Capítulo 9.

Sistema de planeación de trayectorias basado en aprendizaje por refuerzo para vehículo 4WS con LiDAR2D

Robinson Salazar Chavarro[1]
Olmer García-Bedoya[2]

Cítese como: Salazar-Chavarro, R. y García-Bedoya, O. (2023). Sistema de planeación de trayectorias basado en aprendizaje por refuerzo para vehículo 4WS con LiDAR2D. En F. C. Gómez-Meneses, E. M. Moncayo-Torres y T. M. Piamba, (comps.), *Aplicaciones tecnológicas de la Ingeniería Mecatrónica y sus impactos al desarrollo socioeconómico* (pp. 127-143). Editorial UNIMAR. https://doi.org/10.31948/editorialunimar.214.c372

Resumen

El objetivo de este trabajo fue analizar un sistema de inteligencia artificial para el entrenamiento de un agente y simulación con diferentes modelos de aprendizaje por refuerzo en el ambiente, para identificar cuáles son las características de aprendizaje por cada uno de los algoritmos con los diferentes métodos. Los agentes son los responsables de enviar acciones al ambiente y, el ambiente entrega un estado con una recompensa, dependiendo del objetivo que quiere alcanzar el agente.

Después de hacer la evaluación de los modelos con una infraestructura definida para el desarrollo del proyecto, se evaluará la eficiencia del modelo, el tiempo de entrenamiento y la optimización de los recursos físicos, con lo cual se espera entregar la evaluación de los modelos en un ambiente simulado y las características de cada uno para identificar cuál es la eficiencia con los recursos de infraestructura y la optimización de las respuestas obtenidas por cada uno de los modelos.

Palabras clave: aprendizaje por refuerzo; robot; aprendizaje profundo.

[1] Estudiante de Maestría en Ingeniería y Analítica de Datos. Correo: robinson.salazarch@gmail.com
[2] Docente de Maestría en Ingeniería y Analítica de Datos. Correo: olmerg@gmail.com

Reinforcement learning-based trajectory planning system for 4WS vehicle with LiDAR2D

Abstract

The objective of this work was to analyze an artificial intelligence system for agent training and simulation with different reinforcement learning models in the environment, to identify which are the learning characteristics for each of the algorithms with the different methods. The agents are responsible for sending actions to the environment and, the environment delivers a state with a reward, depending on the goal that the agent wants to achieve.

After evaluating the models with a defined infrastructure for the development of the project, the efficiency of the model, the training time, and the optimization of the physical resources will be evaluated, with which it is expected to deliver the evaluation of the models in a simulated environment and the characteristics of each one to identify the efficiency with the infrastructure resources and the optimization of the responses obtained by each one of the models.

Keywords: reinforcement learning; holonomic robot; deep learning.

Sistema de planejamento de trajetória baseado em aprendizagem por reforço para veículo 4WS com LiDAR2D

Resumo

O objetivo deste trabalho foi analisar um sistema de inteligência artificial para treinamento e simulação de agentes com diferentes modelos de aprendizagem por reforço no ambiente, para identificar quais são as características de aprendizagem de cada um dos algoritmos com os diferentes métodos. Os agentes são responsáveis por enviar ações para o ambiente e o ambiente fornece um estado com uma recompensa, dependendo da meta que o agente deseja alcançar.

Após a avaliação dos modelos com uma infraestrutura definida para o desenvolvimento do projeto, serão avaliados a eficiência do modelo, o tempo de treinamento e a otimização dos recursos físicos, com os quais se espera entregar a avaliação dos modelos em um ambiente simulado e as características de cada um para identificar a eficiência com os recursos de infraestrutura e a otimização das respostas obtidas por cada um dos modelos.

Palavras-chave: aprendizagem por reforço; robô; aprendizado profundo.

Introducción

El presente proyecto tiene como referencia, el programa de un vehículo eléctrico para estudiar la movilidad dentro de edificios y vías de bajo tráfico vehicular. Surge del trabajo colaborativo entre docentes y estudiantes de la Facultad de Artes y Diseño y la Facultad de Ciencias Naturales e Ingenierías de la Universidad Jorge Tadeo Lozano de Bogotá. Se centra en el desarrollo de vehículos eléctricos, tanto tripulados como autónomos, que respondan a necesidades concretas según el entorno, permitiendo la integralidad de las disciplinas que ofrece la universidad al campo automotriz.

En la primera fase del proyecto se entregó el diseño del prototipo y sus primeros pilotos, sobre la estructura real y la construcción del prototipo de movilidad en un contexto local. El ambiente permitirá ejecutar pruebas con los modelos seleccionados para evaluar la eficiencia del aprendizaje y efectividad del modelo ejecutado.

Este proyecto tiene como finalidad, evaluar diferentes modelos de aprendizaje por refuerzo para la comparación de la eficiencia en un ambiente simulado y futuras implementaciones en vehículos autónomos. El aprendizaje por refuerzo es una herramienta que permite entrenar agentes para realizar tareas de forma repetitiva y, después de una serie de iteraciones, tener un camino óptimo para el desempeño de la actividad. En la actualidad, el desarrollo de las tecnologías y la capacidad de cómputo a la que se puede acceder permite ejecutar simuladores para evaluar los diferentes algoritmos de *machine learning* (aprendizaje automático) con el propósito de realizar la implementación o pruebas de efectividad de los algoritmos.

Se toma como referencia el proyecto de Rana et al. (2021), '*Bayesian controller fusion*' (BCF), el cual ya es entregado por una serie de librerías, para facilitar la integración con los ambientes de simulación.

Desarrollo

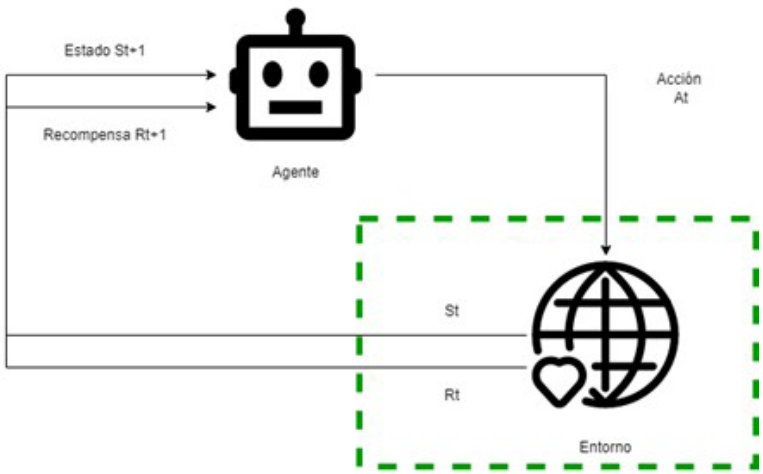
Aprendizaje por refuerzo

Aprendizaje por Refuerzo (RL proveniente del inglés *Reinforcement Learning*) es un enfoque de la ciencia del aprendizaje automático (*Machine learning*). Es un modelo de aprendizaje conductual por el cual el algoritmo aprende a través de la prueba y error; a raíz de esto se le otorga una retroalimentación sobre sus acciones, lo que permite identificar si el agente está tomando buenas decisiones o no. Este tipo de algoritmos busca simular la naturaleza del aprendizaje y cómo se pone en práctica en seres cognitivos. Uno de los elementos a tener en cuenta es el entorno en el que se realiza el entrenamiento

del modelo. En la documentación del aprendizaje por refuerzo se habla de un componente en común identificado como el agente; este es el individuo que interactúa con el ambiente para obtener información y generar conocimiento. El conocimiento del agente progresa, teniendo en cuenta los episodios que se ejecute; según el modelo, después de determinado número de episodios no mejorará el resultado del agente en el ambiente. La Figura 1 presenta la interacción del agente con el ambiente.

Figura 1

Entrada al entorno y retroalimentación al agente con la información obtenida



El agente realiza una acción (A) al entorno y el entorno tiene dos salidas: un estado (O) y una recompensa (R), las cuales son la base para la nueva entrada del agente. Algunos modelos de entrenamiento agregan un ruido a la salida para mejorar la capacidad de aprendizaje del agente. Se debe tener en cuenta los siguientes elementos en RL:

- **Políticas**: son utilizadas para definir un comportamiento del agente sin considerar el estado. Pueden ser implementadas con diferentes métodos.
- **Señal de recompensa**: valor entregado por el entorno después de realizar una acción; define la meta del agente; el objetivo es que el agente obtenga la mayor recompensa.
- **Función de valor:** determina qué tan buena fue la acción y/o estado, considerando el cálculo con varios parámetros.
- **Modelo del entorno**: permite predecir estados y recompensas futuras.

Actor crítico

El entrenamiento de la red neuronal se trabajó con el método del actor - crítico, donde se puede simular una serie de estados y recompensas; lo que hace este método es calcular cuál es el mejor resultado posible, a partir de las recompensas obtenidas; esto permite que las redes neuronales disminuyan el tiempo de entrenamiento y obtengan mejores resultados. La fórmula para calcular este método es:

$$(2.5)\ \Delta\theta = \alpha * \Delta\theta\ (\log \pi\ (St, At, \theta)) * Q\ (St, A) \textbf{(2.5)}$$

Una forma sencilla de explicar cómo funciona el método es imaginar un videojuego donde tú eres el que está jugando y el que maneja el control con otra persona; al lado se indica dónde debes tener cuidado y cuáles son los errores que has cometido. De esta forma se puede asumir que los dos están jugando; por lo tanto, los dos están aprendiendo. Las redes neuronales tienen una política de la red crítica; además, se debe medir las acciones tomadas; para eso se utiliza las siguientes ecuaciones:

Políticas

Se representa con la función:

$$\textbf{(2.6)}\ \pi\ (s, a, \theta)\ (2.6)$$

Medición de acciones

Se representa con la función:

$$\textbf{(2.7)}\ q\hat{}\ (s, a, w)\ (2.7)$$

Al inicio, el actor toma acciones aleatorias; el crítico evalúa la acción y proporciona una retroalimentación; el actor actualiza sus políticas y, de igual manera lo hace el crítico; de esta forma, las dos neuronas mejoran su aprendizaje con base en la experiencia de cada episodio. El pseudocódigo se representa en la Figura 2.

Figura 2

Pseudocódigo Actor Crítico

Actor–Critic with Eligibility Traces (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Initialize S (first state of episode)
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
  $\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
  $I \leftarrow 1$
  Loop while S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + I\nabla_{\mathbf{w}}\hat{v}(S,\mathbf{w})$
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla_{\boldsymbol{\theta}} \ln \pi(A|S,\boldsymbol{\theta})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

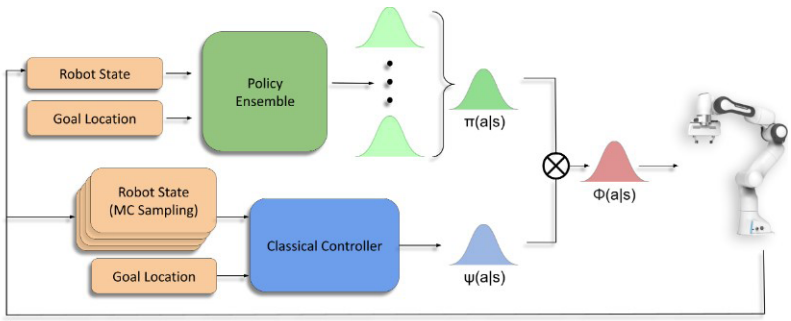
Fuente: García-Pascual (2021).

Estado del Arte

Con el fin de establecer una guía de conocimientos aplicados sobre Aprendizaje por Refuerzo y Ambientes para simulación, se puede apreciar las siguientes investigaciones, cuya base de códigos fuentes utilizados y desarrollo de este proyecto fue el trabajo de Rana et al. (2021), que se enfocó en el entrenamiento con la combinación de diferentes modelos para desarrollar una serie de librerías y códigos fuentes para el manejo de diferentes ambientes 2d y 3d. La Figura 3 presenta la arquitectura que se utilizó en el proyecto BCF. La finalidad de este proyecto fue el desarrollo de un control híbrido para las acciones que realiza un brazo en los ambientes de simulación.

Figura 3

Arquitectura Proyecto BCF



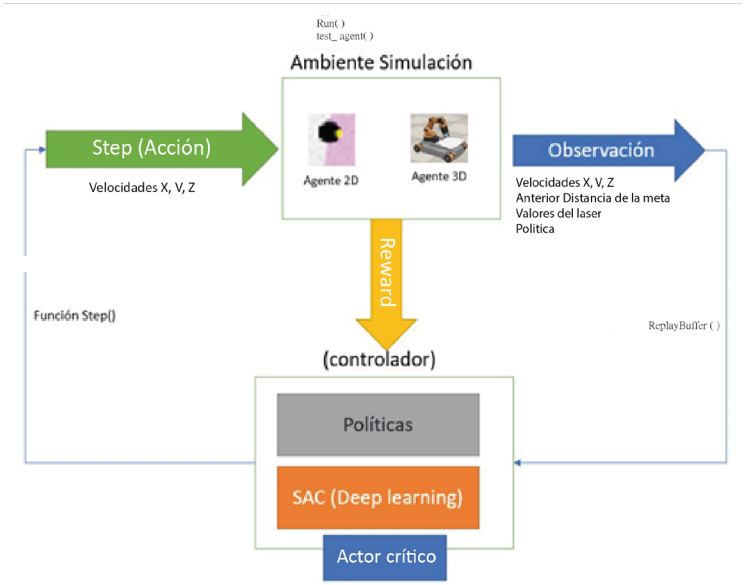
Fuente: Rana et al. (2021).

Arquitectura de la solución

El proyecto del vehículo eléctrico plantea realizar un vehículo omnidireccional capaz de realizar y recibir comandos de velocidad en los ejes lineales **X**, **Y** y **Y** en el eje angular z a través de un mecanismo de *four-wheel-steering*. La dificultad encontrada con los modelos de la bibliografía de referencia consistió en que el simulador bidimensional solo permitía velocidad lineal en x y velocidad angular en z, debido a que esta tenía una configuración diferencial en el vehículo. Esto planteó como reto, probar el proceso de aprendizaje por refuerzo profundo con la evaluación de las acciones de un robot omnidireccional. A continuación, se describe la arquitectura de la solución desarrollada para generar el ambiente de simulación y, de esta forma, poder realizar los entrenamientos y validación de la red neuronal.

Figura 4

Arquitectura Proyecto



La Figura 4 presenta la arquitectura trabajada; se relaciona las entradas de cada componente y las salidas respectivas. En esta arquitectura se aprecia cómo cada ambiente de simulación fue testado con un simulador bidimensional realizado en la librería Box2D y con un simulador en 3D llamado *CoppeliaSim* conectado a Python por la librería *pyrep*. Para realizar este cambio fácilmente, la arquitectura de la solución consiste en una clase llamada *Environment*, que implementa el entorno de simulación, una clase *agent* que efectúa el controlador entrenado por aprendizaje, por refuerzo y, un *Main* donde se realiza un ciclo de simulación.

En este ciclo, el agente determina una acción a través de la política π basada en la observación; luego, el *environment* realiza un *step* a partir de la acción del controlador y este retorna la observación y la función de *rewards*. Este proceso es repetido en una cantidad de pasos durante los cuales se guarda toda la información para efectuar el proceso de entrenamiento y luego reiniciar el *environment* y volver a iniciar el ciclo hasta que la función de *rewards* sea exitosa.

Es de resaltar en esta arquitectura que, el agente entrenado con aprendizaje por refuerzo consiste en una red neuronal implementada en *pytorch* entrenado con la GPU (*Graphics Processing Unit:* Unidad de procesamiento gráfico) del computador.

Infraestructura

En el desarrollo de este proyecto se cuenta con los siguientes componentes físicos para el desarrollo de la simulación: Computador portátil MSI con un procesador Intel I5 de octava generación a 1.8GHz; 16 GB de memoria Ram; tarjeta graficadora Nvidia Geforce MX150 con 2gb dedicada; disco duro estado sólido 256 GB y un disco duro externo estado sólido de 512. El computador cuenta con un sistema operativo Windows 10 Pro para todo el procesamiento de texto e investigación. La ejecución del modelo se realizó en un sistema Ubunto 20.04 en el disco externo para la ejecución de todo el proceso de entrenamiento.

Simulación

Escenario de Simulación

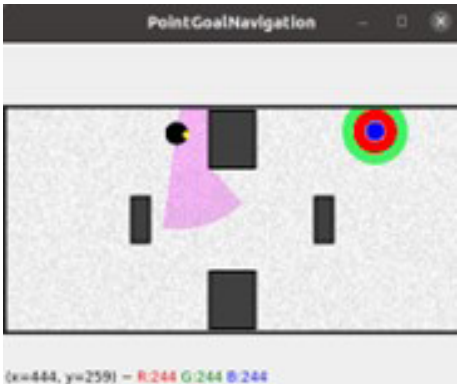
Se debe considerar que una de las finalidades de la robótica es obtener el mayor rendimiento en el menor tiempo posible y con el menor consumo de recursos; el entrenamiento del modelo garantizará la navegación segura y la obtención de los objetivos del robot. La finalidad de la simulación es validar las capacidades de cómputo requeridas por un robot para el desarrollo de las acciones en tiempo real y en diferentes ambientes: el Escenario 2d con modelo cinemático del vehículo y el Escenario 3D con el *youbot* permitieron el entrenamiento de una red neuronal que dejó llegar al objetivo en el menor tiempo, evitando los diferentes obstáculos que había en cada uno de los escenarios. Se aclara que, para los dos ambientes de simulación se tuvo las mismas iteraciones y dimensiones, con la finalidad de comparar el rendimiento del modelo con los mismos recursos.

Escenario de Simulación 2D

Atendiendo el proyecto BCF y las librerías PyRep se realizó el perimétrico de ambiente Box2d y Python; adicional, se configuró 'Anaconda', con el fin de centralizar la interfaz gráfica y poder realizar la consolidación de los códigos fuentes. La Figura 5 presenta el ambiente donde se muestra el robot buscando llegar a la meta con los diferentes obstáculos.

Figura 5

Robot desde el punto inicial. Información proyecto



El robot cuenta con una LiDAR que tiene un rango de 180 grados para identificar los diferentes obstáculos y generar retroalimentación a la red neuronal con la información, generando una nueva acción en el ambiente. Después de un periodo de tiempo el robot aprendió a llegar a la meta, como se evidencia en la Figura 6.

Figura 6

Robot llegando a la meta dinámica. Información proyecto

Al analizar la Figura 6 se puede demostrar una evolución del robot, teniendo en cuenta que el ambiente se genera de una forma aleatoria en el punto de inicio y la ubicación de la meta para validar el aprendizaje del robot.

Resultados

Escenario de Simulación 2D

Después de realizar toda la configuración del ambiente y las diferentes librerías, se realizó el proceso de entrenamiento, dividido en diferentes etapas para el entendimiento del funcionamiento del ambiente y la red neuronal.

Se hizo varias corridas de simulación, las cuales pueden ser clasificadas según el periodo de duración en: 12, 24, 48, 72 horas. Como resultado, se almacenó el entrenamiento de cada una de las corridas de forma acumulativa e individual para validar cuál sería el mejor periodo de tiempo y cuándo la red neuronal tendría un entrenamiento adecuado para lograr el objetivo en el menor tiempo y sin tener impactos en los obstáculos. Las Figuras 7 y 8 fueron generadas después de la simulación, para validar el comportamiento del modelo con el entrenamiento.

Figura 7

Gráfica red sin entrenamiento después de simulación

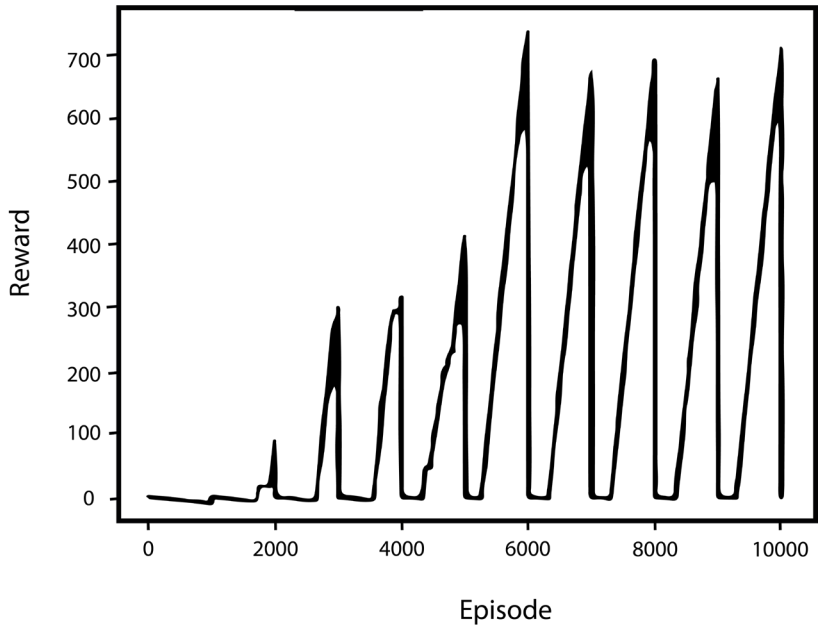
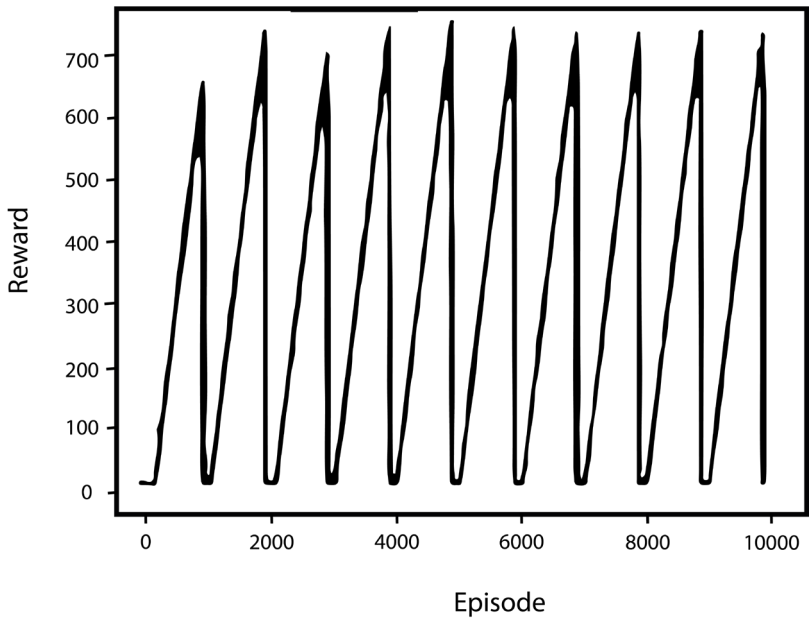


Figura 8

Gráfica red con entrenamiento después de simulación



Analizando las Figuras 7 y 8 se puede evidenciar el tiempo que le lleva al robot obtener una recompensa y, la irregularidad para lograr llegar a la meta; sin embargo, la Figura 8 evidencia un robot que puede llegar de una forma rápida a la meta y, además, de forma constante, indicando la evolución de la red neuronal para el entendimiento del ambiente simulado. Esto se consigue después de realizar más de un millón de iteraciones en los periodos de tiempo mencionados en el párrafo anterior.

Observando la información de la simulación 2D se puede concluir que, al realizar el entrenamiento de una red neuronal, esta puede generar la navegación, considerando los obstáculos y encontrando su objetivo. Se debe poner atención en que, el mayor desafío está en la configuración de ambiente y capacidad de cómputo.

El entrenamiento requiere tener claridad sobre cuáles serían los valores que se entregará a la red neuronal y cuál será la salida esperada. Dependiendo de estos valores, el proceso de entrenamiento puede ser un éxito o un fracaso. Se identificó que el ambiente 2D de simulación no tenía en cuenta las dimensiones de robot; por lo tanto, al variar la dimensión, no podía llegar a la meta.

El tiempo de entrenamiento se relaciona de forma directa con el número de iteraciones; no obstante, después de que el modelo está entrenado, se requiere un valor constante de iteraciones menor para lograr el objetivo,

pero, al variar las dimensiones del ambiente, la simulación varía el número de iteraciones requeridas. El ambiente tiene en cuenta los valores de la velocidad en X, Y, Z para el entrenamiento.

La recompensa obtenida es una variable que atiende la red neuronal para entender qué sucede en el ambiente de simulación. Se debe observar que, para este ambiente solo se tiene tres valores válidos posibles, teniendo en cuenta la observación. Los valores que se trabajó indicaban éxito, falla o, fin del ciclo de iteraciones.

Escenario de Simulación 3D

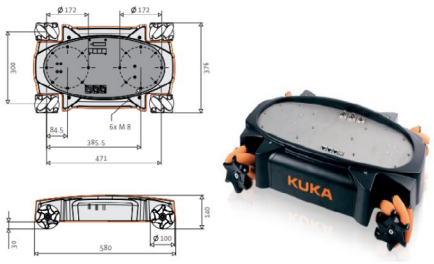
En este modelo se propone un ambiente de simulación 3D con *CoppeliaSim*; el ambiente permite tener características diferentes y agregar tipos de robots más complejos, teniendo en cuenta las limitantes del ambiente utilizado en la simulación 2D.

El ambiente proveyó elementos como una Láser Scanner 2d y el movimiento del robot en ángulos, que no se podía manejar en el ambiente anterior; también permitió generar diferentes tipos de obstáculos y cambiar las dimensiones del área donde se efectuó la simulación. Lo que se esperaba con este ambiente era mejorar la capacidad de la red neuronal al momento de realizar las acciones en el ambiente simulado.

Es un tipo de robot que se desarrolló para realizar diferentes acciones; tiene diferentes componentes, incluyendo un brazo que le permite tomar y mover objetos desde un punto a otro. Para nuestra simulación, el brazo no fue tenido en cuenta; solamente la plataforma que permite movimiento en las cuatro ruedas y que tiene un componente adicional que permite un movimiento en 45 grados, según la necesidad. La Figura 9 presenta la estructura de la base del robot.

Figura 9

Imagen de la base móvil



Esta es la estructura que se desplazó en el ambiente simulado; el robot en el ambiente *CoppeliaSim* tuvo el brazo, pero este no proporcionó datos para la red neuronal. Cabe aclarar que esta parte no es alcance de este proyecto.

Láser Scanner 2d es un tipo de sensor habilitado en *CoppeliaSim* que permite capturar la información del ambiente en 180 grados y en tiempo real. De esta forma fueron alimentadas las observaciones entregadas a la red neuronal. Facilitó la configuración de la longitud y ángulo de captura de datos; es decir, se pudo limitar la distancia del sensor y los grados. Para simulación, la longitud fue definida en 19 metros y no se tuvo en cuenta los obstáculos que estuvieran a 0.05 metros.

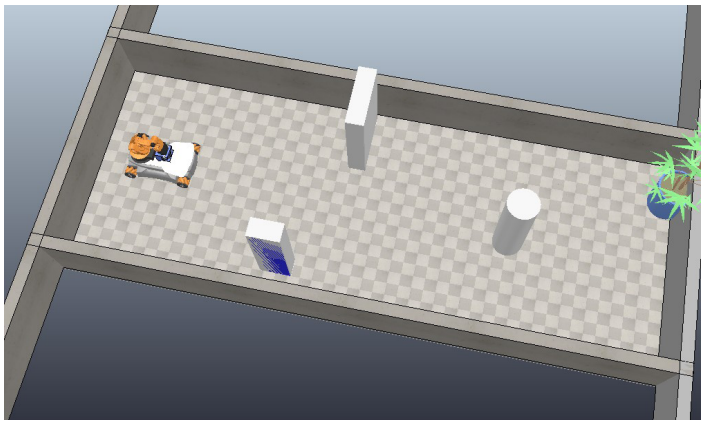
Ambiente 3d CoppeliaSim

Fueron generados diferentes ambientes de simulación en *CoppeliaSim* con la finalidad de evaluar el entrenamiento de la red neuronal en diversas condiciones. Se debe considerar que el entrenamiento fue progresivo; el ambiente con menor espacio generaba menor complejidad, entendiendo como espacio, el área que debía cubrir el robot para llegar a su meta y complejidad y el número de obstáculos que tenía cada ambiente. Para los entrenamientos iniciales se mantuvo constante la longitud y radio de cobertura del Láser Scanner 2d.

Para todos los ambientes, la meta estuvo representada por una matera; esta meta, dependiendo del ambiente, podía tener una posición dinámica o estática para igualar las acciones del ambiente 2D. La Figura 10 presenta el ambiente 3D de simulación.

Figura 10

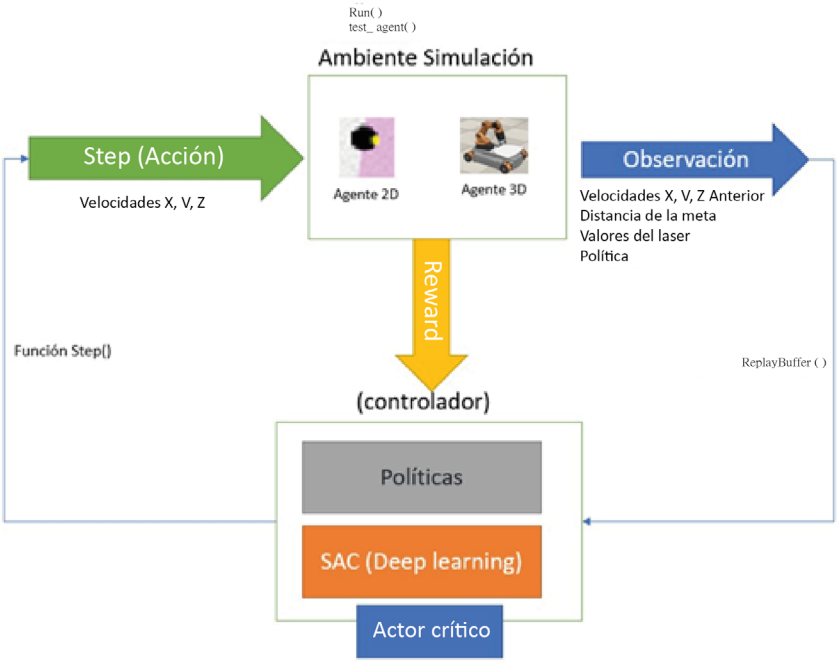
Ambiente 1 CoppeliaSim



Para el modelo de simulación 2d, los valores de recompensa fueron 1,-1 y cero; los valores no afectaron el desempeño del simulador a nivel de velocidad, aunque, los valores en el simulador 3d debieron ser ajustados debido al robot utilizado. La Figura 11 presenta las diferentes funciones que se debe tener en cuenta en el proceso de entrenamiento.

Figura 11

Arquitectura Funciones

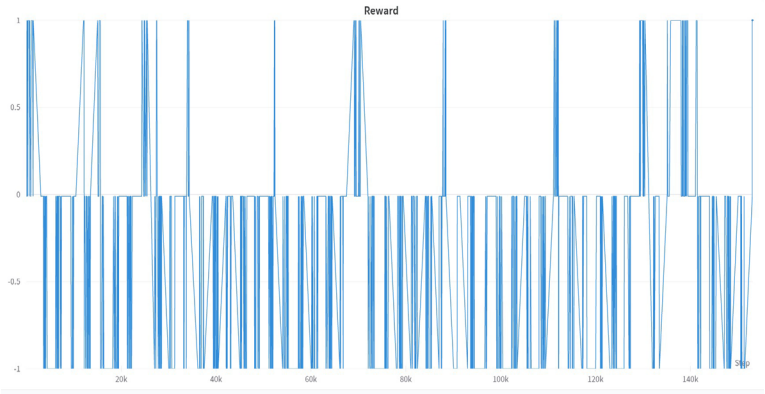


Resultados Escenario de Simulación 3D

Se realizó la configuración de *python* con el nuevo simulador; se debió ajustar el código fuente atendiendo las variables utilizadas en la simulación 2d y, adicionar las variables del ambiente 3d incluyendo la lectura del láser escáner 2d. Igualmente, la nueva forma de movimiento que tenía el robot; a partir de esto se entendió el nuevo funcionamiento de *CoppeliaSim*. Se trabajó con la misma red neuronal. Se hizo varias corridas de simulación; los periodos de tiempos usados en el ambiente 3D fueron: 12, 24, 48, 72 horas. La Figura 12 permite apreciar los resultados de la simulación.

Figura 12

Imagen Red en Entrenamiento

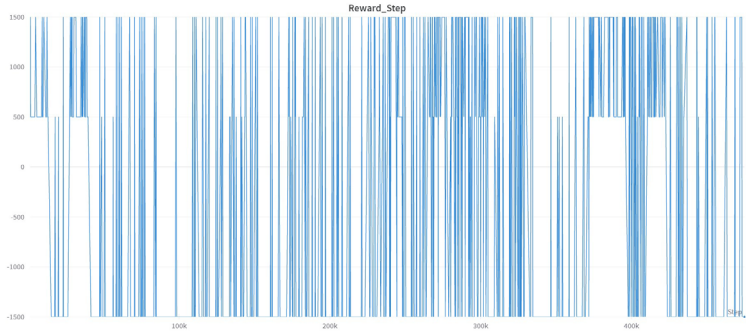


Este primer acercamiento del modelo simulado fue exitoso, teniendo en cuenta que el robot lograba el objetivo, pero, no se tenía un movimiento fluido del mismo en el ambiente. Para mejorar el comportamiento del robot se debió afinar los valores de recompensa; de esta forma se pudo solucionar el problema de movimiento y, el entrenamiento de la red neuronal fue efectivo.

La Figura 13 evidencia el comportamiento de las recompensas obtenidas por el robot al iniciar el entrenamiento y después de ajustar los rangos en las recompensas para obtener un comportamiento mejorado en el simulador 3d.

Figura 13

Reward ajustado para el simulador



Análisis Simulación 3D

Los principales retos en el ambiente 3D de simulación fueron: la comprensión de la relación recompensa vs. velocidad del robot, la cual se afectaba de

forma directa. A menor recompensa, menor velocidad reflejada en el robot y, a mayor recompensa, mejor desempeño. Adicionalmente, al manejo de las nuevas propiedades del ambiente con la identificación de los objetos con el Láser Scanner 2d y propiedades como coalición, visible entre otras con estas nuevas propiedades, se debió hacer el análisis de los datos obtenidos con el Láser Scanner 2d y, el manejo de otras propiedades para identificar colisión con un objeto y distancia de la meta.

En la segunda simulación, fue responsabilidad de la red neuronal entregar el valor para la velocidad en X, Y y Tangencial requeridas para el movimiento del robot.

Conclusiones

Este documento permite identificar las características que se debe considerar al momento de efectuar el entrenamiento de una red neuronal DDPG (*Deep Deterministic Policy Gradient*) con un actor crítico que permita aprender a navegar por diferentes escenarios y llegar a un objetivo. Se puede inferir que no cualquier tipo de red neuronal se puede utilizar para resolver un problema de navegación; en nuestro caso se utilizó DDPG, teniendo en cuenta las entradas y salidas requeridas para las acciones en el ambiente.

Al iniciar este proceso se pensó que el mayor factor que se debía controlar era la velocidad; después de realizar todo el proceso de simulación, se puede colegir que, para el ambiente 2D de simulación esto es cierto; sin embargo, para el ambiente 3D de simulación, teniendo en cuenta el tipo de robot, esto no aplica, pues la red neuronal aprendió no solo a evitar los obstáculos, sino a controlar la velocidad del robot en todas sus direcciones.

Los diferentes ambientes permitieron, después del entrenamiento, validar si el cambiar la posición del objetivo en el mismo escenario afectaría el aprendizaje. Como resultado se puede sostener que, después de que la red neuronal esté entrenada, esto no afecta el rendimiento. Aumentar las dimensiones y obstáculos de ambiente simulado tampoco generó un cambio significativo en los tiempos de respuesta y obtención del objetivo después del entrenamiento.

Al inicio de este proyecto se pensó que, si el láser tenía una mayor longitud podría llegar a entrenar de una forma más efectiva la red neuronal, pero, se hizo pruebas con diferentes longitudes del láser y se pudo evidenciar que el factor que afectaba el entrenamiento era la distancia que determinaba si el robot estaba en colisión con otro objeto del ambiente, y no la distancia al objeto.

Si bien durante este proyecto se identificó que hay diferentes formas de calcular variables, la recomendación es utilizar las funciones que estén definidas dentro de las librerías utilizadas, para garantizar que los cálculos sean realizados de acuerdo con el ambiente de simulación.

Respecto al entrenamiento, se pudo evidenciar que es progresivo; con mayor cantidad de iteraciones es más efectivo el proceso de navegación, aunque, teniendo en cuenta los resultados obtenidos en este proyecto, el simulador utilizado puede influir en el tiempo de entrenamiento; es decir, a mayor cantidad de variables que se pueda obtener de la simulación, mayor será el tiempo que tarde el entrenamiento. En la simulación 2d se puede evidenciar que después de un millón de iteraciones, el entrenamiento se mantiene; no hay una mejora significativa.

Antes de realizar las simulaciones en los diferentes ambientes trabajados, es posible llegar a pensar que el manejo de las librerías para el desarrollo puede ser transversal en cualquier simulador; sin embargo, al hacer la implementación, se hizo el ajuste del código para poder manejar el simulador 3d.

Los tipos de robots manejados en las simulaciones y sensores de captura de datos del ambiente tienen un gran impacto al momento de efectuar la captura de información y acciones en el ambiente. En el ambiente 2D el robot tenía movimientos limitados a velocidad en X, Y, Z, mientras que el robot en el ambiente 3D permitía tener movimientos con relación a las cuatro llantas de forma independiente y ángulos distintos, además de tener una relación entre las mismas llantas, que se vieron afectadas por la recompensa obtenida en cada acción; este factor no se identificó en el ambiente 2D de simulación.

Se pudo identificar que cada episodio puede tener un número de *steps* determinado al iniciar el entrenamiento; no obstante, este número cambiará después de que la red neuronal esté entrenada, pues se optimizará el tiempo en el que el robot encuentre el objetivo. En este proyecto se realizó la variación de esta variable en diferentes simulaciones, para validar la capacidad de aprendizaje del robot.

Referencias

García-Pascual, M. (2021). *Aprendizaje por refuerzo profundo con Open AI Gym* [Tesis de Pregrado, Universidad Autónoma de Madrid]. https://repositorio.uam.es/bitstream/handle/10486/698285/garcia_pascual_mario_tfg.pdf?sequence=1

Rana, K., Dasagi, V., Haviland, J., Talbot, B., Milford, M., & Sünderhauf, N. (2021). Bayesian controller fusion: Leveraging control priors in deep reinforcement learning for robotics. https://doi.org/10.48550/arXiv.2107.09822