La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

Aprendizaje por refuerzo profundo en un robot móvil para la evasión de obstáculos y alcance de objetivos

Alfaro López Roberto de Jesús, Ramos Arreguín Juan Manuel, Marco Antonio Aceves Fernández, Efrén Gorrostieta Hurtado, Saúl Tovar Arriaga, Jesús Carlos Pedraza Ortega

Universidad Autónoma de Querétaro, Facultad de Ingeniería, Cerro de las Campanas s/n, Centro Universitario, C.P. 76010, Santiago de Querétaro, Querétaro.
robertoalfaro14@hotmail.com

Resumen

En el presente proyecto se describe el desarrollo de un sistema de navegación autónoma para un robot móvil utilizando aprendizaje por refuerzo. Se explica el empleo del algoritmo DQN para el proceso de entrenamiento, complementado con una arquitectura de red neuronal diseñada para interpretar eficazmente los datos recopilados por un LiDAR 2D y los sensores de orientación. Además, se presenta una metodología específica para el entrenamiento por refuerzo en tareas de navegación, la cual se enfoca en la descomposición de las tareas fundamentales: orientación y evasión de obstáculos. El documento concluye con una exposición de los resultados obtenidos mediante esta metodología, identificando áreas de mejora y proponiendo estrategias para abordar dichos retos en investigaciones futuras.

Palabras clave: Aprendizaje por refuerzo, red neuronal, DQN, robot móvil, LiDAR.

1. Introducción

Actualmente, la humanidad ha sido testigo de grandes avances tecnológicos en ciertas áreas gracias a la aplicación de métodos de inteligencia artificial, avances que hace apenas algunas décadas se pensaban casi imposibles. Por ejemplo, modelos como Dall-e, Midjourney o Stable Difussion han demostrado que los ordenadores son capaces de desempeñar tareas de creación de arte visual, por lo que la utilidad de ellos ya no se limita solo a procesos aritméticos o tareas monótonas. En paralelo, sistemas como GPT-4 o Bard, creados para generación de texto, están redefiniendo la interacción humano-máquina, pues han probado ser capaces de entender y responder al contexto de conversaciones complejas, algo que, se pensaba, estaba reservado únicamente para el ser humano. Sin embargo, a pesar de que la línea entre lo que se considera exclusivamente “humano” se ha comenzado a desdibujar, hay tareas que, a pesar de su naturaleza aparentemente más simple, no han dado un salto tan disruptivo en su proceso de resolución, este es el caso del área de la movilidad autónoma.

La movilidad autónoma ha sido, por décadas, uno de los principales objetos de estudio de la ingeniería, debido a la gran cantidad de beneficios que traería consigo, como mejoras en la seguridad y eficiencia vial, aumento en la accesibilidad para personas con limitaciones físicas, exploración de zonas inaccesibles o incluso tareas de rescate. Sin embargo, a pesar de que se han logrado grandes y numerosos avances en ella, aún resultan insuficientes como para hacer de estos sistemas algo común y totalmente seguro [1], esto debido a que, a pesar de que es una tarea muy natural y simple para cualquier ser vivo (básicamente llegar a un objetivo mientras se evaden los obstáculos del entorno),

realmente presenta varios desafíos a nivel técnico. Algunos de estos desafíos, tienen que ver con el desempeño de los sensores que se les otorgan a las máquinas, ya sea en su calidad de obtención de datos o velocidad de operación; otros se refieren a los algoritmos para la detección de los obstáculos y otros más a su método de planificación de ruta y seguimiento de trayectoria. Se observa pues, que la tarea puede dividirse en muchas partes y además, su eficiencia depende de muchos factores, muchos de ellos ambientales (el lugar o zona en donde se esté llevando a cabo la tarea) y es aquí donde radica su complejidad [1]–[3].

Es normal pues que, debido a esta naturaleza del problema, en la que se deben resolver tareas pequeñas con miras a resolver una tarea aún mayor, y en la que además existe, una gran variabilidad en cuanto al hardware y los vehículos a controlar, haya una muy grande cantidad de algoritmos y métodos que se han desarrollado para intentar resolver cada una de las etapas de este, algunos de ellos mencionados en [1], [3]. Debido a ello, es necesario dar al menos un breve repaso por algunos trabajos que pueden resultar útiles, con el fin visualizar de mejor manera la evolución del problema y su estado actual.

Muchos proyectos se enfocan en mejorar la percepción del autómata, buscando que este detecte los obstáculos de mejor manera, se localice mejor en el entorno o haga predicciones sobre sus acciones o las de los objetos a su alrededor. Entre estos trabajos destaca el desarrollado por Eppenberger y colaboradores, en este se propone un nuevo método de detección de objetos clasificándolos en estáticos y dinámicos y, además, se predice su posible trayectoria a futuro. Utilizan métodos de deep learning tales como redes neuronales convolucionales (CNN) y algoritmos de clustering para abordar el problema [4].

Igualmente, Zhou y su equipo proponen una técnica de detección y evasión de obstáculos potenciada por el trabajo en conjunto de dos algoritmos distintos, una red neuronal convolucional (CNN) y un método propio de procesamiento de imágenes de LiDAR. En este trabajo, se alcanzó una tasa de evasión de obstáculos de 90% en ambos métodos por separado, y el 100% al trabajar en conjunto [5].

Siguiendo otro enfoque, también ha habido esfuerzos de otros científicos para crear y mejorar técnicas o algoritmos de planeación de ruta y/o evasión de obstáculos. Entre estos trabajos destacan por ejemplo, el desarrollado por Soumic Sarkar y colegas, este propone cambios a una técnica de evasión de obstáculos llamada PFF (*Potential Field Force*), la cual es muy común en robótica, pero que posee la conocida desventaja de los mínimos locales, en la que el robot suele quedar atascado debido a que las fuerzas repulsivas y atractivas de los obstáculos se cancelan entre sí; su propuesta otorga una mejora a dicha técnica permitiendo al robot recuperarse ante estas situaciones [6].

Otro trabajo que aborda la evasión de obstáculos y planeación de ruta es el propuesto por Song y compañeros, en este se implementan algoritmos de lógica difusa para otorgarle a un USV (un bote no tripulado) la capacidad de llegar a una meta siguiendo un camino que evite colisiones; para la percepción del robot utilizaron un sensor LiDAR [7].

En esta misma corriente sobre lógica difusa, debe mencionarse también, el proyecto presentado por Cerbaro en el que se trabaja igualmente con un LiDAR, pero aplicado a un robot terrestre de servicio y analizando distintos enfoques para la aplicación del algoritmo [8].

Llegados a este punto, se considera crucial destacar algo que ya se mencionó en párrafos anteriores, y es que, los trabajos antes citados dependen de diversos algoritmos trabajando en conjunto para resolver un problema. Esto resulta a veces en sistemas muy complejos y que además no otorgan una correcta capacidad de generalización en caso de que el robot/vehículo se encuentre en situaciones ambientales distintas para el que fue pensado. Esto sin mencionar que las técnicas a menudo son poco intuitivas y que, además, en general, los seres vivos (que realizan la tarea evasión de obstáculos y alcance de objetivos de manera casi perfecta) no realizan cálculos demasiado complejos para llegar a un destino, ni mapean el entorno de manera tan exhaustiva, como sí lo requieren en cambio, ciertas

técnicas[2], [3]. En general los seres vivos suelen aprender mediante ensayo y error, es aquí entonces, donde entra en juego el área de la inteligencia artificial llamada aprendizaje por refuerzo.

El aprendizaje por refuerzo permite a un agente desarrollar estrategias para resolver una tarea a partir de diversas interacciones con el entorno, de esta manera, no siempre se necesitará de múltiples algoritmos que resuelvan cada subproblema, sino que, más bien, el comportamiento se adaptará a las diversas observaciones con que el modelo de inteligencia artificial se familiarice durante su entrenamiento. Ejemplos exitosos de los resultados de los que son capaces estas técnicas son los conocidos modelos ChatGPT y AlphaGo. Como es normal, el aprendizaje por refuerzo se ha visto como una manera prometedora de abordar el problema de la navegación autónoma [2], por lo que será necesario repasar igualmente algunos trabajos relevantes.

El primero de ellos que vale la pena mencionar, fue dirigido por Duguleana y Mogan. En este proyecto, se desarrolló una solución de Q-learning junto con una red neuronal para planeación de camino para un robot. Los resultados fueron probados tanto en una simulación de realidad virtual y en el mundo real, dando resultados satisfactorios [3].

Por su parte, Xie y su equipo, igualmente presentaron un trabajo de aprendizaje por refuerzo para un robot. El objetivo de este era lograr que el autómata, a través de solamente una cámara monocular RGB, pudiera navegar en un entorno sin ayuda de información 3D adicional ni algoritmos de planeación de ruta. Para alcanzar el objetivo utilizaron una arquitectura de red neuronal basada en capas convolucionales para las imágenes de entrada, y entrenada con el método D3QN (Deep Double Q network) [9].

Resulta destacada también la labor de Song y colegas. Ellos proponen un nuevo método llamado MDRLAT (Multimodal Deep Reinforcement Learning Method with Auxiliary Task) para tareas de evasión en entornos interiores. Utilizaron una red neuronal con una rama convolucional para imágenes y otra convolucional para datos de LiDAR, para el aprendizaje se usó el método *Dueling Deep Q Network* [10].

Por último, un trabajo sumamente interesante es el realizado por Surmann y colaboradores. En este se implementó el algoritmo GA3C (Asynchronus Advantage Actor Critic) para entrenar una red neuronal con el objetivo de evadir obstáculos. La red consistía en capas convolucionales 1D, las cuales tenían como entrada datos de un LiDAR 2D y de odometría del robot. Además, cabe resaltar que el equipo desarrolló su propio simulador 2D con el fin de acelerar el entrenamiento. El modelo se probó en el mundo real utilizando el LiDAR e incluso una cámara de profundidad (con ayuda de una reducción de dimensionalidad de la nube de puntos) las pruebas resultaron exitosas [11].

Habiendo revisado ya diversos métodos con los que se ha pretendido resolver el problema de la movilidad autónoma; se puede observar que aquellos basados en algoritmos de inteligencia artificial, especialmente aquellos de aprendizaje por refuerzo, se perfilan para resolver el problema de una manera ligeramente más intuitiva y poderosa, pues permiten a las máquinas aprender de manera similar a como lo hacen los seres vivos, a través de la experiencia y adaptación a distintas situaciones; y además, sin depender de un diseño excesivamente meticuloso y predefinido para cada subproblema específico [2], [3].

Debido a esto, en el presente documento se presenta la implementación de un modelo de navegación autónoma para un robot móvil. El modelo es una red neuronal de creación propia, y se entrena usando el algoritmo DQN (*Deep Q Network*). El entrenamiento se lleva a cabo a través de una metodología ligeramente distinta a la habitual, siendo el entrenamiento más gradual, entrenando primero la orientación y posteriormente la evasión, aunque todo esto se revisará a profundidad más adelante.

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

2. Aprendizaje por refuerzo

El aprendizaje por refuerzo es un área dentro del Machine Learning, en el cual un agente autónomo aprende a realizar una tarea a través de la interacción con un entorno [12]. La principal diferencia con el aprendizaje supervisado y no supervisado es que, en este, el modelo aprende gracias a una serie de castigos o recompensas que se le otorgan según si sus acciones son buenas o malas. En este sentido, las decisiones deberán estar orientadas a maximizar una recompensa acumulativa a lo largo del tiempo [12]. En pocas palabras, un agente aprende a través de "prueba y error".

Formalmente, el proceso del aprendizaje por refuerzo se puede describir mediante un proceso de decisión de Markov (MDP). Un MDP es una manera de describir un problema de decisión a través de los elementos mostrados en (1) [2].

$$< S, A, R(s,a), P(s'|s,a), \gamma > \quad (1)$$

Donde:

- ***S***: Es un conjunto de estados, es decir, las diferentes situaciones que puede tener el entorno.
- ***A***: Es un conjunto de acciones, que son todas las que el agente puede realizar.
- ***R(s, a)***: Es una función de recompensas, encargada de otorgar una puntuación al agente según sus acciones.
- ***P(s' | s, a)***: Es una función de transición, la cual describe cómo cambia el entorno ante las acciones del agente.
- ***γ***: Factor de descuento que se encarga de ajustar la importancia de las recompensas a lo largo del tiempo.

La interacción agente-entorno en un MDP se puede visualizar en la figura 1.

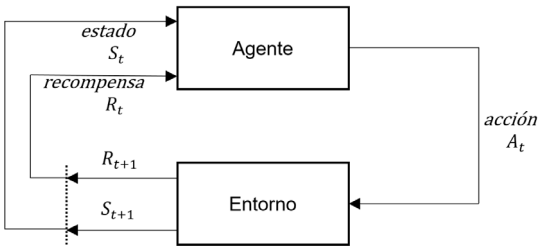

Figura 1. Interacción agente-entorno en aprendizaje por refuerzo.

Un aspecto clave en el RL es la noción de política (π), esta es una función que determina qué acción tomará el agente ante un estado del entorno. Igual de importante resulta también el concepto de "función de valor". Se manejan dos tipos, la función de valor de estado *V(s)* la cual representa la recompensa anticipada que un agente puede esperar a partir de un estado específico al seguir π; y la función de valor *Q(s, a)*, que indica la recompensa esperada a partir de un estado, pero al tomar una acción específica e igualmente siguiendo π. El objetivo final del RL es encontrar una política óptima $\boldsymbol{\pi^*}$ que maximice la recompensa esperada acumulada a lo largo del tiempo.

2.1 Q-Learning

Uno de los algoritmos más populares del RL es el Q-Learning. El objetivo de este método es buscar estimar la función de valor de acción óptima ***Q*(s, a)***; es decir, lograr que el agente tome la mejor

acción posible ante cada uno de los diferentes estados del entorno. La forma en que se logra esto, es estimando la recompensa esperada acumulativa (o sea la recompensa total hasta el final de un episodio) que se tendrá al realizar la acción ***a*** en el estado *s*, si se sigue la política óptima desde ese estado en adelante [12]. Estas estimaciones se realizan mediante la ecuación de Bellman para el Q-Learning mostrada en (2).

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (2)$$

Donde:

- ***α***: Es la tasa de aprendizaje, que indica cuánto de la nueva estimación Q contribuirá a la actualización.
- ***r***: Es la recompensa inmediata después de tomar la acción.
- ***γ***: Es el factor de descuento.

Normalmente en este método, se guardan en una tabla todas las observaciones que el agente ha visto en el entrenamiento, para después acceder a ella y revisar qué acción le resultó más beneficiosa a lo largo del tiempo. Una ventaja del Q-Learning es que puede aprender la función de valor de acción óptima ***Q*(s, a)***, sin requerir un modelo del entorno [12].

Por último, un aspecto importante en el algoritmo tiene que ver con cómo el agente adquiere experiencia, es decir, qué política seguirá al principio el agente mientras aprende a realizar la tarea. Un consenso muy popular consiste en que el agente elija acciones aleatorias al principio y que, en el transcurso del entrenamiento comience a comportarse según lo aprendido, a lo primero se le llama "exploración" y a lo segundo "explotación". Generalmente este comportamiento se implementa usando una política *ε-greedy*. Aquí, con probabilidad *ε*, el agente elige una acción al azar (exploración), y con probabilidad 1− *ε*, elige la acción que maximiza el valor actual *Q(s, a)* (explotación) [12].

2.2 Deep Q-Network (DQN)

Se mencionó en la subsección anterior, que en el Q-Learning se suelen almacenar todas las observaciones/estados vistos durante el entrenamiento en una tabla, junto con cada uno de los valores de acción *Q(s, a)* de cada uno de ellos. Esto permite que para tomar una decisión se acceda a la tabla y se escoja la acción con el valor más alto, la cual sería la mejor posible ante esa situación, en caso de que el algoritmo haya convergido correctamente. Este método funciona muy bien ante entornos cuyo espacio de estados no sea demasiado grande, pues en caso contrario, resultaría impráctico y poco factible almacenar una tabla tan grande; sin mencionar que el entrenamiento sería demasiado largo [12]. Fue en respuesta a esta problemática que surgió DQN.

En el algoritmo DQN, se aprovechan las capacidades de las redes neuronales de aproximar funciones, para realizar una estimación de la función de valor de acción *Q(s, a)* para cada estado [2]. Esta técnica permite prescindir de una tabla de dimensiones excesivas, solucionar tareas cuyos entornos poseen espacios de estados muy grandes (como imágenes) y además acelerar el entrenamiento, pues una red neuronal puede calcular todos los valores de acción de un estado al mismo tiempo, como se ve en figura 2.

Para mejorar el entrenamiento de la DQN, se utilizan dos técnicas: *"Experience Replay"* y *"Target Network"* [12].

- ***Experience Replay*:** Es un búfer de repetición. A medida que el agente adquiere experiencias, se almacenan los diversos estados, acciones, recompensas y siguientes estados que se visitaron *(s, a, r, s')*. Esto tiene dos propósitos, en primer lugar, reducir la correlación de acciones consecutivas al tomar *batches* aleatorios de este búfer para estimar los valores de acción; y segundo, poder beneficiarse de experiencias pasadas para seguir aprendiendo.

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

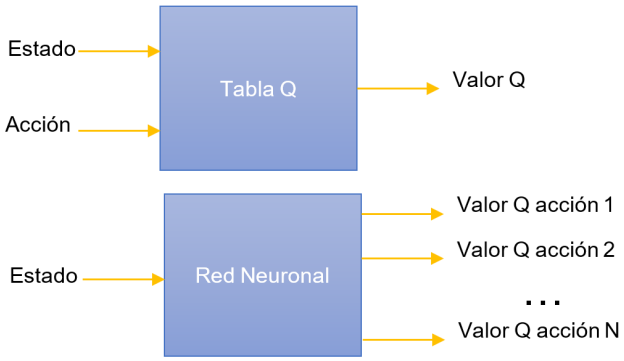

Figura 2. Una red neuronal puede predecir todos los valores de acción de un estado.

- ***Target Network*:** Esta es una segunda red que sirve como un ancla para estimar el valor Q. Y es que, si se utiliza la red principal para estimar el valor Q actual y el valor Q esperado, el entrenamiento resulta inestable. Por eso se usa esta segunda red, la cual se actualiza de manera más lenta según el usuario lo decida.

Se observa pues, que DQN puede resultar muy útil para implementación de movilidad autónoma en robots, pues esta tarea se realiza en entornos con espacios de estado muy grandes, en los que una tabla Q simplemente no sería suficiente. Es por esta razón que en el presente trabajo se decidió utilizar este algoritmo. A continuación, en las siguientes secciones se describirá a detalle el proceso llevado a cabo para implementar el aprendizaje por refuerzo en un robot móvil a través de herramientas de simulación.

3. Configuración del robot para simulación

Una característica de entrenar agentes con aprendizaje por refuerzo es que, en general, estos suelen tardar un tiempo considerable en aprender su tarea, dependiendo en gran medida del tamaño de sus espacios de estado y acción, así como del algoritmo que se esté utilizando. Normalmente esto no es un gran problema en caso de que el agente vaya a desempeñarse en entornos virtuales, pero en caso de los agentes robóticos, cuyo entorno es físico, puede resultar un gran inconveniente, no sólo porque el aprendizaje en el mundo real es aún más lento ya que el programador tiene que reacomodar el entorno al terminar un solo episodio de entrenamiento; sino, además, porque en el proceso de aprendizaje el robot podría dañarse tanto a sí mismo, como a otros seres u objetos. Es por esto que resultan de gran ayuda los simuladores, pues estos permiten llevar a cabo entrenamientos sin riesgos de daño físico al robot o a terceros, además de que aceleran el aprendizaje al reiniciar todo el entorno de manera automática cuando se inicia un nuevo episodio. Debido a esto, el desarrollo de todo el proyecto se llevó a cabo usando el simulador Gazebo, junto con el sistema operativo de robots ROS. Es importante mencionar, que este proyecto se ha hecho con miras a trasladar el modelo entrenado resultante a la realidad, debido a ello, en esta sección, se detallará la configuración que se aplicó en el robot virtual para que la simulación de este, fuera lo más fiel posible al hardware real.

El modelo del robot que se está utilizando en el proyecto es el ROSMASTER R2 de la marca Yahboom. Este robot móvil posee una configuración estilo Ackermann, lo que significa que las llantas delanteras controlan la dirección y no poseen motores, mientras que las llantas traseras se encargan de la tracción, pero no tienen capacidad de giro; esta es la configuración usada por prácticamente todos los automóviles. Para trasladar este carro de la realidad a la simulación, se utilizó un archivo URDF (usado para describir robots a través de enlaces y articulaciones) provisto por la misma empresa; sin embargo, se tuvieron que hacer numerosos cambios en él, debido a problemas al simular su

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

movimiento. Entre estos cambios se cuentan, la eliminación de los enlaces de dirección delanteros por otros más simples, la eliminación de la parte superior que correspondía a la pantalla y algunos cambios en las matrices de inercia. El URDF resultante se muestra en la figura 3, junto con una fotografía de referencia del robot real.

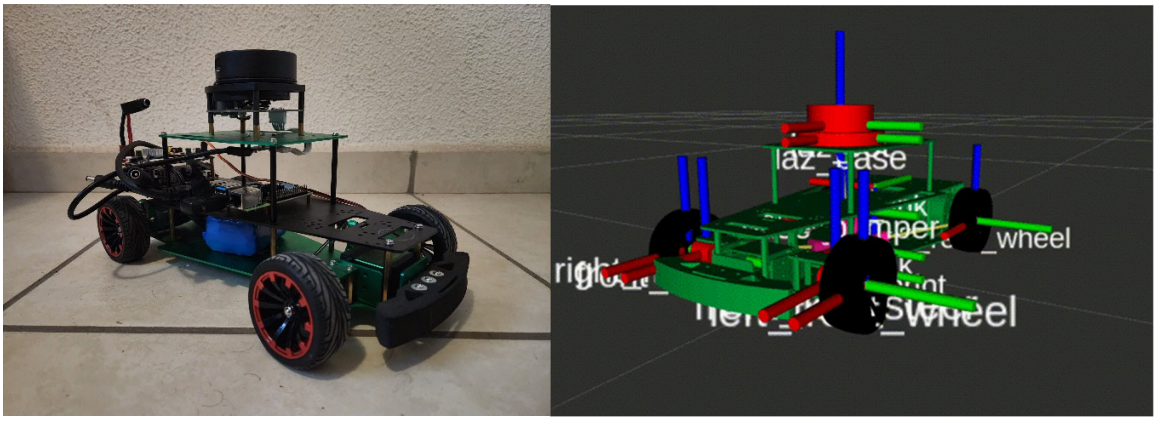

Figura 3. ROSMASTER R2 FÍSICO (izquierda) y descripción en URDF (derecha).

Con respecto a los sensores, se utilizó un sensor LiDAR 2D para la percepción, un sensor IMU para la odometría y un *"bumper"* para detectar colisiones; todos ellos se configuraron con ayuda de los *plugin* ya incluidos en Gazebo. El *bumper*, no requirió demasiado trabajo de configuración, pues se utilizaba solamente con el fin práctico de avisar al nodo de entrenamiento que se debía castigar al robot si chocaba con algún objeto, además, no era algo que se usaría en la realidad sino solamente en el entorno virtual. Por su parte el sensor IMU se configuró con una velocidad de operación de 50 Hz, recordar que este sensor entrega datos de orientación y posición del autómata.

Requirió un poco más de ajustes en cambio, el sensor LiDAR. El LiDAR que se utilizará en el robot físico es el RPLIDAR A1M8 de la marca Slamtec, el cual en términos generales posee las características de la tabla 1 [13].

Tabla 1. Especificaciones del RPLIDAR A1M8 [13].

Característica	Unidad	Valor/Rango
Tasa de escaneo	Hz	5.5
Rango de Distancia	Metro (m)	0.15-12
Ángulo de medición	Grados	0-360
Resolución	mm	<1% de la distancia

Estas mismas configuraciones fueron ajustadas en el sensor virtual, aunque la resolución se estandarizó a un valor fijo de 0.015 metros y la tasa de escaneo a 6 Hz. También vale la pena mencionar, que el sensor LiDAR real suele entregar de 270 a 310 puntos por escaneo, por lo que, para fines prácticos, en la simulación se le asignó un valor fijo de 281. Por último, se añadió un ruido gaussiano con un valor de desviación estándar de 0.01, con el fin de simular las condiciones reales de operación

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

y hacer al sistema más robusto ante las variabilidades que encontrará cuando se traspase al entorno físico.

4. Arquitectura de la red neuronal y preprocesamiento de los datos

Como se mencionó anteriormente, el algoritmo que se usó para entrenar el agente fue DQN, por lo que, una vez configurado el robot, fue necesario diseñar una red neuronal adecuada para el tipo de tarea y datos que se manejarían. El diagrama de la red neuronal se muestra en la figura 4.

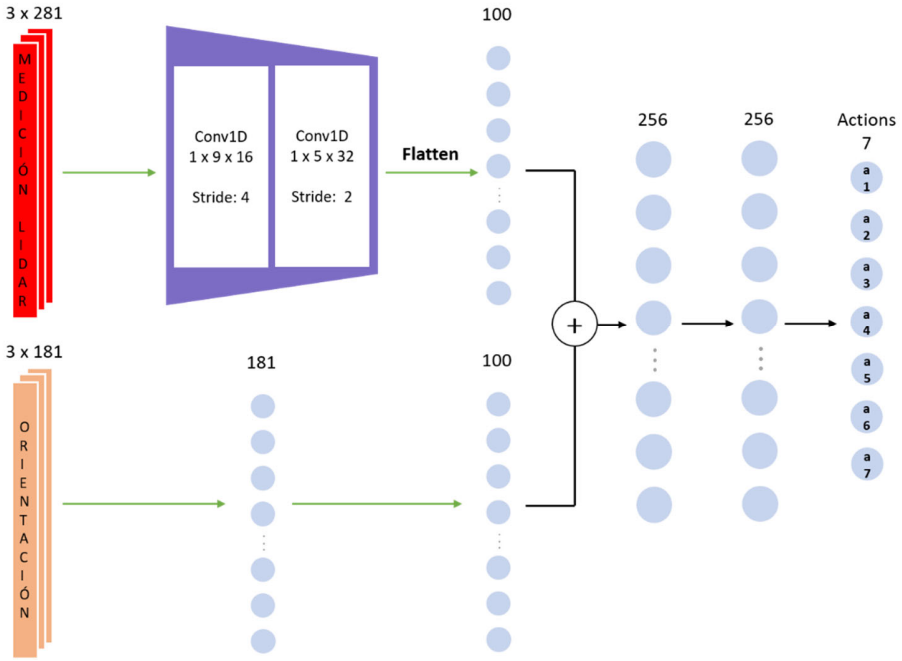

Figura 4. Arquitectura de la red neuronal del robot. Basado en [11].

Como se ve en la imagen, la red neuronal está compuesta de dos ramas de entrada diferentes, la primera posee dos capas convolucionales con tamaños de kernel de 9 y 5, así como 16 y 32 canales respectivamente, esta rama finaliza con una última capa densa de 100 neuronas. Por su parte, la segunda rama de la red es una FCN (*Fully Connected Network*) de dos capas, de 181 y 100 cada una. Las salidas de ambas ramas se concatenan finalmente, para servir de entrada a otra red neuronal densa de dos capas, de 256 neuronas ambas y con 7 salidas, correspondientes a cada una de las acciones de las que dispondrá el robot. Todas las capas usan como función de activación ReLU.

Con respecto a las entradas de la red, la primera rama se alimenta de las últimas tres mediciones que el LiDAR proporcionó del entorno; y la segunda, de la orientación que el robot tenía hacia la meta al realizar cada una de las mediciones. Sin embargo, para cada una de las entradas se realizó un preprocesamiento que ayudaría a que los datos fueran más adecuados para la red neuronal [11]:

- **Preprocesamiento de los datos del LiDAR:** Se realizaron dos acciones distintas. En primera, se sustituyen los valores *inf* (que surgen cuando el láser no encuentra algún obstáculo dentro de su rango de medición) por valores -1. Después de esto, se aplica una normalización a los datos dividiéndolos entre la distancia máxima de medición del sensor que es de 12 m.

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

- **Preprocesamiento de la orientación:** La orientación del robot se mide en grados, siendo el valor mínimo de -180° (izquierda) y máximo de 180° (derecha). Para poder introducir estos datos a la red, primero se codificó el ángulo a código binario usando un vector de 180 elementos, de tal manera que la posición donde estuviera el 1 correspondería al valor que debía girar en grados. Además de esto, se agregó un bit más que indicara la dirección del ángulo (izquierda o derecha), en este caso un valor de 1 correspondía a girar a la derecha y hacia la izquierda un 0, por lo que cada vector de orientación tiene al final del proceso 181 valores. El vector final se vería más o menos así [0, 1, 0, 0, …, 0], en este caso el vector corresponde a un valor de -1°, por lo que el robot debería girar levemente a la izquierda.

Por último, si se observa la red, hay 7 salidas. Estas 7 salidas corresponden a un valor fijo de velocidad lineal y angular y se probaron con anticipación de tal manera que fueran adecuadas para que el robot pudiera moverse en el entorno sin demasiadas restricciones. El conjunto de velocidades se muestra en la tabla 2.

Tabla 2. Set de velocidades del robot.

Velocidad (m/s)	1	2	3	4	5	6	7
Lineal	0.8	0.8	0.8	1.0	0.5	0.5	0.5
Angular	0.0	0.6	-0.6	0.0	0.0	0.4	-0.4

5. Función de recompensas

Uno de los elementos más importantes del aprendizaje por refuerzo es la función de recompensas. Este sistema de incentivos y penalizaciones es lo que capacita al agente para aprender y optimizar su desempeño en la tarea asignada. La correcta definición de esta función puede potenciar en gran medida al modelo, influyendo en su aprendizaje tanto o más que, incluso, el mismo algoritmo que se utilice para el entrenamiento. La función de recompensas que se definió después de una larga cantidad de pruebas es (3).

$$Q = r_1 + r_2 + r_3 + r_4 \tag{3}$$

Donde:

- *r1*: Es la recompensa por velocidades y tiene como propósito castigar al agente en caso de doblar, esto con el objetivo de que el robot procure seguir en la medida de lo posible una trayectoria recta, evitando oscilaciones innecesarias. Se define en (4), en donde v_l es la velocidad lineal y v_a la velocidad angular:

$$r_1 = \begin{cases} 0.01 & si\ v_l \neq 0\ y\ v_a = 0 \\ -0.01 & si\ v_a \neq 0 \end{cases} \tag{4}$$

- *r2*: Es el castigo por proximidad a los obstáculos y se otorga al traspasar el umbral de un metro. El objetivo es evitar que el agente se acerque demasiado a los objetos. Se calcula con la ecuación (5), donde d_{min} es la medición más pequeña que hay en el último escaneo láser, o sea la distancia más pequeña al obstáculo más próximo:

$$r_2 = -(1 - d_{min}) \quad si\ d_{min} < 1\ metro \tag{5}$$

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

- *r3*: Es la recompensa dada según el nivel de orientación que el agente tiene hacia el objetivo; es positiva si el ángulo actual es menor al del anterior *step* y negativa si es mayor (lo que significa que podría desviarse de la trayectoria deseada). Se calcula usando (6).

$$r_3 = (|\theta_{last}| - |\theta_{current}|) * 10 \tag{6}$$

- *r4*: Es la recompensa o castigo que se da sólo hasta el final del episodio y sólo en caso de que se haya llegado a la meta o se haya colisionado con un obstáculo (7).

$$r_4 = \begin{cases} -20 & si\ hubo\ colisión \\ +20 & si\ se\ alcanzó\ el\ objetivo \end{cases} \tag{7}$$

Esta es la función de recompensas que se definió y usó en todas las etapas del entrenamiento, se observa que es relativamente simple y, como se verá más adelante, permite al agente aprender muy bien la tarea.

6. Entrenamiento del agente

Uno de los principales aportes del presente documento, es la metodología usada para el entrenamiento del robot. Si se revisan los artículos mencionados en la introducción, se muestra una tendencia clara de colocar al agente a resolver la tarea de navegación directamente en el entorno final, y si bien se ha comprobado que es un método funcional, también es cierto que, al menos para nuestro caso, hizo que se experimentaran grandes retrasos de aprendizaje pues el agente se estancaba en mínimos locales y, en ocasiones, incluso nunca llegaba a la meta, perdiéndose así hasta por 4000 episodios de entrenamiento, momento en el cual se decidió seguir la metodología que se mostrará a continuación.

La idea es relativamente simple, y es hacer que cada una de las ramas de la red tenga un entrenamiento propio, empezando con la rama encargada de que el robot aprenda a orientarse y terminando con la sección convolucional responsable de la detección de los obstáculos. De esta manera se busca que, al llegar al entrenamiento con obstáculos, la red posea ya la capacidad de llegar a la meta y solamente le quede ajustar los pesos necesarios en la rama correspondiente para ajustar su comportamiento a los obstáculos que encuentre en el entorno. Un esquema de este método de entrenamiento se muestra en la figura 5.

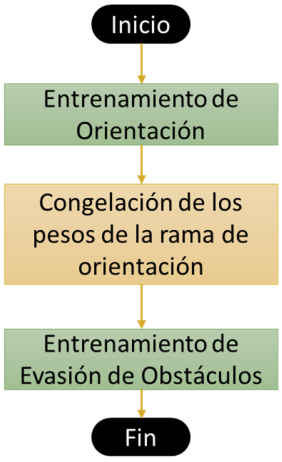

Figura 5. Proceso de entrenamiento del robot.

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

Se observa que, como se mencionó antes, cada una de las ramas tiene su propio proceso de entrenamiento, una vez que la red neuronal ha alcanzado una orientación precisa, se opta por fijar los parámetros correspondientes a esa rama específica. Esta decisión se fundamenta en la premisa de que los patrones esenciales para el reconocimiento de ángulos apropiados ya han sido capturados. Por lo tanto, cualquier ajuste adicional de estos parámetros en la fase de entrenamiento para evasión no solo resultaría redundante, sino que también podría introducir variabilidad no deseada, obstaculizando así la optimización eficiente de los parámetros restantes.

Como último punto, ambos entrenamientos se ejecutaron en una laptop equipada con 12 GB de RAM y una tarjeta gráfica Nvidia GTX 1050. La totalidad de ambos procesos tardó 4 días en completarse, aunque esto pudo optimizarse mucho deshabilitando la interfaz gráfica del simulador Gazebo.

6.1 Entrenamiento de Orientación

El entrenamiento de la fase de orientación se llevó a cabo en un entorno sin obstáculos, de tal manera que no hubiera objetos que estimularan continuamente a la rama encargada de las mediciones láser y así añadieran ruido no deseado. Los hiperparámetros usados y la configuración del entrenamiento se muestran en la tabla 3.

Tabla 3. Configuración e hiperparámetros de entrenamiento de orientación.

Configuración/Hiperparámetro	Valor
Umbral al objetivo	0.2 m
Límite de *timesteps* para un episodio	2,000
Recompensa promedio objetivo (últimos 10 episodios)	16
Learning rate	$1x10^{-6}$
Factor de descuento γ	0.99
ε inicial	1.0
ε *decay*	0.999987
ε mínimo	0.02
Tamaño de *experience replay*	40,000
Batch size	64
Timesteps para sincronización del *target network*	2,000

Es importante añadir, que en este entrenamiento se buscó que el robot hiciera un recorrido entre dos objetivos distintos, esto tenía el propósito de que el agente estimulará todos los ángulos posibles que la red podía recibir. Además, se colocó un umbral bajo de alcance de la meta con un valor de 20 centímetros lo cual permitiría al modelo ser suficientemente preciso para la tarea. La red terminó el entrenamiento después de los 1750 episodios, obteniendo la gráfica de la figura 6.

Por último, al finalizar este primer entrenamiento se analizó el desempeño de la red en un recorrido distintito y con un objetivo más a alcanzar. Una de las trayectorias generadas se muestra en la figura 7, así como la tasa de éxitos.

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

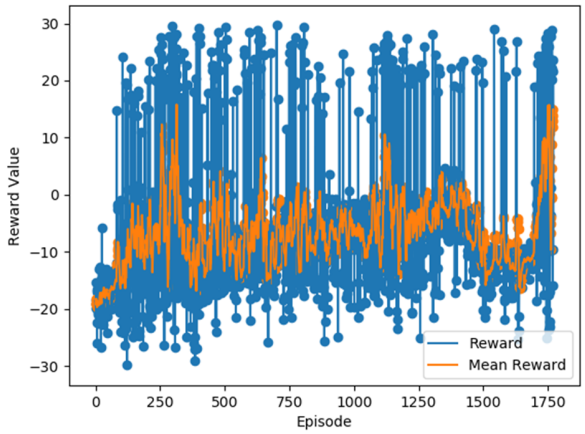

Figura 6. Recompensas por episodio y recompensas promedio del entrenamiento de orientación.

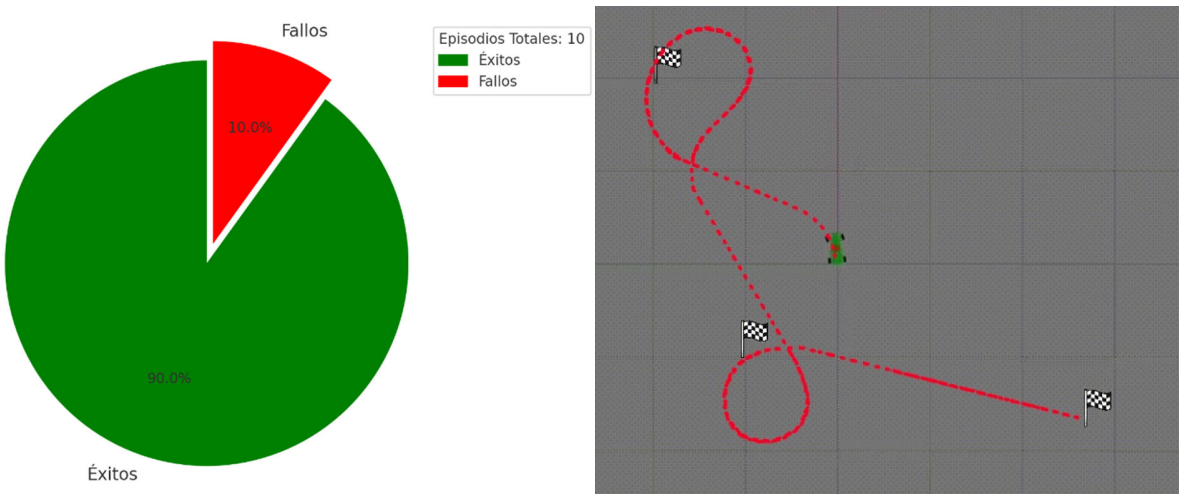

Figura 7. Porcentaje de éxitos después de 10 episodios (izquierda) y trayectoria generada para tres puntos en el espacio (derecha).

Se puede observar que el desempeño de la red es muy bueno, teniendo un porcentaje de éxitos del 90%. Por otra parte, aunque la trayectoria generada que se muestra en la imagen no es totalmente perfecta, sí es lo suficientemente buena en términos de eficiencia, careciendo de cambios demasiado abruptos y oscilaciones, además se aprecia que el modelo es capaz de recuperarse en caso de rebasar el objetivo.

6.2 Entrenamiento de Evasión de Obstáculos

La fase de entrenamiento de evasión se desarrolló en un *world* de Gazebo (archivo usado por Gazebo para describir el entorno tridimensional) donde se dispusieron múltiples obstáculos con una gran variedad de formas y tamaños, siempre asegurándose de que todos fueran detectables (en altura) por el LiDAR del robot. En el diseño de dicho entorno virtual se incluyeron zonas con amplios espacios abiertos que añadirían elementos de incertidumbre al sistema de percepción del robot. Esta característica buscó no solo enseñar al robot a esquivar obstáculos, sino también a mantener sus

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

habilidades de orientación en lugares abiertos que se había aprendido en la etapa anterior, de tal manera que pudiera navegar tanto en áreas pobladas de obstáculos, hasta en zonas completamente despejadas. Es muy importante añadir también que, cómo se mostró en la figura 5, durante este entrenamiento se congelan los pesos de la rama de orientación. Los parámetros utilizados para este entrenamiento se presentan en la tabla 4.

Tabla 4. Configuración e hiperparámetros de entrenamiento de evasión.

Configuración/Hiperparámetro	Valor
Umbral al objetivo	0.3 m
Límite de *timesteps* para un episodio	3,000
Recompensa promedio objetivo (últimos 10 episodios)	23
Learning rate	$1x10^{-5}$
Factor de descuento γ	0.99
ε inicial	1.0
ε *decay*	0.999987
ε mínimo	0.02
Tamaño de *experience replay*	40,000
Batch size	64
Timesteps para sincronización del *target network*	2,000

Debido a la complejidad de la tarea de evasión, en este entrenamiento no se buscó que el vehículo realizará algún recorrido, por lo que, para añadir variabilidad a los datos, se probó llegar al mismo objetivo, pero desde distintos puntos del mapa. Además, como se muestra en la tabla, se subió el umbral 0.3 m. Se observa en la figura 8 que, al igual que en la fase anterior, el entrenamiento terminó después de los 1750 episodios:

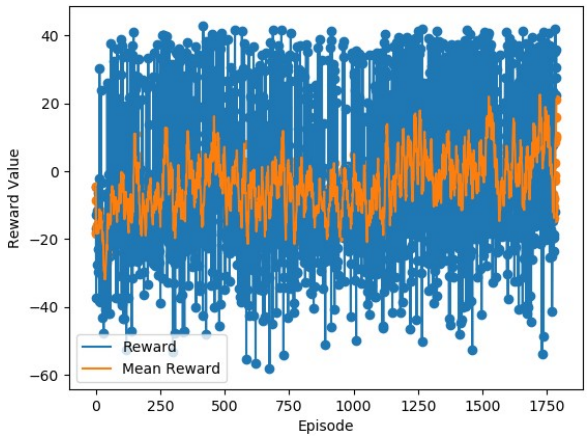

Figura 8. Recompensas por episodio y recompensas promedio del entrenamiento de evasión.

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

Los resultados de este entrenamiento se analizarán con mayor profundidad en la sección siguiente, sin embargo, para visualizar el *world* de entrenamiento y observar un breve ejemplo de trayectoria generada se coloca a continuación la figura 9.

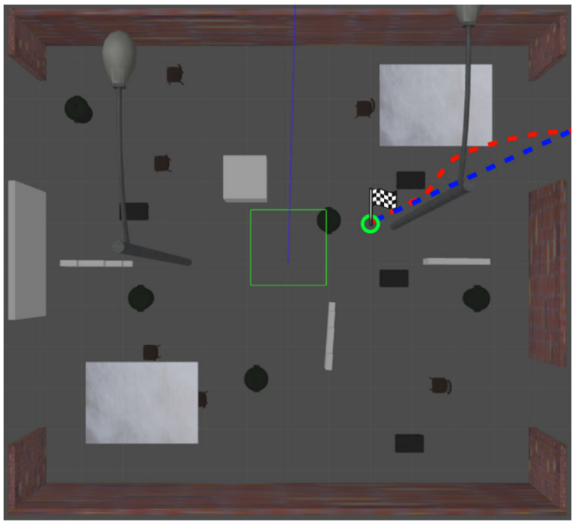

Figura 9. Trayectoria generada (rojo) contra la trayectoria ideal (azul).

7. Resultados

Para evaluar el modelo final resultante, se creó un entorno virtual distinto al del entrenamiento que permitiera medir el desempeño de la red neuronal ante observaciones que no había visto antes. De esta manera no se examinaría solamente la tasa de victorias de un episodio, sino que se obtendría una percepción aproximada de la capacidad de generalización del modelo. Los datos obtenidos al realizar esta evaluación se muestran a continuación, en donde se presentan las trayectorias generadas por el robot para un objetivo específico, así como su tasa de éxitos (para 10 episodios) y una comparación con la trayectoria ideal para cada uno de estos ejemplos.

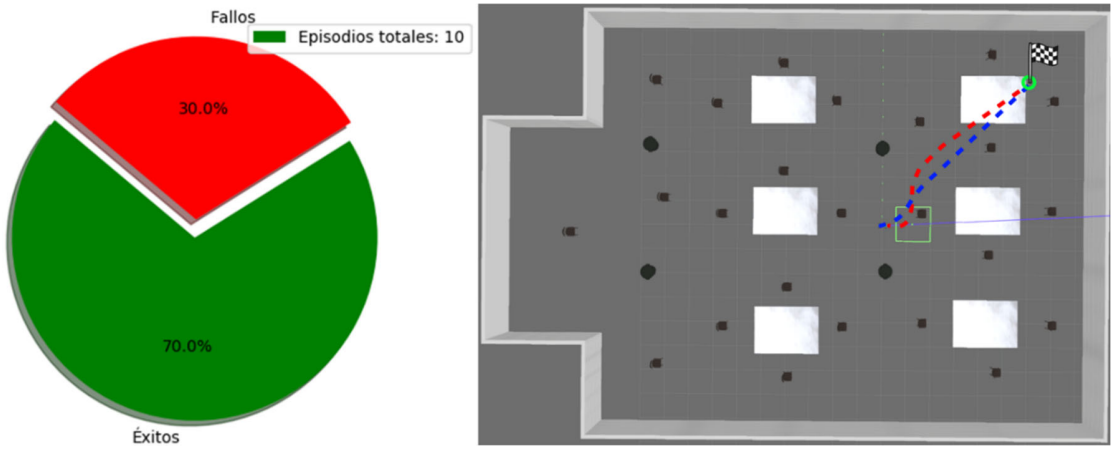

Figura 10. Trayectoria generada para el objetivo en x=6 y=6 (rojo) contra la trayectoria ideal (azul).

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

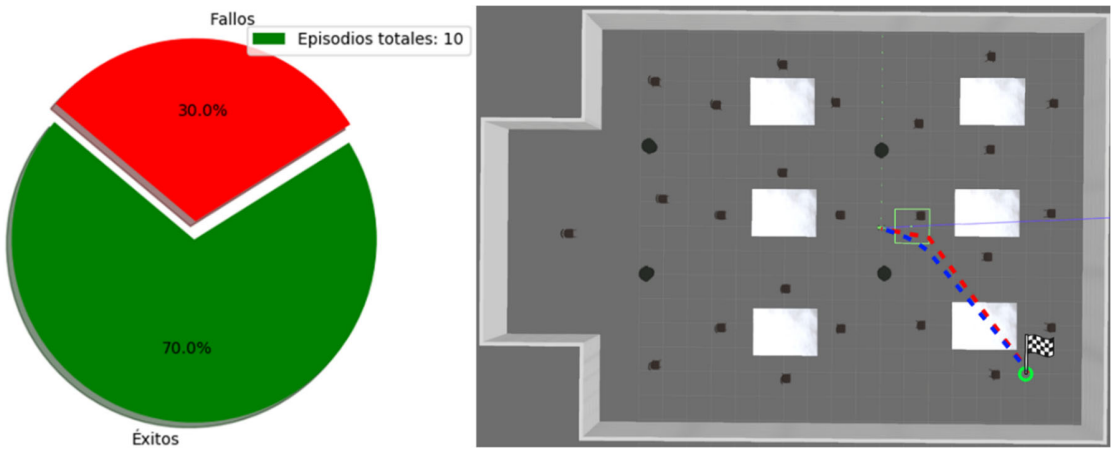

Figura 11. Trayectoria generada para el objetivo en x=6 y=-6 (rojo) contra la trayectoria ideal (azul).

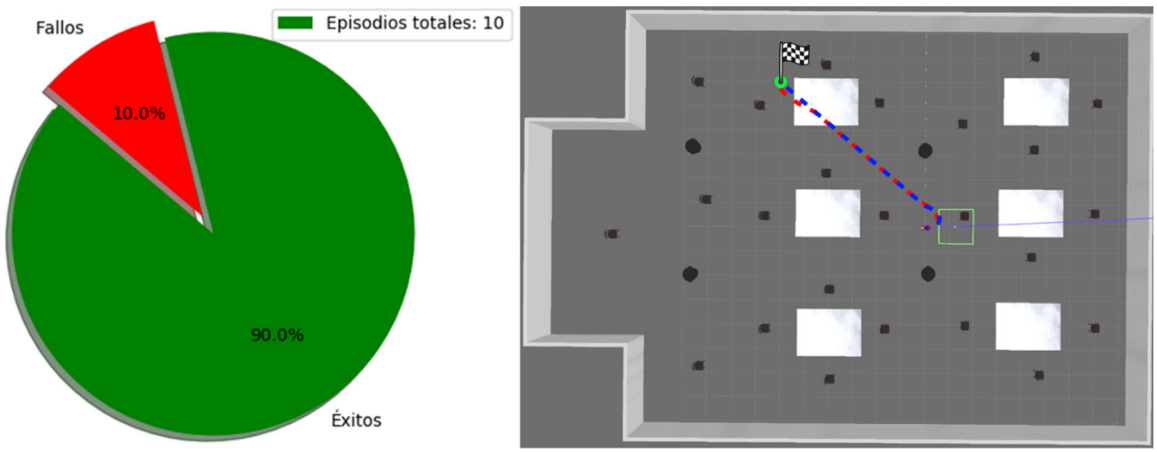

Figura 12. Trayectoria generada para el objetivo en x=-6 y=6 (rojo) contra la trayectoria ideal (azul).

Se puede observar que el robot posee un desempeño aceptable y una gran capacidad de generalización ante entornos desconocidos. Sin embargo, por otra parte, también se logró observar que el autómata llega a tener deficiencias en su precisión al llegar a cualquier meta. Esto último suele ser la principal causa de colisiones con obstáculos debido al intento de autocorrección del robot al haber sobrepasado el objetivo final o a que la meta se encuentra muy cerca de un objeto. Esto se muestra gráficamente en las figuras 13 y 14.

El principal factor que contribuye a la deficiencia mencionada reside en la configuración estructural del robot, específicamente su diseño Ackerman. Esta estructura añade una complejidad adicional a la tarea de navegación debido a sus limitaciones inherentes, como la incapacidad del robot para rotar sobre su propio eje y cambiar de dirección. Esta restricción se traduce en una maniobrabilidad limitada en espacios confinados, ya que el robot requiere de un área más amplia para efectuar giros, tal como se ilustra en la figura 15, lo que lo hace más propenso a colisiones. Se verá en el siguiente capítulo que esta limitación puede abordarse de muchas maneras, sin embargo, esto sobrepasa el alcance del presente proyecto.

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

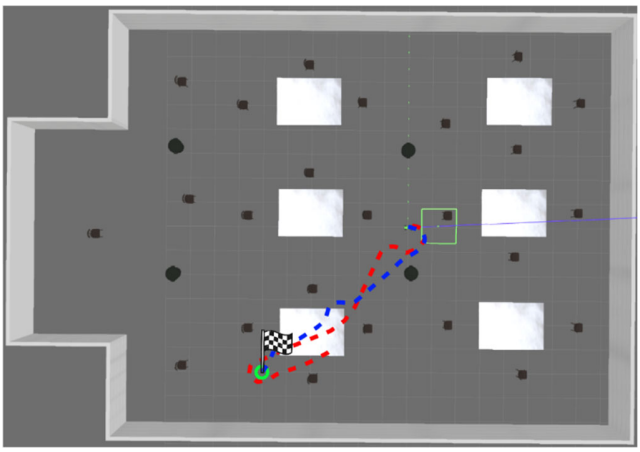

Figura 13. Colisión generada en la trayectoria para el objetivo en x=-6 y=-6 (rojo) contra la trayectoria ideal (azul).

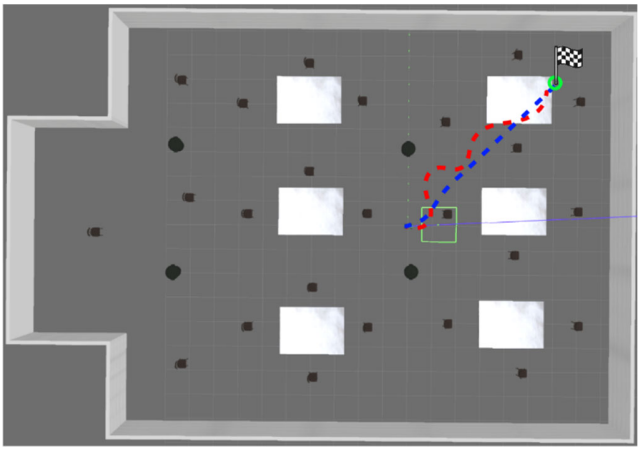

Figura 14. Colisión generada en la trayectoria para el objetivo en x=6 y=6 (rojo) contra la trayectoria ideal (azul).

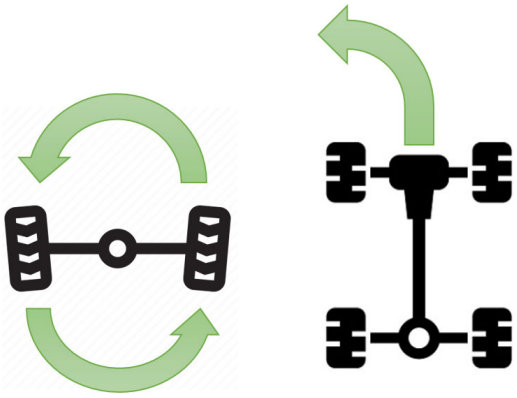

Figura 15. Diferencias en el giro de un robot diferencial y un robot con estructura Ackermann.

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

Adicional a esto, se encontró un área adicional de mejora, pues a menudo el robot suele generar trayectorias ligeramente distintas (a veces muy distintas) para alcanzar un objetivo, lo que puede hacerlo ligeramente impredecible, ejemplos de esto se muestran en las figuras16 y 17.

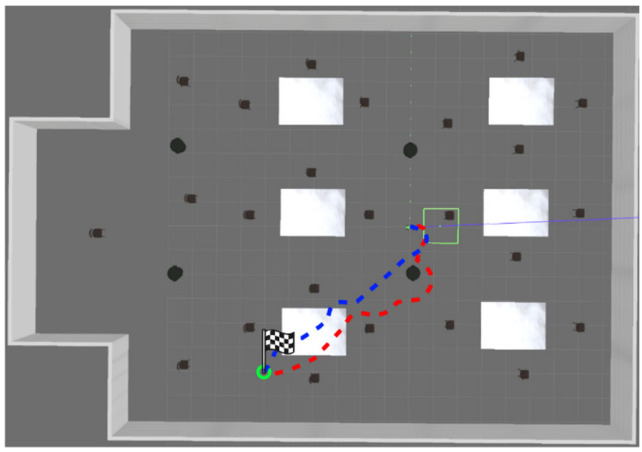

Figura 16. Trayectoria generada para el objetivo en x=-6 y=-6 (rojo) contra la trayectoria ideal (azul). Comparar con figura 13.

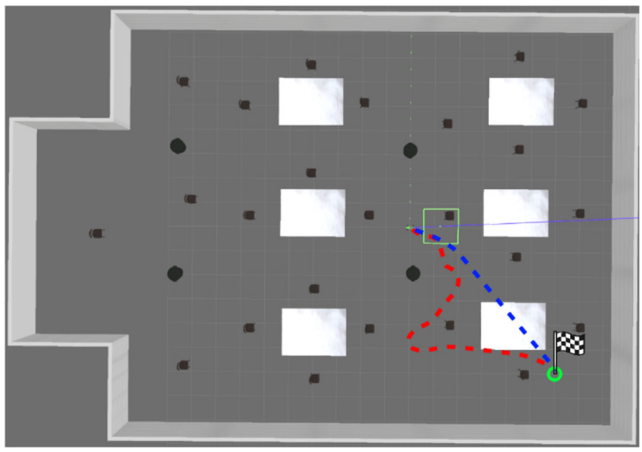

Figura 17. Trayectoria generada para el objetivo en x=6 y=-6 (rojo) contra la trayectoria ideal (azul). Comparar con figura 11.

Por último, se decidió probar el modelo con una tarea muy retadora para el robot, para la cual no se había entrenado directamente, esta es la creación de una trayectoria para el alcance de tres objetivos distintos, el resultado y la trayectoria generada se muestran en la figura 18.

La Mecatrónica en México, Septiembre 2023, Vol. 12, No. 3, páginas 56 – 75
Disponible en línea en www.mecamex.net/revistas/LMEM
ISSN: 2448-7031, Asociación Mexicana de Mecatrónica A.C

Figura 18. Trayectoria generada para tres objetivos distintos (rojo) contra la trayectoria ideal (azul).

8. Conclusiones

En el presente proyecto se ha presentado una metodología específica para el entrenamiento por refuerzo de un robot móvil para la tarea de navegación. Así mismo, se describió una función de recompensas que permite enseñar a un agente a generar trayectorias correctas hacia un objetivo. Con todo esto, se logró obtener un sistema que permitiera a un robot ser capaz de llegar a un objetivo al tiempo que evade los obstáculos del entorno.

Además, se han identificado y analizado las principales áreas de mejora del sistema. Estas incluyen, primordialmente, dos aspectos: la precisión del robot para llegar exactamente al objetivo, limitada por su diseño Ackermann, y la notable variabilidad en la generación de trayectorias. El primer aspecto a menudo resulta en colisiones cuando el robot se encuentra cerca de su destino, mientras que el segundo introduce un nivel de impredecibilidad en su comportamiento, lo que podría limitar su aplicabilidad en ciertas situaciones o entornos.

Como propuesta para abordar dichas áreas de oportunidad, se propone como trabajo futuro lo siguiente: la extensión del espacio de acciones del robot añadiendo una mayor cantidad de velocidades disponibles que permitan al agente una mejor precisión para llegar a un objetivo (por ejemplo, dando reversa). O en su defecto, se propone también el uso de algoritmos de aprendizaje por refuerzo que permitan establecer espacios de acción continuos, de tal manera que el robot supere los limitantes del espacio de acciones discreto y pueda modificar las velocidades de manera más flexible.

Por último, como mejora independiente las áreas anteriores, se propone la extensión de la percepción del robot usando cámaras, de tal forma que se pueda capturar la información de obstáculos que no se encuentren en el mismo plano de percepción del LiDAR.

Referencias

[1] R. Cimurs, J. H. Lee, and I. H. Suh, “Goal-Oriented Obstacle Avoidance with Deep Reinforcement Learning in Continuous Action Space,” Electronics (Basel), vol. 9, no. 3, p. 411, Feb. 2020, doi: 10.3390/electronics9030411.

[2] H. Jiang, H. Wang, W.-Y. Yau, and K.-W. Wan, “A Brief Survey: Deep Reinforcement Learning in Mobile Robot Navigation,” in 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA), IEEE, Nov. 2020, pp. 592–597. doi: 10.1109/ICIEA48937.2020.9248288.

[3] M. Duguleana and G. Mogan, “Neural networks based reinforcement learning for mobile robots obstacle avoidance,” Expert Syst Appl, vol. 62, pp. 104–115, 2016, doi: https://doi.org/10.1016/j.eswa.2016.06.021.
[4] T. Eppenberger, G. Cesari, M. Dymczyk, R. Siegwart, and R. Dubé, “Leveraging Stereo-Camera Data for Real-Time Dynamic Obstacle Detection and Tracking,” Jul. 2020.
[5] C. Zhou, F. Li, W. Cao, C. Wang, and Y. Wu, “Design and implementation of a novel obstacle avoidance scheme based on combination of CNN-based deep learning method and liDAR-based image processing approach,” Journal of Intelligent & Fuzzy Systems, vol. 35, no. 2, pp. 1695–1705, Aug. 2018, doi: 10.3233/JIFS-169706.
[6] S. Sarkar, S. N. Shome, and S. Nandy, “An Intelligent Algorithm for the Path Planning of Autonomous Mobile Robot for Dynamic Environment,” 2010, pp. 202–209. doi: 10.1007/978-3-642-15810-0_26.
[7] H. Song, K. Lee, and D. H. Kim, “Obstacle Avoidance System with LiDAR Sensor Based Fuzzy Control for an Autonomous Unmanned Ship,” in 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS), IEEE, Dec. 2018, pp. 718–722. doi: 10.1109/SCIS-ISIS.2018.00119.
[8] J. Cerbaro, D. Martinelli, A. S. de Oliveira, and J. A. Fabro, “WaiterBot: Comparison of Fuzzy Logic Approaches for Obstacle Avoidance in Dynamic Unmapped Environments Using a Laser Scanning System (LiDAR),” in 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), IEEE, Nov. 2020, pp. 1–6. doi: 10.1109/LARS/SBR/WRE51543.2020.9307098.
[9] L. Xie, S. Wang, A. Markham, and N. Trigoni, “Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning,” Jun. 2017.
[10] H. Song, A. Li, T. Wang, and M. Wang, “Multimodal Deep Reinforcement Learning with Auxiliary Task for Obstacle Avoidance of Indoor Mobile Robot,” Sensors, vol. 21, no. 4, p. 1363, Feb. 2021, doi: 10.3390/s21041363.
[11] H. Surmann, C. Jestel, R. Marchel, F. Musberg, H. Elhadj, and M. Ardani, “Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments,” arXiv preprint arXiv:2005.13857, 2020, doi: https://doi.org/10.48550/arXiv.2005.13857.
[12] J. Torres, Introducción al aprendizaje por refuerzo profundo: teoría y práctica en Python. Kindle Direct Publishing, 2021.
[13] SLAMTEC, “RPlidar A1,” https://www.slamtec.ai/home/rplidar_a1/.

Biografía de Autores

Alfaro López Roberto de Jesús. Realizó sus estudios de Ingeniería Mecatrónica en la Universidad Politécnica del Centro. Actualmente está culminando su Maestría en Ciencias en Inteligencia Artificial en la Universidad Autónoma de Querétaro. Apasionado por el área de la IA dedica gran parte de su tiempo a su implementación en la industria especialmente en las ramas del Reinforcement Learning y Deep Learning.

Juan Manuel Ramos Arreguín. Es Ingeniero en Comunicaciones y Electrónica por parte de la Facultad de Ingeniería Mecánica, Eléctrica y Electrónica (FIMEE) de la Universidad de Guanajuato. En la misma institución obtuvo el Grado de Maestría en Ingeniería Eléctrica. Cuenta con Doctorado en el Programa Interinstitucional de Ciencia y Tecnología (PICyT), en CIDESI, Querétaro. Pertenece al SNI nivel 1, y cuenta con reconocimiento al perfil deseable PRODEP. Actualmente es profesor de tiempo completo en la Facultad de Ingeniería de la Universidad Autónoma de Querétaro.

Marco Antonio Aceves Fernández. Obtuvo su licenciatura en ingeniería telemática en la Universidad de Colima, México. Obtuvo su grado de maestría y doctorado en la Universidad de Liverpool, Inglaterra, en el campo de sistemas inteligentes, éste último en el 2005. Es profesor de tiempo completo en la facultad de ingeniería de la Universidad Autónoma de Querétaro, México. Es miembro del sistema nacional de investigadores (SNI) desde 2009, así como miembro de la Academia Mexicana de computación (AMEXCOMP) y la Academia Mexicana de Ciencias (AMC). También es presidente

honorario en la Asociación Mexicana de Sistemas Embebidos (AMESE). Sus intereses de investigación incluyen sistemas inteligentes y embebidos.

Efrén Gorrostieta Hurtado. Es profesor de tiempo completo en la Universidad Autónoma de Querétaro desde 1995 donde imparte cursos en la Facultad de Ingeniería a nivel licenciatura y posgrado. Obtuvo su grado de Doctor en Ciencias opción Mecatrónica y la Maestría en Ciencia y Tecnología en el Centro de Ingeniería y Desarrollo Industrial (CIDESI) en Querétaro, México. Así mismo, obtuvo su licenciatura en Ingeniería Eléctrónica en el Instituto Tecnológico de Estudios Superiores y Occidente. Es miembro del Sistema Nacional de Investigadores nivel 1 y perfil PRODEP. Su área de especialización se encuentra en Sistemas Inteligentes y Robots Caminantes. Ha impartido conferencias a nivel nacional e internacional, publicado varios artículos indizados en el JCR en el área de sistemas embebidos, robótica e inteligencia artificial. Es miembro del Cuerpo Académico de Optimización y Computación Avanzada.

Saúl Tovar Arriaga. Obtuvo su Doctorado en Ciencias Biomédicas por la Universidad de Erlangen-Nuremberg, Alemania, su Maestría en Ciencias Biomédicas por la Universidad de Siegen, Alemania, y es Ingeniero en electrónica por el Instituto Tecnológico de Querétaro. Es profesor de tiempo completo de la de la Facultad de Ingeniería de Universidad Autónoma de Querétaro y actualmente es coordinador de la Maestría en Ciencias en Inteligencia Artificial. Sus intereses de investigación incluyen diagnóstico médico automático e Inteligencia artificial. Es miembro del Sistema Nacional de Investigadores Nivel 1 y perfil PRODEP.

Jesús Carlos Pedraza Ortega. Realizó sus estudios de Maestría en la FIMEE, Universidad de Guanajuato. Obtuvo el Doctorado en Ingeniería Mecánica con especialidad en Robótica - Sistemas de Reconstrucción 3D en la University of Tsukuba en Japón. Ha impartido cursos en los tres niveles de estudios (Licenciatura, Maestría y Doctorado) desde 1997, actualmente en la Universidad Autónoma de Querétaro. Es Senior Member por la IEEE y es miembro de la Academia Mexicana de Ciencias y Academia Mexicana de Computación. Sus líneas de investigación son sistemas de reconstrucción 3D, inteligencia artificial aplicada a sistemas de visión, entre otros.