los algoritmos del estado del arte de Active SLAM que utilizan DRL. Las principales contribuciones de este proyecto son:

- Implementación de dos agentes de SLAM activo basados en DRL capaces de tomar decisiones casi óptimas en entornos desconocidos.
- Evaluación en entornos cuya complejidad excede la presentada por la literatura del estado del arte [4] [5] [2].
- Normalización y disminución del cómputo sensorial debido al uso de solamente 5 datos obtenidos del campo frontal con un LiDAR (láser de medición y detección de objetos, por sus siglas en inglés), lo que reduce la dimensión del espacio de observación. Esto tiene como ventajas: evita el problema de la “Maldición de la dimensionalidad” [6], reduce el costo computacional y, de este modo, resulta en mayor rapidez de entrenamiento y convergencia.

De acuerdo al conocimiento de los autores, este es el primer enfoque de SLAM activo con DRL que incluye una función de recompensa basada en incertidumbre e información del mapa sobre entornos topológicamente complejos.

Los resultados obtenidos experimentalmente muestran mejoras en la performance de los agentes presentados respecto a los enfoques existentes a pesar de disminuir la dimensión del espacio de observación, aumentando la precisión de los mapas obtenidos, sobre entornos desconocidos de mayor complejidad. Incluso, también, superando los resultados de un agente de exploración basado en fronteras [7].

Esta sección proporcionó una breve pero completa introducción al problema abordado. A continuación, se presenta la estructura del resto del documento de tesis. En el Capítulo 3 se revisan los principales conceptos teóricos sobre SLAM en robots móviles, Aprendizaje por Refuerzo y Aprendizaje por Refuerzo Profundo. Luego, en el Capítulo 4 se estudia la literatura asociada a SLAM activo con Aprendizaje por Refuerzo Profundo. En el Capítulo 5 se define formalmente el problema y se propone una solución basada en el uso de dos funciones recompensa distintas con mejoras a las presentadas en el Estado del Arte. Más adelante, en el Capítulo 6 se muestran los experimentos realizados en simulación y se reportan los resultados

Capítulo 2. Introducción

obtenidos tanto en fase de validación del entrenamiento como en fase de evaluación. Finalmente, en el Capítulo 7 se presentan las conclusiones de los resultados obtenidos en esta Tesis, y adicionalmente, se exponen varias propuestas de desarrollo a futuro motivadas por el trabajo realizado.

Este trabajo dio lugar a una publicación científica que fue aceptada y presentada en la conferencia IEEE Latin American Robotics Symposium (LARS-SBR 2022).

Capítulo 3

Marco teórico

Este capítulo de marco teórico permitirá al lector conocer los conceptos básicos y relacionados de Localización y Mapeo Simultáneos y Aprendizaje por Refuerzo para el entendimiento del desarrollo de este proyecto. Por último, se presentan dos implementaciones de Exploración que fueron utilizadas para evaluar la solución propuesta en esta Tesis.

3.1. Localización y Mapeo Simultáneos

SLAM (Localización y Mapeo Simultáneos, por sus siglas en inglés) [8] aborda el problema de un robot que navega en un entorno desconocido, comenzando en un lugar con coordenadas conocidas. Su movimiento es incierto, lo que dificulta gradualmente determinar sus coordenadas globales. Mientras deambula, el robot percibe su entorno. El problema de SLAM es el problema de construir un mapa del entorno mientras se determina simultáneamente la posición del robot en relación con este mapa.

Formalmente, SLAM se puede presentar en terminología probabilística [8]. En ese sentido, hay dos formas principales del problema SLAM [9], que tienen la misma importancia práctica. Uno se conoce como el problema SLAM en línea que implica estimar probabilidad posterior sobre la pose actual junto con el mapa, el cual puede formularse como:

$$p(x_t, m | z_{1:t}, u_{1:t}) \quad (3.1)$$

donde x_t es la pose en el tiempo t , m es el mapa y $z_{1:t}$ y $u_{1:t}$ son las medidas y los controles, respectivamente. Este problema se denomina problema SLAM en línea ya que solo involucra la estimación de variables que persisten en el tiempo t .

El segundo problema de SLAM se denomina problema de SLAM completo. En SLAM completo, se busca calcular la probabilidad posterior sobre toda la ruta $x_{1:t}$

Capítulo 3. Marco teórico

junto con el mapa, en lugar de solo la pose actual x_t , el cual puede formularse como:

$$p(x_{1:t}, m | z_{1:t}, u_{1:t}) \quad (3.2)$$

3.1.1. Clasificaciones de SLAM

El problema de SLAM cuenta con varias clasificaciones, a continuación se presentarán algunas posibles clasificaciones.

Offline vs. Online

En el caso del SLAM online se procesa la información en el mismo robot mientras este navega en el entorno. Por otro lado, en el offline se realiza SLAM sobre un conjunto de datos que previamente fueron obtenidos con algún robot, tanto de la medida de sus sensores como las medidas de odometría [8].

Estático vs. Dinámico

SLAM estático considera que el entorno no cambia con el transcurso del tiempo. Es más utilizado ya que simplifica el problema. Por otro lado, SLAM dinámico se acerca más a la realidad considerando ambientes cambiantes. Suelen ser más robustos que los estáticos [8].

Pasivo vs. Activo

En los algoritmos de SLAM pasivo [8], es otro agente quien se encarga de controlar el robot, mientras que el algoritmo de SLAM está puramente observando. La gran mayoría de algoritmos son de este tipo y brindan al diseñador del robot la libertad de implementar los controladores de movimiento arbitrarios.

SLAM activo [10] es un problema de toma de decisiones en el que se elige la trayectoria de un robot tanto para mejorar los resultados del mapeo y localización, como para realizar otras tareas como la cobertura o la exploración.

El robot explora activamente su entorno en busca de generar un mapa preciso [8]. Los métodos de SLAM activo tienden a producir mapas más precisos en menos tiempo, pero limitan el movimiento del robot. Existen técnicas híbridas de SLAM activo y SLAM pasivo donde el algoritmo de SLAM sólo controla la dirección de los sensores y otra entidad se encarga de la dirección del movimiento del robot.

3.1.2. Técnicas básicas de SLAM

En esta sección se presentan dos técnicas que se utilizan con frecuencia en el problema de SLAM y son relevantes para este proyecto: la utilización de filtros de partículas y los mapas de cuadrículas.

3.1. Localización y Mapeo Simultáneos

Filtro de partículas

En el contexto de la robótica móvil, los filtros de partículas a menudo se usan para rastrear la posición del robot [1]. Los filtros de partículas son implementaciones no paramétricas del filtro recursivo de Bayes. Utilizan un conjunto de muestras ponderadas y pueden representar distribuciones arbitrarias. Las muestras se extraen de una distribución propuesta. Después de determinar los pesos de importancia que explican el hecho de que la distribución objetivo es diferente de la distribución de la propuesta, el paso de remuestreo reemplaza las partículas con un peso bajo por partículas con un peso de gran importancia.

Uno de los filtros de partículas más utilizados es RBPF (Filtro de partículas Rao-Blackwellized, por sus siglas en inglés) utilizado para construir mapas de entornos [11] mediante la estimación de la pose del robot independientemente del proceso de estimación del mapa. Esto se muestra en la ecuación (3.3):

$$p(x_{1:t}, m | z_{1:t}, u_{1:t-1}) = p(m | x_{1:t}, z_{1:t}) p(x_{1:t} | z_{1:t}, u_{1:t-1}) \quad (3.3)$$

donde $x_{1:t}$ es la trayectoria del robot, $z_{1:t}$ es la secuencia de medidas del sensor, $u_{1:t-1}$ es la entrada de control en el paso de tiempo anterior y m es el mapa del entorno.

Mapas de cuadrículas

Existen diferentes tipos de modelos para representar el entorno que se usan con frecuencia en la robótica móvil. Los más comunes son mapas de características, mapas geométricos y mapas de cuadrícula [1].

Un mapa de características almacena un conjunto de características detectadas en el entorno. Las características típicas son líneas y esquinas cuando se utilizan sensores de proximidad. Para cada característica, estos mapas almacenan la información de la función junto con una coordenada y, finalmente, una medida de incertidumbre. Esto se puede realizar mediante una lista de características o mediante el uso de estructuras de datos más eficientes como los árboles.

Los mapas geométricos representan todos los obstáculos detectados por el robot como objetos geométricos, como círculos o polígonos. Este tipo de representación es comparablemente compacta y necesita solo unos pocos recursos de memoria. Los mapas de la cuadrícula discretizan el entorno en las llamadas celdas de la cuadrícula. Cada celda almacena información sobre el área que cubre.

Los más utilizados con mayor frecuencia son los mapas de cuadrícula de ocupación que almacenan para cada celda, un valor único que representa la probabilidad de que esta celda esté ocupada por un obstáculo. La ventaja de las cuadrículas es que no confían en características predefinidas que deben extraerse de los datos del sensor. Además, ofrecen un acceso de tiempo constante a las celdas de la cuadrícula y proporcionan la capacidad de modelar áreas desconocidas (no observadas), que es

Capítulo 3. Marco teórico

una característica importante en el contexto de la exploración. Sin embargo, tienen las desventajas de los errores de discretización y de requerir muchos recursos de memoria.

3.1.3. Conceptos varios relacionados a SLAM

En esta sección se presentan conceptos varios relacionados a SLAM.

Pose

La pose representa la ubicación y orientación del robot mediante el siguiente vector: (x, y, z, X, Y, Z, W) . Los primeros tres valores representan la posición del robot, donde x, y, z son las coordenadas en los ejes cartesianos. Los últimos cuatro valores representan la orientación del robot, donde X es la rotación según el eje x , Y es la rotación sobre el eje y , Z es la rotación sobre el eje z , W es un escalar que almacena la rotación alrededor del vector (X, Y, Z) .

Matriz de covarianza

La matriz de covarianza Σ es de tamaño $(3N + 3) \times (3N + 3)$ [9] cuya entrada (i, j) es la covarianza entre la variable X_i y X_j , es decir $\Sigma_{ij} = Cov(X_i, X_j)$. En particular, cuando $i = j$, es decir, la diagonal de la matriz Σ , obtenemos $\Sigma_{ii} = Cov(X_i, X_i) = Var(X_i)$.

Las matrices de covarianza son una forma de describir la relación entre una colección de variables [9]. Un único valor de covarianza describe la relación entre dos variables. Son una herramienta para estimar el posible error en un valor numérico y para predecir un valor numérico.

Grillas de ocupación

Las grillas de ocupación (Occupancy Grids) representan el espacio como una grilla que divide el entorno en un conjunto de celdas, donde cada celda tiene asociado un valor que representa su ocupación. Este valor puede ser absoluto (libre u ocupado) pero en la práctica suele representar una noción de probabilidad de que la celda se encuentre ocupada [12]. Estos mapas son usualmente utilizados en sistemas con sensores de distancia como sonares o sensores laser [13] [14] [15] [16].

A diferencia de otras representaciones, la salida generada por un algoritmo que utiliza grillas de ocupación es muy similar a los planos utilizados por los seres humanos.

Además, la naturaleza métrica de esta representación la hace adecuada para tareas de planificación de caminos.

3.1. Localización y Mapeo Simultáneos

Debido a que la creencia sobre la ocupación de cada celda se representa con una probabilidad, el modelo permite manejar la incertidumbre naturalmente. Esto permite incluir manejo de:

- Ruido causado por los sensores.
- Ruido causado por dinámismos no modelados, como personas moviéndose.
- Incertidumbre de la posición del robot que afecta el conocimiento sobre el mapa relativo al robot.

En general estas grillas son utilizadas para representar entornos en un plano de dos dimensiones. Sin embargo, se han realizado trabajos que extienden esta representación a tres dimensiones. La precisión de la grilla de ocupación determina la calidad del mapa obtenido y con esta se disminuyen los errores producidos por la discretización de un ambiente continuo. Sin embargo, al aumentar el número de celdas el costo computacional de actualización del mapa aumenta, al igual que la memoria necesaria para almacenar el mapa. Al utilizar grillas de ocupación, es necesario encontrar un punto de equilibrio entre la precisión del mapa y el poder de cómputo y espacio de memoria disponible.

Mapas topológicos

Los mapas topológicos constan de una relación de conectividad entre lugares distinguibles. Su representación consta de un grafo donde los nodos son lugares distinguibles mediante sensado directo y las aristas representan la navegabilidad entre dos de estos lugares. Estos mapas son compactos en su representación debido a su falta de información métrica detallada del entorno.

En la figura 3.1 se muestra un mapa topológico, donde los nodos son representados con puntos verdes y las aristas con líneas azules.

Entropía del Mapa

La entropía del mapa [17] corresponde a la suma de la entropía de todas las celdas c en el mapa m y cuantifica las incertidumbres en el mapa.

Se representa mediante la Ecuación (3.4):

$$H(m) = \sum_{c_f \in m} p(c_f) \log(p(c_f)) + \sum_{c_o \in m} (1 - p(c_o)) \log(1 - p(c_o)) \quad (3.4)$$

donde c_f corresponde a las celdas desconocidas y desocupadas en el mapa y c_o a las ocupadas.

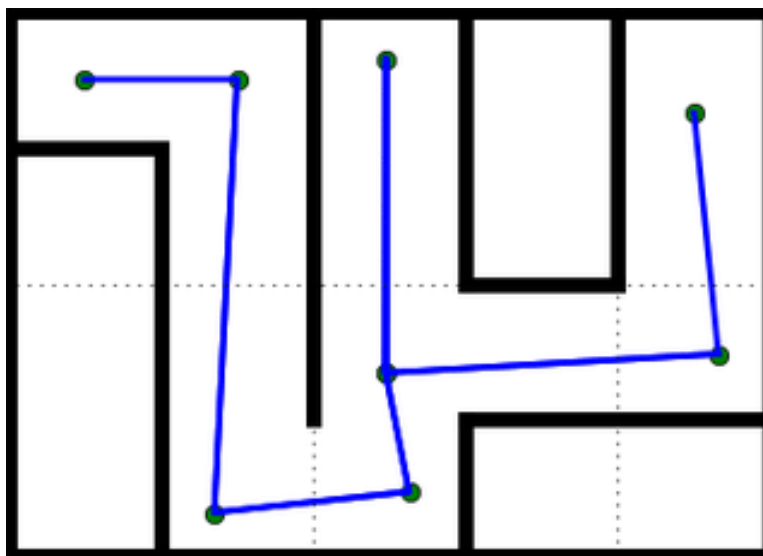


Figura 3.1: Mapa topológico

Cierre de ciclos

Cerrar ciclos refiere a la situación en que el robot debe poder reconocer cuando pasa por un lugar que ya ha sido visitado [9]. Esta información es útil para reducir la entropía en la estimación del mapa. Sin cierre de ciclos, la entropía del mapa crece sin límites.

Cada vez que un robot que usa un filtro de partículas explora un nuevo terreno, todas las muestras tienen más o menos el mismo peso de importancia ya que la medición más reciente es típicamente consistente con la parte del mapa construida a partir de las observaciones inmediatamente anteriores [1]. Típicamente, la incertidumbre sobre la pose del robot aumenta. Sin embargo, tan pronto como reinicie el terreno conocido, los mapas de algunas partículas son consistentes con la medición actual y algunos no lo son. En consecuencia, los pesos de las muestras difieren en gran medida. Debido al paso de remuestreo, las partículas poco probables generalmente se eliminan y, por lo tanto, la incertidumbre sobre la pose del robot disminuye. Un ejemplo típico se muestra en la figura 3.2.

En las dos imágenes de la izquierda, el robot explora el nuevo terreno y aumenta la incertidumbre del conjunto de muestras. En la imagen derecha, el robot viaja a través del terreno conocido y las partículas poco probables han desaparecido. Un aspecto a tener en cuenta que este efecto es mucho más pequeño si el robot solo mueve hacia atrás unos pocos metros para volver a visitar las áreas escaneadas previamente. Esto se debe a que el mapa asociado con una partícula generalmente es localmente consistente. Las inconsistencias surgen en su mayoría cuando el robot

3.1. Localización y Mapeo Simultáneos

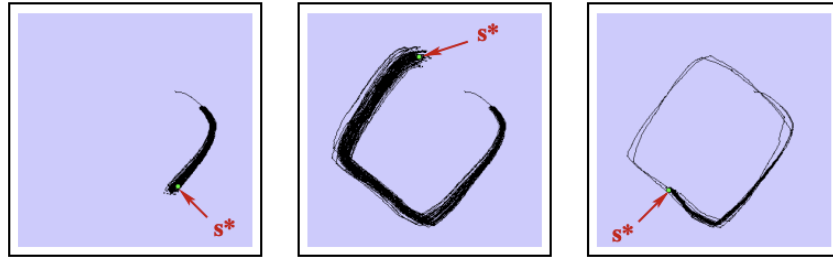


Figura 3.2: Evolución de un conjunto de partículas y el mapa de la partícula más probable (aquí marcada como s^*) en tres pasos de tiempo diferentes. En las dos imágenes de la izquierda, el vehículo viajó a través de un terreno desconocido, de modo que la incertidumbre aumentó. En la imagen derecha, el robot reinició el terreno conocido para que las muestras que representen trayectorias improbables desaparecieran [1].

reinicia las áreas exploradas hace algún tiempo. Por lo tanto, los lugares de visita vistos más atrás en la historia tienen un efecto más fuerte en las diferencias entre los pesos de importancia y, por lo general, también en la reducción de la incertidumbre en comparación con los lugares observados recientemente.

3.2. Aprendizaje automático

En esta sección se presentan los conceptos relacionados con aprendizaje automático.

3.2.1. Proceso de decisión de Markov

Proceso de decisión de Markov (MDP, por sus siglas en inglés) [9] es una tupla $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ donde \mathcal{S} es el conjunto de estados posibles, \mathcal{A} es el conjunto de acciones, \mathcal{P} describe la dinámica del entorno, es decir, la probabilidad de transición de un estado s_t al siguiente estado s_{t+1} al aplicar una acción a_t y una función de recompensa \mathcal{R} . Por lo general, \mathcal{P} y \mathcal{R} son desconocidos de antemano y deben aprenderse por ensayo y error.

3.2.2. Aprendizaje por refuerzo

Aprendizaje por refuerzo (RL, por sus siglas en inglés) [9] tiene como objetivo resolver un problema de toma de decisiones secuencial mediante el aprendizaje de las acciones a realizar a través de la interacción con un entorno desconocido. Al agente RL, es decir, el que toma las decisiones, no se le enseña cuál es el mejor comportamiento, sino que tiene que inferirlo recibiendo solo un valor escalar, la recompensa, por cada acción realizada. La interacción entre el agente y el entorno se puede modelar como un MDP.

3.2.3. Aprendizaje por refuerzo profundo

Los métodos de Aprendizaje por refuerzo profundo (DRL, por sus siglas en inglés) [18] son obtenidos al utilizar redes neuronales profundas para aproximar cualquiera de los siguientes componentes de RL: función de valor, $\hat{v}(s; \theta)$ o $\hat{q}(s, a; \theta)$, política $\pi(a|s; \theta)$ y modelo (función de transición de estado y función de recompensa). Aquí, los parámetros θ son los pesos en redes neuronales profundas.

3.2.4. Política

Una política π (del inglés policy) es un mapeo de acción-estado. Un “estado” es un formalismo utilizado en Inteligencia Artificial que representa el estado del mundo, es decir, cuál es la idea del mundo del agente. La acción es, naturalmente, qué acción debe tomar en ese estado. Una política simplemente asigna acciones a estados.

3.2. Aprendizaje automático

Dentro de la política

Los métodos dentro de la política (del inglés on-policy) estiman el valor de una política mientras la usan para el control.

Fuera de la política

En los métodos fuera de política (del inglés off-policy), la política utilizada para generar comportamiento, denominada política de comportamiento, puede no estar relacionada con la política que se evalúa y mejora, denominada política de estimación.

3.2.5. Implementaciones de RL

A continuación se presentan varias implementaciones basadas en RL.

Gradiente de política determinista profundo

Gradiente de política determinista profundo (DDPG, por sus siglas en inglés) [19] es un algoritmo de RL libre de modelos y fuera de la política con una estructura de actor-crítico. El actor, generalmente representado por una red neuronal, elige una acción para cada estado dado s_t . Por otro lado, el crítico, representado también por una red neuronal, evalúa las actuaciones del actor estimando la función de valor de acción del estado Q como ocurre en el algoritmo DQN [20]. DDPG puede verse como la extensión de DQN para espacios de acción continua.

Red Q profunda

Para elegir la mejor acción, la función de valor es estimada y empleada por muchos algoritmos RL. La función de valor contiene información sobre la calidad y conveniencia de los estados. Red Q profunda (DQN, por sus siglas en inglés) [20] estima la función de valor de acción de estado Q , es decir, una función que mide qué tan bueno es elegir una determinada acción en un estado dado, por medio de una red neuronal. Es común emplear aproximadores de funciones cuando el espacio de estados y/o el espacio de acción son continuos y no pueden representarse mediante una tabla de consulta. En particular, la red Q está entrenada para minimizar el error cuadrático medio entre la función de valor de acción de estado verdadero y su predicción.

Red Q recurrente profunda

DQN ha demostrado tener éxito en muchas aplicaciones, sin embargo, la mayoría de ellas se basan en la suposición de Markov de que el futuro es independiente

Capítulo 3. Marco teórico

del pasado dado el presente, es decir, la información transportada por el estado en el paso de tiempo actual es suficiente para elegir la mejor acción. En muchas situaciones, el estado del entorno no es directamente observable, es decir, en el caso de los MDP parcialmente observables (por ejemplo, dos lecturas de sensores pueden parecer iguales, pero corresponden a dos situaciones diferentes), y DQN tiene dificultades para aprender la política óptima. Para abordar este problema, se introdujo Red Q recurrente profunda (DRQN, por sus siglas en inglés) [21]. DRQN extiende DQN a entornos parcialmente observables al agregar una red neuronal recurrente (RNN, por sus siglas en inglés) en la Q-Network.

Optimización de políticas de región de confianza

Optimización de políticas de región de confianza (TRPO, por sus siglas en inglés) es un algoritmo similar a los métodos de gradiente de política natural [22] y es efectivo para optimizar grandes políticas no lineales como las redes neuronales [23].

Ventaja Asíncrona Actor-Crítico

Ventaja Asíncrona Actor-Crítico (A3C, por sus siglas en inglés), es un algoritmo de gradiente de política que mantiene una política π y una estimación de la función de valor. A continuación se presenta la decodificación de las diferentes partes del nombre del algoritmo:

Asíncrono: a diferencia de otros algoritmos populares de DRL como Deep Q-Learning, que usa un solo agente y un solo entorno, este algoritmo usa múltiples agentes y cada agente tiene sus propios parámetros de red y una copia del entorno. Estos agentes interactúan con sus respectivos entornos de forma asincrónica, aprendiendo con cada interacción. Cada agente está controlado por una red global. A medida que cada agente adquiere más conocimiento, contribuye al conocimiento total de la red global. La presencia de una red global permite que cada agente tenga datos de entrenamiento más diversificados.

Actor-crítico: a diferencia de algunas técnicas más simples que se basan en métodos de iteración de valor o métodos de gradiente de política, el algoritmo A3C combina las mejores partes de ambos métodos, es decir, el algoritmo predice tanto la función de valor $V(s)$ como la función de política óptima $\pi(s)$. El agente de aprendizaje utiliza el valor de la función Valor (Crítico) para actualizar la función de política óptima (Actor). Nótese que aquí la función política significa la distribución probabilística del espacio de acción. Para ser exactos, el agente de aprendizaje determina la probabilidad condicional $P(a|s; \theta)$, es decir, la probabilidad parametrizada de que el agente elija la acción a cuando se encuentra en el estado s .

Ventaja: por lo general, en la implementación de Política de Gradiente, el valor de devoluciones descontadas (γr) para decirle al agente cuáles de sus acciones fueron

3.2. Aprendizaje automático

gratificantes y cuáles fueron penalizadas. Al usar el valor de Ventaja en su lugar, el agente también aprende cuánto mejores fueron las recompensas de lo que esperaba. Esto le da al agente una visión nueva del entorno y, por lo tanto, el proceso de aprendizaje es mejor. La métrica de ventaja viene dada por la siguiente expresión:

$$\text{Advantage: } A = Q(s, a) - V(s)$$

Ventaja Actor-Crítico

Ventaja Actor-Crítico (A2C, por sus siglas en inglés) es una versión síncrona del método de gradiente de políticas A3C. Como alternativa a la implementación asincrónica de A3C, A2C es una implementación sincrónica y determinista que espera a que cada actor termine su segmento de experiencia antes de actualizar, promediando todos los actores.

Optimización de políticas próximas

El algoritmo de Optimización de políticas proximales (PPO, por sus siglas en inglés) combina ideas de A2C (tener varios trabajadores) y TRPO (utiliza una región de confianza para mejorar el actor).

La idea principal es que después de una actualización, la nueva política no debe estar muy alejada de la política anterior. Para eso, PPO usa el recorte para evitar una actualización demasiado grande [24].

PPO es un método de gradiente de política y se puede usar para entornos con espacios de acción discretos o continuos. Entrena una política estocástica de forma dentro de la política. Además, utiliza el método de actor crítico. El actor asigna la observación a una acción y el crítico da una expectativa de las recompensas del agente para la observación dada. En primer lugar, recopila un conjunto de trayectorias para cada época tomando muestras de la última versión de la política estocástica. Luego, se calculan las recompensas por llevar y las estimaciones de ventajas para actualizar la política y ajustar la función de valor. La política se actualiza a través de un optimizador de ascenso de gradiente estocástico, mientras que la función de valor se ajusta a través de algún algoritmo de descenso de gradiente. Este procedimiento se aplica durante muchas épocas hasta que se soluciona el entorno [25].

3.3. Implementaciones de Exploración

En esta sección se presenta la idea teórica detrás de dos implementaciones de Exploración utilizadas como agentes de comparación para la solución propuesta en esta Tesis.

3.3.1. Exploración basado en fronteras

En [7] se introduce un enfoque para exploración basada en el concepto de fronteras, regiones y los límites entre el espacio abierto y el espacio sin explorar. Avanzando en nuevas fronteras un robot puede extender su mapa a nuevos territorios hasta que se haya explorado el entorno en su totalidad. Una ventaja de la propuesta es la habilidad de explorar tanto espacios abiertos como lugares estrechos, desordenados, con paredes y obstáculos en orientaciones arbitrarias. La exploración tiene el potencial de liberar a los robots de esta limitación, entendiéndose por exploración al acto de moverse a través de un entorno desconocido mientras construye un mapa que puede ser utilizado para posterior navegación. Una buena estrategia de exploración es aquella que genera un mapa completo y correcto, o lo suficientemente aproximado, en un tiempo razonable.

La meta es desarrollar estrategias de exploración para los entornos complejos típicamente encontrados en edificios de oficinas del mundo real.

3.3.2. Exploración con cierres de ciclos activo

En [26] se presenta un enfoque integrado que combina exploración con SLAM. El método presentado utiliza una versión basada en una grilla y en cada punto considera acciones para el cerrado activo de ciclos durante la exploración. Volviendo a recorrer áreas previamente visitadas el robot reduce el error en su localización y, de esta forma, aprende los mapas de forma más precisa.

La contribución de este estudio es un algoritmo integrado que genera trayectorias que cierran ciclos de forma activa durante SLAM. Para lograrlo se toma en cuenta la incertidumbre de la pose del robot durante la tarea de exploración. Además, cerrando ciclos activamente, evita que el robot se vuelva demasiado confiado de su posición que es un problema típico del filtro de partículas en este contexto. Como resultado se obtienen mapas más precisos comparados con combinaciones de SLAM y exploración.

Capítulo 4

Trabajos relacionados

En este capítulo se presentan los artículos más relevantes con la temática a abordar cronológico de publicación, realizando un análisis sobre la complejidad de los mapas, sus contribuciones, resultados y conclusiones. Para medir la complejidad de los mapas se realizó una analogía con escenarios del mundo real, por ejemplo, mapas de un hogar. En este sentido se tomó en cuenta la cantidad de representaciones de habitaciones, pasillos y paredes.

4.1. A Deep Reinforcement Learning approach for Active SLAM

En [2] se formula el paradigma SLAM activo en términos de DRL sin modelos, incorporando las funciones de utilidad tradicionales basadas en la teoría del diseño experimental óptimo en las recompensas y, por lo tanto, relajando los cálculos intensivos de los enfoques clásicos. Validan dicha formulación en un entorno de simulación complejo, utilizando una arquitectura de DQN con mediciones láser como entradas de la red. Los agentes capacitados se vuelven capaces no solo de aprender una política para navegar y explorar en ausencia de un modelo del entorno, sino también de transferir sus conocimientos a mapas nunca antes vistos, que es un requisito clave en la exploración robótica.

Función de recompensa

SLAM activo se puede formular como un problema de optimización multiobjetivo [27] donde la función de costo \mathcal{J} debe contener el costo de realizar una trayectoria de colisión libre (\mathcal{F}) y una métrica de la matriz de covarianza matrix (\mathcal{U}), que representa las incertidumbres en las estimaciones de la pose del robot y el mapa. Como se muestra a continuación, ambos términos deben escalarse mediante coeficientes dependientes de la tarea y pueden considerarse en un horizonte de n pasos:

Capítulo 4. Trabajos relacionados

$$\mathcal{J} = \sum_n \beta_n \mathcal{F}_n + \sum_n \alpha_n \mathcal{U}_n \quad (4.1)$$

Dicha formulación se puede generalizar directamente al problema de RL, simplemente diseñando la función de recompensa de la siguiente manera:

$$\mathcal{R}_{\text{aug}} = \mathcal{R}_{\mathcal{F}} + \mathcal{R}_{\mathcal{U}} \quad (4.2)$$

Donde el primer término se refiere a una recompensa completamente extrínseca que da cuenta de las trayectorias libres de colisiones, y el segundo es una recompensa intrínseca, que se puede definir inversamente proporcional a una métrica de la matriz de covarianza:

$$\mathcal{R}_{\mathcal{F}} = \begin{cases} -100 & \text{if collision} \\ 1 & \text{if } \omega = 0 \\ -0,05 & \text{if } \omega \neq 0 \end{cases} \quad (4.3)$$

$$\mathcal{R}_{\text{aug}} = \begin{cases} -100 & \text{if collision} \\ 1 + \tanh\left(\frac{\eta}{f(\Sigma)}\right) & \text{if } \omega = 0 \\ -0,05 + \tanh\left(\frac{\eta}{f(\Sigma)}\right) & \text{if } \omega \neq 0 \end{cases} \quad (4.4)$$

Donde ω es la velocidad angular en rad/s , η es un factor de escala dependiente de la tarea y $f(\Sigma)$ es el criterio de D-optimal y se calcula como:

$$\mathcal{DO} = \exp\left(\frac{1}{l} \sum_{k=1}^l \log(\lambda_k)\right) \quad (4.5)$$

donde λ_k son valores propios de las entradas de la matriz de covarianza.

Evaluación y resultados

La figura 4.1 muestra los escenarios utilizados para la evaluación. En la tabla 4.1 se presenta la complejidad de cada uno de los escenarios. El entrenamiento se realizó únicamente en el Env. 1, mientras que la evaluación se realizó sobre Env. 1, Env. 2 y Env. 3.

Se desarrollaron tres agentes (DQN, DDQN y D3QN). La Figura 4.2 muestra la performance de cada agente en los tres escenarios durante la fase de evaluación en términos de tasa de éxito (SR), número de pasos por episodio y recompensa por episodio.

Se puede ver que todos los agentes se comportan bien en el Env.1 (es el mismo en el que se entrenaron) y también en el Env.2 ya que tiene una topología similar al de entrenamiento. Sin embargo, para el caso del Env.3, ninguno de los agentes

4.1. A Deep Reinforcement Learning approach for Active SLAM

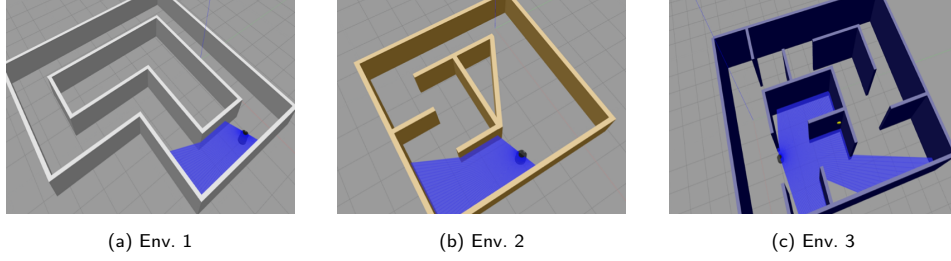


Figura 4.1: Entornos utilizados para entrenamiento y evaluación [2]

Escenario	Habitaciones	Pasillos	Paredes
(a)	1	1	7
(b)	1	2	6
(c)	1	4	7

Tabla 4.1: Complejidad de los escenarios presentados en la Figura 4.1

entrenados con $\mathcal{R}_{\mathcal{F}}$ fue capaz de recorrer todo el mapa, dejando algunas áreas como la esquina superior derecha y el callejón central sin visitar. Para mostrar el impacto que tiene no conocer el entorno a priori, se produjo otro experimento donde se le permite al agente entrenado de forma más simple almacenar información del segundo escenario durante unos pocos episodios antes de ser evaluado en él (representado como DQN^{\dagger}). Por último, se realizó otro experimento utilizando la recompensa definida como \mathcal{R}_{aug} . El agente D3QN se volvió a entrenar en el primer entorno durante 250 episodios. Este caso está representado como $D3QN^{\ddagger}$. Lo más notable de este último experimento es que aparece un valor distinto de cero en la tasa de éxito para el Env.3. Esto significa que el agente es capaz de completar el mapa en algunas ocasiones (26 %), hecho que ninguno de los agentes había logrado debido a la complejidad topológica respecto al entorno de entrenamiento.

Los mejores mapas obtenidos para los tres entornos en la etapa de evaluación se muestran en la Figura 4.3.

Los círculos rojos representan la posición inicial, las flechas rojas la trayectoria seguida, las estrellas negras los cierres de ciclo y las elipses amarillas la incertidumbre de la pose del robot en 2D.

Conclusiones

Este trabajo presenta un enfoque para resolver SLAM activo utilizando DRL sin modelo embebiendo métricas de incertidumbre en la función de recompensa. Los resultados muestran que se logró cumplir con el objetivo en entornos realistas con cierta complejidad utilizando sensores y la física del entorno. Los agentes entre-


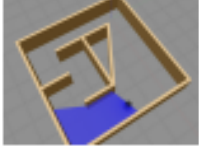

		Agent	SR(%)	Steps	R
	Env. 1	DQN	100	500 ₍₀₎	352.14 _(0.1)
		DDQN	100	500 ₍₀₎	337.56 _(2.2)
		D3QN	100	500 ₍₀₎	355.31 _(0.3)
		D3QN [†]	100	500 ₍₀₎	300.1 _(9.3)
	Env. 2	DQN	72.8	312 ₍₁₃₎	95.55 _(14.4)
		DDQN	100	500 ₍₀₎	192.52 _(3.0)
		D3QN	89.3	419 _(9.3)	196.5 _(5.7)
		DQN [†]	100	500 ₍₀₎	269.98 _(4.4)
	Env. 3	DQN	0	170 ₍₁₅₎	-24.89 _(8.3)
		DDQN	0	253 ₍₁₆₎	-42.15 _(4.8)
		D3QN	0	432 ₍₁₈₎	241.56 _(16.5)
		D3QN [†]	26	459 ₍₉₇₎	261.62 _(57.5)

Figura 4.2: Resultados obtenidos para los agentes DQN, DDQN y D3QN entrenados con $\mathcal{R}_{\mathcal{F}}$. El superíndice [†] refiere a un agente al que se le permitió entrenar brevemente previo a su evaluación en ese escenario, mientras que [‡] refiere a un agente re-entrenado con \mathcal{R}_{aug} [2]

nados fueron capaces de tomar decisiones para reducir la incertidumbre y obtener mejores trayectorias y, también, fueron capaces de transferir ese conocimiento a entornos desconocidos. La principal debilidad de este enfoque radica en la complejidad topológica de los mapas.

4.2. On reward shaping for mobile robot navigation: a RL and SLAM based approach

En [5] se presenta un algoritmo de planificación de rutas sin mapas basado en DRL para robots móviles que navegan en un entorno desconocido que solo se basa en datos de láser sin procesar de 40 dimensiones e información de odometría. El agente es entrenado usando una función de recompensa basada en el conocimiento en línea del mapa del entorno de entrenamiento, obtenido usando el filtro RBPF basado en

4.2. On reward shaping for mobile robot navigation: a RL and SLAM based approach

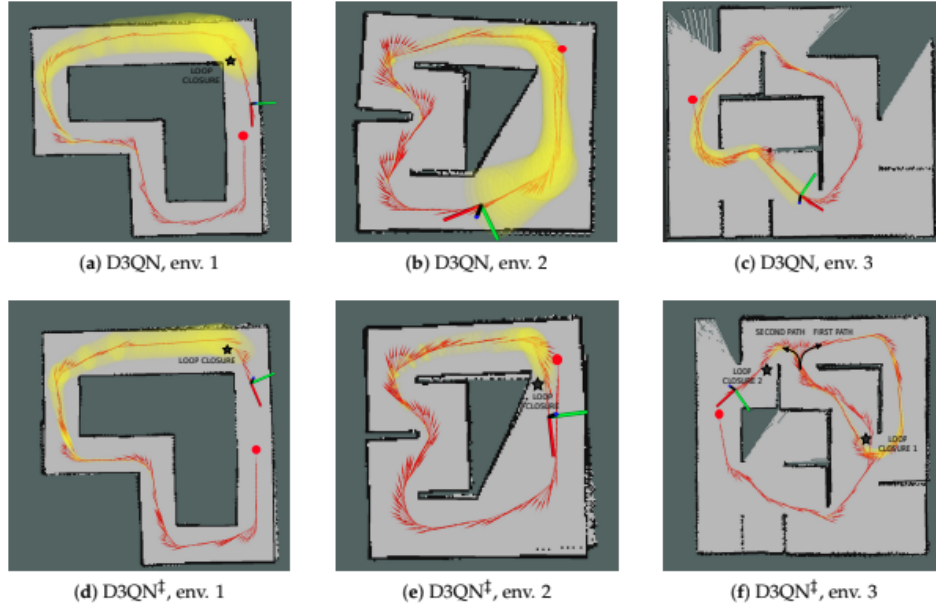


Figura 4.3: Mapas obtenidos para los 3 entornos en la etapa de evaluación [2]

cuadrícula, en un intento de mejorar la conciencia de los obstáculos del agente. El entrenamiento se realiza sobre un entorno simulado complejo y evaluado en otros dos no vistos previamente. Se muestra que la política entrenada utilizando la función de recompensa introducida no solo supera a las funciones de recompensa estándar en términos de velocidad de convergencia, sino que también mejora drásticamente el comportamiento del agente en entornos desconocidos. Además, la política formada en el entorno de simulación puede transferirse directa y correctamente a un robot real.

Función de recompensa

La meta del agente es alcanzar un objetivo deseado en el menor número de pasos de iteración evitando chocar con obstáculos. Para éste propósito, se define una función de recompensa estándar de referencia para poder comparar con la propuesta por los autores. En la misma, llamada RL en este contexto, el agente recibe una penalización proporcional a la exponencial negativa de la distancia Euclidiana entre su posición actual y la meta. Además, se agrega una recompensa dispersa si el agente alcanza el objetivo dentro de una tolerancia predefinida, así como, una penalización si el robot choca con un obstáculo o excede el número máximo de pasos de iteración T permitidos en un solo episodio sin alcanzar el objetivo o un obstáculo. La función detallada anteriormente se formula en la Ecuación 4.6:

Capítulo 4. Trabajos relacionados

$$r(s_t) = \begin{cases} r_{\text{reached}}, & d \leq d_{\text{mín}}, \\ r_{\text{crashed}}, & s_{ts}, \\ 1 - e^{\gamma d}, & \text{otherwise} \end{cases} \quad (4.6)$$

donde γ representa la tasa de caída de la exponencial.

Esta función de recompensa sólo tiene en cuenta la ubicación del objetivo por lo que, el robot no aprende la capacidad de ser consciente de los obstáculos en el entorno, solamente aprende su ubicación al chocar contra ellos.

Para mejorar el rendimiento del aprendizaje y la capacidad de percibir obstáculos, la función de recompensa propuesta se configura en función de la grilla de ocupación 2D que el robot construye durante el entrenamiento. Cada celda dentro de la grilla de ocupación se clasifica como ocupada o libre según un valor de umbral predefinido que determina su probabilidad de ocupación. La incorporación del conocimiento del entorno está ponderada por el nivel de certeza de la distribución de probabilidad posterior del mapa. Además, dado que cada obstáculo dentro del entorno está representado por un cierto número de celdas de ocupadas en la grilla, este término de recompensa se normaliza por el número total de celdas ocupadas en el campo de visión del robot. Las áreas con mayor concentración de obstáculos devolverán recompensas más negativas que las que concentran más espacios libres, asumiendo la misma confianza en el mapa dada por su posterior. Esto se formula en la Ecuación 4.7:

$$r_m(s_t, \text{map}) = \frac{1}{k} \prod_{i=0}^M p(m_i | z_{1:t}, x_{1:t}) \sum_{j=0}^k e^{-c_{j_{\text{min}}}} \quad (4.7)$$

donde $c_{j_{\text{min}}}$ representa la distancia entre el robot y una celda ocupada en su campo de visión. Ahora la función de recompensa no depende puramente del estado, sino del mapa (distancia a las celdas ocupadas), del tiempo, de la secuencia de mediciones láser $z_{1:t}$ y poses $x_{1:t}$. Finalmente, el término dependiente del mapa definido en (4.7) se resta de la recompensa densa dada en (4.6) donde las recompensas dispersas se mantienen como están. La función de recompensa propuesta, llamada RL&SLAM, se presenta en la Ecuación 4.8:

$$r(s_t, \text{map}) = \begin{cases} r_{\text{reached}}, & d \leq d_{\text{mín}} \\ r_{\text{crashed}}, & s_{ts} \\ 1 - e^{\gamma d} - r_m(s_t, \text{map}), & \text{otherwise.} \end{cases} \quad (4.8)$$

De esta forma, la función de recompensa no sólo depende de qué tan lejos esté el agente del objetivo, sino de la distancia a los obstáculos dentro del espacio de trabajo. Para visualizar cómo la función de recompensa varía con el cambio de la

4.2. On reward shaping for mobile robot navigation: a RL and SLAM based approach

posterior del mapa $p(m | z_{1:t}, x_{1:t})$ que aumenta gradualmente a medida que el robot se vuelve más confiado sobre el mapa, la función de recompensa (4.8) incluyendo el término mapa-dependiente se dibuja en dos instantes diferentes como se muestra en la Figura 4.4b y 4.4c. Es posible notar tres áreas de depresión principales correspondientes a los tres obstáculos presentes en el entorno simple que se muestra en la Figura 4.4a.

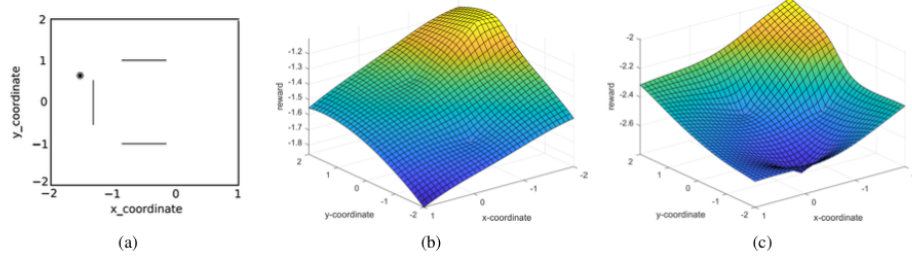


Figura 4.4: La meta está ubicada en $(-1.6, 0.65)$ representada por el círculo negro en (a). La evolución del efecto del término dependiente del mapa sobre la función de recompensa, cuando $p(m | z_{1:t}, x_{1:t}) = 0.5$ en (b) y $p(m | z_{1:t}, x_{1:t}) = 1.0$ en (c). Es posible notar las depresiones de la función de recompensa, por lo que la penalización recibida por los agentes se vuelve más severa cuanto más crece la posterior del mapa [5]

Evaluación y resultados

Para validar la efectividad del enfoque propuesto, se entrena al robot en el entorno *Env-1* mostrado en la Figura 4.5a. Para garantizar que el robot pueda alcanzar el objetivo sin tener ningún sesgo por una trayectoria, la posición inicial del robot p_0 y la ubicación de destino g , son seleccionadas al azar desde distribuciones uniformes al comienzo de cada episodio. Para evaluar más a fondo las capacidad de generalización de la política aprendida a entornos desconocidos, el agente es evaluado en *Env-2* y *Env-3*, mostrados en la Figuras 4.5b y 4.5c respectivamente.

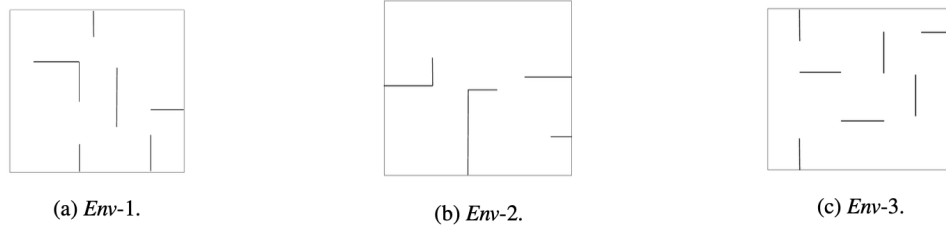


Figura 4.5: El robot es entrenado en *Env-1* ilustrado en (a). Luego, el rendimiento de la política es evaluado en los entornos *Env-2* (b) y *Env-3* (c). Todos los entornos tienen un tamaño de $5.5\text{m} \times 4\text{m}$ [5]

En la tabla 4.2 se presenta la complejidad de cada uno de los escenarios

Capítulo 4. Trabajos relacionados

Escenario	Habitaciones	Pasillos	Paredes
(a)	4	0	13
(b)	2	0	9
(c)	1	0	7

Tabla 4.2: Complejidad de los escenarios presentados en la Figura 4.5.

La Figura 4.6 muestra la evolución de la tasa de colisión respecto a la cantidad de episodios. En el inicio de la fase de entrenamiento ambos agentes muestran un rendimiento similar. Sin embargo, luego de aproximadamente 580 episodios, el robot empezó a mejorar el mapa que está construyendo y, como resultado, el término de la probabilidad $p(m | z_{1:t}, x_{1:t})$ tiene mayor impacto en la función de recompensa. El agente entrenado con la función de recompensa propuesta logra mejor entendimiento y consciencia sobre la presencia de obstáculos y, de este modo, logra encontrar caminos libres de colisiones hacia el destino en menor cantidad de episodios. Más aún, al final del entrenamiento, el enfoque propuesto alcanza una tasa de éxito de 93 % comparada con 68 % del enfoque estándar. Además, si bien ambos agentes fueron entrenados durante 2000 episodios, el del enfoque propuesto logró una reducción de un 36.9 % en la cantidad de iteraciones requeridas para lograr la convergencia. Estos resultados muestran que incorporar conocimiento del mapa en la función de recompensa mejora el rendimiento del algoritmo estándar reduciendo drásticamente la cantidad de colisiones durante el entrenamiento y, por lo tanto, mejorando la tasa de aprendizaje.

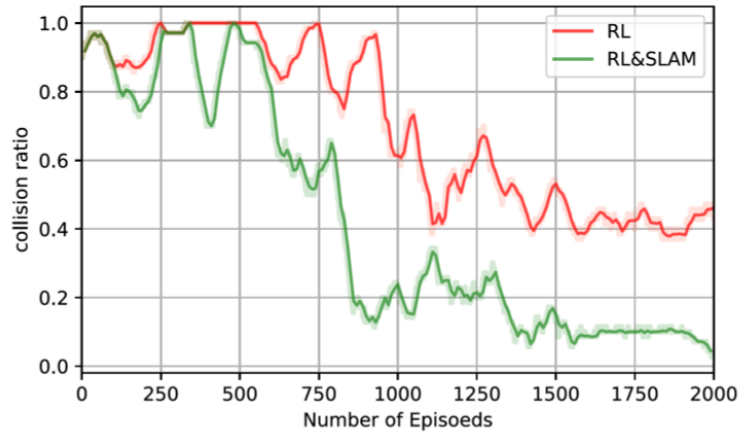


Figura 4.6: Tasa de colisiones respecto a la cantidad de episodios en etapa de entrenamiento [5]

Luego de la fase de entrenamiento, se evalúa la política aprendida en ambos enfoques sobre el mismo conjunto de 100 objetivos al azar. Primero, se efectúa la

4.2. On reward shaping for mobile robot navigation: a RL and SLAM based approach

evaluación en el mismo entorno *Env-1* y los resultados se muestran en la Figura 4.7. Luego, se transfiere la política a nuevos entornos desconocidos *Env-2* y *Env-3* y los resultados se muestran en la Figura 4.8. El enfoque propuesto no sólo alcanza una mejor tasa de éxito que los otros métodos, sino que además tiende a ejecutar una menor cantidad de acciones promedio para alcanzar los objetivos deseados. Además de aprender a explorar, debido a la mejora en la detección de obstáculos el agente puede encontrar trayectorias más cortas, suaves y eficientes hacia los objetivos.

	approach	success ratio %	number of actions (mean \pm std)
<i>Env-1</i>	RL	68 %	128.9 \pm 183.8
	RL & SLAM	93 %	86.2 \pm 68.1
	move_base	73 %	104.2 \pm 72.61

Figura 4.7: Evaluación de RL, RL & SLAM y move_base en el mismo entorno de entrenamiento *Env-1* [5]

Una vez que la política del enfoque planteado entrenada en *Env-1* es transferida a *Env-2* y *Env-3*, el rendimiento del agente aumenta significativamente tanto en la tasa de éxito como en el número de acciones ejecutadas como se puede ver en la Figura 4.8.

Si bien la estructura de *Env-2* es más sencilla y con menos obstáculos que la del entorno de entrenamiento *Env-1*, el agente entrenado utilizando la función de recompensa propuesta logra una mayor tasa de éxito 84 % contra 61 % y alcanza rendimientos ligeramente superiores a move_base, pero lo más interesante es que la cantidad de acciones realizadas disminuye drásticamente un 64.2 % y 24.1 % respectivamente.

En un entorno más fraccionado como *Env-3*, el agente propuesto mejora drásticamente la tasa de éxito, 76 % contra 31 % y 56 %, mientras que la cantidad de acciones excede por un 11.2 % las de move_base.

	approach	success ratio %	number of actions (mean \pm std)
<i>Env-2</i>	RL	61 %	149.7 \pm 208.9
	RL & SLAM	84 %	85.5 \pm 68.9
	move_base	82 %	109.6 \pm 107.34
<i>Env-3</i>	RL	31 %	99.6 \pm 115.5
	RL & SLAM	76 %	86.4 \pm 101.2
	move_base	56 %	75.2\pm62.15

Figura 4.8: Evaluación de RL, RL & SLAM y move_base en el entornos desconocidos *Env-2* y *Env-3* [5]

Capítulo 4. Trabajos relacionados

Conclusiones

Este trabajo presenta un planificador de rutas basado en DRL y SLAM para navegar en entornos desconocidos entrenado con una función de recompensa que toma en cuenta el conocimiento del mapa adquirido únicamente durante la fase de entrenamiento. Los experimentos probaron que el enfoque propuesto es capaz de alcanzar mayor velocidad de convergencia, mayor tasa de éxito durante la fase de entrenamiento, y mejor calidad de la trayectoria gracias a la reducción de la cantidad promedio de acciones tomadas. Más importante aún, el rendimiento del agente en entornos desconocidos es muy bueno. Gracias a la incorporación del conocimiento de los obstáculos en la función de recompensa, el agente puede relacionar de mejor forma las lecturas obtenidas con el sensor con buenos o malos comportamientos, ya sea al acercarse a los objetivos o a los obstáculos. Esto prueba que el agente aprendió cómo navegar de forma segura en presencia de obstáculos. Más aún, la política entrenada en el entorno de simulación puede ser transferida con éxito al robot real.

4.3. Entropy-based exploration for mobile robot navigation: a learning-based approach

En [28] se presenta una estrategia de exploración de un planificador DRL para aprender comandos de velocidad continua a partir de datos obtenidos de sensores y resolver tareas de navegación en espacios cerrados (desconocidos). Aborda un enfoque en el que la entropía del mapa (construido de forma online utilizando RBPF SLAM) se utiliza durante la fase de entrenamiento para modelar la función de recompensa. Las políticas aprendidas con este enfoque no solo permiten hacer una buena generalización a entornos y destinos desconocidos, sino que también pueden transferirse directamente, sin ningún ajuste adicional, a un robot real. Además, el espacio de acciones es continuo para obtener maniobras más avanzadas y mayor suavidad en las trayectorias.

En este trabajo se refuerza la conexión entre RL y SLAM sacando provecho del conocimiento almacenado en el mapa. En particular, se usa la entropía del mapa para mejorar las habilidades de exploración del planificador y su capacidad para escapar a mínimos locales que ocurren cuando los entornos son más complejos y realistas.

Función de recompensa

El objetivo del agente es generar comandos de velocidad para poder controlar el robot. El mismo tiene que ser capaz de llegar a su destino evitando colisionar con

4.3. Entropy-based exploration for mobile robot navigation: a learning-based approach

obstáculos y quedar atrapado en mínimos locales. Para conseguir estos 3 objetivos, la función de recompensa propuesta es igual a la suma (con pesos) de tres términos diferentes:

- **Enfocado en el destino (TD):** El agente es recompensado si está más cerca del destino que en el paso anterior.

$$r_1(s_t) = \|p_{t-1}^{x,y} - g\|_2 - \|p_t^{x,y} - g\|_2 \quad (4.9)$$

- **Enfocado en evitar obstáculos (OA):** Penalización inversamente proporcional a la distancia a los obstáculos al cuadrado. Cabe mencionar que la distancia a los obstáculos es calculada en tiempo real utilizando la información del láser, no se asume como conocida de antemano.

$$r_2(s_t) = \frac{1}{(\|p_t^{x,y} - o\|_2)^2} \quad (4.10)$$

- **Enfocado en la entropía del mapa:** El agente es recompensado si explora el entorno o si navega por espacios abiertos y/o inexplorados anteriormente.

$$r_3(s_t) = \sum_{c_f \in m} p(c_f) \log(p(c_f)) + \sum_{c_o \in m} (1 - p(c_o)) \log(1 - p(c_o)) \quad (4.11)$$

donde c_f corresponde a las celdas desconocidas y desocupadas en el mapa y c_o a las ocupadas. Además, se agrega una recompensa dispersa $r_{reached}$ si el agente alcanza el objetivo dentro de un umbral de distancia predefinido y una penalización $r_{crashed}$ si el robot colisiona con un obstáculo o si excede la cantidad máxima de pasos T en un episodio. De esta manera, la función de recompensa se define como:

$$R(s_t, m) = \begin{cases} r_{reached}, & d \leq d_{\min}, \\ r_{crashed}, & s_{ts}, \\ \lambda^g r_1(s_t) - \lambda^o r_2(s_t) - \lambda^H r_3(s_t), & \text{otherwise} \end{cases} \quad (4.12)$$

donde λ^g , λ^o , λ^H son escalares para darle peso a los términos r_i .

Evaluación y resultados

Utilizando la misma arquitectura de red y configuración de parámetros, se comparan los resultados entre 3 agentes con las siguientes características:

- TD: entrenado con la función de recompensa 4.9
- TD+OA: entrenado con la función de recompensa 4.9 + 4.10
- Propuesto: entrenado con la función de recompensa 4.12

Capítulo 4. Trabajos relacionados

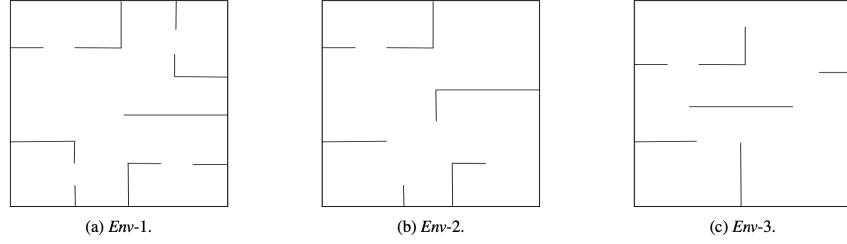


Figura 4.9: Ambientes de entrenamiento y evaluación [28]

A su vez, se definen tres entornos de prueba también: *Env-1* es donde se entrena a cada agente y luego la performance de las políticas son evaluadas en los entornos *Env-2* y *Env-3* (desconocidos de antemano).

En la tabla 4.3 se presenta la complejidad de cada uno de los escenarios.

Escenario	Habitaciones	Pasillos	Paredes
(a)	0	6	6
(b)	0	1	5
(c)	2	3	11

Tabla 4.3: Complejidad de los escenarios presentados en la Figura 4.9

Después de haber entrenado los tres agentes en el entorno *Env-1*, las políticas aprendidas se evalúan en el mismo conjunto de 100 objetivos generados al azar. La Figura 4.10 muestra la comparación de porcentaje de éxito, colisiones, timeouts (episodios finalizados sin alcanzar el objetivo ni colisionar) y el cantidad de episodios que tomaron:

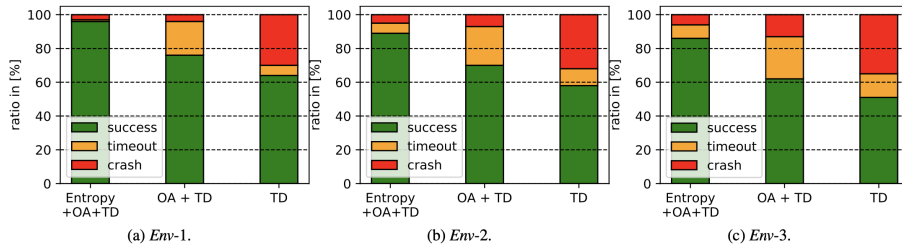


Figura 4.10: Comparación de evaluación de los 3 agentes en los respectivos ambientes [28]

El planificador entrenado con la recompensa TD pudo alcanzar el objetivo el 64 % de las veces y, en la mayoría de los episodios fallidos, colisionó con un obstáculo. El entrenamiento con la función de recompensa TD+OA logra mejores resultados ya que alcanza el objetivo un 76 % de las veces y reduce las colisiones a solamente el 3 %

4.3. Entropy-based exploration for mobile robot navigation: a learning-based approach

de las ocasiones. Sin embargo, en entornos complejos y con mínimos locales, no es suficiente solamente evitar obstáculos de forma eficiente. Esto se refleja en el incremento de “timeouts” respecto al agente TD (de 4 % a 21 %). Por otro lado, el agente entrenado con la función de recompensa propuesta es el único que logra alcanzar buenos rendimientos y llega al objetivo el 96 % de las veces.

Por otro lado, al momento de evaluar la capacidad de generalizar hacia escenarios no entrenados, se obtuvieron resultados consistentes con lo visto durante el entrenamiento. El agente entrenado con TD aprende a ser prudente y se mueve lejos de los obstáculos penalizando el largo de la trayectoria. El agente entrenado con TD+OA es menos prudente y aprende trayectorias más cortas pero, en el caso de escenarios con mínimos locales (como lo es el *Env-2*), queda atrapado y su rendimiento no es mejor que el del agente TD. El agente entrenado con la recompensa propuesta es capaz de aprender trayectorias más cortas respecto a los otros gracias a la exploración dada por el término de entropía. Incluso en escenarios desconocidos, el agente es capaz de escapar de mínimos locales, por lo tanto logró incorporar e integrar de forma efectiva las capacidades de navegación y exploración.

Finalmente, la figura 4.11 muestra la comparación de las trayectorias generadas por el planificador RL propuesto contra las generadas por *move_base* sobre el mismo conjunto de objetivos en el escenario *Env-1*. Ambos pueden alcanzar todos los objetivos planteados, sin embargo, la distancia total recorrida por el agente del enfoque propuesto es menor (14,7m vs 16,3). Además, la trayectoria generada por *move_base* no parece ser tan suave como la otra.

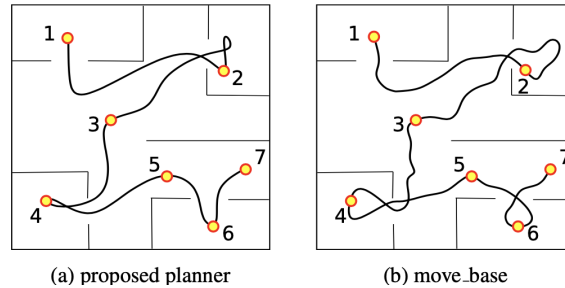


Figura 4.11: Comparación con algoritmo *move_base* [28]

Conclusiones

Aprendiendo mejores habilidades de exploración, comparado con una función de recompensa enfocada en el destino (TD) y una función recompensa enfocada en el destino y evitar obstáculos (TD+OA), el agente entrenado con el enfoque propuesto logra mejores resultados tanto en el escenario de entrenamiento (tasa de éxito 96 % contra 64 % y 76 % respectivamente) como en 2 escenarios desconocidos (éxito

89 % y 84 % contra 58 %, 51 % y 68 %, 62 % respectivamente). Además, el agente entrenado con el enfoque propuesto logra rendimientos similares a los alcanzados por *move_base*.

4.4. Curiosity-driven RL agent for mapping unknown indoor environments

En [4], proponen el uso de RL y SLAM para navegar, explorar y construir mapas de entornos interiores desconocidos de manera eficiente basado en las lecturas sensoriales a bordo del robot y la información proveniente del algoritmo SLAM, como la estimación de la pose y la completitud del mapa.

El agente, para seleccionar acciones óptimas, está capacitado para sentir curiosidad por el mundo. Este concepto se traduce en la introducción de una función de recompensa impulsada por la curiosidad que anima al agente a dirigir al agente hacia áreas desconocidas e invisibles del mundo y el mapa. Dicha función de recompensa se basa en la novedad de la posición visitada por el robot, lo que permite un rápido aprendizaje de la política de exploración y su generalización a entornos no entrenados. Además, la función de recompensa propuesta también es independiente del tipo de mapa, ya sea mapa basado en características o basado en ubicación, construido utilizando el algoritmo SLAM.

Función de recompensa

El objetivo es resolver el problema de SLAM activo utilizando RL para seleccionar las mejores acciones para explorar los entornos interiores y construir sus mapas. Los autores suponen que este problema tiene un horizonte finito, es decir, asumen la existencia de un estado terminal alcanzable en un número finito de pasos, correspondiente a la exploración total del entorno y la construcción completa del mapa. El algoritmo de RL, en este caso DDPG, se basa en 80 lecturas 2D LiDAR, la estimación de la pose del robot procedente del algoritmo RBPF de SLAM, la acción anterior realizada por el agente, el porcentaje del mapa a explorar y los pasos de tiempo restantes antes del final del episodio. Por último, en DDPG, tanto la función de la política como la de valor se aproximan mediante redes neuronales.

Proponen una función de recompensa que combina una bonificación cuando el entorno está completamente explorado y el mapa completo, una penalización cuando ocurren colisiones y una recompensa por curiosidad que premia la exploración de lugares desconocidos. La función de recompensa propuesta, llamada *Curiosity*, se muestra en la ecuación 4.13.

4.4. Curiosity-driven RL agent for mapping unknown indoor environments

$$R(s_t) = \begin{cases} r_{mapCompleted}, & c_t > \bar{c} \\ r_{crashed}, & z_t < z_{min} \\ r_t^c, & otherwise. \end{cases} \quad (4.13)$$

dónde $r_{mapCompleted}$ es la bonificación otorgada al agente cuando el porcentaje de completitud del mapa c_t está por encima de un umbral \bar{c} ; $r_{crashed}$ es una penalización negativa por acercarse demasiado, de acuerdo con un umbral fijo z_{min} , a obstáculos según la lectura actual de LiDAR z_t ; y r_t^c es el término de curiosidad que premia la exploración.

Para computar la función de recompensa planteada, se define un conjunto $X_t^{(x,y)}$ de tamaño finito M para almacenar las posiciones novedosas. Una posición es novedosa si se encuentra al menos a una distancia k de todas las posiciones previamente visitadas por el robot, siendo k una constante del algoritmo. En cada paso del algoritmo se calcula el término de la curiosidad como el promedio de las distancias Euclidianas entre la pose estimada actual del robot $X_t^{(x,y)}$ y cada una de las posiciones $X_{1..M}^{(x,y)} = X_1^{(x,y)}, \dots, X_M^{(x,y)}$ en el conjunto de posiciones novedosas de acuerdo a la Ecuación 4.14

$$r_t^c = \frac{\alpha}{M} \sum_{i=1}^M d(X_t^{(x,y)}, X_i^{(x,y)}) \quad (4.14)$$

donde α es una constante que escala el término de curiosidad de la recompensa y regula la urgencia de alcanzar posiciones novedosas, M es la cantidad de posiciones novedosas y d es la distancia Euclidiana. Si r_t^c supera un umbral dado, la posición es agregada al conjunto de posiciones novedosas. Si el conjunto se llena, se descarta una posición anterior al azar. De este modo, cualquier posición fácil de alcanzar rápidamente deja de ser de interés ya que no genera ninguna recompensa, mientras que se incentiva a alcanzar posiciones lejanas a las conocidas, al menos a una distancia k . Esta recompensa lleva al robot hacia áreas desconocidas.

Evaluación y resultados

Para estudiar el efecto de la función de recompensa sobre el rendimiento y la habilidad de generalización de los agentes, los mismos son entrenados en el entorno *Env-1*, y luego evaluados en otros dos entornos desconocidos, *Env-2* y *Env-3* mostrados en la Figura 4.12.

En la tabla 4.4 se presenta la complejidad de cada uno de los escenarios.

Cada agente es entrenado en el entorno *Env-1* durante la misma cantidad de episodios siguiendo dos estrategias diferentes. En la primera, la posición inicial del agente se mantiene durante todo el entrenamiento; mientras que en la segunda, al comienzo de cada episodio se ubica al robot en una posición al azar entre cuatro posibles.

Capítulo 4. Trabajos relacionados

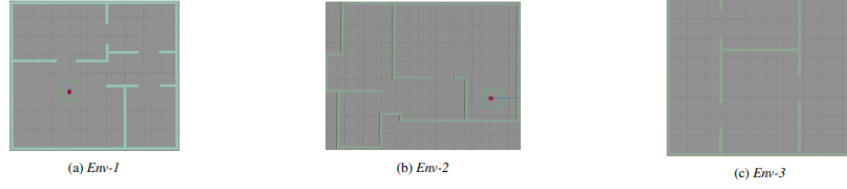


Figura 4.12: Ambientes de entrenamiento y evaluación [4]

Escenario	Habitaciones	Pasillos	Paredes
(j)	3	0	9
(k)	2	0	8
(l)	1	2	6

Tabla 4.4: Complejidad de los escenarios presentados en la Figura 4.12.

Luego del entrenamiento, cada agente se evalúa en los entornos *Env-2* y *Env-3*, desconocidos a priori, y se almacena la completitud del mapa, el largo y la calidad de la trayectoria ejecutada. Para cada entorno de evaluación, se seleccionan cuatro posibles ubicaciones iniciales y se realizan tres experimentos por posición.

Para poder medir los resultados, se compara contra funciones de recompensa de la literatura definidas en las Ecuaciones 4.15 - 4.17.

Primero se compara contra una función de recompensa dispersa, llamada *Sparse*:

$$R(s_t) = \begin{cases} r_{mapCompleted}, & c_t \geq C \\ r_{crashed}, & l_t \leq l_{min} \\ 0, & otherwise. \end{cases} \quad (4.15)$$

Luego, se compara contra una recompensa basada en la ganancia de completitud del mapa, llamada *Oracle*:

$$R(s_t) = \begin{cases} r_{mapCompleted}, & c_t \geq C \\ r_{crashed}, & l_t \leq l_{min} \\ c_t - c_{t-1}, & otherwise. \end{cases} \quad (4.16)$$

Por último, se compara con una función de recompensa basada en la entropía, llamada *Information-gain*, similar a la utilizada en [28]:

$$R(s_t) = \begin{cases} r_{mapCompleted}, & c_t \geq C \\ r_{crashed}, & l_t \leq l_{min} \\ H_t - H_{t-1}, & otherwise. \end{cases} \quad (4.17)$$

En el escenario en que el robot comienza en una posición fija, el agente entrenado con la recompensa orientada en curiosidad supera las otras en términos de tasa de

4.4. Curiosity-driven RL agent for mapping unknown indoor environments

completitud del mapa, convergiendo rápidamente luego de apróx. 150 episodios, y el número de acciones comparado a los otros agentes. La función de recompensa propuesta motiva rápidamente al agente a escoger acciones que pueden orientar al robot hacia ubicaciones desconocidas en el entorno, tan lejos de las conocidas como sea posible.

Por otra parte, en el caso en que el agente cambia de posición inicial, el entrenamiento con la función en la Ecuación 4.17 y el entrenamiento con la función en la Ecuación 4.16, alcanzan un rendimiento similar, si no apenas superior, al del agente entrenado con la función de recompensa basada en curiosidad (propuesta) en términos de completitud del mapa durante el entrenamiento. Estos dos métodos sacan mayor provecho de la posición inicial del robot al azar. Sin embargo, si se analizan las acciones tomadas en cada episodio de entrenamiento, la función de recompensa del enfoque propuesto mejora la tasa de acciones tomadas respecto a las demás.

En cuanto a la capacidad de generalización, comparan los agentes entrenados con las funciones de recompensa mencionadas con un agente que realiza exploración basada en fronteras. En la Figura 4.13 se muestran los resultados en términos de completitud del mapa y largo de la trayectoria de cada agente iniciando en cuatro posiciones distintas.

	approach	map-completeness % (mean \pm std)	traj.length (m) (mean \pm std)
<i>Env-2</i>	Sparse	55.86 \pm 17.44	11.86 \pm 3.35
	Oracle	65.28 \pm 18.53	15.89 \pm 4.4
	Information-gain	93.48 \pm 8.94	24.31 \pm 5.87
	Curiosity	99.09 \pm 0.65	17.65 \pm 5.14
	Frontier	98.45 \pm 0.366	15.68 \pm 3.6
<i>Env-3</i>	Sparse	37.08 \pm 7.88	10.7 \pm 2.25
	Oracle	66.83 \pm 11.29	18.87 \pm 10.33
	Information-gain	88.03 \pm 6.54	18.78 \pm 4.04
	Curiosity	92.58 \pm 6.69	25.23 \pm 7.93
	Frontier	99.15 \pm 0.29	25.38 \pm 4.08

Figura 4.13: Resultados de generalización en entornos *Env-2* y *Env-3* [4]

En ambos entornos, el agente entrenado con la función de recompensa propuesta alcanza mejores resultados en términos de completitud del mapa y traza trayectorias más cortas y suaves que el resto de los agentes. Del mismo modo, logra un rendimiento comparable con el de exploración basada en fronteras respecto a completitud del mapa y largo de la trayectoria, pero cabe destacar que el agente propuesto tiene un costo computacional más bajo.

Conclusiones

La clave del éxito de este enfoque es transformar el problema de la exploración de entornos desconocidos en el problema de visitar lugares novedosos del mundo y el mapa. Esto se hace moldeando la función de recompensa, utilizada para entre-

Capítulo 4. Trabajos relacionados

nar al agente de RL, para alentar su curiosidad hacia lugares no vistos. El agente propuesto, entrenado con la función de recompensa impulsada por la curiosidad, supera en términos de generalización en entornos no entrenados, completitud del mapa, longitud y suavidad de la trayectoria, a los agentes entrenados con funciones de recompensa comúnmente utilizadas para tales tareas. El enfoque propuesto logra un rendimiento comparable al método de exploración basado en fronteras [7], pero con un menor costo de cálculo. Además, el enfoque no se limita a mapas de cuadrícula de ocupación, este es el caso de la exploración basada en fronteras, sino que puede hacer frente a cualquier tipo de representación cartográfica y algoritmo SLAM. Esto se debe al hecho de que el algoritmo de entrenamiento y prueba solo requiere la estimación de la pose del robot y la integridad del mapa.

4.5. Resumen

Los enfoques presentados anteriormente muestran que los agentes entrenados son capaces no solamente de aprender una política para navegar y explorar ante la ausencia de un modelo del entorno sino de transferir el conocimiento a mapas desconocidos, un requerimiento clave para resolver SLAM Activo. Por otro lado, en todos los casos, su principal debilidad radica en la dificultad topológica de los entornos; no logrando buenos resultados en entornos medianamente complejos.

El entorno más complejo de los presentados es el (c) de Placed et. al [2] y los autores no logran completar el mapa en la mayoría de los casos.

Capítulo 5

Declaración del problema y Solución propuesta

En este capítulo se define formalmente el problema de esta tesis y se propone una solución basada en el uso de dos funciones recompensa distintas con mejoras a las presentadas en el Estado del Arte.

5.1. Declaración del problema

En este trabajo, se estudia el problema de exploración activa y mapeo en un entorno interior de dos dimensiones del estilo de una vivienda. El problema se modela como un Proceso de Decisión de Markov Parcialmente Observable (POMPD, por sus siglas en inglés) [3] definido por una tupla de seis elementos $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \omega)$ donde: (i) \mathcal{S} es el espacio de estados que no puede ser observado, (ii) \mathcal{A} se define como un espacio discreto de tres acciones, (iii) \mathcal{P} es la probabilidad de transición, (iv) \mathcal{R} es la función de recompensa, (v) \mathcal{O} se define como un espacio de observación continuo basado en láser (LiDAR), y (vi) ω es el modelo de observación. La meta del robot es maximizar el retorno esperado como se muestra en la ecuación 5.1,

$$E[\sum_{t=0}^{\infty} \gamma^t r_t] \quad (5.1)$$

donde r_t es el retorno del robot en el tiempo t y γ es el factor de descuento.

Si bien existen soluciones que modelan el problema usando POMDP, en la mayoría de los casos no lo hacen con mapas complejos ni con función de recompensa que consideren la incertidumbre de la pose como en este trabajo presentado.

5.2. Solución propuesta

En esta sección se presenta la propuesta de solución - Localización y Mapeo Simultáneo Activo Profundo. La solución se basa en una DNN entrenada con un paradigma RL. El sistema tiene como objetivo aprender una política π , que lleva al robot a explorar el entorno y mantener baja la incertidumbre de la pose. En las sub secciones de esta sección se muestran los componentes principales de la solución propuesta.

5.2.1. Función de recompensa

El diseño de la función de recompensa es un aspecto importante de RL, ya que debe incorporar el conocimiento específico de la tarea que el agente utiliza para aprender el comportamiento óptimo. Aquí se presentan dos funciones de recompensa diferentes.

La primera (5.2) es:

$$\mathcal{MC}(s_t) = \begin{cases} r_{crashed} & \text{si colisiona} \\ c_t - c_{t-1} & \text{de lo contrario} \end{cases} \quad (5.2)$$

donde $r_{crashed}$ es un escalar negativo ($r_{crashed} = -100$) si se alcanza el estado terminal de colisión y c_t es la completitud del mapa en el momento t y c_{t-1} es la completitud del mapa en el momento $t - 1$ [29]. La completitud del mapa se calcula en cada paso calculando el porcentaje de celdas cuyo valor es diferente a -1 (es decir, celdas que son conocidas por el robot y que pueden estar ocupadas o desocupadas) en la cuadrícula de ocupación.

La segunda (5.3) [2] es:

$$\mathcal{UB}(s_t) = \begin{cases} r_{crashed} & \text{si colisiona} \\ 1 + \tanh\left(\frac{1}{f(\Sigma)}\right) & \text{de lo contrario} \end{cases} \quad (5.3)$$

donde $r_{crashed}$ es un escalar negativo ($r_{crashed} = -100$) si se alcanza el estado terminal de colisión y $f(\Sigma)$ es el criterio D-optimal [2].

Dado que el valor de este criterio tiende a infinito, se ha utilizado la función \tanh para acotarlo en el intervalo $[0, 1]$.

El uso de la función de recompensa (5.2) da como resultado agentes con trayectorias más cortas, por lo que la exploración es más rápida, pero a costa de la precisión del mapa resultante. Por otro lado, el uso de esta función de recompensa (5.3) da como resultado agentes con una generación mapas resultantes más precisos.

5.2.2. Espacio de observación

De los datos sensados proporcionados por los 360 haces de luz disponibles, se consideraron (en el espacio de observación) sólo cinco frontales correspondientes a

5.2. Solución propuesta

los siguientes ángulos: 0° , 45° , 90° , 270° , 315° . En la figura 5.1 se muestra cómo se distribuyen estos haces.

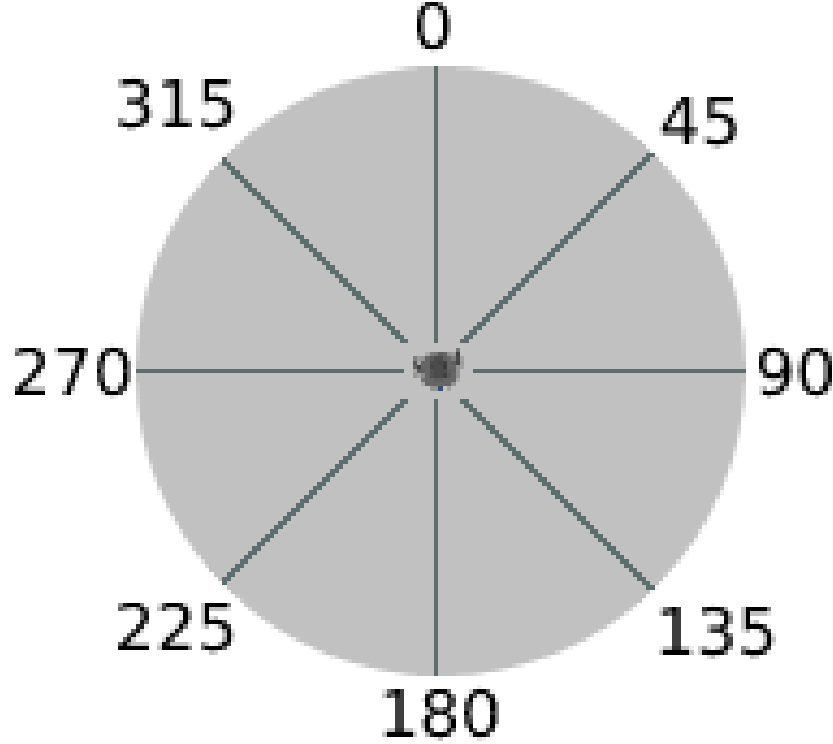


Figura 5.1: Distribución de los principales haces de luz

Sea $\mathcal{O} = [O_0, O_{45}, O_{90}, O_{270}, O_{315}]$ el vector del espacio de observación donde cada entrada de \mathcal{O} es el valor de distancia detectado por el haz de luz desde la posición x , y max_{range} el valor de distancia máxima que podría detectar el haz de luz, se considera \mathcal{O}_N como el vector del espacio de observación normalizado, donde cada una de las entradas de \mathcal{O} es dividido por max_{range} y posteriormente se redondea con dos cifras significativas, como resultado, todas sus entradas estarán entre 0 y 1. En el caso que $O_x = inf$ (es decir, el láser no impacta contra ninguna pared) entonces se reemplaza su valor por $O_x = max_{range}$.

Normalizar el espacio de observación es muy importante porque aumenta la velocidad de convergencia, evita la divergencia de parámetros y permite un ajuste de hiperparámetros más fácil [30]. Experimentalmente, no se pudo lograr la convergencia sin normalización.

5.2.3. Espacio de acción

Para mantener la sencillez del sistema y la fase de entrenamiento, se decidió utilizar un espacio de acción discreto. Las acciones definidas fueron:

- forward: la velocidad lineal es 0,5 m/s y la velocidad angular es 0.
- turnleft: la velocidad lineal es 0,05 m/s y la velocidad angular es 0,3 m/s.
- turnright: la velocidad lineal es 0,05 m/s y la velocidad angular es $-0,3$ m/s.

5.2.4. Espacio de estado

El agente selecciona acciones en base a la información contenida dentro del vector de estado s_t en el tiempo t (ecuación 5.4):

$$s_t = [O_t, a_{t-1}, r_{t-1}] \quad (5.4)$$

donde O_t es el vector del espacio de observación normalizado, a_{t-1} es la acción previa elegida por el robot y r_{t-1} es:

- el porcentaje (normalizado y redondeado con dos cifras significativas) previo de completitud del mapa en el caso del agente con función de recompensa \mathcal{MC} .
- el valor anterior de entropía (normalizada y redondeada con dos cifras significativas) en el caso del agente con función de recompensa \mathcal{UB} .

s_t se considera terminal cuando al menos una de las entradas de O_t vale 0.

5.2.5. Módulo de toma de decisiones

La toma de decisiones se realiza a través de DRL utilizando el algoritmo PPO [24].

Experimentalmente, se descubrió que PPO tenía el entrenamiento más estable y los resultados más confiables, por lo tanto, se eligió PPO como el algoritmo DRL.

Para la implementación de este módulo se utilizó Stable-Baselines3 [31] que es un conjunto de implementaciones de algoritmos de aprendizaje por refuerzo en PyTorch (biblioteca de aprendizaje automático). Se eligió Stable-Baselines3 por ser la librería de código abierto más avanzada en cuanto a implementaciones de algoritmos para aprendizaje por refuerzo.

No discutiremos detalles sobre su implementación ya que no hicimos modificaciones sobre el código. Los hiperparámetros utilizados son los predeterminados proporcionados por dicha librería, los principales se pueden ver en la tabla 5.1.

5.2. Solución propuesta

Tabla 5.1: Principales hiperparámetros de entrenamiento y simulación

Parámetro	Valor
max_n_steps_per_episode	1000
training_total_timesteps	6000000
learning_rate	0,0003
batch_size	64
n_epochs	10
gamma	0,99
gae_lambda	0,95
clip_range	0,2

5.2.6. Arquitectura

La arquitectura general presentada en la figura 5.2 muestra cómo se ha utilizado el framework ROS Noetic Ninjemys [32] para conectar los entornos de simulación (3D Gazebo robotics simulator [33]) con el algoritmo de SLAM pasivo (Gmapping [34]) y el módulo de toma de decisiones, utilizando la librería de openai_ros [35], basada en OpenAI [36] Gym. Para el algoritmo DRL se utilizó Stable-Baselines3 [31].

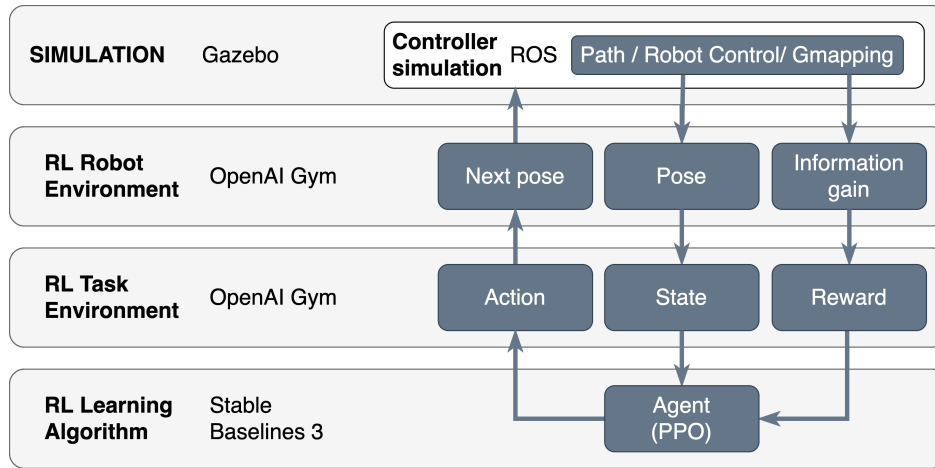


Figura 5.2: Arquitectura de la solución propuesta

5.2.7. Histórico de agentes

Definir los parámetros de la tupla POMDP y determinar el diseño de la recompensa óptima es parte de la solución propuesta. Para encontrar estos parámetros se hicieron evaluaciones preliminares en múltiples agentes. El tiempo de estas evaluaciones varía dependiendo de la cantidad de pasos de tiempo utilizados. A medida que el vector de estados se torna más complejo, mayor es la cantidad de pasos de tiempo que se necesitan para que el agente comience a presentar resultados de aprendizaje.

En las tablas 5.2 y 5.3 se puede visualizar el histórico de los distintos agentes previos que fueron entrenados hasta llegar a la solución propuesta. Para cada agente se presenta la red utilizada, el espacio de observación, la función de recompensa, el tiempo de entrenamiento y los resultados obtenidos. El espacio de acción utilizado para todos los casos fue el mismo que el que finalmente se utilizó en la solución propuesta.

Los primeros tres agentes muestran resultados muy malos, logran resolver problemas muy simples como ir en línea recta, girar en círculos o una combinación de ellos, pero no logran generalizar a otro tipo de problemas. El agente 4 aprende a explorar pero no consigue girar en 90 grados; objetivo que sí alcanza el agente 5 que es un re-entrenamiento del anterior. A partir del agente 6 en adelante se modificaron los cinco láseres frontales para probar otro ángulo de visión. A su vez, con el mismo espacio de observación, se toman en cuenta distintas combinaciones de métricas para el diseño de la función de recompensa entre las cuales están: cobertura del mapa, entropía normalizada de la pose del robot y última acción realizada por el mismo. En todos ellos se consiguen resultados al menos aceptables, logrando que explore de forma correcta pero variando en velocidad de convergencia. Los agentes 9 y 10 son los que logran un buen comportamiento en esta etapa de entrenamiento.

Por último, en la tabla 5.4 se puede visualizar la información de los agentes de la solución propuesta. El resultado obtenido por ambos es muy bueno, comportándose bien en términos de exploración sin colisiones, generalización a entornos desconocidos y, en el caso del último, tendencia activa al cerrado de ciclos.

5.2. Solución propuesta

Red	Estado	Recompensa	Tiempo	Resultado
QL	5 láseres sin normalizar (0, 30, 60, -30, -60)	+10 si avanza o rota; -100 si co- lisiona	6 ho- ras	Malo. El agente apren- de a ir en línea recta.
DQN	5 láseres sin normalizar (0, 30, 60, -30, -60)	+10 si explora; -100 si colisiona	6 ho- ras	Malo. El agente apren- de a girar en círculos.
DQN	5 láseres sin normalizar (0, 30, 60, -30, -60); co- verage del mapa sin nor- malizar	+10 si explora; -100 si colisiona	18 ho- ras	Malo. El agente apren- de a ir de punta a pun- ta y es como que gi- ra en círculos prolon- gados.

Tabla 5.2: Histórico de agentes previos entrenados usando QL o DQN.

Capítulo 5. Declaración del problema y Solución propuesta

Estado	Recompensa	Tiempo	Resultado
5 láseres normalizados (0, 30, 60, -30, -60); coverage del mapa normalizado; última acción	+1 si explora; -10 si colisiona	24 horas	Apenas aceptable. El agente aprende a explorar. No sabe girar en 90 grados.
(Re entrenamiento del agente 4)		6 horas	Aceptable, aprendió a girar en 90 grados.
5 láseres normalizados (0, 45, 90, -45, -90); coverage del mapa normalizado; última acción	+1 explora; -10 si colisiona	24 horas	Aceptable. El agente aprende a explorar.
5 láseres normalizados (0, 45, 90, -45, -90); coverage del mapa normalizado	+1 explora; -10 si colisiona	24 horas	Aceptable. El agente aprende a explorar. Se comporta igual que el que almacena la última acción pero converge más rápido.
5 láseres normalizados (0, 45, 90, -45, -90); coverage del mapa normalizado; entropía normalizada	+1 explora; +1 si reduce entropía; -10 si colisiona	24 horas	Aceptable. El agente aprende a explorar. Tarda mucho en converger.
5 láseres normalizados (0, 45, 90, -45, -90)	la misma que Placed y Castellanos, a excepción que el D-opt es $\exp(\text{entropía})$.	24 horas	Bueno. No convergió pero se comporta bien y explora.
5 láseres normalizados (0, 45, 90, -45, -90)	la misma que Placed y Castellanos	48 horas	Bueno. No convergió pero se comporta bien y explora.

Tabla 5.3: Histórico de agentes previos entrenados usando PPO.

5.2. Solución propuesta

Estado	Recompensa	Tiempo	Resultado
5 láseres normalizados (0, 45, 90, -45, -90); acción previa; porcentaje normalizado previo	Basado en completitud. diferencia de la completitud del mapa si no colisiona, -100 si colisiona.	96 horas	Muy bueno. Se comporta bien y explora sin colisionar nunca.
5 láseres normalizados (0, 45, 90, -45, -90); acción previa; entropía normalizada previa	Basado en incertidumbre. $1 + \tanh\left(\frac{1}{\bar{f}(\Sigma)}\right)$ si no colisiona, -100 si colisiona.	120 horas	Muy bueno. Se comporta bien, explora sin colisionar nunca y tiende a cerrar ciclos.

Tabla 5.4: Agentes finales usando PPO.

Capítulo 6

Experimentos y Resultados

En este capítulo se presentan los experimentos simulados realizados y los resultados de los mismos. En la sección 6.1 se presenta cómo se realizó la etapa de entrenamiento, cómo se realizaron los experimentos de simulación y qué tipos de entornos se utilizaron. Por último, en la sección 6.2 se presentan los resultados obtenidos en la fase validación del entrenamiento y la fase de evaluación de los agentes en entornos desconocidos.

6.1. Experimentos

Con el fin de evaluar el enfoque propuesto, se han realizado varios experimentos en un entorno de simulación 3D Gazebo robotics simulator sobre el Robot Operating System (ROS), con un robot móvil ROBOTIS TurtleBot3-Burger, controlado a través de comandos de velocidad y equipado con un láser LiDAR de 360 haces de luz. El alcance de LiDAR está entre 0,12 metros y 3,5 metros, para valores superiores a 3,5 metros el valor obtenido del haz es “inf” y para valores inferiores a 0,12 metros el valor obtenido del haz es “NaN”.

6.1.1. Entornos

Se han utilizado cuatro entornos diferentes (figura 6.1) durante las etapas de entrenamiento y prueba. El primero, donde se entrenó al agente, consiste en un laberinto con varios obstáculos y alta simetría. El segundo es un laberinto simple, el tercer y cuarto entorno tienen una topología más compleja y también son muy simétricos, incluidos giros consecutivos y callejones sin salida. Los últimos tres entornos descritos se han utilizado durante la fase de evaluación únicamente.

Es importante mencionar que *Env-3* y *Env-4* presentan mayor complejidad que los entornos utilizados en los artículos del estado del arte [4] [5] [2].

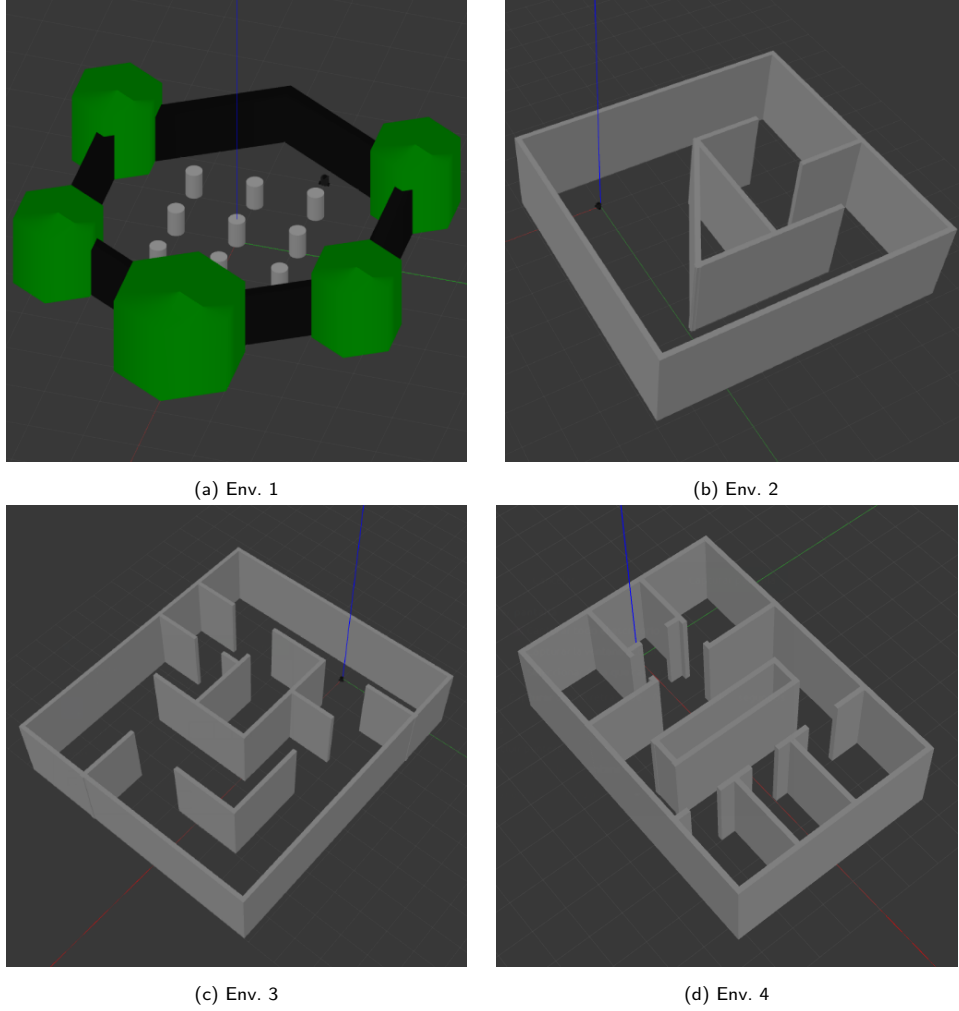


Figura 6.1: Entornos utilizados en este trabajo

En la Tabla 6.1 se desglosa la cantidad de habitaciones, pasillos y paredes de cada uno de ellos. Se puede ver que los escenarios (c) y (d) combinan varias habitaciones con pasillos y giros de 90° , lo que representa una mayor dificultad para un robot que nunca se enfrentó a dichos entornos anteriormente.

6.1.2. Entrenamiento

Una etapa de entrenamiento/prueba consta de una cierta cantidad de episodios. Cada episodio se compone de pasos de tiempo (del inglés *timesteps*) o decisiones, en los cuales, en cada episodio, el robot se mueve hasta que ocurre una colisión (es decir, se acerca a cualquier obstáculo más allá de un umbral definido de 0,12 m) o

6.1. Experimentos

Escenario	Habitaciones	Pasillos	Paredes
(a)	0	9	0
(b)	0	1	5
(c)	2	4	11
(d)	6	4	18

Tabla 6.1: Complejidad de los escenarios presentados en la Figura 6.1.

1000 pasos (es decir, decisiones) se consumen.

Los agentes fueron entrenados en *Env-1*, con un total de 6×10^6 pasos de tiempo y 1000 pasos máximos por episodio. En la figura 6.2 se muestran las gráficas de recompensa acumulada del entrenamiento. El número de episodios se asigna en el eje x (episodio_número) y la recompensa por cada episodio se asigna en el eje y (episodio_recompensa).

6.1.3. Evaluación

Los dos agentes entrenados fueron evaluados en *Env-2*, *Env-3* y *Env-4* en función de la longitud de la ruta, el tiempo, el promedio de incertidumbre de la distribución sobre la pose del robot y el porcentaje de completitud de mapa de cada episodio.

Para el cálculo de la completitud del mapa de cada episodio se utilizaron grillas de ocupación generadas por el algoritmo de exploración basado en fronteras en cada uno de los entornos utilizados. Una vez finalizado el episodio, se compara la grilla de ocupación generada por el agente vs la grilla de ocupación generada por la exploración basado en fronteras, dando así el porcentaje de completitud del mapa.

Además, ambos agentes son comparados con exploración basada en fronteras [7] y un agente de SLAM Activo avanzado (EALC) desarrollado por los escritores de esta tesis que combina exploración y cierre de ciclos activo basado en [26].

Exploration and Active Loop Closing (EALC)

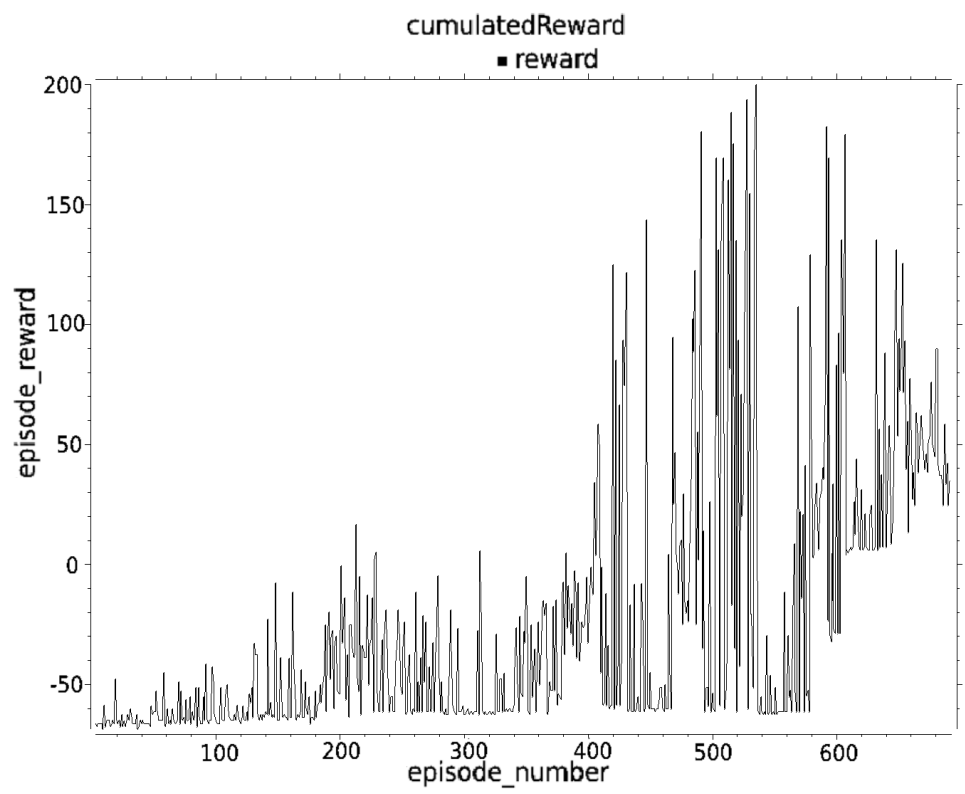
Este agente surge de la necesidad de contar con un agente de SLAM activo avanzado, el cual realiza un análisis del mapa topológico, de la entropía y del mapa métrico para tomar decisiones tales como si debe explorar, cerrar ciclos de forma activa o re-explorar poses del mapa topológico con entropía alta.

EALC utiliza exploración basado en fronteras para explorar el mapa. Paralelamente, se almacena el mapa topológico $\mathcal{G}^{[x]}$, donde x_0 es la pose inicial del robot. Cada nodo $x^{[t]}$ de \mathcal{G} tiene asignada la entropía de la pose, la distancia real a $x^{[t-1]}$ y la distancia topológica a $x^{[t-1]}$. Si existe al menos una frontera para explorar (brindada por el exploración basado en fronteras), entonces, para determinar si un

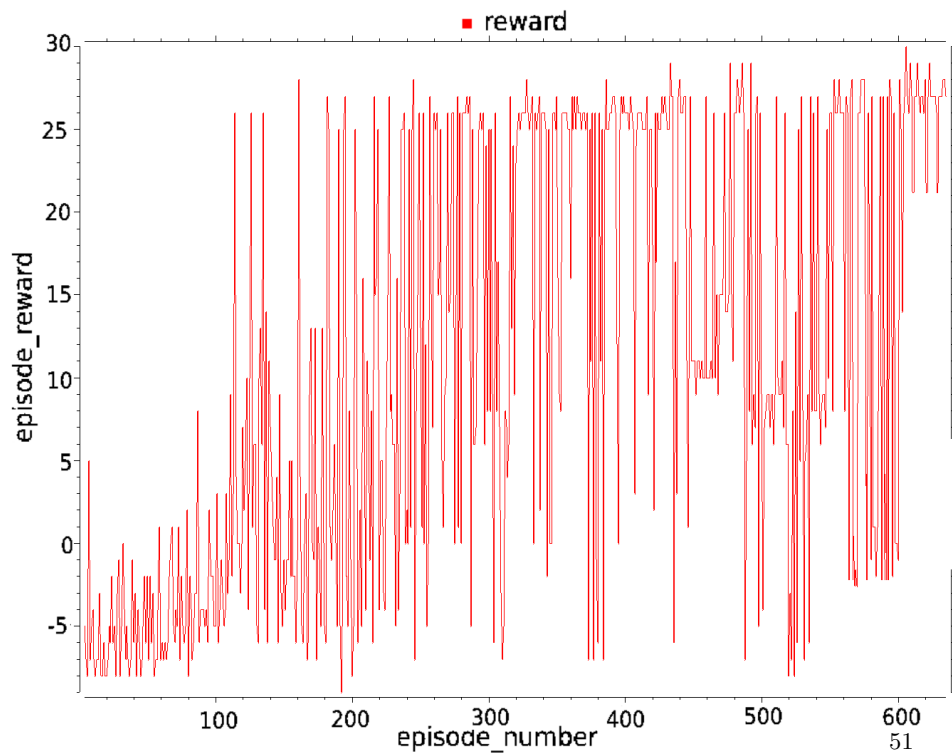
Capítulo 6. Experimentos y Resultados

ciclo puede cerrarse o no, se calcula el conjunto \mathcal{I} de poses de interés, que contiene todos los nodos que están cerca de la pose actual x_t para la distancia real pero están muy lejos dada la $\mathcal{G}^{[x]}$ topológica y la entropía es mayor que la entropía de x_t . Si $\mathcal{I} \neq \emptyset$, entonces el robot va al nodo con la distancia real más cercana y luego la pose se almacena en $\mathcal{G}^{[x]}$; de lo contrario, el robot va hacia la pose proporcionada por exploración basado en fronteras. Finalmente, si no hay una frontera que explorar, el robot vuelve a explorar cada pose de la inversa de $\mathcal{G}^{[x]}$ donde la entropía era mayor que la entropía promedio.

Se utilizó el paquete ROS explore-lite [37] para la implementación de exploración basado en fronteras.



(a) Agente basado en completitud
computedReward



(b) Agente basado en incertidumbre

Figura 6.2: Recompensa acumulada

6.2. Resultados

En esta sección se presentan tanto los resultados obtenidos en la fase de validación y los resultados obtenidos en la fase de evaluación de los agentes entrenados.

6.2.1. Resultados de validación

Después del entrenamiento, el agente basado en completitud y el agente basado en incertidumbre fueron evaluados 10 veces cada uno dentro del *Env-1* como una forma de validar la fase de entrenamiento. La tabla 6.2 muestra los resultados y la figura 6.3 muestra los mapas con sus respectivas rutas generadas.

Agente	Longitud de la ruta (m)	Tiempo (s)	Incertidumbre promedio	Completitud del mapa (%)
Completitud	12.83	105.61	4.78	98.9
Incertidumbre	20.14	164.71	4.59	100

Tabla 6.2: Resultados de entrenamiento en *Env-1*.

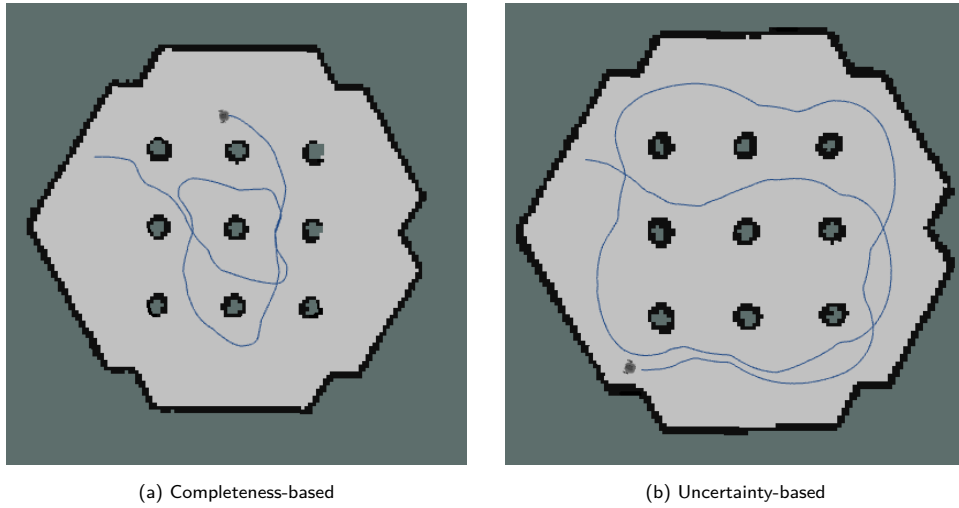


Figura 6.3: Entornos de entrenamiento

De estos resultados se puede decir que ambos agentes consiguen completar el mapa por completo evitando colisiones.

El tiempo que tarda en terminar el mapa y la longitud de la ruta generada por el agente basado en incertidumbre es mayor que la generada por el agente de basado en completitud. Sin embargo, la incertidumbre media es menor.

6.2. Resultados

Incluso en este primer mapa, se puede ver la diferencia en los objetivos de los agentes. El agente basado en completitud trata de completar el mapa lo más rápido posible, mientras que el agente basado en incertidumbre vuelve a visitar los lugares ya vistos para reducir la incertidumbre.

6.2.2. Resultados de evaluación

Los agentes fueron evaluados 10 veces cada uno dentro de *Env-2*, *Env-3* y *Env-4*. La tabla 6.3 muestra los resultados en *Env-2*, *Env-3* y *Env-4*. La figura 6.4 muestra los mapas con sus respectivas rutas generadas en *Env-2*. La figura 6.5 muestra los mapas con sus respectivas rutas generadas en *Env-3*. La figura 6.6 muestra los mapas con sus respectivas rutas generadas en *Env-4*.

Env	Agente	Longitud de la ruta (m)	Tiempo (s)	Incertidumbre promedio	Completitud del mapa (%)
2	Frontier	25.02	134.69	4.67	100
	EALC	55.61	293.49	4.05	100
	Completitud	20.34	181.53	4.76	100
	Incertidumbre	30.69	241.42	4.57	100
3	Frontier	76.04	397.68	4.20	100
	EALC	113.01	717.91	4.05	100
	Completitud	37.32	294.19	4.78	95.3
	Incertidumbre	139.15	921.52	4.51	100
4	Frontier	29.95	179.33	4.72	100
	EALC	50.92	293.77	4.11	99.28
	Completitud	42.67	325.77	4.50	100
	Incertidumbre	63.41	412.12	4.52	100

Tabla 6.3: Resultados de la evaluación en *Env-2*, *Env-3* y *Env-4*.

A partir de estos resultados, se podría decir que ambos agentes lograron aprender a realizar SLAM activo sobre entornos complejos y la generalización de capacidades a mapas no vistos.

El agente basado en completitud realiza una ruta más corta y tarda menos tiempo en completar el mapa. Sin embargo, el agente basado en incertidumbre tiene una incertidumbre de pose promedio más baja, esto se debe a que visita con frecuencia posiciones ya visitadas y por lo tanto, frecuentemente cierra ciclos. Dando como resultado la generación de mapas más precisos.

Esta diferencia en las rutas se relaciona con los diferentes objetivos de los agentes, el agente basado en completitud intenta completar los mapas lo más rápido que



Figura 6.4: Evaluación en *Env-2*

puede mientras que el basado en incertidumbre intenta no perderse en el mapa que está generando.

Aunque no existe una herramienta para comparar los mapas, el lector puede hacer un análisis visual 6.4 y notará un mejor mapa generado por el agente basado en la incertidumbre. A su vez, notará que los mapas generados por ambos agentes son mejores que los mapas generados en [2].

6.2. Resultados

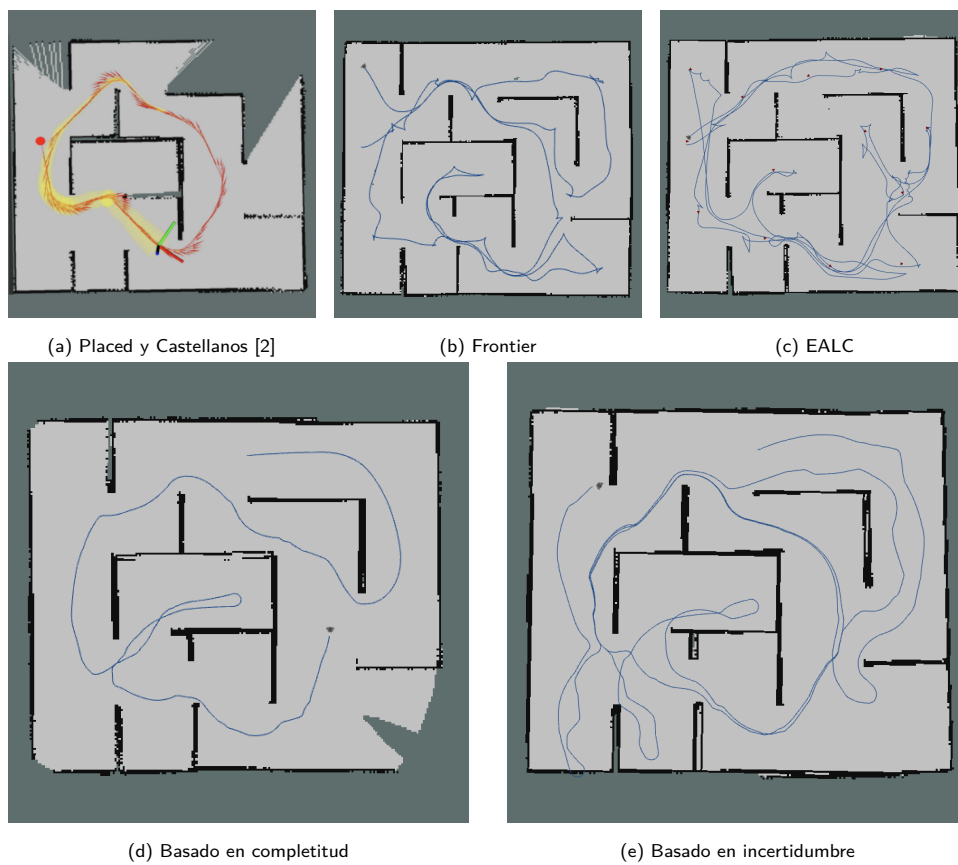


Figura 6.5: Evaluación en *Env-3*

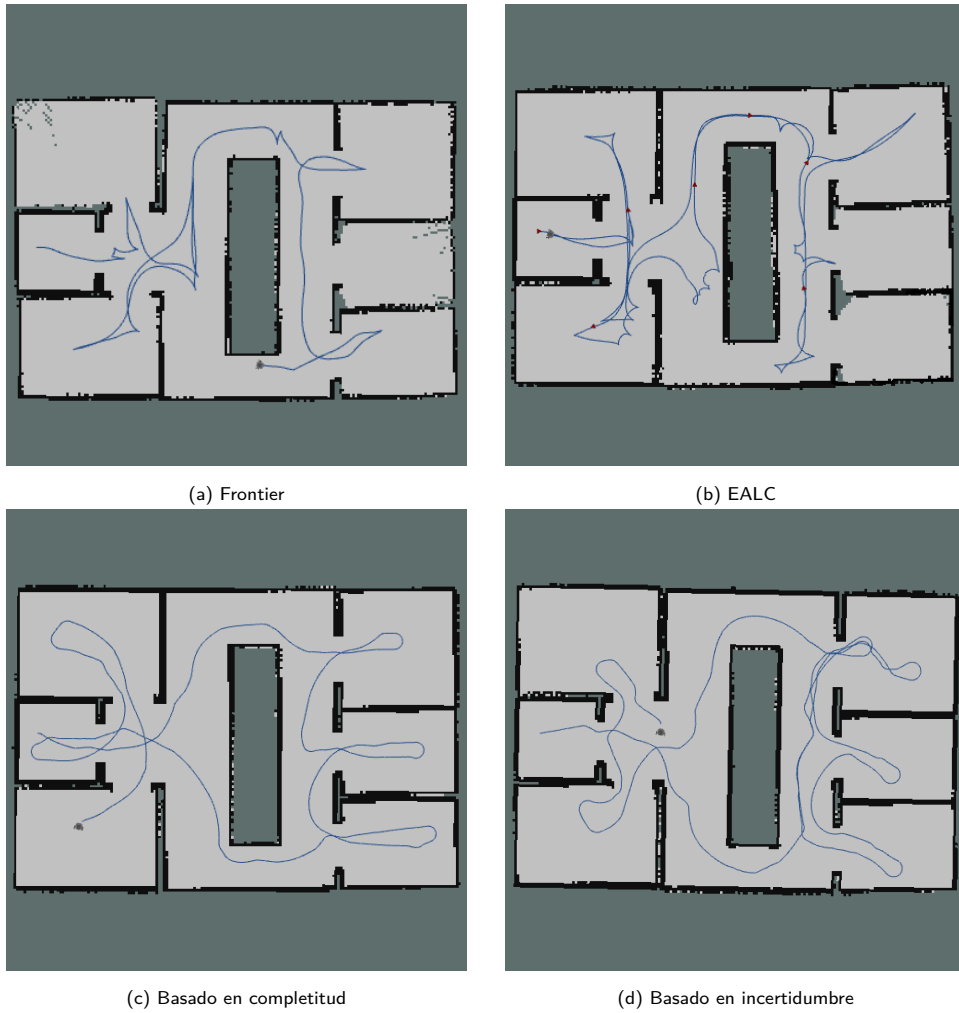


Figura 6.6: Evaluación en *Env-4*

Capítulo 7

Conclusiones y Trabajos futuros

Este es el último capítulo del documento, la sección 7.1 se utiliza para presentar las principales fortalezas, debilidades y conclusiones resultantes del trabajo de investigación realizado durante el desarrollo de esta tesis. Por último, en la sección 7.2 se presentan varias líneas de trabajo a futuro que extienden la propuesta de este documento.

7.1. Conclusiones

En este documento se presentó un enfoque novedoso en el diseño de las funciones de recompensa para resolver el paradigma SLAM activo mediante el uso de DRL sin modelo en escenarios de mayor complejidad a los utilizados en el estado del arte. Se ha integrado una arquitectura PPO de última generación sobre el simulador Gazebo y sobre un back-end ligero se integró el algoritmo SLAM pasivo que permitió la obtención de la incertidumbre del robot y, por lo tanto, el cálculo de D-Optimal. Por lo tanto, se pudo realizar SLAM activo en entornos complejos con modelos realistas para el robot, los sensores y la física del entorno. Hasta donde los autores de este trabajo saben, no existen agentes de SLAM activo entrenados con DRL que hayan sido evaluados con buenos resultados en escenarios de la dificultad presentada en este trabajo.

Se presentaron dos agentes DRL con diferentes enfoques en la función de recompensa, uno basado en la completitud del mapa y el otro en la incertidumbre del mismo. Ambos agentes aprendieron una política que les permite evitar colisiones para completar el mapa, pero el primero de ellos (agente con una función de recompensa basada en la completitud del mapa) aprendió caminos cortos para completar el mapa. Aprender un camino corto implica que el tiempo para completar el mapa sea menor, pero esto no significa que el mapa resultante sea mejor. El segundo (agente con una función de recompensa basada en la incertidumbre) obtiene tiene

Capítulo 7. Conclusiones y Trabajos futuros

un mejor mapa como resultado, esto se debe a que el agente trata de no perderse en el mapa (su función de recompensa usa el criterio D-Optimal y la entropía de la pose está en el espacio de observación). Gracias a esto, el agente no solo aprende a evitar colisiones y completar el mapa, sino que también aprende a cerrar ciclos.

Algo a destacar es el caso de *Env-4*, donde debido al diseño del mapa, el robot tiende a cerrar ciclos involuntariamente al entrar y salir de las habitaciones. Esto produce que ambos agentes se comporten de manera similar en términos de entropía promedio y mapa resultante, manteniendo una ruta más corta y menos tiempo para el agente de completitud del mapa. Este es el único caso especial en el que el agente de completitud del mapa tiene un mejor rendimiento general que el agente basado en la incertidumbre.

Además, los agentes fueron entrenados en un solo entorno y demostraron, a posteriori, la capacidad de transferir el conocimiento adquirido a mapas nunca antes vistos, lo que es un requisito clave para resolver SLAM activo.

7.2. Trabajos futuros

Durante el desarrollo de este proyecto de investigación surgieron diversos aspectos sobre los cuales se puede extender la propuesta. Esta sección expone las líneas de desarrollo que son consideradas más relevantes.

Un punto a tener en cuenta es la creación de un agente que incorpore las dos propuestas, buscando mantener una incertidumbre baja sobre la pose y a su vez generar un mapa con un alto nivel de precisión.

Otro aspecto a considerar es evaluar nuestros agentes en entornos reales.

Además, sería interesante realizar pruebas con otros algoritmos de DRL buscando optimizar el rendimiento de los agentes.

Por último, otro conjunto de pruebas que puede producir mejores agentes es probar diferentes combinaciones de hiper parámetros. Este es uno de los métodos que se usa exhaustivamente en aprendizaje reforzado para mejorar los agentes y que nosotros casi no utilizamos.

Referencias

- [1] C. Stachniss, *Robotic Mapping and Exploration*, 01 2009, vol. 55.
- [2] J. A. Placed and J. A. Castellanos, “A deep reinforcement learning approach for active slam,” *Applied Sciences*, vol. 10, no. 23, 2020. [Online]. Available: <https://www.mdpi.com/2076-3417/10/23/8386>
- [3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: <http://incompleteideas.net/book/the-book-2nd.html>
- [4] N. Botteghi, R. Schulte, B. Sirmacek, M. Poel, and C. Brune, “Curiosity-driven reinforcement learning agent for mapping unknown indoor environments,” *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. V-1-2021, pp. 129–136, 2021. [Online]. Available: <https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/V-1-2021/129/2021/>
- [5] N. Botteghi, B. Sirmacek, M. Khaled, M. Poel, and S. Stramigioli, “On reward shaping for mobile robot navigation: A reinforcement learning and slam based approach,” arXiv.org, WorkingPaper, 2020.
- [6] G. V. Trunk, “A problem of dimensionality: A simple example,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 3, pp. 306–307, 1979.
- [7] B. Yamauchi, “A frontier-based approach for autonomous exploration,” in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 1997, pp. 146–151.
- [8] Springer-Verlag, Berlin, and Heidelberg, *Springer Handbook of Robotics*. Springer, 2008. [Online]. Available: <https://www.springer.com/gp/book/9783319325507>
- [9] W. Burgard and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*, 01 2005.

Referencias

- [10] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.
- [11] K. P. Murphy, “Bayesian map learning in dynamic environments,” in *In Neural Info. Proc. Systems (NIPS)*. MIT Press, 2000, pp. 1015–1021.
- [12] H. P. Moravec, “Sensor fusion in certainty grids for mobile robots,” in *Sensor Devices and Systems for Robotics*, A. Casals, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1989, pp. 253–276.
- [13] A. Eliazar and R. Parr, “Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks.” 01 2003, pp. 1135–1142.
- [14] G. Grisetti, C. Stachniss, and W. Burgard, “Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters,” *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, Feb 2007.
- [15] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, “An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements,” vol. 1, 11 2003, pp. 206 – 211 vol.1.
- [16] B. Steux and O. E. Hamzaoui, “tinyslam: A slam algorithm in less than 200 lines c-language program,” in *2010 11th International Conference on Control Automation Robotics Vision*, 2010, pp. 1975–1979.
- [17] S. Thrun, W. Burgard, and D. Fox, “A probabilistic approach to concurrent mapping and localization for mobile robots,” *Auton. Robots*, vol. 5, no. 3–4, p. 253–271, Jul. 1998. [Online]. Available: <https://doi.org/10.1023/A:1008806205438>
- [18] S. S. Mousavi, M. Schukat, and E. Howley, “Deep reinforcement learning: An overview,” in *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016*, Y. Bi, S. Kapoor, and R. Bhatia, Eds. Cham: Springer International Publishing, 2018, pp. 426–440.
- [19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: <http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15>
- [20] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” 12 2013.

Referencias

- [21] M. Hausknecht and P. Stone, “Deep recurrent q-learning for partially observable mdps,” 07 2015.
- [22] “Naturalpolicygradients,” https://www.andrew.cmu.edu/course/10-703/slides/Lecture_NaturalPolicyGradientsTRPOPPO.pdf, (Accedido 2022-10-22).
- [23] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization,” 2015. [Online]. Available: <https://arxiv.org/abs/1502.05477>
- [24] “Ppo - stable baselines3 1.5.1a8 documentation,” <https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html>, (Accedido 2022-02-16).
- [25] “Proximal policy optimization,” https://keras.io/examples/rl/ppo_cartpole/, (Accedido 2022-02-16).
- [26] C. Stachniss, D. Hahnel, and W. Burgard, “Exploration with active loop-closing for fastslam,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 2, 2004, pp. 1505–1510 vol.2.
- [27] H. Carrillo, I. Reid, and J. A. Castellanos, “On the comparison of uncertainty criteria for active slam,” in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 2080–2087.
- [28] N. Botteghi, M. Khaled, B. Sirmacek, and M. Poel, “Entropy-based exploration for mobile robot navigation: A learning-based approach,” 2020, planning and Robotics Workshop, PlanRob 2020, PlanRob 2020 ; Conference date: 22-10-2020 Through 23-10-2020.
- [29] N. Botteghi, B. Sirmacek, M. Poel, and C. Brune, “Reinforcement learning helps slam: Learning to build maps,” *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 43, no. B4, pp. 329–336, Aug. 2020, xXIVth ISPRS Congress 2020, ISPRS 2020 ; Conference date: 04-07-2020 Through 10-07-2020. [Online]. Available: <http://www.isprs2020-nice.com>
- [30] “Normalization,” <https://machinelearningmastery.com/using-normalization-layers-to-improve-deep-learning-models/>, (Accedido 2022-10-30).
- [31] “Stable-baselines3 docs - reliable reinforcement learning implementations — stable baselines3 1.5.1a8 documentation,” <https://stable-baselines3.readthedocs.io/en/master/>, (Accedido 2022-02-16).

Referencias

- [32] “Ros,” <https://www.ros.org/>, (Accedido 2022-02-16).
- [33] “Gazebo,” <https://gazebo.org/home>, (Accedido 2022-02-16).
- [34] “Turtlebot3 slam - ros wiki,” http://wiki.ros.org/turtlebot3_slam, (Accedido 2022-02-16).
- [35] “Openai ros - ros wiki,” http://wiki.ros.org/openai_ros, (Accedido 2022-02-16).
- [36] “Gym documentation,” <https://www.gymnasium.ml/>, (Accedido 2022-02-16).
- [37] “Explore lite - ros wiki,” http://wiki.ros.org/explore_lite, (Accedido 2022-02-16).

Siglas

A2C Advantage Actor-Critic.

A3C Asynchronous Advantage Actor-Critic.

DDPG Deep Deterministic Policy Gradient.

DQN Deep Q-Network.

DRL Deep Reinforcement Learning.

DRQN Deep Recurrent Q-Network.

LiDAR Light Detection And Ranging.

MDP Markov Decision Process.

POMPD Partially Observable Markov Decision Process.

PPO Proximal Policy Optimization.

RBPF Rao-Blackwellized Particle Filtering.

RL Reinforcement Learning.

RNN Recurrent Neural Network.

SLAM Simultaneous Localization and Mapping.

SPLAM Simultaneous Planning Localization and Mapping.

TRPO Trust Region Policy Optimization.

Índice de tablas

4.1. Complejidad de los escenarios presentados en la Figura 4.1	21
4.2. Complejidad de los escenarios presentados en la Figura 4.5.	26
4.3. Complejidad de los escenarios presentados en la Figura 4.9	30
4.4. Complejidad de los escenarios presentados en la Figura 4.12.	34
5.1. Principales hiperparámetros de entrenamiento y simulación	41
5.2. Histórico de agentes previos entrenados usando QL o DQN.	43
5.3. Histórico de agentes previos entrenados usando PPO.	44
5.4. Agentes finales usando PPO.	45
6.1. Complejidad de los escenarios presentados en la Figura 6.1.	49
6.2. Resultados de entrenamiento en <i>Env-1</i>	52
6.3. Resultados de la evaluación en <i>Env-2</i> , <i>Env-3</i> y <i>Env-4</i>	53

Índice de figuras

2.1. Areas básicas de la navegación en robots móviles	4
3.1. Mapa topológico	12
3.2. Evolución de un conjunto de partículas y el mapa de la partícula más probable (aquí marcada como s^*) en tres pasos de tiempo diferentes. En las dos imágenes de la izquierda, el vehículo viajó a través de un terreno desconocido, de modo que la incertidumbre aumentó. En la imagen derecha, el robot reinició el terreno conocido para que las muestras que representen trayectorias improbables desaparecieran [1].	13
4.1. Entornos utilizados para entrenamiento y evaluación [2]	21
4.2. Resultados obtenidos para los agentes DQN, DDQN y D3QN entrenados con $\mathcal{R}_{\mathcal{F}}$. El superíndice \dagger refiere a un agente al que se le permitió entrenar brevemente previo a su evaluación en ese escenario, mientras que \ddagger refiere a un agente re-entrenado con \mathcal{R}_{aug} [2]	22
4.3. Mapas obtenidos para los 3 entornos en la etapa de evaluación [2] . .	23
4.4. La meta está ubicada en (-1.6,0.65) representada por el círculo negro en (a). La evolución del efecto del término dependiente del mapa sobre la función de recompensa, cuando $p(m z_{1:t}, x_{1:t}) = 0,5$ en (b) y $p(m z_{1:t}, x_{1:t}) = 1,0$ en (c). Es posible notar las depresiones de la función de recompensa, por lo que la penalización recibida por los agentes se vuelve más severa cuanto más crece la posterior del mapa [5]	25
4.5. El robot es entrenado en <i>Env-1</i> ilustrado en (a). Luego, el rendimiento de la política es evaluado en los entornos <i>Env-2</i> (b) y <i>Env-3</i> (c). Todos los entornos tienen un tamaño de 5.5m x 4m [5]	25
4.6. Tasa de colisiones respecto a la cantidad de episodios en etapa de entrenamiento [5]	26
4.7. Evaluación de RL, RL & SLAM y move_base en el mismo entorno de entrenamiento <i>Env-1</i> [5]	27

Índice de figuras

4.8. Evaluación de RL, RL & SLAM y move_base en el entornos desconocidos <i>Env-2</i> y <i>Env-3</i> [5]	27
4.9. Ambientes de entrenamiento y evaluación [28]	30
4.10. Comparación de evaluación de los 3 agentes en los respectivos ambientes [28]	30
4.11. Comparación con algoritmo <i>move_base</i> [28]	31
4.12. Ambientes de entrenamiento y evaluación [4]	34
4.13. Resultados de generalización en entornos <i>Env-2</i> y <i>Env-3</i> [4]	35
5.1. Distribución de los principales haces de luz	39
5.2. Arquitectura de la solución propuesta	41
6.1. Entornos utilizados en este trabajo	48
6.2. Recompensa acumulada	51
6.3. Entornos de entrenamiento	52
6.4. Evaluación en <i>Env-2</i>	54
6.5. Evaluación en <i>Env-3</i>	55
6.6. Evaluación en <i>Env-4</i>	56

Esta es la última página.
Compilado el jueves 23 febrero, 2023.