CONTROL DE DESPLAZAMIENTO CON EVASIÓN DE OBSTÁCULOS PARA UN ROBOT MÓVIL EN CONFIGURACIÓN DIFERENCIAL, USANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL

CRISTIAN MAURICIO NAVARRO LEÓN

Directores:
I.E. LEONARDO ENRIQUE SOLAQUE GUZMAN, Ph.D.
I.E. ALEXANDRA ELIZABETH VELASCO VIVAS, Ph.D.

UNIVERSIDAD MILITAR NUEVA GRANADA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA MECATRÓNICA
BOGOTÁ D.C., SEPTIEMBRE DE 2020

CONTROL DE DESPLAZAMIENTO CON EVASIÓN DE OBSTÁCULOS PARA UN ROBOT MÓVIL EN CONFIGURACIÓN DIFERENCIAL, USANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL

CRISTIAN MAURICIO NAVARRO LEÓN

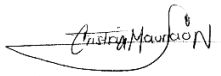
Directores:
I.E. LEONARDO ENRIQUE SOLAQUE GUZMAN, Ph.D.
I.E. ALEXANDRA ELIZABETH VELASCO VIVAS, Ph.D.

UNIVERSIDAD MILITAR NUEVA GRANADA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA MECATRÓNICA
BOGOTÁ D.C., SEPTIEMBRE DE 2020

Bogotá D. C., octubre de 2020

DEDICATORIA

A mi madre, por ser mi fuente eterna de amor y fortaleza
A mi padre, por enseñarme el camino, ser luz y guía desde el cielo
A mi padrino, por siempre transmitirme su sabiduría
A mi madrina, por su apoyo incondicional



Cristian

AGRADECIMIENTOS

Agradezco especialmente a todos los docentes del programa de ingeniería mecatrónica de la universidad, quienes realizan un excelente trabajo en la formación de grandes personas y fueron fundamentales en mi formación como ingeniero. Al ingeniero Danilo Moreno por su ayuda en la fase final de simulación sobre ROS-Gazebo la cual fue imprescindible para llevar a término la implementación requerida. A mis tutores el Dr. Leonardo Solaque y la Dra. Alexandra Velasco quienes siempre estuvieron dispuestos a guiar el camino para el desarrollo de este trabajo y desempeñaron una excelente labor.

CONTENIDO

Lista de figuras

Lista de tablas

RESUMEN

En este trabajo, se realizó el diseño y simulación de un controlador de desplazamiento mediante aprendizaje por refuerzo, que permite a una plataforma en configuración diferencial la navegación en un entorno desconocido.

Se presenta una aproximación gradual a la plataforma diferencial Ceres-Agrobot a partir de la navegación en una cuadrícula y el control de nivel en un modelo dinámico para luego realizar la implementación de dos técnicas de aprendizaje por refuerzo sobre el modelo dinámico de la plataforma. Se implementa una simulación haciendo uso de ROS y Gazebo para validar el comportamiento de las señales de control generadas por el controlador, dichas señales son aplicadas sobre el modelo dinámico de la plataforma diferencial Pioneer 3DX en Gazebo.

Los resultados evidencian la capacidad del aprendizaje por refuerzo de parametrizar sobre la marcha, el proceso de toma de decisiones en una plataforma diferencial en función del objetivo a cumplir de la navegación.

INTRODUCCIÓN

La evasión y prevención de colisiones en plataformas móviles se ha convertido en un área de interés en la robótica y viene siendo tema de estudio [1]. En cuanto a navegación de plataformas robóticas móviles en escenarios urbanos dinámicos, se han logrado realizar pruebas exitosas en vías públicas alrededor del mundo aplicando conducción autónoma [2]. Tal nivel de ejecución y afianzamiento ha permitido direccionar el interés a la conducción autónoma en tareas dedicadas a la agronomía [3] y explotar las prestaciones que una plataforma robótica brinda.

La necesidad de agilizar y reducir tiempos en las tareas asociadas a la producción de alimentos bajo la premisa de optimizar recursos ha llamado a la robótica a cumplir en términos de eficiencia dentro del agro [4]. Tareas como supervisión de cultivos, erradicación de malezas, riego, fertilización y otras aplicaciones han empezado a implementarse, haciendo necesario pensar en la seguridad de la plataforma durante su desplazamiento. Desarrollos similares exitosos en entornos urbanos [2] han sido facilitados debido a los avances tecnológicos en comunicación y procesamiento de la información [5].

En cuanto a vehículos terrestres dedicados al agro, hay un mayor nivel de complejidad, al tratarse de terrenos que varían sus condiciones (rigidez, humedad) y que además están condicionados por características físicas como textura, estructura, profundidad. A este inconveniente se suma la interacción con agentes estáticos y/o dinámicos como lo son personas, animales, siembra, maquinaria, los cuales deben ser cuidados de algún daño o colisión [5]. Así, se establece Inteligencia Artificial como una alternativa de solución basándose en que puede reflejar el comportamiento humano en contextos de toma de decisiones para la prevención y evasión de daño sobre equipos, personas y cultivo.

De esta manera, se abre un campo de aplicación en donde se puede aportar al usar técnicas que faciliten la labor de navegar en un entorno evitando colisiones y sean alternativas de solución al problema de conducción segura de una plataforma dedicada al agro aplicando Inteligencia Artificial.

1. PROBLEMA

1.1. Descripción

La navegación de plataformas robóticas supone un problema puntual cuando se trata de una plataforma agrícola trabajando en el entorno no estructurado que esto supone, debido a que involucra el uso de rutas que no afecten su movilidad y eviten chocar con obstáculos estáticos y dinámicos durante la operación al interior de un cultivo (personas, maquinaria, cultivo) [5].

Frente a esta problemática, se busca aportar con la presente propuesta, y a la vez, soportar parcialmente los desarrollos que enfrenta el Grupo de Investigación y Desarrollo de Aplicaciones Mecatrónicas (GIDAM) del programa de ingeniería mecatrónica de la Universidad Militar Nueva Granada, en donde se busca dotar de autonomía y desplazamiento seguro a la plataforma robótica para agricultura de precisión Agrobot CERES (Figura 1)

Figura 1. Ceres-AgroBot desarrollado por GIDAM (Diseño CAD y construcción)

Agrobot CERES está compuesta por un sistema no holónomo con tracción diferencial eléctrica (48V) en el frente del robot, y dos ruedas traseras sin tracción. Tiene una capacidad de carga de 100 kg de abono sólido, 20 litros para fumigación y un sistema de remoción de maleza. Posee en su zona frontal, un sistema de visión artificial que integra cámaras del tipo térmica, multiespectral, RGB y estéreo.

Basándose en la revisión de desarrollos en el tema, se ha encontrado que las técnicas Deep Learning son una alternativa para la conducción segura y la evasión de obstáculos. Su capacidad de inferir y tomar decisiones como las que caracterizan al humano [6], es imprescindible al momento de desplazarse dentro de un cultivo con múltiples obstáculos dinámicos y estáticos, previniendo daño de equipos o del cultivo. De esta manera, se logra establecer un acercamiento a la conducción autónoma con evasión de obstáculos integrando el uso de Inteligencia Artificial - Deep Learning a un robot dedicado a labores de agricultura.

1.2. Planteamiento

Lo descrito antes, lleva a plantearnos la siguiente pregunta de investigación: ¿Hasta qué punto se puede implementar una técnica de Inteligencia artificial en la plataforma diferencial PIONEER 3DX, para que, durante la realización de una misión, se desplace de manera segura y evadiendo obstáculos?

2. OBJETIVOS

2.1. Objetivo General

Implementar un control de desplazamiento usando una técnica de inteligencia artificial, que permita a una plataforma en configuración diferencial evadir obstáculos en un entorno de trabajo que permita la simulación realista de su despliegue y comportamiento.

2.2. Objetivos Específicos

1. Evaluar y seleccionar al menos dos técnicas de Deep-Learning usadas para el desplazamiento de plataformas móviles, que se enfoquen en la evasión de obstáculos con base en métricas como error de seguimiento, tiempo de estabilización, etc.

2. Implementar y verificar la técnica que mejor desempeño presente en función de las métricas establecidas, dentro de un entorno que permita validar dichas métricas y su dinámica (Simulink), así como el comportamiento del algoritmo desarrollado.

3. Integrar y validar el funcionamiento de la técnica seleccionada sobre la plataforma en configuración diferencial Pioneer 3DX, usando herramientas que permitan la simulación dinámica realista del algoritmo, como lo permite la integración de ROS y Gazebo.

3. ANTECEDENTES

La autoconducción, comprendida como un tipo de conducción donde el conductor (humano) delega parcial o totalmente el manejo, aparece alrededor de la década de los 80. En un marco internacional, se han realizado aportes enfocados a vehículos no tripulados, inteligentes y autónomos [7], [8]. Los avances desde el contexto investigativo y de desarrollo tecnológico, se vienen centrando en lograr vehículos autónomos, tal como es el caso de la industria automotriz y Google. Estas empresas, han integrado técnicas que permiten autonomía y a su vez, implementando jerarquías [9], [10]. Cabe resaltar que, a la fecha, aún no se ha desarrollado un vehículo totalmente autónomo [11].

En la búsqueda de autonomía de plataformas para realizar misiones, se debe determinar y seguir una trayectoria, que permita llevar el vehículo desde una configuración inicial a una configuración final, involucrando en la toma de estas acciones, decisiones sujetas a reglas de conducción establecidas a priori (como acciones de parar, acelerar o cambiar de carril), intervención en la escena de agentes externos como tráfico y restricciones dinámicas del mismo sistema. En la literatura se encuentra que para los vehículos urbanos se establecen cinco posibles acciones: 1) Seguimiento, 2) Aceleración, 3) Adelantamiento, 4) Salida, y 5) Conducción libre. Además, a estas acciones se incorpora la incertidumbre en las intenciones de otros agentes dinámicos como peatones, vehículos, bicicletas o luces de tráfico [12].

La incertidumbre en el comportamiento de elementos dinámicos es compleja e impredecible, haciendo necesario interpretar y predecir conductas en términos de espacio y tiempo [13]. Una aproximación para integrarlas y plantear soluciones es mediante Inteligencia Artificial, donde técnicas como: aprendizaje de máquina basada en modelos y regresiones Gaussianas, redes neuronales recurrentes o planeamientos probabilísticos, son usadas [13], [14]. Aplicaciones reportadas en la literatura como las de Google y ejecuciones de cambio de carril, muestran integración del comportamiento de vehículos, personas u objetos en movimiento [15].

La Figura 2 muestra el proceso de interpretación, detección y predicción usado para la toma de decisiones, aplicado a cualquier método que se desee emplear sea probabilístico o de inteligencia artificial:

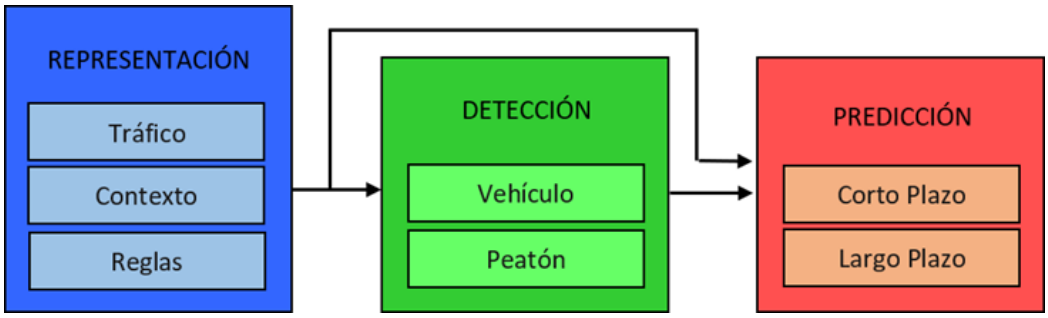


Figura 2. Representación, detección y predicción de eventos.
Adaptada de [13]

En la toma de decisiones (Figura 2), la representación de escena tiene como principal objetivo determinar qué cambios deben ser tomados en cuenta. Tales cambios relacionan aspectos como semáforos, enfoque de visión en manejo, segmentación de líneas de tráfico, direccionamiento dinámico de participantes en escena. Este proceso siempre debe aplicar reglas de manejo que identifiquen el conjunto de alternativas a ejecutar como lo son continuar en la vía, efectuar giros o realizar maniobras como cruzar o cambiar carriles.

Las etapas de representación y predicción brindarán las pistas necesarias para estimar el movimiento, permitiendo que el vehículo realice evasión de obstáculos y estableciendo una trayectoria hasta un determinado objetivo como se pretende realizar en este trabajo. Todo este conjunto de acciones debe estar descrito en función de la variación temporal y de la arquitectura mecánica del vehículo [12].

En el desarrollo de la Inteligencia Artificial, se han alcanzado desarrollos importantes en materia de evasión de obstáculos tanto estáticos como dinámicos. Técnicas basadas en lógica difusa han aplicado reglas de comportamiento a vehículos, y se han complementado con la adición de redes neuronales, algoritmos genéticos y controladores PID clásicos [16], [17], [18], [19]. Desarrollos más avanzados como Deep Learning han sido usados con objetivo de evadir peatones mediante aprendizaje bio-inspirado en ojos humanos y percepción de cercanía, integrando sensores ultrasónicos y una cámara a la entrada de una red neuronal profunda [20]. Tal aplicación ha generado procesos de decisión que generan comandos de control a partir de imágenes [6].

Se han evidenciado aplicaciones en robótica de servicio que resuelven el proceso de decisión de Markov mediante Deep Reinforcement Learning entrenado en varias etapas y múltiples desarrollos tolerantes a fallos donde la sensórica ha sido usada de manera independiente [10]. Tales modelos neurales se han usado para control de movimiento, ajuste de percepción y mejora de desplazamiento sobre la marcha libre de colisiones en un entorno local (cerca a robot) [3], [21].

El aprendizaje reforzado Q ha implementado el uso de funciones recompensa debido a la posición relativa de agentes, obstáculos y objetivos a ser evadidos, evidenciando gran precisión sobre la experimentación [22]. Las funciones de recompensa al ser combinadas con algoritmos basados en el comportamiento de colonias de abejas (Ant-Q) ha logrado buscar caminos óptimos y libres de colisiones en problemas de alto costo computacional que, mediante Q-Learning mejoraron la navegación en ambientes con obstáculos dinámicos generados aleatoriamente [23] , [24].

La aplicación desarrollada en, establece un punto de partida al presentar un contexto similar a lo que se plantea desarrollar en este trabajo, como la identificación de la posición de un agente con relación a los obstáculos de un entorno y su posterior evasión [22]. También el uso de técnicas de aprendizaje por refuerzo esenciales como el Q-Learning para la navegación sobre un entorno del cual no se tenga información a priori, dicha técnica se pretende usar en su forma de representación básica y haciendo uso de la representación por neuronales (DQN) con la idea de generar señales de control no a partir de imágenes como en sino a partir de información recibida del entorno, que serán aplicadas al modelo dinámico que relaciona la arquitectura mecánica del vehículo como se determina realizar en [6], [12], [24].

4. JUSTIFICACIÓN

El aumento en la demanda alimenticia ha requerido un incremento a nivel internacional en la capacidad productiva e inclusión de la optimización en procesos de siembra y cultivo aplicando técnicas de control y automatización sobre las plantaciones, bajo la filosofía de producir más optimizando los recursos disponibles, lo cual enmarca la impacto que puede tener la agricultura de precisión en términos sociales, siendo llamada a sostener el crecimiento de la población mundial [4], [25].

Se ha dado entonces lugar al desarrollo tecnológico en robótica aplicada a la agricultura de precisión, donde se han evidenciado aplicaciones y robots enfocados en el cuidado y supervisión de cosechas, además de vehículos con desplazamiento autónomo, que en términos de rentabilidad y sostenibilidad han logrado un aumento de la productividad [4], [26]. Tal aumento ha generado que aproximadamente un 80% de maquinaria agrícola incorpore funcionalidades de agricultura de precisión alrededor del mundo asociando sus bondades al aumento de rendimientos en los insumos, seguridad alimentaria y al aumento en ganancias para los agricultores [27], [28].

A pesar de las bondades que presenta la robótica aplicada a la agricultura, a nivel nacional aún no se alcanza el nivel de confianza suficiente para generalizar estas tecnologías y empezar una implementación en mayor escala sobre plataformas móviles en el agro, que permita generar impacto en términos sociales cerrando la brecha tecnológica hacia la agricultura de precisión, la tecnificación del campo y en aspecto económico la explotación de la capacidad agropecuaria que el país siempre ha tenido [26], [29], [28].

Con el objetivo de contribuir en el impacto social y económico nacional, el grupo de investigación GIDAM de la Universidad Militar Nueva Granada, busca lograr automatizar las tareas que realiza un agricultor durante la cosecha e incorporar ciertas funcionalidades a una plataforma móvil autónoma. Es donde el presente trabajo busca aportar una solución haciendo uso de la inteligencia artificial al problema de desplazamiento autónomo de la plataforma, incorporando un modelo que permite la percepción y toma de decisiones que realiza el cerebro humano, en este caso un trabajador del agro [6].

5. MARCO TEORICO

Para el desarrollo de la propuesta es necesario hacer un recorrido por algunos conceptos que permiten establecer una idea más clara de las herramientas que se plantean y esperan ser usadas en el transcurso:

La **auto conducción** aparece alrededor de la década de los 80, comprendida como un tipo de manejo sobre un vehículo de cualquier tipo (terrestre, aéreo, acuático o submarino) donde el conductor (humano) delega parcial o totalmente el manejo sobre un sistema con el objetivo de planear una trayectoria o **path planning** como punto de partida, y establecer la selección de una ruta entre una posición inicial hasta una marca objetivo [7], [12]. Un término similar es la **planificación de movimiento**, la cual constituye el trazado de un camino para alcanzar dicho objetivo en función de la variación temporal y de la configuración mecánica del mismo.

En referencia a todos los procesos nombrados para la conducción autónoma, la **inteligencia artificial** como aproximación al comportamiento cerebral, a la representación de conocimiento y a mecanismos evolutivos ha brindado la capacidad de interacción con otros agentes, extracción de estructuras de información y también ha incorporado métodos de interpretación con predicción espacio-temporal de su comportamiento [13]. Una de las alternativas que ha surgido para dicha predicción, ha sido el aprendizaje profundo por refuerzo o **Deep-Reinforcement Learning**, consistente en resolver problemas de percepción asociados a toma de decisiones de manera similar a como lo realizaría un cerebro humano, basándose en recompensas, generalmente expresadas en valores numéricos asociadas al comportamiento deseado dentro de un entorno determinado [6], [30].

Las pruebas de algoritmos y el acercamiento a implementaciones sobre plataformas robóticas involucran el uso de entornos de trabajo como **ROS (Robot Operating System)**, que ofrecen capacidades para manejo de software o dispositivos de bajo nivel, funcionalidad, comunicación entre procesos y procesamiento en múltiples núcleos [31]. Para estas pruebas el uso de entornos de simulación como **Gazebo** se ha tornado importante, debido a que permite la prueba rápida de algoritmos y diseño de estos, involucrando sistemas dinámicos, sensores y modelos de comunicación que permite la creación de simulaciones muy realistas [32].

5.1. Aprendizaje Automático (Machine Learning)

El machine learning o aprendizaje automático, establece para una máquina el aprendizaje de tareas basándose en la experiencia obtenida de múltiples repeticiones [33], de forma similar a como lo hacen humanos y animales [34]. Este aprendizaje reflejado en algoritmos emplea métodos computacionales que permiten adaptar su rendimiento a medida que la cantidad de información disponible y suministrada nutre el aprendizaje [34].

Los algoritmos empleados para el aprendizaje son capaces de encontrar patrones y comportamientos en volúmenes de datos, con los cuales se realizan predicciones o toma de decisiones (1). Múltiples aplicaciones para el machine learning han sido implementadas en campos como: búsquedas en línea, finanzas, detección de fraudes, diagnósticos médicos, comercio de acciones, entre otros [35]. Según la necesidad que se tiene para implementar un algoritmo de aprendizaje automático, se distinguen tres tipos:

5.1.1 Aprendizaje supervisado:

El principal objetivo es realizar modelos que permitan realizar predicciones con cierto nivel de incertidumbre, usando pares de datos conocidos donde a partir de un dato de entrada, el segundo dato es resultado o respuesta del primero, se dice que el algoritmo se encarga de encontrar la estructura interna de la información para realizar predicciones [36]. Aplicaciones incluyen técnicas de clasificación como diagnósticos médicos (benignos o malignos) y técnicas de regresión como las usadas para predecir fluctuaciones de temperatura o diferentes variables [34].

5.1.2 Aprendizaje no supervisado:

Su funcionamiento es similar a los modelos supervisados, ajustando su modelo a partir de los datos de entrada con la salvedad de no involucrar datos de respuesta o resultados [37].Esto permite encontrar patrones ocultos lo cual usualmente es usado para agrupar en varios conjuntos, datos con características similares [36],las aplicaciones vigentes incluyen análisis de secuencias genéticas, investigación de mercados y reconocimiento de objetos [34].

5.2. Aprendizaje por Refuerzo (Reinforcement Learning - RL)

A diferencia de los otros modelos de aprendizaje automático, el aprendizaje por refuerzo utiliza datos de un entorno dinámico, con el objetivo de encontrar las acciones que generan un resultado óptimo [38]. Este proceso se realiza mediante la interacción con el entorno y se asemeja al modelo conductista que empleamos los humanos, en el cual recompensas o penalizaciones son aplicadas en función de las acciones realizadas [37] aplicaciones del aprendizaje por refuerzo incluyen procesos en donde la toma de decisiones sea imprescindible para lograr un objetivo [36].

En este caso, un conjunto de datos estáticos son empleados como entradas o salidas según el caso para extraer un modelo que permite realizar predicciones una vez la estructura de los datos ha sido descifrados [36], [37], [38].Cuando se trata de datos y entornos dinámicos el aprendizaje por refuerzo permite desplegar un agente con la misión de explorar, interactuar y aprender del entorno [38].

Cuando se despliega un agente, este empieza a ejecutar acciones, que no son determinadas explícitamente, sino que son suministradas por un sistema de ensayo y error, donde se recibe una recompensa o una penalización en el proceso de resolver el problema (exploración) [39]. En cada uno de estos intentos, las acciones que son ejecutadas no afectan únicamente el desempeño actual de la misión, sino que afectan en cierta medida el ensayo siguiente y los subsecuentes [34] haciendo que progresivamente el resultado final en cada prueba sea cada vez más acertado.

La aplicación entonces del aprendizaje por refuerzo se puede asociar al concepto de aprendizaje conductista, donde con el fin de generar un aprendizaje autónomo se saca provecho de un proceso de ensayo-error y la asociación de acciones inmediatas con un resultado que se desencadenara en el término de mediano a largo a plazo [40]. Teniendo en cuenta el concepto y entrando a un campo practico, se entiende que el agente debe ser capaz de recibir información del entorno en donde se desempeña para explorar, debe sentarse un conjunto de reglas que le indiquen como alcanzar el objetivo y las acciones que deben ejecutarse en consecuencia [34].

5.2.1 Elementos del aprendizaje por refuerzo

Hasta el momento los elementos que han sido nombrados son los dos fundamentos del aprendizaje por refuerzo, el agente encargado de realizar la exploración y el entorno destinado para tal [41]. Para cumplir esto, el agente debe estar recibiendo constantemente información de su estado actual con relación al entorno, este conjunto de datos recibe el nombre de observaciones y pasan a tener un papel importante ya que van a determinar el comportamiento o acciones que van a ser ejecutadas luego de ser interpretadas [39], [34].

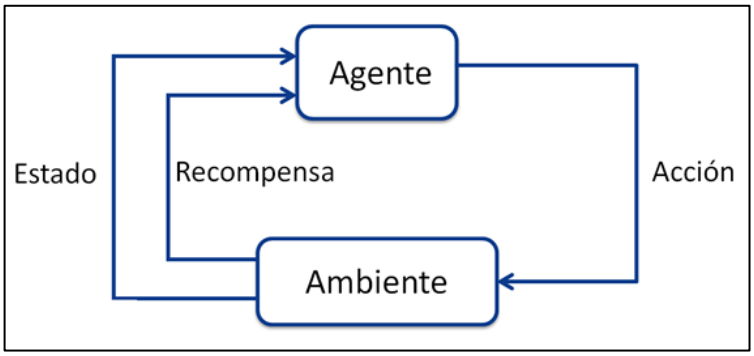


Figura 3. Diagrama aprendizaje por refuerzo.
Recuperada de [41]

Como se aprecia en la Figura 3, el agente está recibiendo constantemente información de dos fuentes y actuando en consecuencia. La primera instancia del proceso realiza una observación para extraer toda la información del entorno (estado), según lo observado el agente en consecuencia decide que acción se va a ejecutar para modificar el estado del ambiente, cuando se establece dicho cambio se genera por consecuencia una recompensa [39]. En este ciclo entonces se parte de un estado inicial para ir a otro estado con nueva información, donde el agente interpretara según la recompensa, que acciones debe seguir ejecutando y cuales debe evitar [38].

5.2.2 La política y la recompensa

Concretamente la política o norma, establece el comportamiento del agente dependiendo de la información recibida traduciéndola en acciones de salida, es la función que se encarga de optimizar el desempeño para obtener la mayor recompensa posible en determinado momento [41]. La recompensa entonces es el objetivo a maximizar en el largo plazo y es donde está definido el objetivo para el que se ha diseñado el algoritmo de aprendizaje, es la que indica numéricamente que tan buena o mala resulto una acción ejecutada y se puede interpretar como el estímulo para entrenar el agente [34].

5.2.3 Analogía al control convencional

Generalmente al hablar de sistemas dinámicos en ingeniería, se asocia el termino control, consistente en estabilizar dicho sistema y hacer que cumpla un objetivo, para tal, el controlador se encarga de corregir errores y de eliminar perturbaciones presentes haciendo uso de una realimentación [38]. El problema general aparece cuando no se puede adquirir o estimar adecuadamente la información necesaria, usualmente en sistemas no lineales o modelos complejos con múltiples estados, complicando los requerimientos para diseñar una ley de control [42].

El aprendizaje por refuerzo como analogía al control convencional permite entonces mezclar toda la información de los estados provenientes del sistema, de las señales de control a ser ejecutadas y toda la complejidad envuelta en cualquier modelo representativo del sistema para interpretarla en un único bloque que, recibe toda la información disponible, la procesa y entrega una o varias señales de control [38].

A nivel general se sigue un procedimiento general para aplicar el aprendizaje por refuerzo en la resolución de un problema, el cual se muestra a continuación:

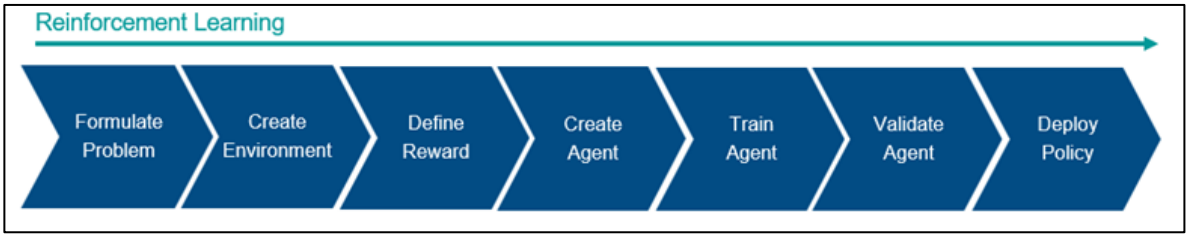


Figura 4. Diagrama de flujo para Aprendizaje por Refuerzo
Tomada de [43].

6. DESARROLLO

En este capítulo se pretende documentar todo el proceso realizado con el objetivo de implementar un control de desplazamiento sobre la plataforma diferencial haciendo uso del aprendizaje por refuerzo. Se describe todo el curso de aproximación a partir de la navegación en una cuadricula, posterior a la implementación del modelo dinámico de un tanque de agua para control de nivel, hasta la implementación de dos técnicas diferentes de aprendizaje sobre el modelo dinámico del robot para la navegación en un entorno desconocido. Se muestra a continuación (Figura 5) un esquema general del desarrollo:

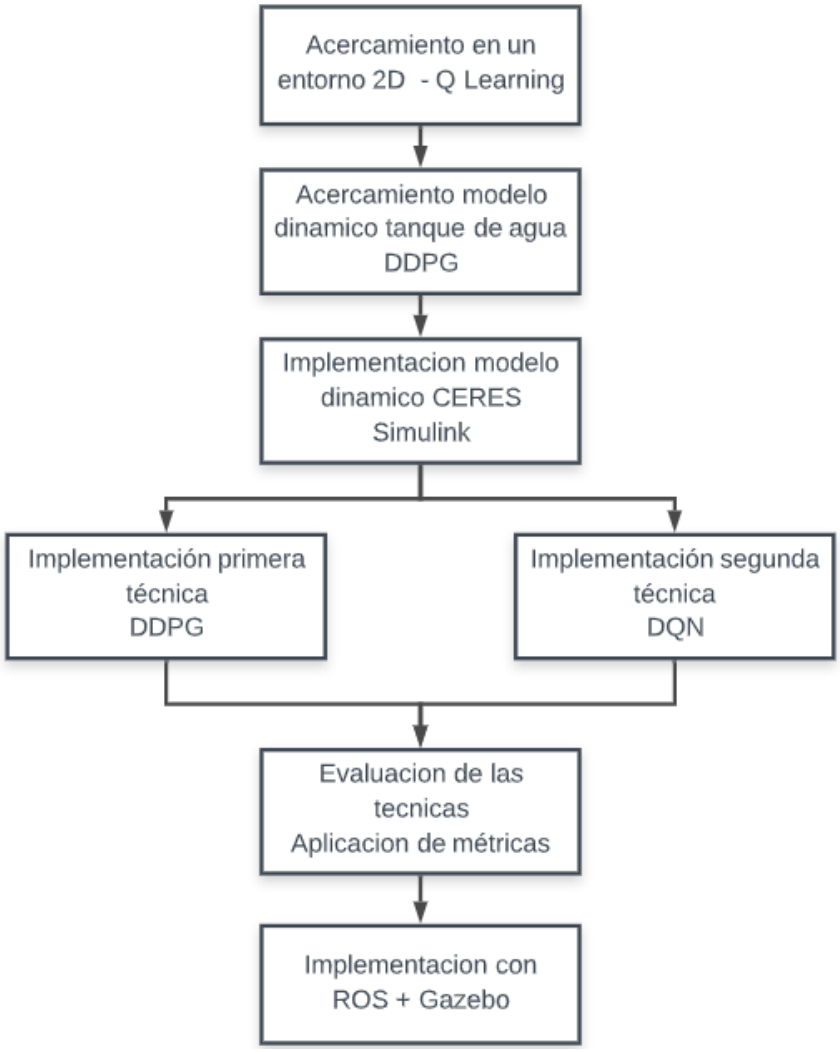


Figura 5. Esquema general de desarrollo

6.1. Aproximación RL en 2D Grid (Q-LEARNING)

Como primera aproximación al aprendizaje por refuerzo y para empezar el enfoque hacia la navegación de un robot, se utilizará el Toolbox de Reinforcement Learning para generar un espacio en dos dimensiones discretizado en una cuadricula.

Q-Learning Agent

Para realizar la primera aproximación se hace uso del algoritmo Q-Learning, el cual consiste en ser model-free, online y off-policy. Al hablar de model-free se hace referencia a que el agente no necesita tener información a priori del entorno, por lo cual realizará un proceso de exploración completo, por las áreas de baja y alta recompensa, el aprendizaje online dicta que estamos obteniendo información que puede ser variable, de manera continua para entrenar el modelo y no se está utilizando un conjunto de datos estáticos como se realizaría en un algoritmo offline [38], [44]. Finalmente, al hablar de un algoritmo off-policy se determina que la acción a tomar se aproxima independientemente de la función de comportamiento que este siendo optimizada [34]. Para la implementación del agente se siguieron los pasos del diagrama de flujo indicado en la Figura 4, documentación y funciones que reposan en [45]

6.1.1 Creación del entorno

Para la creación del entorno, se establece un espacio de trabajo fijo de 10 x 10 casillas simulando un posible entorno de trabajo de la plataforma de 10 metros y que sea fácilmente escalable a distintas dimensiones. En él, se ubicarán obstáculos de forma totalmente aleatoria en cada episodio de entrenamiento, a fin de probar la capacidad del agente para desplazarse en un entorno totalmente desconocido, donde adicionalmente se añaden 2 casillas más tanto horizontal y verticalmente para delimitar el espacio de trabajo. El punto inicial y final serán constantes para todo el proceso de entrenamiento.

Se tienen entonces en consideración los siguientes aspectos:

- Dimensiones de la cuadrícula
- Punto o estado inicial y final
- Ubicación de los obstáculos

Para la creación de la cuadrícula se utiliza una función a la cual se le deben indicar la cantidad de filas y de columnas para el entorno, lo propio se debe hacer para definir el punto inicial, el punto final (TerminalState) y la ubicación de los obstáculos, todo se indica pares coordenados (x,y). En el caso de los obstáculos se indica un vector que contiene todas las coordenadas que se desean tener como obstáculos.

Todo el procedimiento para la creación del entorno se resume en la creación una función, que tiene como parametros la cantidad de filas y columnas en la cuadrícula, las posiciones iniciales y finales para la navegación y el porcentaje de obstáculos que se desean añadir aleatoriamente sobre el entorno. El procedimiento completo para crear el entorno se resume en la Figura 6.

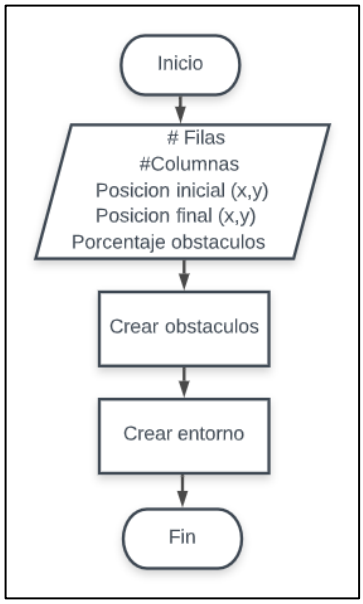


Figura 6. Creación entorno Q-Learning

Cuando se ha definido en su totalidad el entorno y se ha creado, este es asignado a una variable que contiene todo el objeto y sus parámetros. El entorno creado para navegación con obstáculos se muestra en la Figura 7.

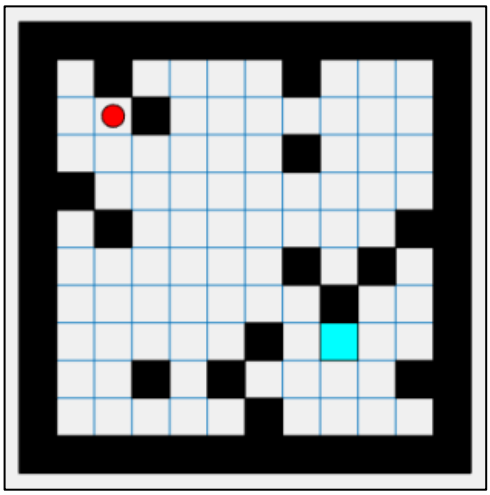


Figura 7. Entorno 2D Grid para RL

En la Figura 7, se interpretan en color negro todos los bordes de la cuadrícula y los obstáculos, en color azul aguamarina el punto final para la navegación y como un punto rojo el agente posicionado en las coordenadas iniciales indicadas.

Ahora, para establecer el comportamiento que se desea obtener del agente en la navegación se deben establecer las recompensas en el proceso de exploración, para tal se debe modificar la matriz de recompensa asociadas al entorno, vale la pena aclarar que esta matriz es de 3 dimensiones definidas por: Número Estados x Número Estados x Número Acciones. El número de estados en este caso corresponde a todas las posibles casillas donde puede estar el agente, es decir 144 debido a las dimensiones 10 x 10 previamente establecidas más todas las casillas correspondientes a los bordes para un total de 144, el número de acciones es 4 debido indicando las direcciones dentro de la cuadrícula, es decir arriba, abajo, izquierda y derecha.

Para establecer las recompensas, se debe modificar el parámetro en donde se establecen las recompensas que se muestran en la Figura 8.

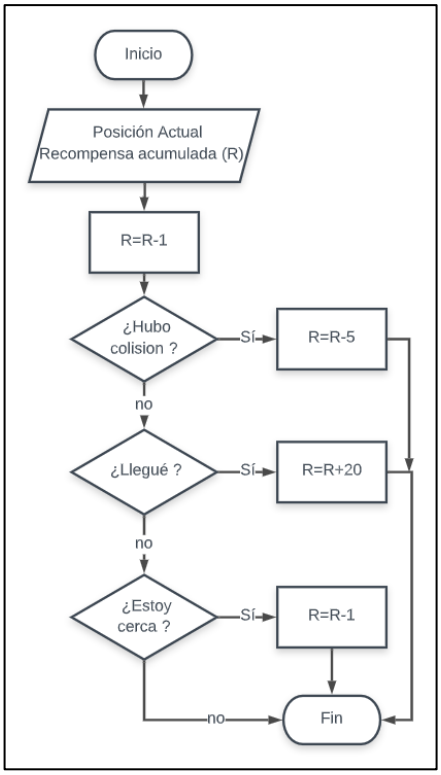


Figura 8. Recompensas navegación cuadrícula

Como punto de partida se penaliza cualquier movimiento con un valor de -1 y el hecho de que el agente este en la misma posición de un obstáculo (colisión). Las

recompensas se otorgan en caso de que el agente llegue a la posición final y cuando se encuentre en una casilla adyacente (cerca).

6.1.2 Creación del agente

Para la creación del agente, hay que tener en cuenta que la representación de todo el sistema reposa en tablas y matrices, por lo que se debe crear una con las dimensiones de las observaciones y las acciones, que será la información utilizada para aproximar la recompensa a largo plazo, esta tabla recibe el nombre de crítico y también se debe asociar una constante de aprendizaje. El proceso general para la creación del agente se establece en la Figura 9.

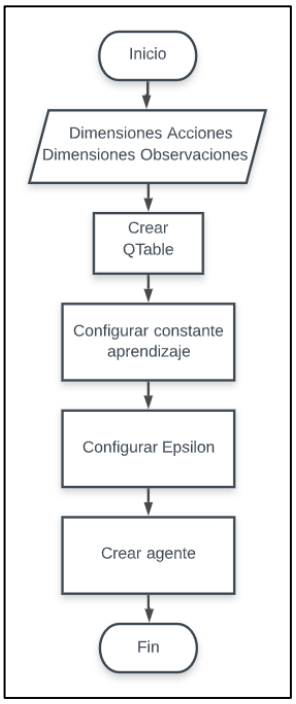


Figura 9. Creación agente Q-Learning

Vale la pena aclarar que en la creación de la QTable se establece con que velocidad se realizara el entrenamiento. Cuando el valor de esta constante es muy pequeño, el aprendizaje será demorado, pero encontrará soluciones al comportamiento optimas, contrario a valores grandes que ejecutaran un entrenamiento más rápido, pero puede generar resultados que no sean óptimos, este valor siempre se coloca entre 0 y 1 [46].

Para el método de exploración se utilizará el método Epsilon-Greedy, que nos permitirá realizar una exploración “codiciosa” es decir permitirá al agente navegar por todo el espacio pero tendiendo siempre a repetir con mayor frecuencia las

acciones que le generen una mayor recompensa, entre mayor sea la constante asignada para Épsilon, el agente navegará más rápida y aleatoriamente el entorno [47], [48].

6.1.3 Entrenamiento

Para el entrenamiento del agente se deben realizar una serie de especificaciones generales, en este caso se indica la cantidad de episodios a realizar, los pasos que se deben realizar en cada episodio y la manera de detener el entrenamiento.

Una vez se inicia la ejecución del algoritmo, haciendo uso del entorno generado para el agente, internamente el algoritmo realiza lo siguiente [49]:

Antes de comenzar, se inicializa el crítico (Q) con valores iniciales aleatorios para el estado y la acción (S,A). Para cada episodio se realiza la observación (S), es decir se lee en qué posición se encuentra el agente en la cuadrícula y se ejecutan iterativamente los siguientes pasos, hasta que el agente se encuentre en el estado final:

1. Para la observación actual (S), se selecciona aleatoriamente una acción (A) con la probabilidad indicada por la exploración épsilon-greedy indicada por el crítico (Q), si no se cumple dicha probabilidad se selecciona la acción que el crítico determina como la mejor:

$$A = \max \mathrm{Q}\,(S, A) \tag{1}$$

2. Ejecutar la acción (A) determinada, luego observar la recompensa (R) y la nueva observación (S') que será el nuevo estado.

3. Si S' coincide con las coordenadas finales o estado terminal, la función objetivo (y) toma el valor de R y el episodio se termina, si no coincide la función toma el siguiente valor:

$$y = R + \gamma \max\ \mathrm{Q}(S, A) \tag{2}$$

   Donde:

   - γ es el factor de descuento, debe tomar valores entre 0 y 1, afectando el comportamiento del agente entre el corto y largo plazo, entre más grande es el factor las recompensas a largo plazo tendrán mayor relevancia, valores cercanos a 0 otorgan mayor importancia a las recompensas inmediatas o de la iteración actual [46].

4. Se calcula el valor del crítico (Q) para la siguiente iteración, mediante:

$$\Delta Q = y - Q(S, A) \tag{3}$$

5. Se actualiza el crítico haciendo uso del factor de aprendizaje (α):

$$Q(S, A) = Q(S, A) + \alpha * \Delta Q \tag{4}$$

6. Colocar la nueva observación (S') como la observación actual (S)

6.1.4 Resultados

Durante el proceso de entrenamiento, se decide detener el proceso, ya que como se evidencia en la Figura 10, la recompensa promedio (rojo) se ha estabilizado y converge con el valor de la estimación a futuro (verde), es decir que para episodios siguientes ya no se presentará alguna posibilidad de mejora en el comportamiento ni de aumentar la retribución obtenida.

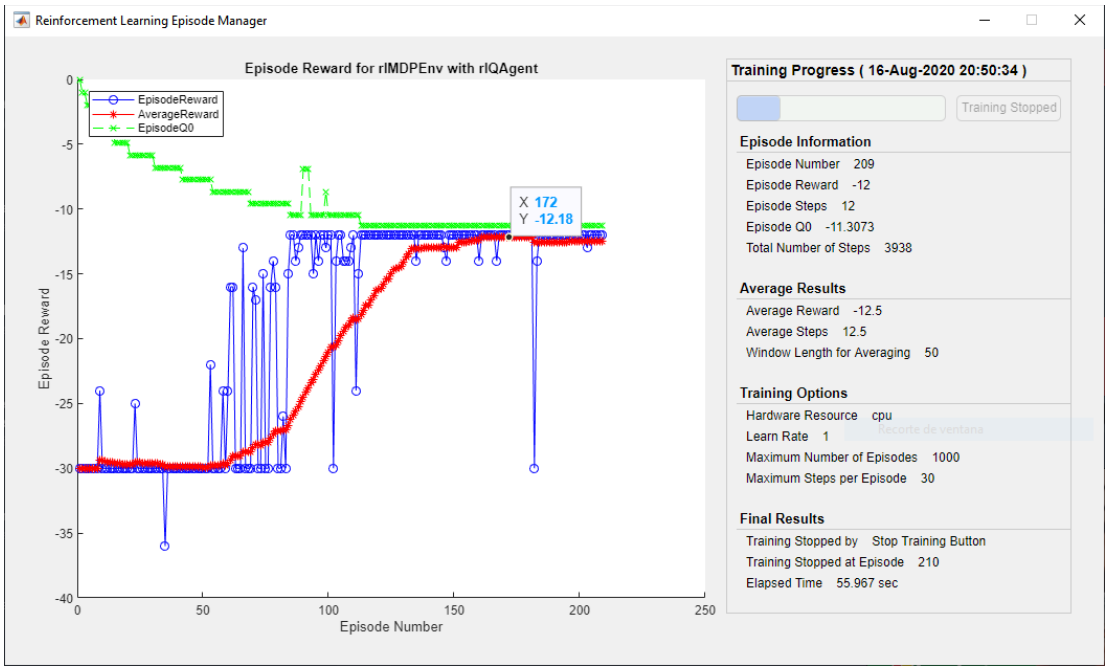


Figura 10. Ventana entrenamiento

Adicionalmente con la visualización del entorno en la Figura 11, se confirma que el agente, para el episodio en donde fue detenido, efectivamente alcanzó el estado terminal.

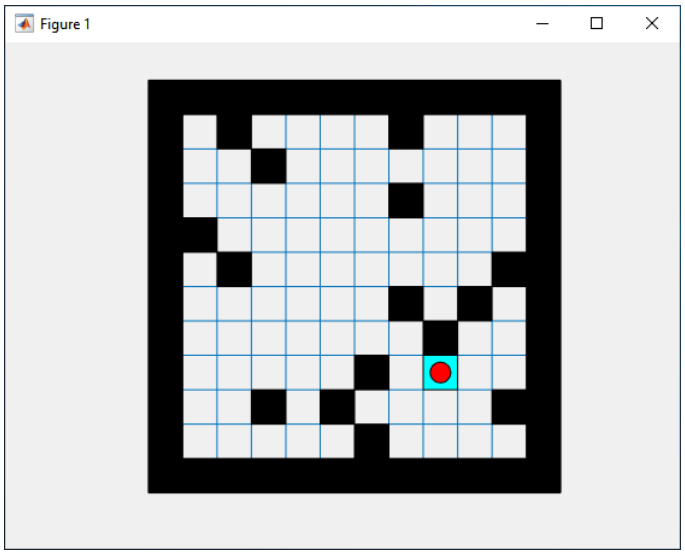


Figura 11. Agente en posición final

6.1.5 Validación

Finalmente, para realizar la validación del agente entrenado y que el comportamiento deseado fue generado, debe realizar el proceso mostrado en la Figura 12.

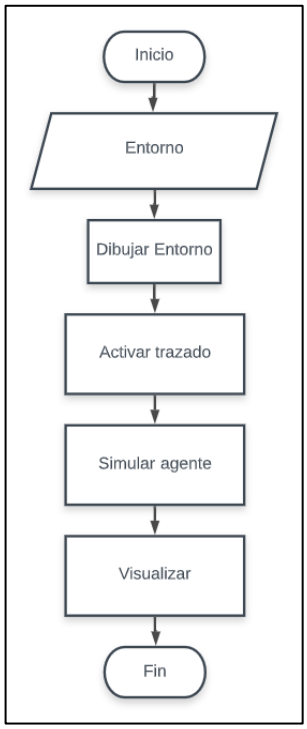


Figura 12. Secuencia para validación Q-Learning

El proceso ilustrado en la Figura 12, permite desplegar nuevamente el agente dentro del entorno en su posición inicial original, para indicarle que ejecute la función de comportamiento entrenada, pero dejando marca sobre cada casilla por la que pasa, el resultado se muestra en la Figura 13.

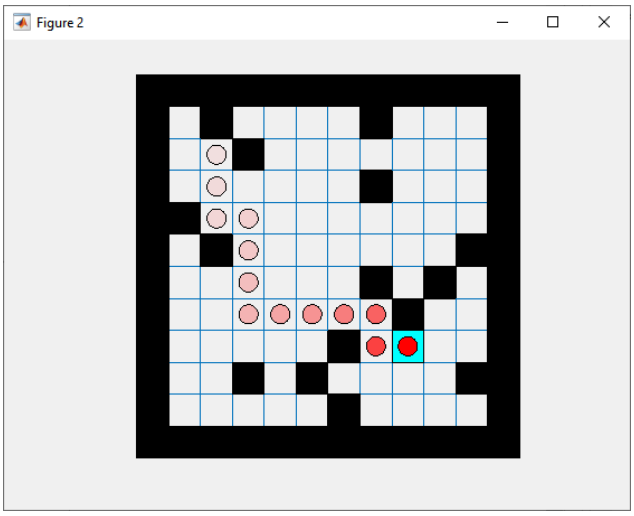


Figura 13. Validación del entrenamiento

Efectivamente como se demuestra en la Figura 13, el agente desplegado con la política de comportamiento adquirida en el entrenamiento realiza la navegación dentro de la cuadrícula, evitando la colisión con los bordes y los obstáculos hasta una posición final indicada.

6.2. Aproximación RL usando modelos dinámicos (Tanque de nivel)

Como primera aproximación al uso de modelos dinámicos, se toma el ejemplo del control de nivel de un tanque de agua mediante un controlador PID, el cual será reemplazado por un agente de aprendizaje por refuerzo conocido como DDPG (Deep Deterministic Policy Gradient), que aprende a realizar dicho control. Adicionalmente esto permitirá familiarizarse en cómo desarrollar un entorno para aprendizaje por refuerzo en Simulink, haciendo uso de modelos dinámicos.

6.2.1 Creación del entorno

El principal cambio que se evidenciará para la creación del entorno es el uso de un modelo dinámico creado en Simulink, para lo cual se usará el modelo dinámico de un tanque de agua. Como primera medida lo que se debe añadir al modelo debe ser el bloque RL Agent, que permitirá asociar directamente como entrada las observaciones, la recompensa y las condiciones para detener el episodio de entrenamiento, así como genera la(s) acción(es) de control como salida.

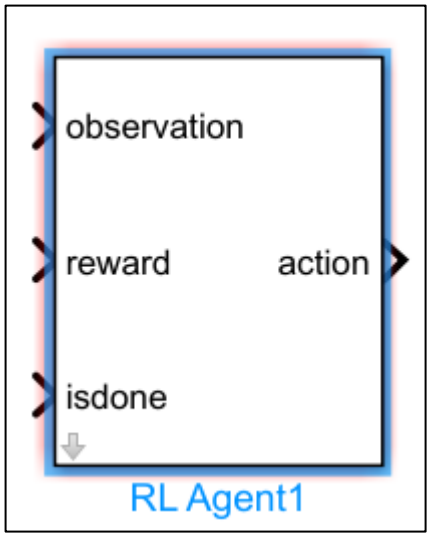


Figura 14. Bloque RL Agent

Como se observa en la Figura 14, hay que definir las observaciones que se asociarán al agente, la recompensa del entorno y las condiciones para finalizar. Al tratarse del control de altura en un tanque, es conveniente observar la altura actual del tanque, el error con respecto a la referencia y realizar la integración de dicha señal de error para realizar la mitigación.

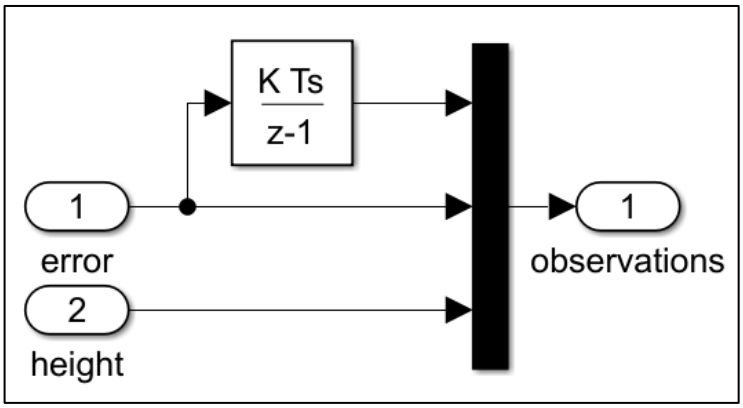


Figura 15. Señales de observación tanque con DDPG

En segundo lugar, se define la señal de recompensa que pasará a ser la sumatoria de las pequeñas acciones que se quieren premiar o penalizar. Como se muestra en la Figura 16, se van a penalizar dos acciones puntuales, la primera en cada periodo de muestreo sumará un valor de -1 si la señal de error no es menor a 0.1m y la segunda en caso de que la altura del tanque supere un valor mayor a 20m o sea 0m lo cual indica el desborde o el vaciado del tanque respectivamente.

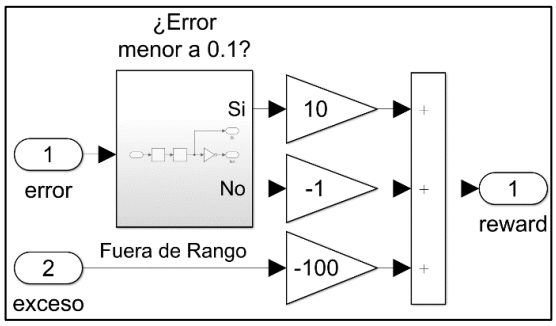


Figura 16. Señal de recompensa para tanque con DDPG

Ahora para indicar durante el proceso de exploración qué acciones se deben seguir tomando exitosamente, se añade una recompensa con un valor de 10, si bien es un valor tomado arbitrariamente es un valor que en el proceso de entrenamiento permitirá evidenciar una parametrización de cierto comportamiento y de la convergencia del aprendizaje.

En la tercera entrada del bloque RL Agent (Figura 14), se coloca entonces la condición para detener el entrenamiento, la cual coindice con la señal 2: exceso, en la Figura 16, es decir al sobrepasar una altura de 20m o el vaciado del tanque, el episodio actual se detiene, se penaliza dicho suceso, se reinicia el proceso y se inicia un nuevo episodio.

Una vez se han definido los bloques que determinan las observaciones, la señal de recompensa y las condiciones para terminar, en conjunto el sistema se visualiza en la Figura 17.

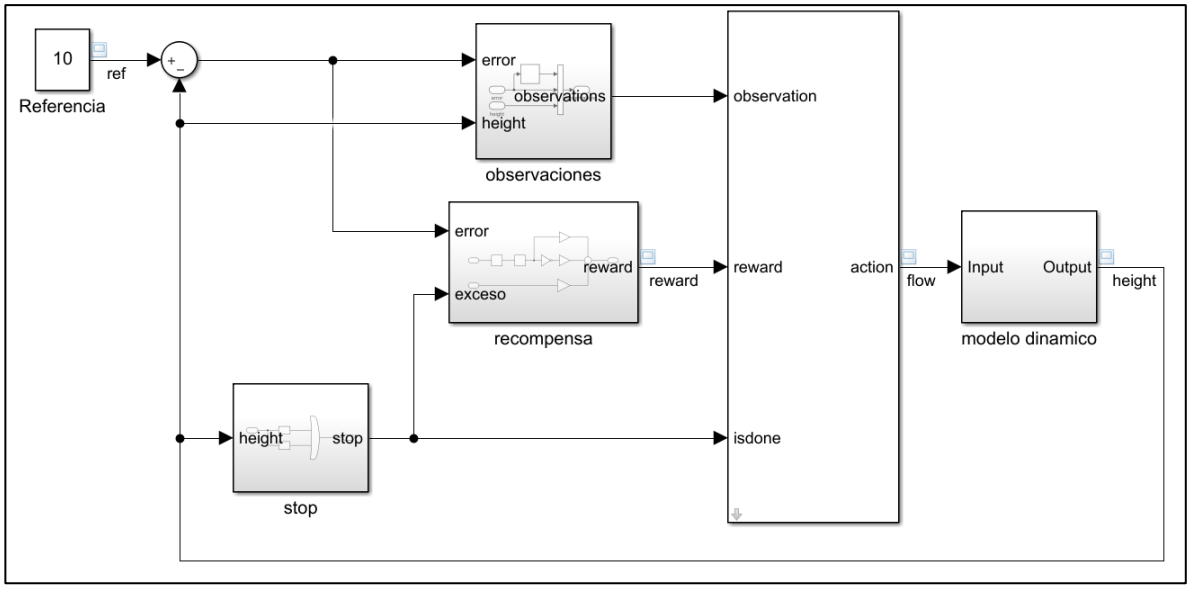


Figura 17. Entorno en Simulink para tanque con DDPG

Luego de que el entorno es creado en Simulink, se debe pasar a configurar un script que determinará los parámetros del agente. Se debe entonces especificar las señales asociadas a la observación y sus respectivas características, indicando el modelo con el que se va a trabajar, las observaciones y las acciones como muestra la Figura 18.

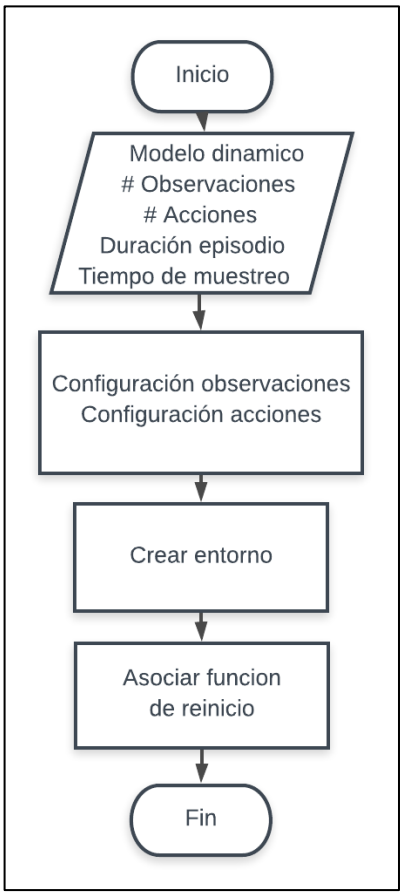


Figura 18. Creación agente para tanque DDPG

Las señales de observación van a ser continuas y se recibirán como un vector de 3 filas, se deben indicar los limites inferiores y superiores para cada señal observada indicados en pares. Para el caso de las acciones asignadas, se determina que la acción es única y toma valores continuos, corresponde al voltaje aplicado para controlar la válvula de flujo de ingreso al tanque.

Vale la pena recalcar que se indica el periodo de muestreo para el entrenamiento y el tiempo total establecido para la simulación. Adicionalmente, se especifica una función de reinicio, la cual cada vez que se termine sin importar porqué, ejecutará dicha función para llevar el modelo a condiciones iniciales para la siguiente

exploración, así como una nueva referencia. La función de reinicio está diseñada para generar una señal de referencia en altura y una condición inicial para la altura del tanque como se indica en la Figura 19.

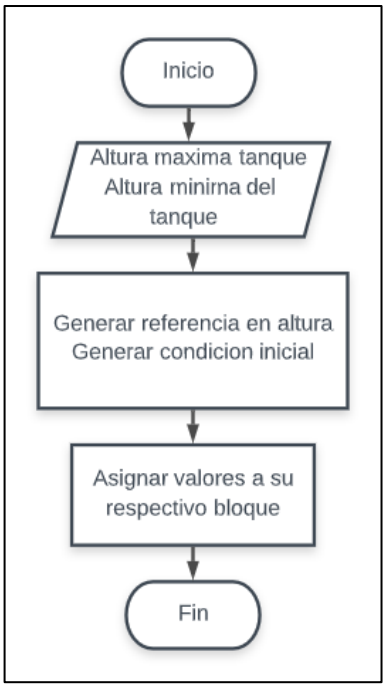


Figura 19. Función de reinicio para tanque

Se genera entonces dos valores aleatorios tanto para la referencia en altura y para la condición inicial de la altura del tanque, dichos valores son asignados a las variables que reposan en cada bloque indicado del modelo en Simulink.

6.2.2 Creación del agente

El algoritmo DDPG, presenta una particularidad con respecto al Q-Learning y es que al estar basado en un método Actor-Critico permite hacer muestreo y recolección de experiencias anteriores para lograr la parametrización de problemas donde hay estados de grandes dimensiones tanto en las observaciones y/o en las acciones [50]. A diferencia de la representación utilizada para Q-Learning que consistía en la creación de una matriz que relacionaba los estados y las acciones, la representación en este caso se realiza mediante una red neuronal tanto para el actor como para el crítico [51].

6.2.2.1 Critico

En el algoritmo, el agente realiza una aproximación a la recompensa en el largo plazo la cual se denomina función de valor y es ejecutada por lo que se conoce como el crítico (Q), el cual hace uso de la información obtenida en cada momento para observaciones y acciones para cumplir dicho fin [52], [53].

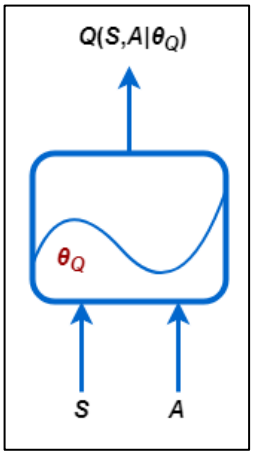


Figura 20. Diagrama del Critico.
Tomada de [54]

Para la creación de una representación mediante redes neuronales, se hace uso de la documentación y los métodos indicados en [54]. Como se muestra en la Figura 20, las entradas al crítico son las observaciones del estado actual (S) y la(s) acción(es) ejecutada(s).

Para realizar la estimación se crean dos redes diferentes (Figura 21), la primera red consiste en dos capas, la primera de 50 neuronas, la segunda de 25 y para la segunda red, se crea una capa sencilla de 25 neuronas debido a que la información de las acciones es más pequeña en relación con la información de las observaciones y siguiendo lineamientos de otras implementaciones realizadas, se establecen redes de menor tamaño teniendo en cuenta que en casos donde se extrae información de imágenes (mayor complejidad y carga computacional) para navegación, se han utilizado capas de 128, 256 y hasta 512 neuronas [55], [56].

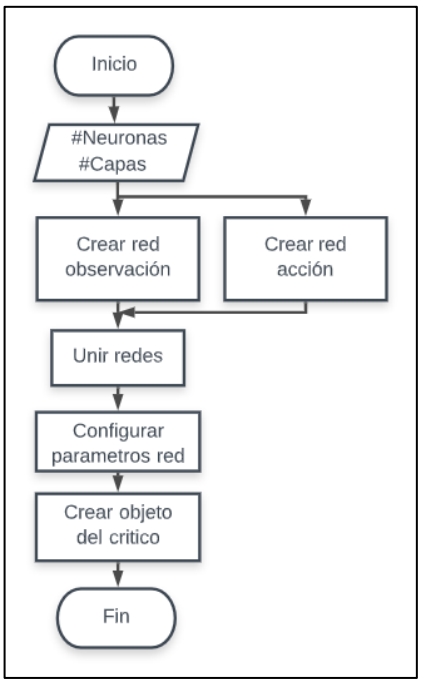


Figura 21. Creación red neuronal Crítico para tanque con DDPG

Como la estimación que se realiza es única y la información que se recibe ya es procesada independientemente en cada camino, solo hace falta unirlos con fin de generar la estimación de una recompensa a largo plazo.

Finalmente se deben ajustar los parámetros de aprendizaje de la red neuronal, la cual se configuro en un valor inicial pequeño menor a 0.1, que permitiera un aprendizaje optimo a pesar de la implicación de un mayor tiempo de entrenamiento. Además, las opciones que se consideran necesarios, como el umbral del gradiente ajustado en un valor de 1 por defecto en el algoritmo que durante el entrenamiento indica que tan grande debe ser la variación entre los parámetros del crítico entre episodios y demostró entregar unos resultados aceptables para el objetivo. En la Figura 22 se puede validar la creación de la red y su estructura completa, iniciando a la izquierda por la red que recibe la información del entorno (estado observaciones) y sus dos capas.

A mano derecha esta la red que recibe la información de las acciones que son ejecutadas y su única capa, la cual se une en el nodo add con la información procesada por la primera red y se pasa por una función de activación que se encarga de entregar la estimación a futuras recompensas.

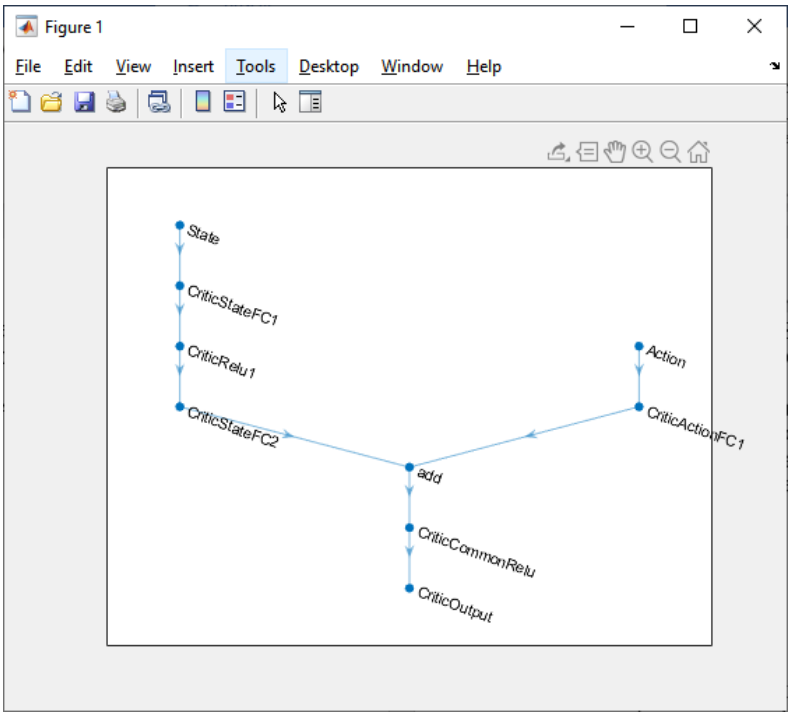


Figura 22. Red neuronal Critico para tanque con DDPG

6.2.2.2 Actor

La segunda parte del agente es conocida como actor (μ) y se encarga, a partir de las observaciones de decidir qué acción(es) elegir en ese específico momento [51], [57].

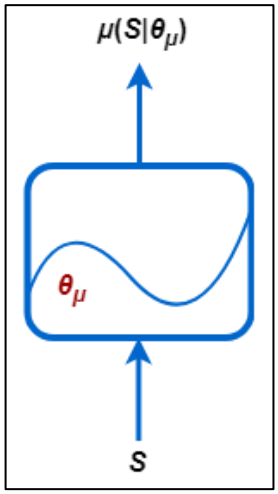


Figura 23. Diagrama del Actor.
Tomada de [54]

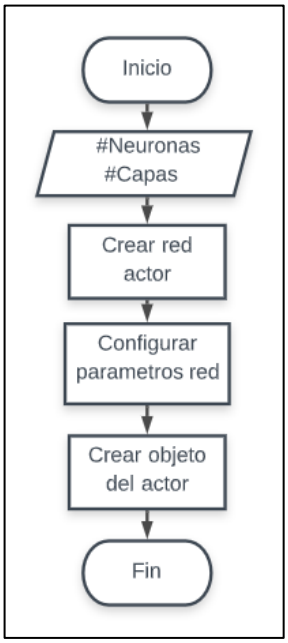


Figura 24. Creación red neuronal Actor para tanque con DDPG

En la Figura 24, se sigue el proceso para la creación de una red neuronal sencilla, de una única capa, las opciones para la red se configuran y se crea el objeto que contiene el actor, su representación y los parámetros configurados. A continuación, se valida la estructura de la red creada para la representación del crítico:

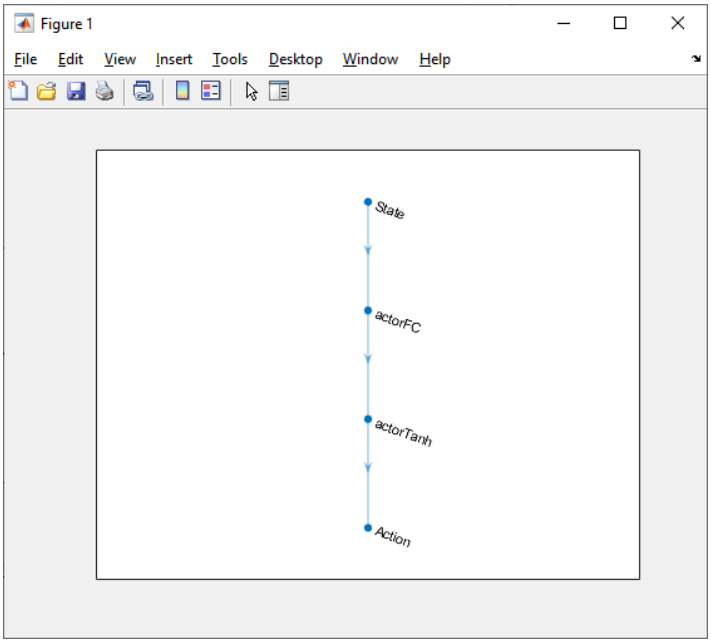


Figura 25. Red neuronal Actor para tanque con DDPG

6.2.2.3 Agente

Una vez se ha creado tanto el actor como el crítico para el agente, lo único que hace falta es configurar los parámetros necesarios, los cuales están descritos en [58]. En la Figura 26, se muestra el proceso de configuración y creación del objeto que contiene el agente a entrenar.

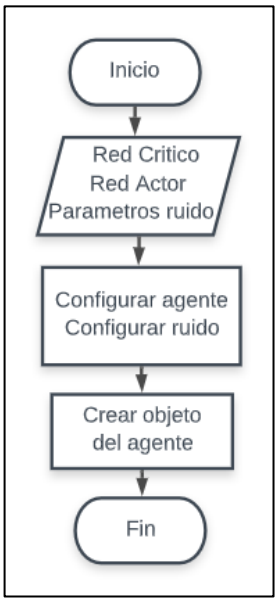


Figura 26. Creación del agente y configuración para tanque con DDPG

En la configuración del agente, se indica el tamaño de un batch que será utilizado para actualizar la del crítico, dicho batch muestrea aleatoriamente todas las experiencias de entrenamiento, para conferir una mayor estabilidad y evitar correlación entre los datos y por lo tanto un sobre entrenamiento [59].

El factor de descuento también se debe incluir, indica que tanto impacto tienen los episodios ejecutados, entre más cercano es el valor a 1 se da prioridad a la recompensa estimada al largo plazo y entra más cercano es a 0 se da prioridad a las recompensas inmediatas [46]. El factor de suavizado utilizado para estabilizar el entrenamiento ya que afecta directamente la forma en que se actualizan los parámetros tanto para el crítico como para el actor, valores grandes hacen el entrenamiento inestable pero rápido, valores pequeños permiten mejores resultados en el entrenamiento pero lo vuelven más demorado [60].

La exploración del agente DDPG, se introduce mediante la adición de ruido a la señal o señales de control que están siendo generadas mediante una función

probabilística, donde progresivamente se va disminuyendo la varianza de dicha función conforme el entrenamiento va avanzando, en medida de que el comportamiento va mejorando y la exploración debe ser menos drástica. La adición de ruido para la exploración se incluye como parámetro dentro del agente.

Finalmente, el objeto que contiene tanto el crítico, el actor, las respectivas representaciones y configuración establecida, se asigna como objeto dentro de una variable que contiene los parámetros indicados.

6.2.3 Entrenamiento

En la Figura 27, se muestra el procedimiento a seguir para ejecutar el entrenamiento y parámetros como la cantidad de episodios, los pasos dentro de cada episodio y las condiciones para dar el entrenamiento como finalizado.

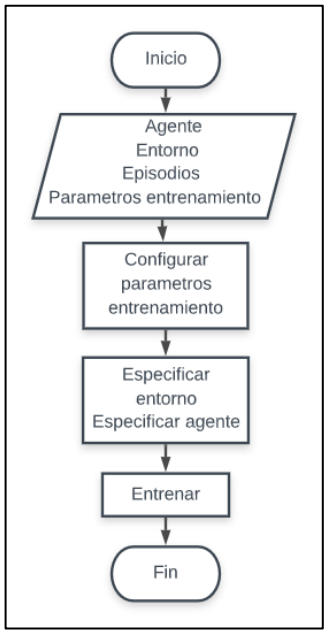


Figura 27. Opciones de entrenamiento para tanque con DDPG

Al indicar la ejecución del algoritmo, internamente el algoritmo que está siendo ejecutado realiza lo siguiente [61]:

Antes de comenzar, se inicializa el crítico (Q) con valores iniciales aleatorios para el estado y la observación (S,A) y sus parámetros (θ_Q), los mismos valores se deben inicializar para los parámetros de la función objetivo ($\theta_{Q'}$). Lo propio se debe realizar con el actor (μ), sus parámetros (θ_μ) y los parámetros de la función objetivo ($\theta_{\mu'}$).

Para cada episodio se realiza la observación del estado (S) es decir el error, su integral y la altura, también se toma el valor de la acción (A) y se ejecutan iterativamente los siguientes pasos:

1. Para la observación actual (S), se toma una acción (A) a la cual se añade ruido para ejecutar la exploración.

$$A = \mu(S) + N \tag{5}$$

   Donde N es una señal de ruido blanco
2. Se aplica la acción determinada (A) sobre el sistema, se toma el valor de la recompensa (R) obtenida y el nuevo estado (S')

3. Se almacena la experiencia en el buffer, para el algoritmo se almacena en un vector que contiene (S,A,R,S').

4. Se toman N muestras aleatorias (S_i,A_i,R_i,S_i') según se ha configurado para colocar en un batch.

5. Si el estado muestreado (S_i') es un estado terminal, la función que estima la recompensa a largo plazo (y) toma el valor de la recompensa (R_i), si no es un estado terminal se computa de la manera siguiente:

$$y_i = R_i + \gamma\, Q' \tag{6}$$

   La función objetivo consiste en la suma de la recompensa actual y la recompensa estimada (Q') multiplicada por el factor de descuento, es decir indicando sus efectos en el largo o corto plazo [46]. Para estimar la futura recompensa (Q'), se toma el valor del estado muestreado en el batch (S_i') se pasa por la función que aproxima el actor y con estos dos valores el crítico ya es capaz de computar una estimación de la recompensa.

6. Se actualizan los parámetros del crítico, minimizando la función de pérdida L para todos los episodios que fueron muestreados, lo cual permite reducir progresivamente el error entre el comportamiento actual y el comportamiento deseado [62].

$$L = \frac{1}{M}\sum_{i=1}^{M}(y_i - Q)^2 \tag{7}$$

7. Para maximizar el valor de la recompensa descontada, se deben actualizar los parámetros del crítico usando la política del gradiente (Policy Gradient), para ajustar continuamente la función que está siendo estimada para seleccionar las acciones (actor) como la función que estima las recompensas en el largo plazo (critico) [63].

$$\nabla\theta_u = \sum_{i=1}^{M} G_C G_a \tag{8}$$

Donde Gc es el gradiente de la salida del crítico con respecto a la acción estimada por la red del actor y Ga es el gradiente de la salida del actor con respecto a sus parámetros.

8. Actualizar los parámetros del actor y del crítico para un nuevo episodio, es aquí donde se efectúa el suavizado que permitirá que el entrenamiento logre un balance entre velocidad y estabilidad [60].

6.2.4 Resultados

Para este caso, al entrenamiento se le colocó un método que lo hace detener automáticamente cuando alcanzara una recompensa promedio de 800 en los últimos 20 episodios.

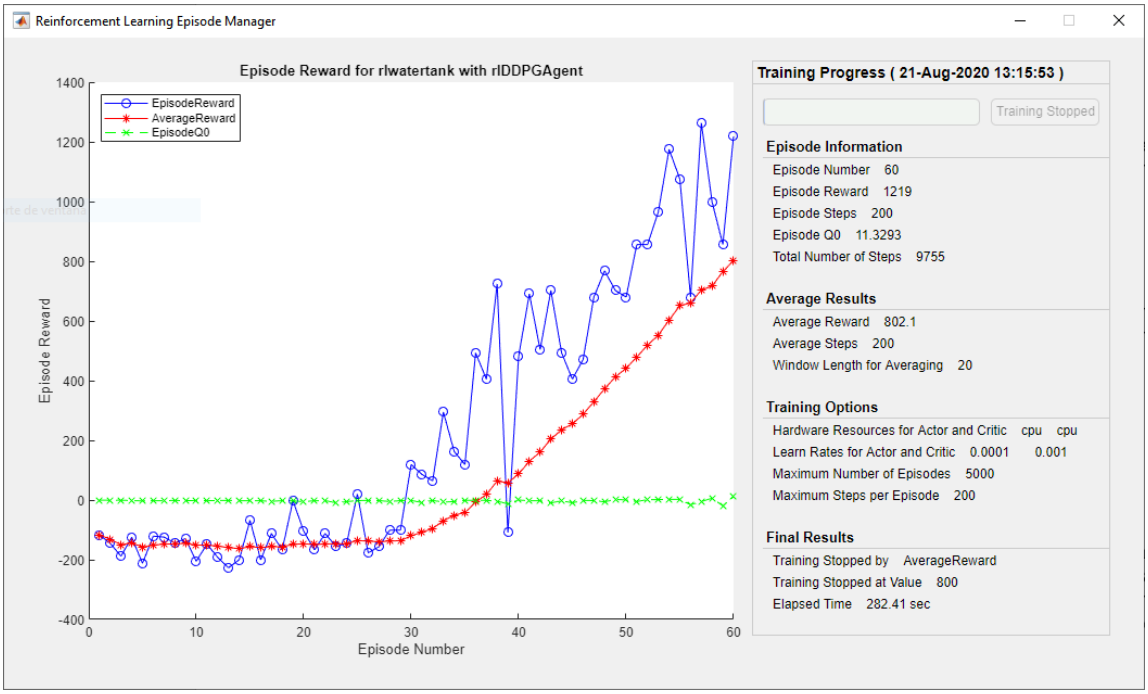


Figura 28. Ventana entrenamiento para tanque con DDPG

En la Figura 28, se evidencia como los valores de la recompensa aumentan progresivamente hasta el punto en que el entrenamiento es finalizado, ya que se determina que el agente ha parametrizado el comportamiento necesario para regular la altura del tanque al nivel establecido aleatoriamente para cada episodio.

6.2.5 Validación

En este caso, la validación del entrenamiento se puede realizar directamente sobre Simulink, ya que es conveniente el poder ver de forma directa las variables de interés como son la altura, la señal de control y la recompensa. Sin embargo, hace falta indicar las opciones de simulación indicado el agente que va a ser utilizado mediante el siguiente proceso:

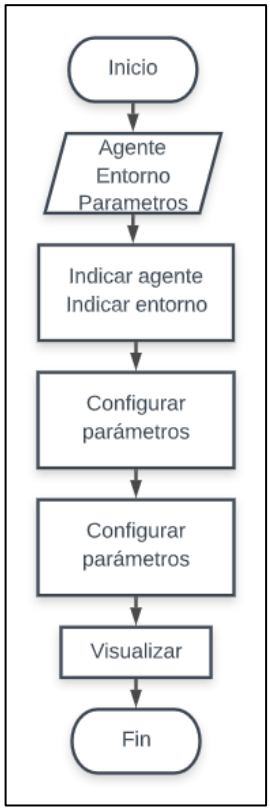


Figura 29. Proceso validación agente para tanque con DDPG

En la Figura 29, se deben indicar como parámetros la cantidad máxima de pasos que se deben ejecutar en el episodio de simulación y la interrupción en cualquier momento que se presente un error. Una vez se carga el agente, en Simulink se puede indicar directamente la referencia que se desea, en la Figura 30 se muestran las señales de referencia vs altura, la recompensa y la señal de control respectivamente, para seguir una referencia de 10 metros:

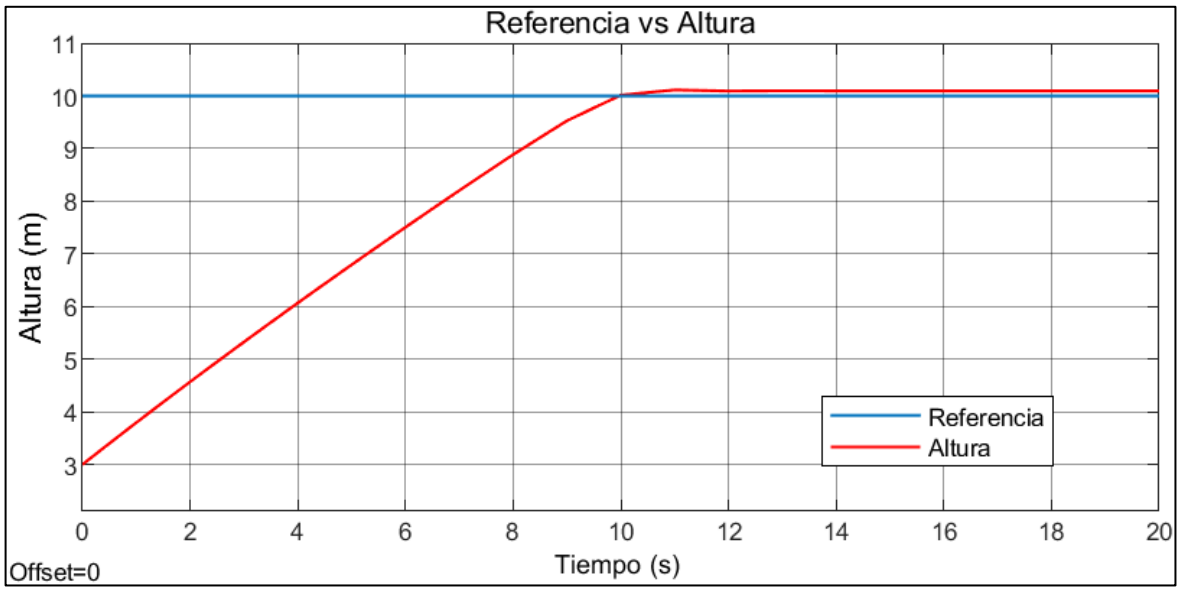


Figura 30. Referencia vs Altura en validación con referencia 10 metros

En la Figura 30 se evidencia el seguimiento a la referencia, así como una diferencia entre las dos con valor 0.1m, la cual corresponde al valor de error que se permitió en la asignación de recompensas. La permisividad del error se puede apreciar de mejor manera en la señal de recompensa para el agente (Figura 31), donde toma un valor de 10 para los casos en que el error toma un valor a 0.1m.

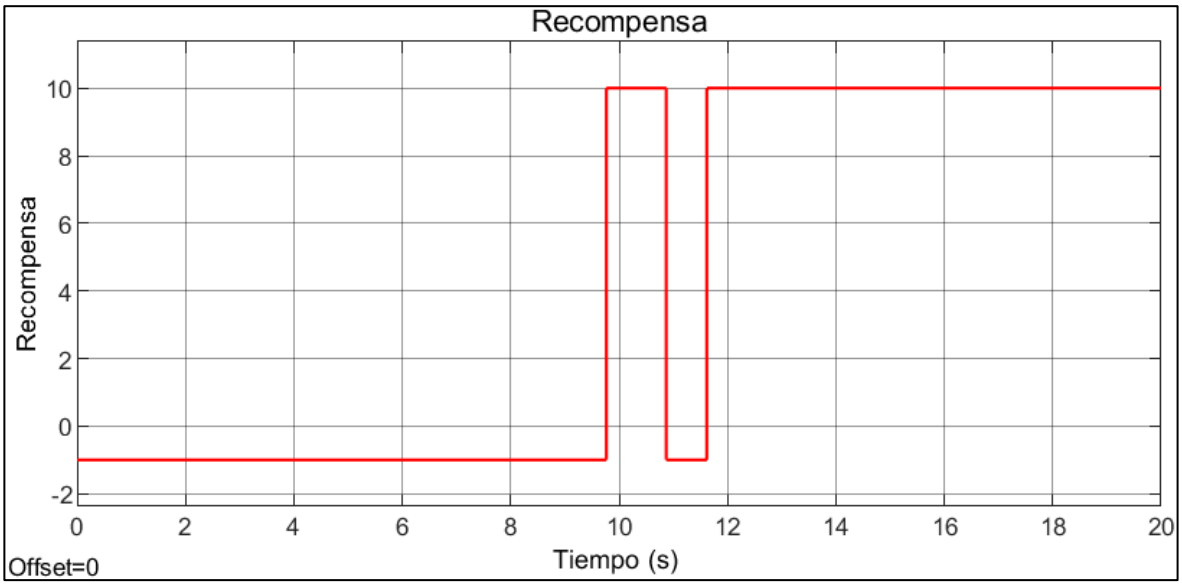


Figura 31. Recompensa en validación con referencia 10 metros

Con respecto a la señal de control, se puede evidenciar en la Figura 32 que toma valores “continuos” para cada periodo de muestreo, lo cual acerca aún más al uso del aprendizaje por refuerzo a problemas prácticos más reales.

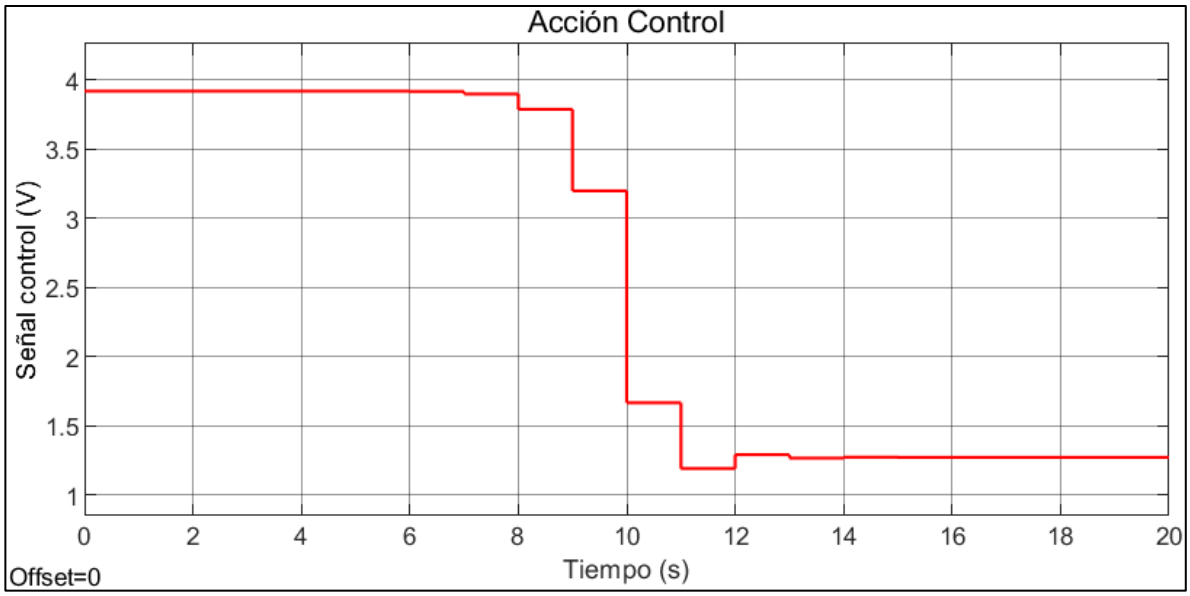


Figura 32. Señal de control con referencia 10 metros

Para la validación con diferentes valores, únicamente se debe cambiar el valor en el bloque “Referencia” y ejecutar la simulación del modelo directamente sobre Simulink.

Efectivamente como se demuestra de las figuras 29 a 31, el agente desplegado con la política de comportamiento adquirida en el entrenamiento realiza el control de nivel en un tanque, generando señales de control en un rango deseado y manteniendo el error inferior a las reglas establecidas para las recompensas.

6.3. Implementación modelo dinámico CERES con control PID para evasión de obstáculos usando DDPG

Para continuar en la implementación de un algoritmo de aprendizaje por refuerzo, se incluirá el modelo del robot CERES sobre la plataforma Simulink y el respectivo controlador PID encargado de realizar el seguimiento de referencias en velocidad lineal y en orientación.

Posterior a la implementación del modelo dinámico con controlador, se entrena un agente usando el algoritmo DDPG para que el robot realice la navegación en un espacio de 20 x 20 metros desde coordenadas iniciales aleatorias hasta el centro del espacio de trabajo, evadiendo obstáculos dinámicos que en este caso son otros

objetos con el mismo modelo dinámico con una trayectoria definida. En este caso el agente en comparación del anterior entregará dos señales de control, correspondientes a las referencias en velocidad y orientación al controlador PID

6.3.1 Creación del entorno

En primera instancia, se deben empezar a definir los componentes que estarán asociados para el aprendizaje, es decir las observaciones, la recompensa y las condiciones para detener el episodio de entrenamiento, así como las acciones de control.

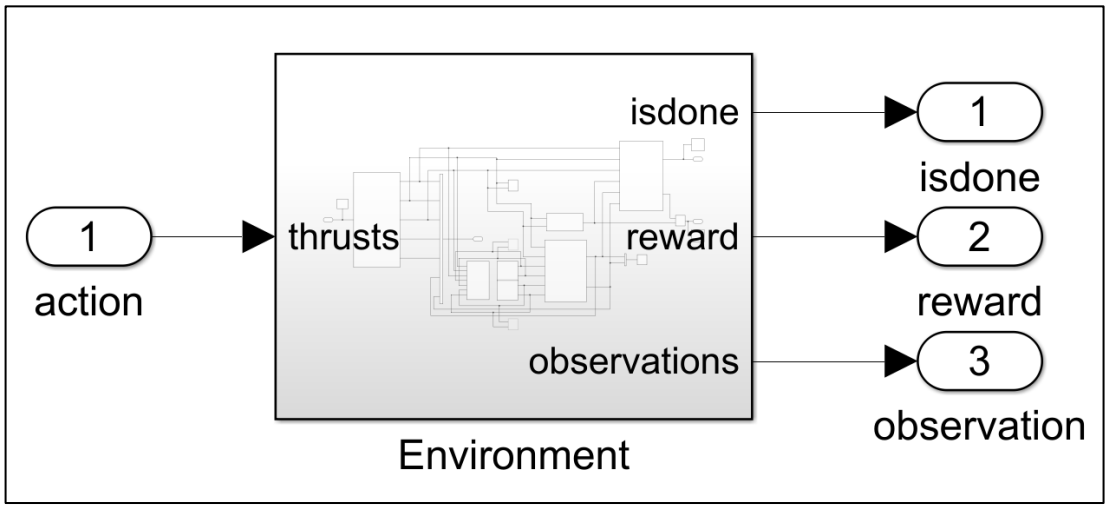


Figura 33. Bloque entorno

En la Figura 34, se definen las observaciones que se asociaran al agente, la recompensa del entorno y las condiciones para finalizar. Al tratarse del control del modelo de un robot móvil, es conveniente observar tanto posición como velocidad en las componentes horizontal y vertical, adicional a la información que se reciba del entorno con respecto a la distancia de los obstáculos.

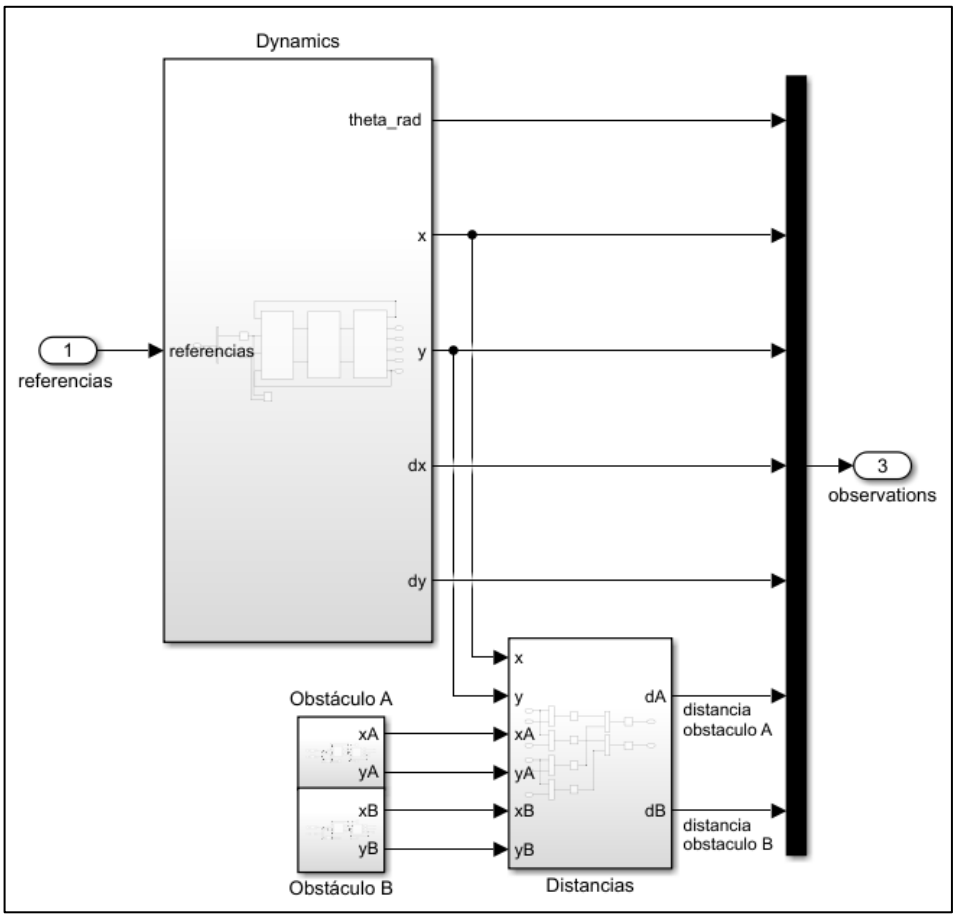


Figura 34. Señales de observación Ceres DDPG

Como se muestra en la Figura 39, se van a penalizar tres acciones puntuales, la primera (Figura 35) penaliza el error con respecto a las coordenadas actuales del robot y de la orientación con respecto a 0, debido a que es el estado terminal del entrenamiento, la segunda (Figura 36) es una penalización por salir del espacio de trabajo del robot es decir el espacio definido entre -10 y 10 para ambos ejes de coordenadas. La penalización final es ejecutada cuando la distancia que se observa a cualquiera de los obstáculos alcanza un valor menor a 80 cm (Figura 37).

Vale la pena aclarar que tanto la segunda y tercera penalización son efectuadas una única vez y son estados terminales del robot, ya que son estados que no se permiten para continuar la navegación (salir del espacio de trabajo y colisionar obstáculos).

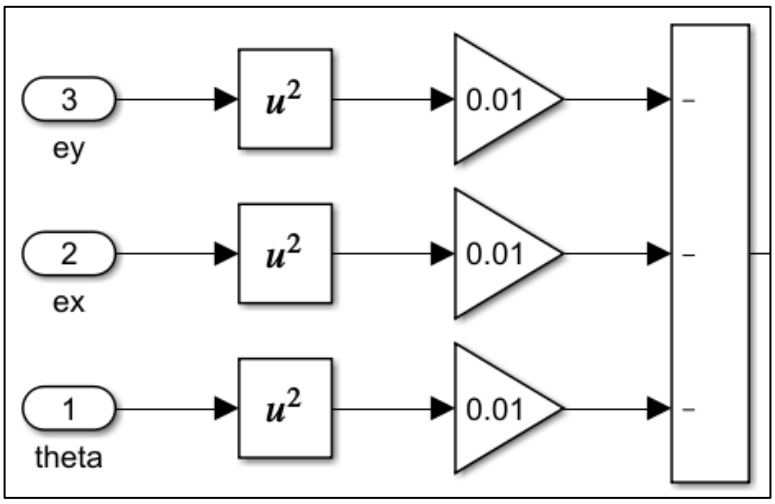


Figura 35. Penalización del error - Ceres DDPG

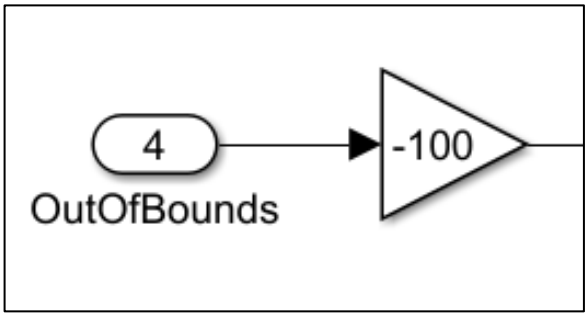


Figura 36. Penalización por salir del espacio de trabajo - Ceres DDPG

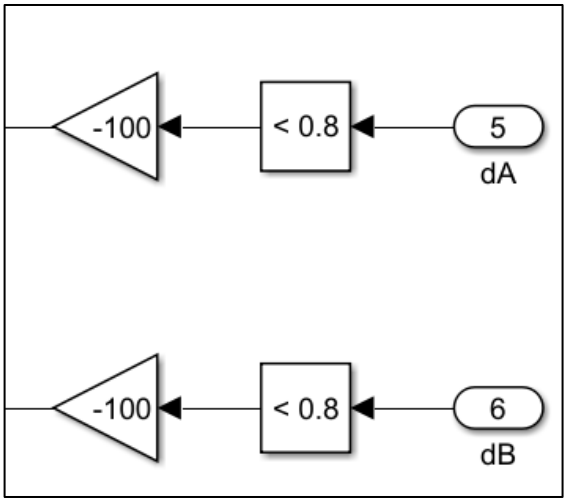


Figura 37. Penalización por cercanía a obstáculos - Ceres DDPG

Para el caso de las recompensas por realizar las acciones deseadas, se establecen dos valores (Figura 38). El primero establece una recompensa por alcanzar valores

de errores de error muy pequeños con respecto a la referencia (0) y el segundo establece una recompensa por alcanzar el estado terminal donde el error es aún más pequeño.

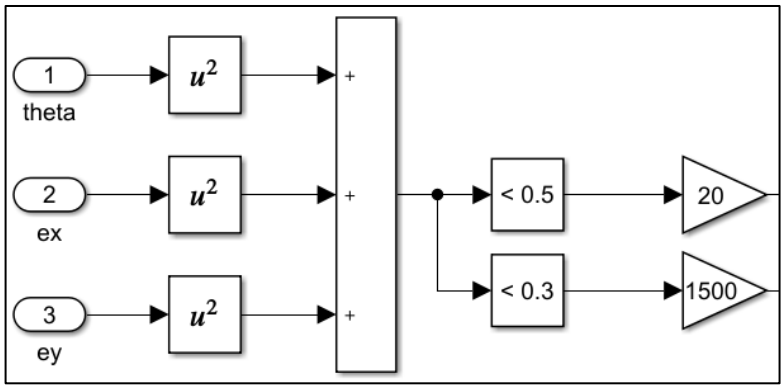


Figura 38. Recompensas - Ceres DDPG

La señal de recompensa completa es entonces la sumatoria de las acciones que se quieren premiar y penalizar:

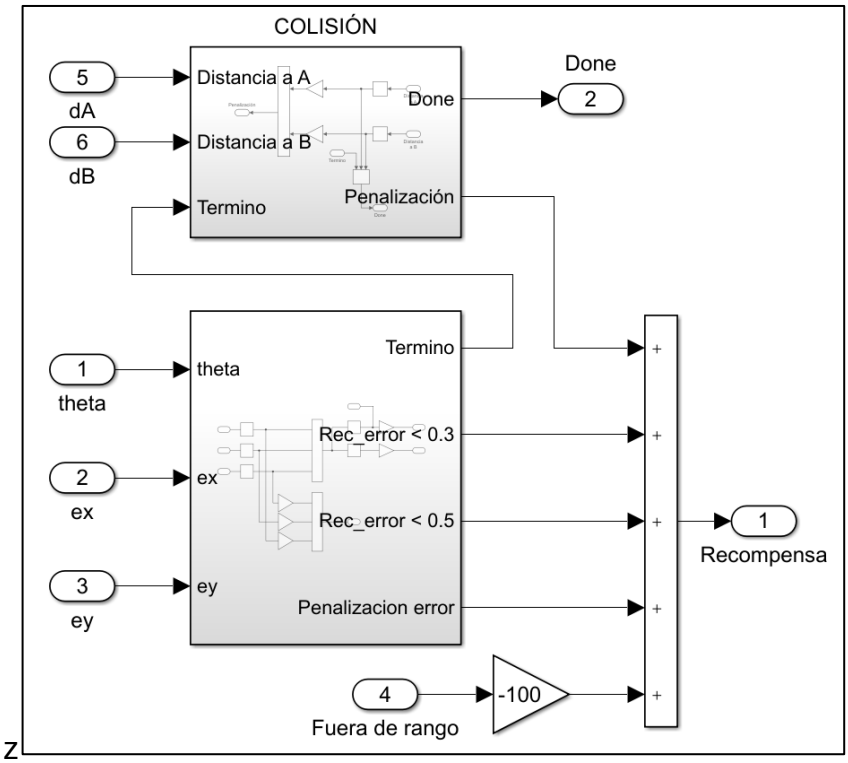


Figura 39. Señal de recompensa - Ceres DDPG

Una vez se han definido los bloques que determinan las observaciones, la señal de recompensa y las condiciones para terminar, en conjunto el sistema se visualiza de la siguiente forma:

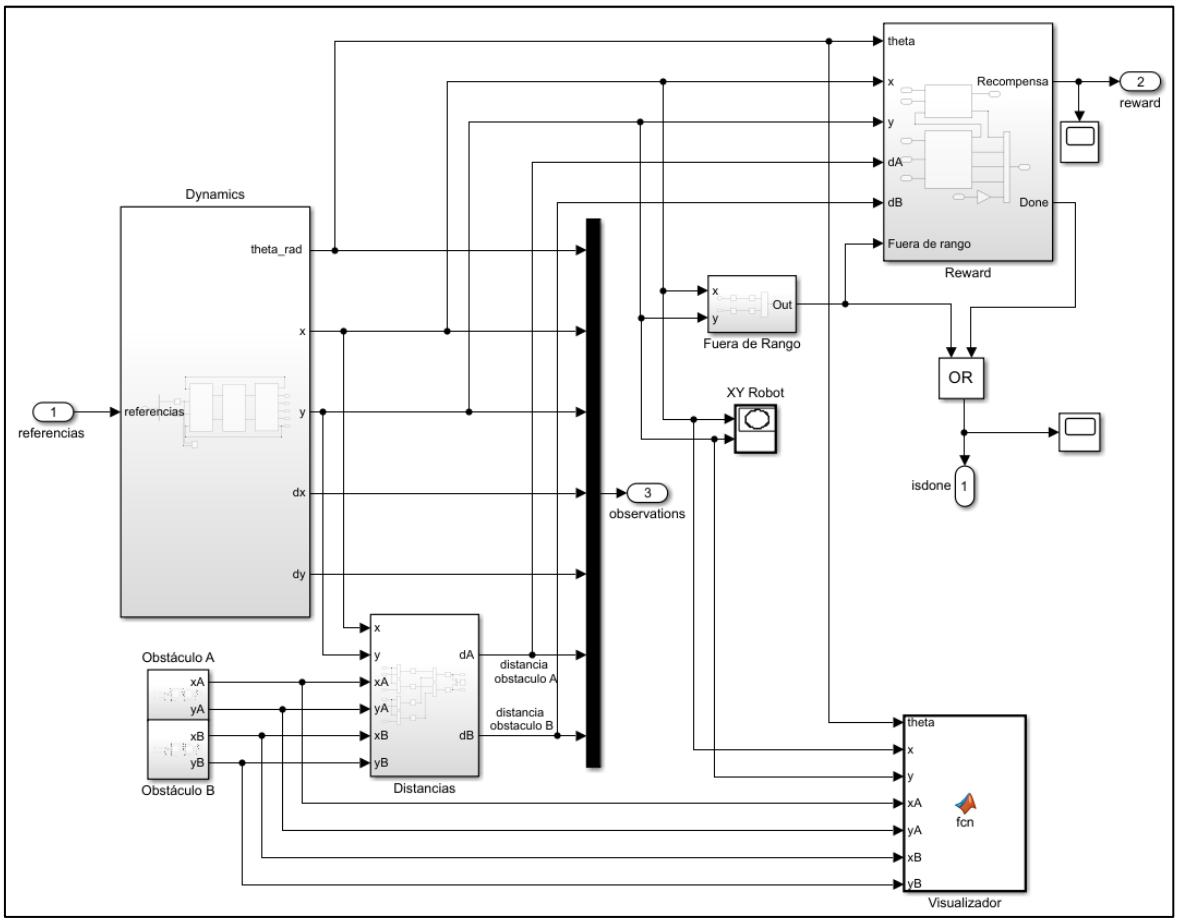


Figura 40. Entorno en Simulink para RL - Ceres DDPG

Luego de que el entorno es creado en Simulink, se debe pasar a configurar el script que determinará los parámetros del agente y permitirá asignar dicho modelo a un bloque RL Agent, para el cual se debe especificar el modelo creado en Simulink con las respectivas observaciones, recompensa y condiciones. El diagrama en la Figura 41 indica el procedimiento a seguir para asociar el modelo con el bloque y realizar su configuración.

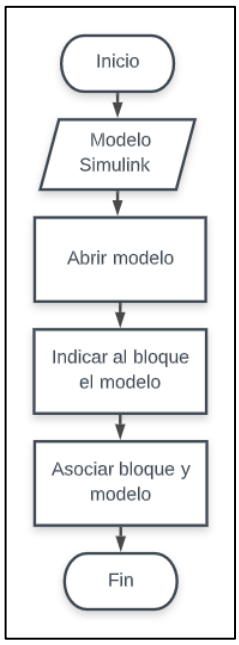


Figura 41. Creación bloque RL Agent - Ceres DDPG

Para definir las observaciones dentro del script, se pide directamente en el script el tamaño de las observaciones contenidas en el modelo Simulink y para efectos prácticos se asigna un nombre que facilitará la visualización en el Workspace si es necesario. Para el caso de las acciones asignadas al agente, el proceso se realiza de manera similar, pero en este caso se deben añadir los límites deseados para aplicar sobre las señales de control tanto en velocidad como en orientación. El modelo asociado con el bloque se muestra en la Figura 42.

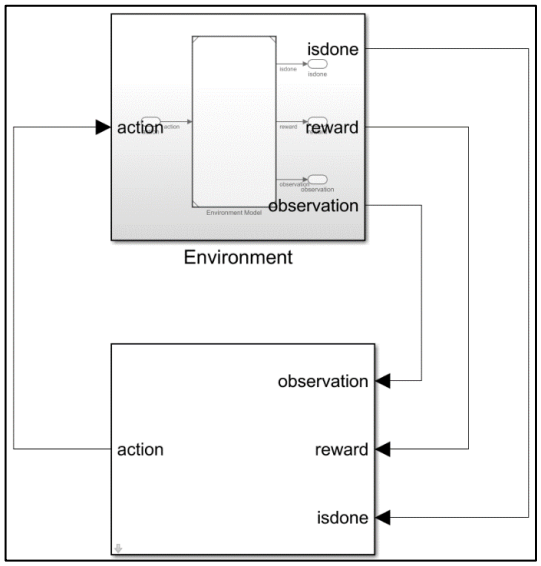


Figura 42. Modelo asociado al bloque RL Agent - Ceres DDPG

Se determina que los límites para la velocidad estarán en un rango de -2 a 2 metros por segundo y para la orientación se determinan entre -π y π radianes, los cuales se deben indicar para terminar la creación del entorno adicional a los valores para el tiempo de muestreo y la duración de la simulación. La creación del objeto que contiene el entorno y lo mencionado con anterioridad se describe en la Figura 43.

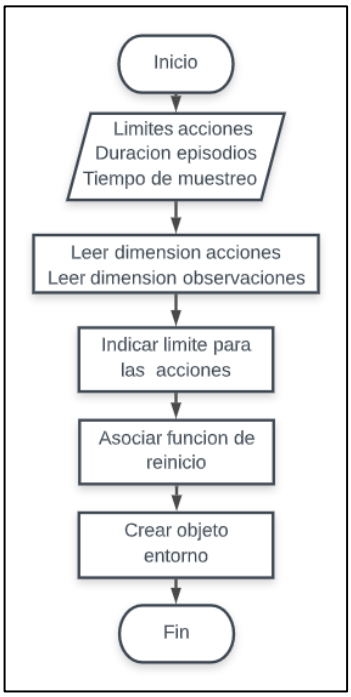


Figura 43. Creación del objeto que contiene el entorno - Ceres DDPG

La función de reinicio está diseñada para generar las condiciones iniciales del robot en posición y orientación, por lo cual genera valores aleatorios entre -9.5 y 9.5 metros para las condiciones iniciales y entre -π y π radianes para la orientación. El valor generado se asigna al integrador que contiene dicha información en el modelo cinemático del robot (Figura 44).

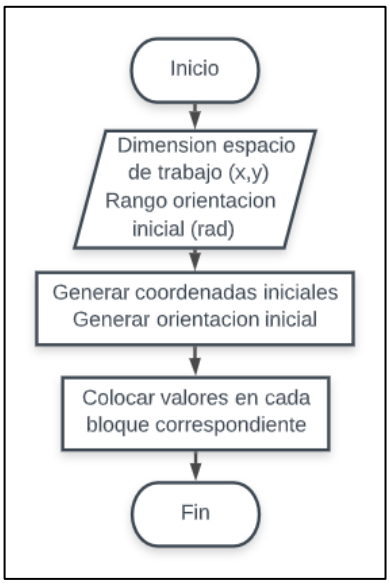


Figura 44. Función de reinicio – Ceres DDPG y DQN

6.3.2 Creación del agente

Como se realizó en el agente para controlar la altura del tanque, en este caso se debe crear la representación del agente mediante redes neuronales, la cual está dividida en crítico y actor [51].

6.3.2.1 Crítico

Como encargado de aproximar la recompensa en el largo plazo o función de valor, el crítico requiere la información en cada momento para observaciones y acciones como se había indicado previamente [52], [53].

Usando la documentación y métodos indicados en [54], a continuación, se muestra cómo se realiza la representación para el crítico:

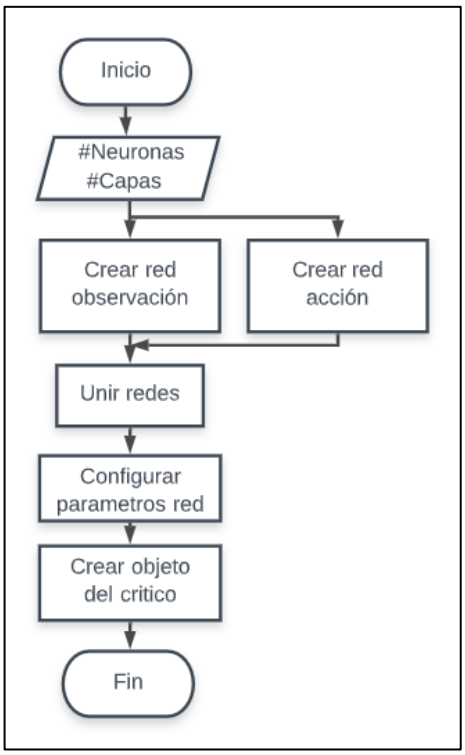


Figura 45. Creación red neuronal para el Critico – Ceres DDPG

Se requiere establecer las observaciones como las acciones en dos caminos independientes de una misma red neuronal, en este caso cada capa cuenta con 80 neuronas cada una, teniendo en cuenta que en casos donde se extrae información de imágenes (mayor complejidad y carga computacional) para navegación, se han utilizado capas de 128, 256 y hasta 512 neuronas [55], [56]. El primer camino (observaciones) define su primera capa donde se toman las observaciones y se pasan por una función de activación relu antes de entrar a la segunda capa, la cual se encargará de mezclar la información proveniente del segundo camino el cual consta de una única capa que lee la información de las acciones ejecutadas.

Como la estimación que se realiza es única y la información que se recibe ya es procesada independientemente en cada camino, solo hace falta unirlos con fin de generar la estimación de una recompensa a largo plazo, para generar una estructura completa como se muestra en la Figura 46.

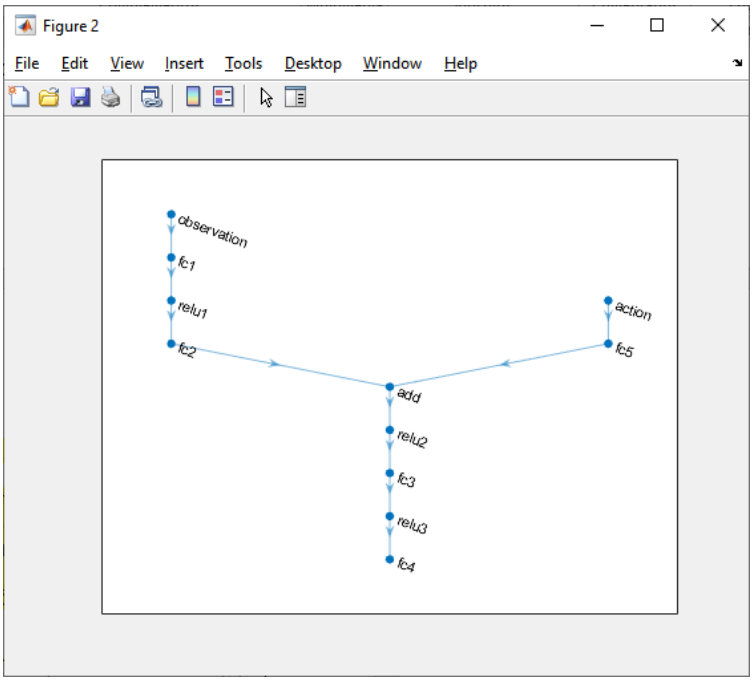


Figura 46. Red neuronal Critico – Ceres DDPG

En cuanto a los parámetros para el aprendizaje de la red neuronal, se establecen la variación del gradiente y un valor pequeño en la razón de aprendizaje (menor a 0.1) para que permita lograr una parametrización de recompensas a largo plazo, al explorar de una manera más minuciosa y en variaciones pequeñas los parámetros que son actualizados constantemente dentro de la red neuronal.

6.3.2.2 Actor

Como se nombró en el ejercicio del tanque del agua, el actor determinará las acciones a tomar en cada momento, para eso se debe crear la representación mediante una red neuronal [51], [57].

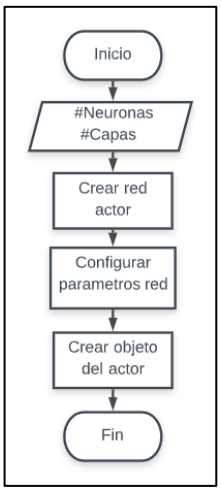


Figura 47. Creación red neuronal para el Actor – Ceres DDPG

Para este caso se utilizará una red neuronal de 3 capas, compuesta por 80 neuronas cada una, con una función de activación relu entre cada una, similar a estructuras expuestas en casos prácticos y que además durante la experimentación mostraron un buen comportamiento con relación al uso de más capas e incluso neuronas representado en la no parametrización de ningún comportamiento o perdida del modelo y en caso de menos capas donde se encontraba una parametrización que conllevaba a colisiones [55], [56]. A la salida se configura una función de activación tanh que permite escalar las señales de control en valores positivos y negativos según corresponda. La estructura completa se muestra en la Figura 48.

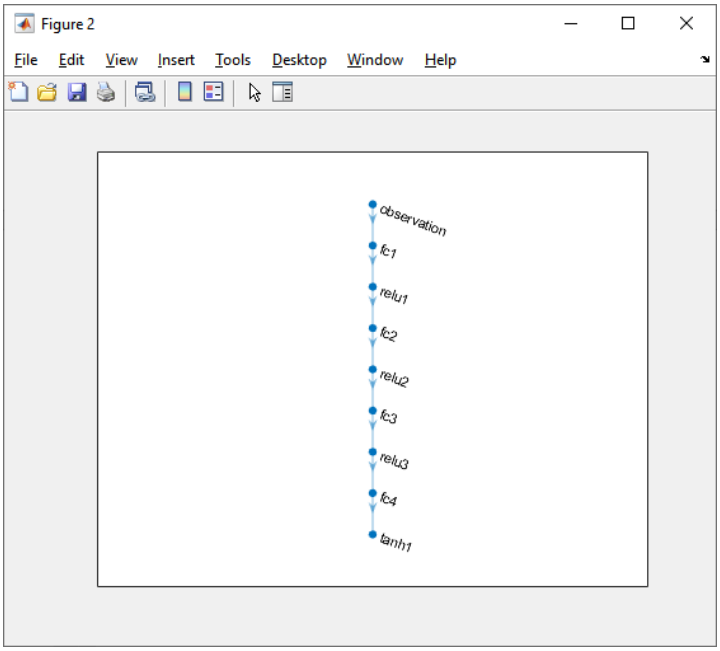


Figura 48. Red neuronal Actor – Ceres DDPG

La configuración de la red completa para la representación del actor sigue los mismos parametros del crítico, la variación se presenta en la creación del objeto que contiene el actor, en la cual se selecciona el nodo de salida de la red neuronal llamado tanh1.

6.3.2.3 Agente

Para establecer totalmente el agente, se deben especificar los parámetros necesarios [58] y hacer la creación del objeto que lo contiene (Figura 49). Se establece nuevamente un buffer de tamaño grande que permite almacenar la totalidad de información respecto a los episodios para evitar correlación entre los datos [59], pero se aumenta el batch de muestreo debido a que en este caso se está trabajando con más información asociada tanto a las acciones y a las observaciones. En el factor de descuento se apuesta por dar un valor muy cercano a 1 para maximizar lo máximo posible las recompensas en el largo plazo [46] y en el factor de suavizado se repite un valor pequeño que ayude a estabilizar el entrenamiento [60].

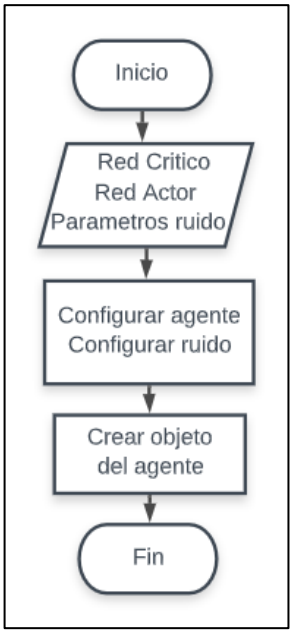


Figura 49. Creación del agente y configuración – Ceres DDPG

La adición de ruido para la exploración se incluye como parámetro dentro del agente antes de crear el objeto que contiene tanto el crítico, el actor, las representaciones y su configuración.

6.3.3 Entrenamiento

Los parámetros configurados para realizar el entrenamiento difieren levemente de los utilizados anteriormente, debido a que al desconocer como se comporte el entrenamiento se desea evidenciar cómo evoluciona en el corto y el largo plazo el establecimiento tanto de las recompensas como de las aproximaciones.

Una vez se colocan los parámetros, se ejecuta la instrucción de entrenamiento, indicando como parámetros el agente, el entorno y las opciones para realizar la iteración del algoritmo [61] según se indica en la Figura 50.

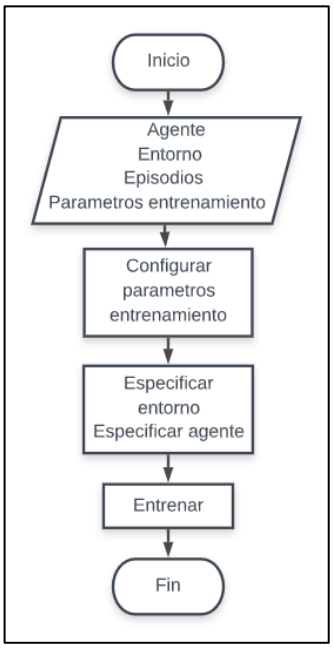


Figura 50. Opciones de entrenamiento – Ceres DDPG

6.3.4 Resultados

La ventana del entrenamiento nos muestra cómo evolucionó el entrenamiento en el corto y en el largo plazo, como se ve en la Figura 51.

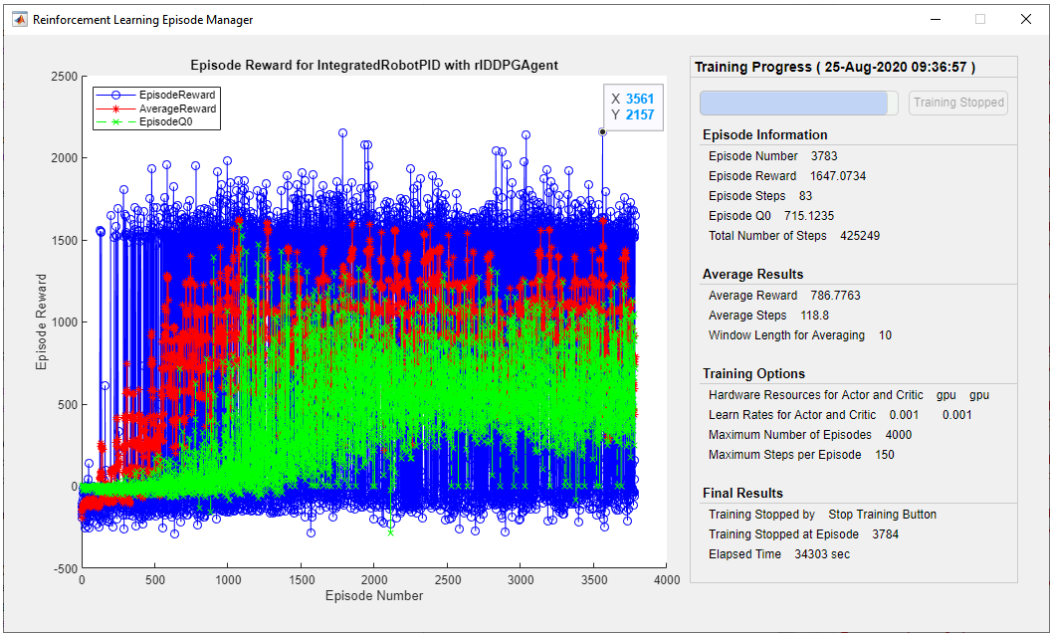


Figura 51. Ventana entrenamiento DDPG Tanque

En la Figura 51 se evidencia, que las recompensas independientes para cada episodio (azul) presentan una dispersión grande a lo largo de todo el entrenamiento y que antes de los 500 episodios la recompensa promedio (roja) presenta un incremento constante, hasta tener cierta estabilidad en una media de 750. Aproximadamente a partir del episodio 1000 la estimación de recompensas futuras (verde) presenta un crecimiento hasta estabilizarse en una media de 600 al final del entrenamiento.

Para la selección de un agente, se realizan pruebas con los episodios independientes (azul) en que las recompensas alcanzaron un valor superior a 1800, en este caso el episodio con la recompensa más alta en el entrenamiento (Figura 51) se encuentra cerca de un valor pico de la recompensa promedio (rojo), por lo cual se toma para realizar la validación

6.3.5 Validación

La validación del entrenamiento se realiza directamente sobre Simulink siguiendo el procedimiento de la Figura 52, indicando las opciones de simulación indicado el

agente que va a ser utilizado, colocando el nombre correspondiente, en este caso fue almacenado con el nombre del episodio correspondiente. Una vez se carga el agente, se ejecuta la validación y se espera la ejecución para ver la simulación dentro del entorno.

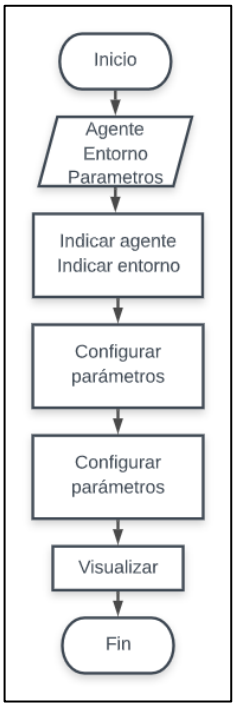


Figura 52. Proceso validación agente para tanque con DDPG

Una vez se realiza todo el proceso para ejecutar el agente y visualizar su comportamiento, se pueden revisar las ventajas que contienen las trayectorias tanto del robot y de los obstáculos para verificar como fue el comportamiento respectivo.

Como se observa en la Figura 53, el agente realiza la navegación desde un punto inicial hacia el objetivo (origen), realizando la evasión de la trayectoria de los obstáculos generados como se muestra en la Figura 54. En la Figura 55 se evidencia que el robot presenta una pequeña diferencia entre la posición final y la posición objetivo (52 centímetros), la cual se debe a la permitividad que se otorgó en las recompensas, ya que la recompensa es alta por llegar a zonas cercanas al origen.

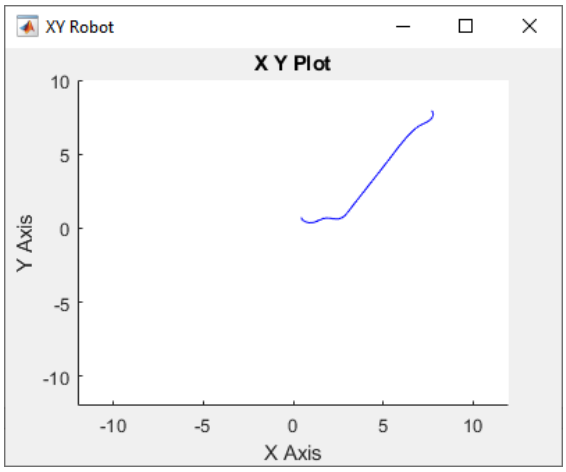


Figura 53. Trayectoria de la navegación

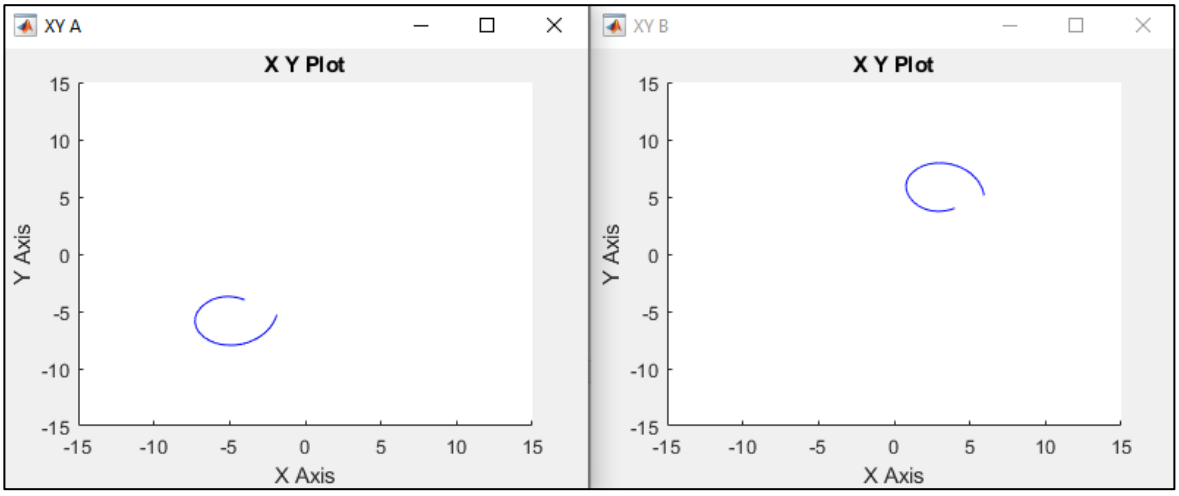


Figura 54. Trayectoria Obstáculos

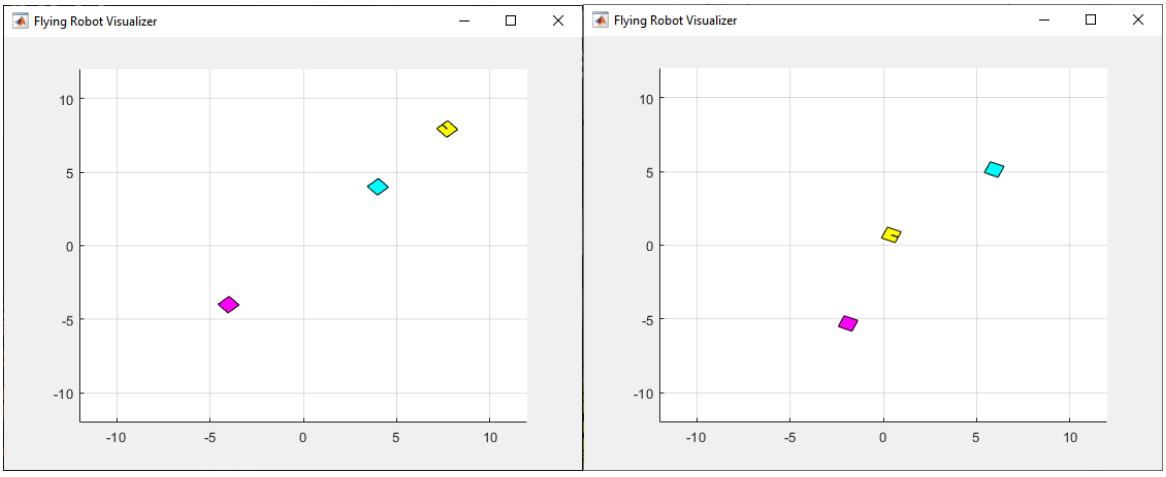


Figura 55. Posición inicial y final robot con obstáculos

En cuanto a las señales de control, se revisa el comportamiento de la referencia entregada por el agente al controlador PID con la orientación que este logra colocar en la plataforma (Figura 56) y lo mismo se revisa para la referencia en velocidad (Figura 57).

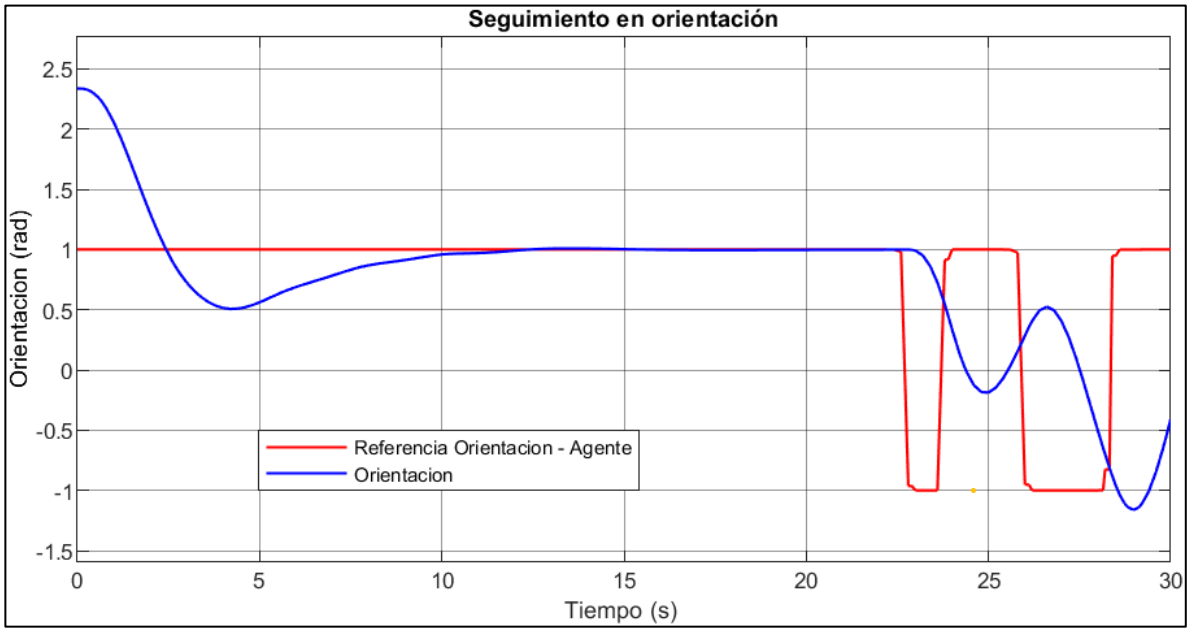


Figura 56. Seguimiento a orientación generada por agente - DDPG

Para la orientación (Figura 56), se evidencia que no hay una gran cantidad de variación en la referencia otorgada para el agente, pero cuando se presenta es demasiado brusca. En el intervalo de tiempo de 20 a 30 segundos se evidencia dicha variación en la referencia (Rojo) al oscilar entre valores de 1 rad a -1 rad (-100%), tal variación intenta ser seguida por el controlador PID, sin un total éxito induciendo señales de error y sobre impulsos que varían en el rango de un 20% a un 50%.

Vale la pena hacer la aclaración que, si bien las señales de referencia entregadas por el agente controlador no son satisfactorias en tiempo de respuesta y estabilidad para el control PID, debido a la inducción de errores y sobre impulsos, siguen siendo funcionales para cumplir el objetivo de evadir los obstáculos y realizar la navegación del agente. Dicha problemática es resuelta más adelante mediante la discretización de las señales en referencia para la velocidad angular.

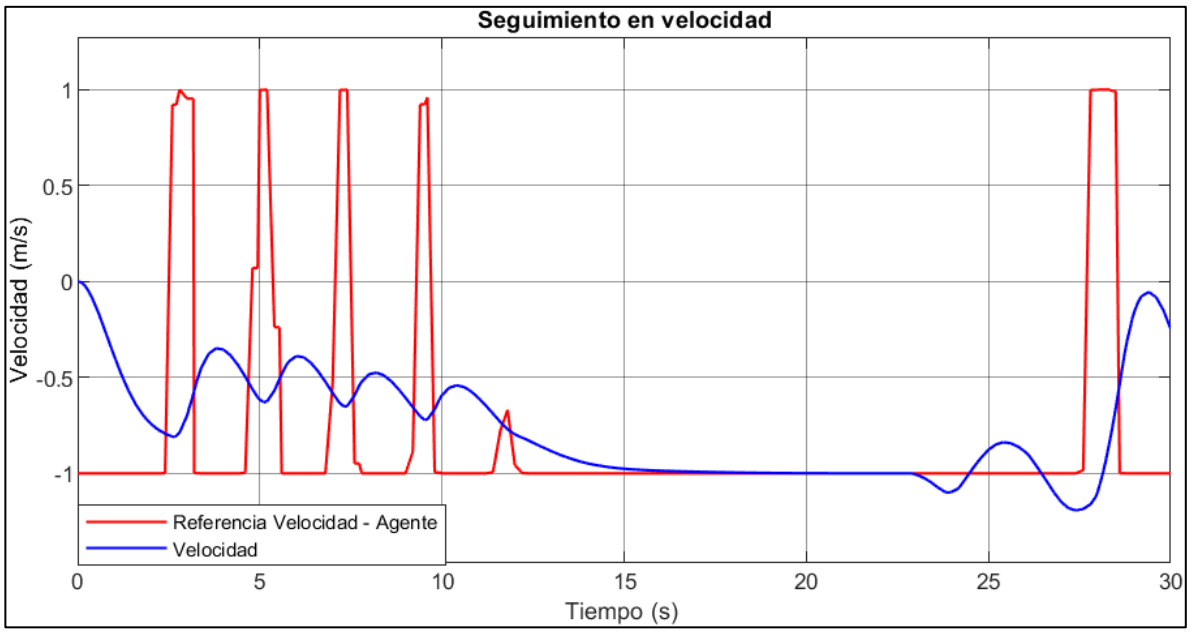


Figura 57. Seguimiento a velocidad generada por agente - DDPG

En el caso de la velocidad (Figura 57), se presenta un comportamiento similar al de la orientación, puesto que las variaciones de la referencia (rojo) son de alrededor de 2m/s (-100%) y se presentan en mayor frecuencia, generando que el controlador PID no tenga el tiempo suficiente para alcanzar un estado estable en referencia cuando ya debe tomar nuevas acciones de control para estabilizar en un valor diferente.

Adicionalmente se toma el agente y se somete a una prueba, donde se le ejecuta 100 veces dentro de la simulación para determinar de una mejor forma su precisión en llegar a la meta evitando colisiones. Se evidencian tres casos de finalización, el cumplimiento de la misión, el no cumplimiento o pérdida y la colisión dentro del entorno.

Tabla 1. Validación agente Ceres DDPG

# Prueba	OK	Cerca	Colisión A	Colisión B	Perdida
TOTAL	69	16	4	3	8

En la Tabla 1, se muestra en cada casilla la cantidad de veces que el vehículo termino satisfactoriamente la prueba, los casos en que termino cerca y se da por terminado el episodio, así como los casos en que se estrelló con el obstáculo A o B y finalmente las ocasiones en que se perdió. Los porcentajes de pérdida (8%) y colisión con algún obstáculo (7%) le dan espacio a un 85% de cumplimiento de la

misión, realizando la evasión y la navegación ya sea hasta el origen o a posiciones cercanas alrededor de 80 centímetros alrededor.

6.4. Implementación modelo dinámico CERES con control PID para navegación usando DQN

En esta ocasión, se trabajará sobre el mismo modelo dinámico con controlador PID, pero implementado sobre un entorno que contiene la geometría de la navegación que se realizaría dentro de un cultivo. Para tal se hará uso del algoritmo para aprendizaje por refuerzo DQN con el objetivo de simplificar las referencias para el control a las mínimas posibles, pero que permita cumplir el objetivo de alcanzar un punto final sin colisionar en el trayecto.

Deep Q-Network – DQN Agent

En este caso se usará un agente de aprendizaje por refuerzo conocido como Deep Q-network o DQN, el cual consiste en model-free, online y off-policy al igual que los dos agentes utilizados con anterioridad. Nuevamente el model-free permite una mejor navegación en un entorno desconocido, ajustando el aprendizaje continuamente mediante un método online y al ser un algoritmo off-policy permite las acciones a tomar son determinadas independientemente de la función de comportamiento aproximada [34], [38], [44].

Previo a establecer el agente, se crea un script que contiene un vector de coordenadas rectangulares el cual determina los bordes del camino en el que el agente va a realizar la exploración, dicho script retorna el vector completo para graficar el entorno y la distancia al punto más cercano del camino al robot (Figura 58).

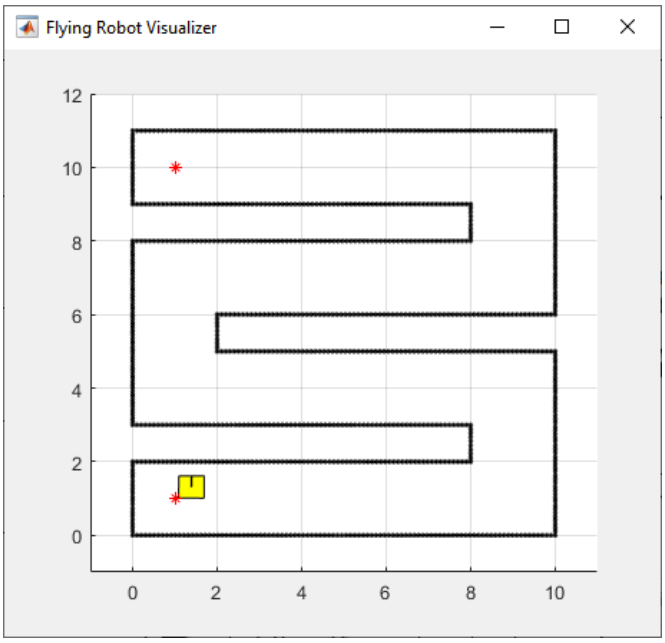


Figura 58. Camino para la navegación del agente

6.4.1 Creación del entorno

Como se realizó en ocasiones anteriores, el primer punto es definir las observaciones, la recompensa, las condiciones para detener entrenamiento y las acciones de control. Las observaciones definidas para este agente son similares a las indicadas para el agente DDPG en la Figura 34, con la variación de que no se indica la distancia a cada uno de los objetos, sino que se indica la distancia al punto del camino más cercano al robot y una referencia en orientación para cada posición en el camino (Figura 59).

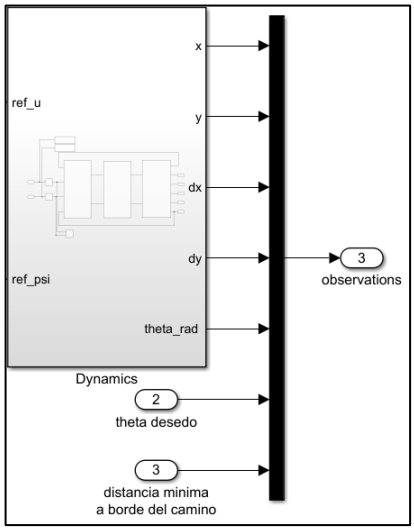


Figura 59. Señales de observación agente – Ceres DQN

Para las recompensas, se establecieron varios bloques independientes que se encargan de penalizar o recompensar según un parámetro determinado. La señal de recompensa completa es entonces la sumatoria de las acciones que se quieren premiar y penalizar como se evidencia en la Figura 60 y se explica posteriormente.

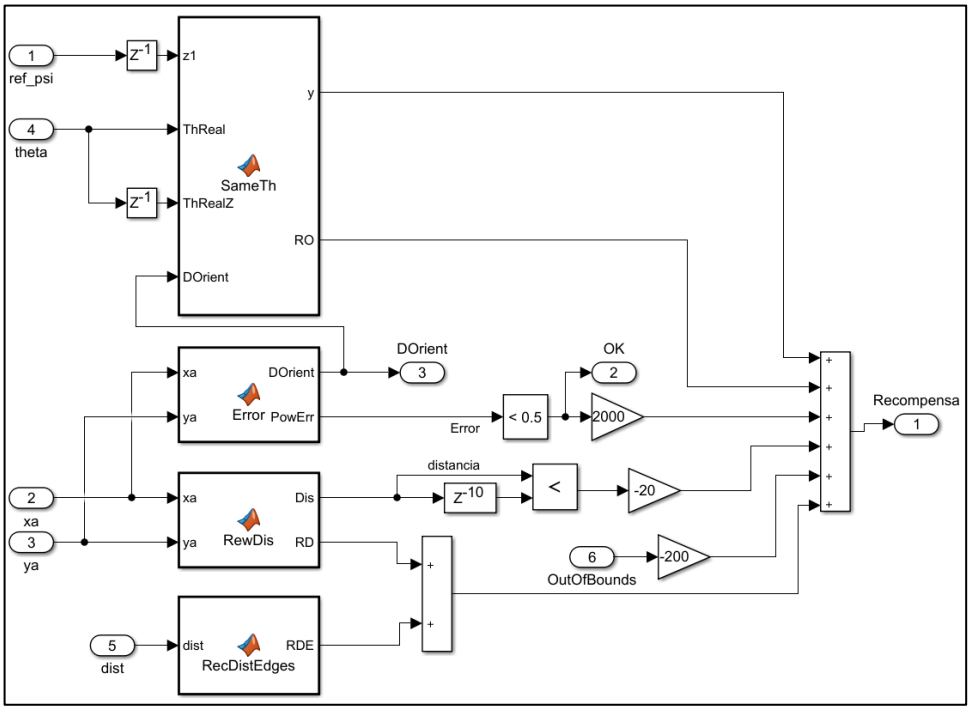


Figura 60. Señal de recompensa DQN – Ceres DQN

El primer parámetro para las recompensas es la distancia a los bordes del camino en el bloque RecDisEdges de la Figura 60, en su interior establece una penalización para distancias menores a 60cm, donde distancias inferiores a 40cm serán penalizadas más severamente, para los valores de distancia mayores, se establece una recompensa la cual se ve incrementada cuando esta distancia supera el valor de 70cm. En la Figura 61 se muestra más claramente el proceso para seleccionar la recompensa o penalizar asignándolo a la variable RDE.

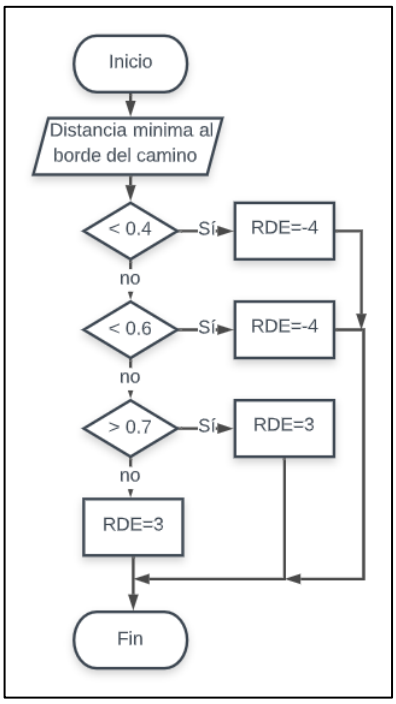


Figura 61. Recompensa por distancia a bordes – Ceres DQN

El segundo parámetro para la recompensa es la distancia recorrida desde el punto inicial, la cual se calcula dentro del bloque RewDis (Figura 60) y es penalizada con un valor de -1 hasta no alcanzar una distancia mayor a 3m, donde progresivamente se otorga un porcentaje de la distancia recorrida como recompensa según la zona en que se encuentra el robot como se indica en la Figura 62, para las zonas que no se indica el porcentaje, se otorga un 40% de la distancia recorrida

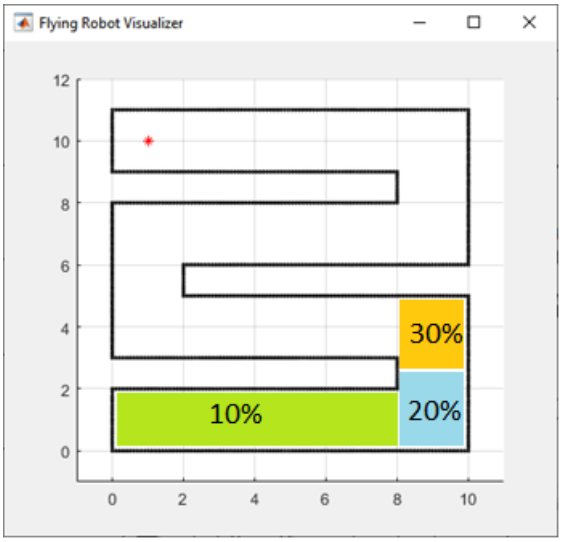


Figura 62. Porcentaje otorgado por distancia recorrida

El bloque Error (Figura 60), no realiza ningún tipo de penalización ni otorga recompensa alguna, pero se encarga de calcular el error actual respecto al punto final para determinar cuando la exploración debe terminar al llegar al estado terminal, adicionalmente se encarga de establecer una referencia en orientación para cada zona del camino como se muestra a continuación:

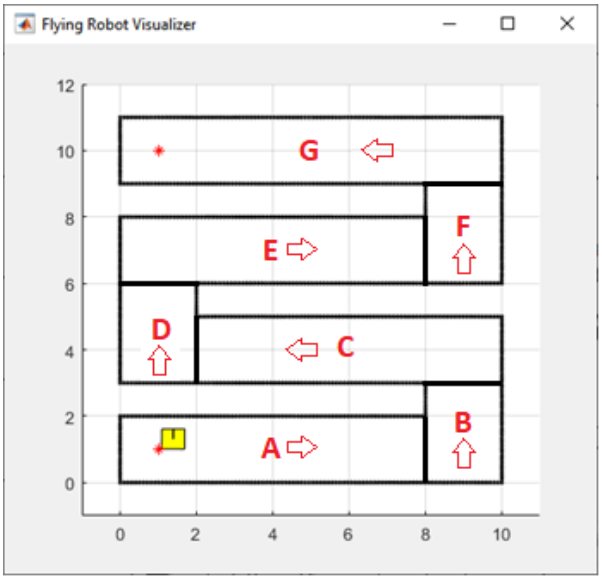


Figura 63. Orientación deseada para cada zona

Con la orientación establecida según la Figura 63, el siguiente bloque SameTh (Figura 60), se encarga de calcular la diferencia entre el valor actual de la orientación del robot y otorgar una recompensa. Cuando la diferencia es menor a 0.18 rad (10° aproximadamente) se otorgará una recompensa por seguir la referencia y se penaliza en el caso de que sea mayor, debido a que es un error de orientación pequeño en comparación a la diferencia entre referencias (1.57 rad – 90°). Adicionalmente cuando el agente genera referencias en orientación sucesivas con el mismo valor también es recompensado para evitar variaciones en la señal de referencia para la orientación.

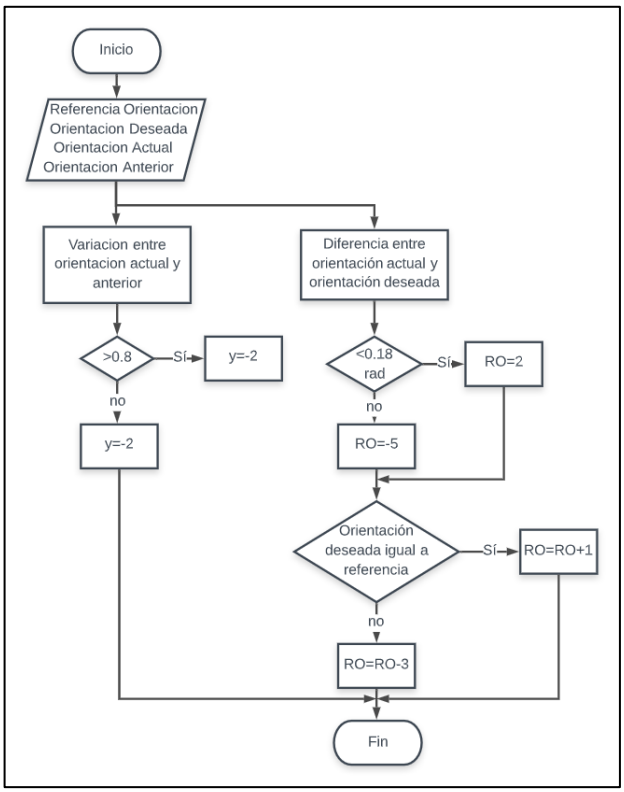


Figura 64. Recompensas por seguimiento a referencias

En la Figura 64 se muestra el proceso para otorgar la recompensa por seguir la orientación deseada (RO) y la recompensa por evitar variaciones bruscas entre cada paso para la orientación del robot (y).

Adicionalmente, se penaliza la distancia actual del robot y su distancia hace 10 periodos de tiempo atrás, equivalente a un tiempo de dos segundos, tiempo suficiente para establecer si el robot está avanzando o retrocediendo, con el fin de que el agente no se dedique a explotar una misma zona recorriéndola sucesivamente y regresando a ella (Figura 65).

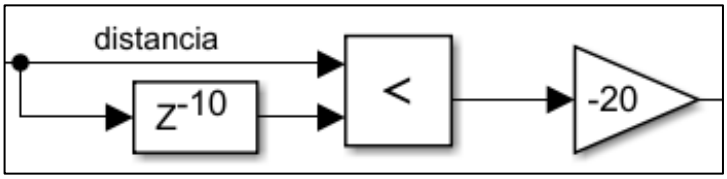


Figura 65. Penalización por explorar la misma zona

Las condiciones para finalizar el episodio son por colisionar con alguno de los bordes del camino o salirse del espacio de trabajo, asignándole una penalización y

una tercera condición de finalización por alcanzar el estado terminal del entrenamiento con su respectiva recompensa.

Una vez se han definido los bloques que determinan las observaciones, la señal de recompensa y las condiciones para terminar, en conjunto el sistema se visualiza de la siguiente forma:

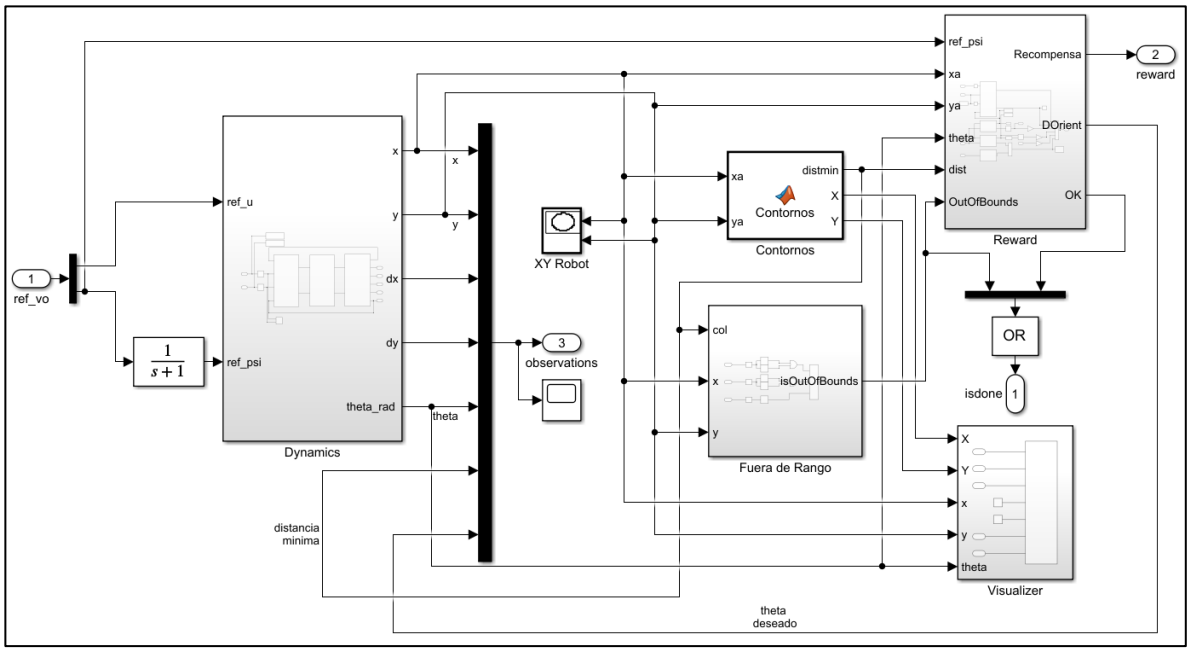


Figura 66. Entorno en Simulink para RL – Ceres DQN

Sobre el diagrama (Figura 66) sobresale una función de transferencia entre el bloque Dynamics y la señal 1 (ref_vo), el cual correspondiente a un filtro pasa bajas de primer orden, que se aplica a la señal de referencia en orientación. El filtro se añade con el objetivo de suavizar las transiciones entre referencias, ya que al simplificar el sistema a 4 referencias en orientación (puntos cardinales) tal transición puede representar una señal de control brusca en el controlador, generando picos y oscilaciones no deseadas.

Una vez el entorno en Simulink está listo, se procede a configurar el modelo dentro de un script, donde se definen los parámetros necesarios.

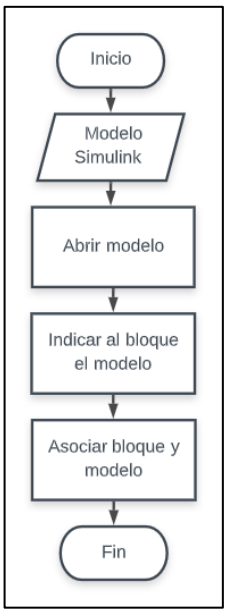


Figura 67. Creación bloque RL Agent – Ceres DQN

En la Figura 67, se abre directamente el modelo y se asocia de manera directa a un bloque RL Agent que contendrá toda la información establecida en el modelo de Simulink. Para definir las observaciones, se especifica la dimensión dentro de los parámetros correspondientes, así mismo se especifican las acciones en pares, donde el primer componente indica la referencia en velocidad lineal y la segunda orientación en radianes (Figura 68).

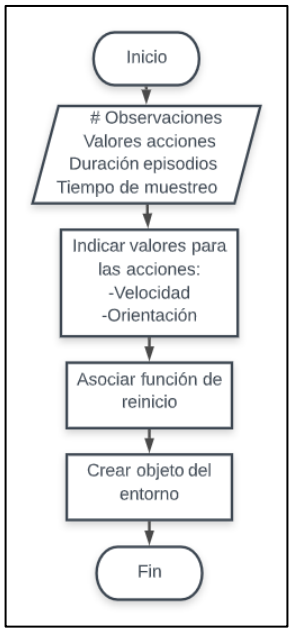


Figura 68. Creación del entorno – Ceres DQN

Finalmente, antes de crear el objeto (Figura 68) se debe especificar la función de reinicio para cada episodio terminado. La función de reinicio será la misma que la del ejercicio anterior y que se muestra en la Figura 44, donde se generan las coordenadas de inicio deseadas y una orientación inicial aleatoria para el robot.

6.4.2 Creación del agente:

El algoritmo DQN, es una variante del algoritmo Q-Learning usado para la navegación en la cuadricula del primer ejercicio, es decir que emplea únicamente un Critico que estima el valor de futuras recompensas. A diferencia de la representación utilizada para el Q-Learning que consistía en la creación de una matriz que relacionaba los estados y las acciones, la representación en este caso se realiza mediante una red neuronal para el crítico que permite solucionar el problema de lidiar con entradas de varias dimensiones además de la no estabilización del entrenamiento utilizando un muestreo de experiencias pasadas [64].

6.4.2.1 Critico

El algoritmo en este caso utiliza dos aproximadores el primero se encarga de estimar la función de valor o la recompensa estimada mediante el crítico (Q) y el segundo, el objetivo critico (Q') se encarga de mejorar la estabilidad del entrenamiento mediante la actualización periódica de los parámetros del crítico [52] , [65].

La representación del crítico (Figura 69) está compuesta por dos caminos independientes, el primero se encarga de tomar toda la información de las observaciones en el estado actual y pasarla por dos capas compuestas por 80 neuronas, tomando referencia casos donde para procesas imágenes con el mismo objetivo, se han utilizado capas de 128, 256 y hasta 512 neuronas [55], [56]. El segundo camino toma la información de las acciones que están siendo ejecutadas para pasarlas por una única capa antes de unir la información, donde nuevamente la información es procesada en dos capas con funciones de activación relu en intermedio.

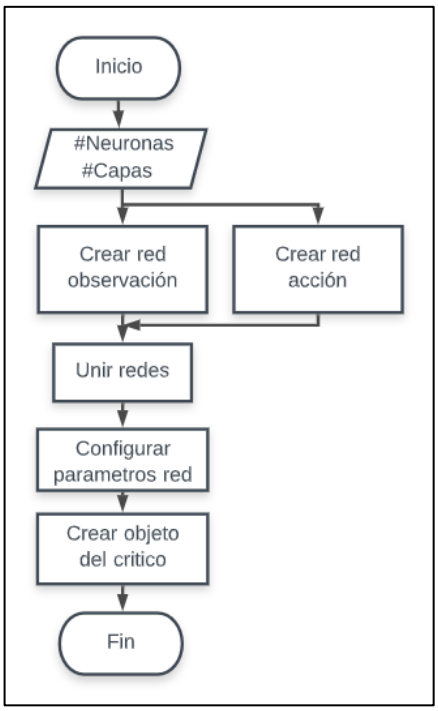


Figura 69. Creación red neuronal para el Critico – Ceres DQN

La representación creada para el crítico, se valida graficando la estructura que la compone, la cual es la siguiente:

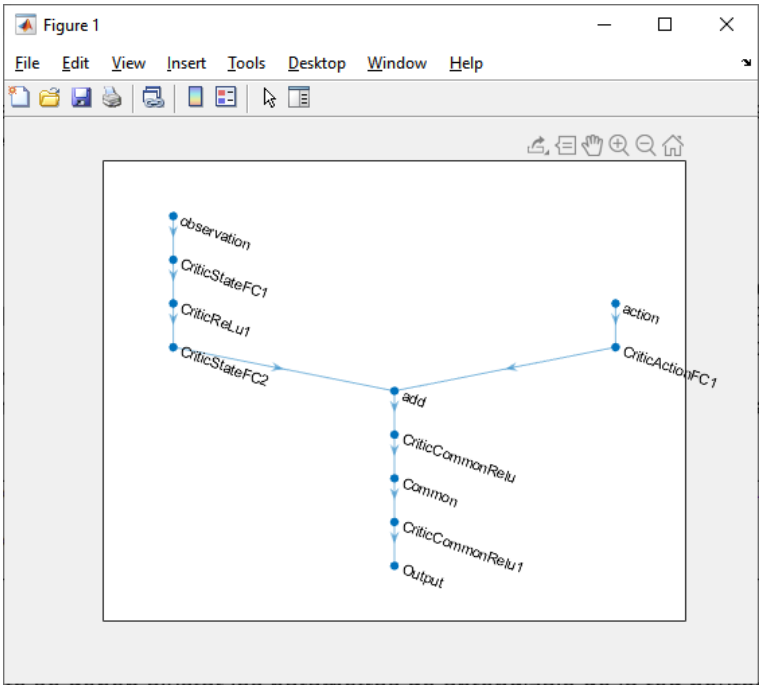


Figura 70. Red neuronal Critico - Ceres DQN

La configuración de los parametros de la red neuronal mantiene los parametros utilizados en el entrenamiento DDPG, debido a los resultados obtenidos.

6.4.2.2 Agente

Como se realizó en los ejercicios anteriores, el paso final consiste en configurar los parametros del agente que serán empleados durante el entrenamiento para actualizar el crítico y sus parametros [58].

El buffer de experiencia se configura en un valor grande que permite en este caso almacenar todos los episodios del entrenamiento para evitar un sobre entrenamiento, el muestreo para actualizar los valores dentro del algoritmo se toma de un batch de 128 muestreando del buffer, el cual es un valor mayor al utilizado en utilizado en otras implementaciones prácticas, pero al no tener problemas de procesamiento se puede establecer en un valor más grande, mejorando el entrenamiento [64], [66].

El factor de descuento se acerca al valor de 1 para dar prioridad a recompensas estimadas al largo plazo y el valor pequeño en el factor de suavizado asegura un entrenamiento lento, pero con mejor estabilidad y resultados [46], [60].

6.4.3 Entrenamiento

En la Figura 71, se indica el procedimiento para ejecutar el entrenamiento, se indica la cantidad de episodios, los pasos dentro de cada episodio y las condiciones para dar el entrenamiento como finalizado.

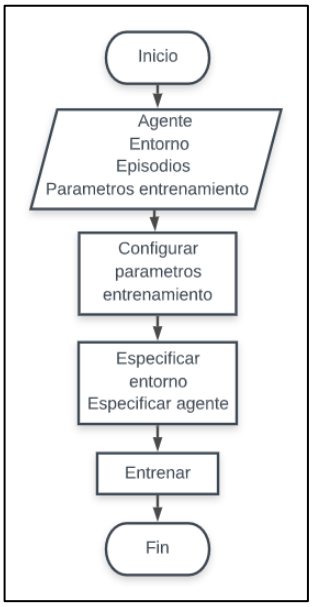


Figura 71. Opciones de entrenamiento – Ceres DQN

Internamente el algoritmo que está siendo ejecutado realiza lo siguiente [65]:

Antes de comenzar, se inicializa el crítico (Q) con valores iniciales aleatorios para el estado y la observación (S,A) y sus parámetros (θ_Q), los mismos valores se deben inicializar para los parámetros de la función objetivo ($\theta_{Q'}$).

Para cada episodio se realiza la observación del estado (S), también se toma el valor de la acción (A) y se ejecutan iterativamente los siguientes pasos:

1. Para la observación actual (S), se selecciona aleatoriamente una acción (A) con la probabilidad indicada por la exploración épsilon-greedy, si no se cumple dicha probabilidad se selecciona la acción que el crítico determina como la mejor:

$$A = \max \mathrm{Q}\,(S, A) \tag{9}$$

   El valor en este caso para épsilon se tomará como 1, para permitir una exploración totalmente aleatoria del entorno, el cual ira decayendo progresivamente en una taza de 0.005 al final de cada entrenamiento, lo cual establece un nuevo valor para cada entrenamiento de:

$$Epsilon = Epsilon\,(1 - EpsilonDecay) \tag{10}$$

2. Ejecutar la acción (A) determinada, luego observar la recompensa (R) y la nueva observación (S') que será el nuevo estado.

3. Guardar en el buffer de experiencia los valores observados (S, A, R, S')

4. Si S' coincide con algún estado terminal, la función objetivo (y) toma el valor de R y el episodio se termina, si no coincide la función toma el siguiente valor:

$$y = R + \gamma \max\ \mathrm{Q}(S, A) \tag{11}$$

   Donde:

   - γ es el factor de descuento configurado anteriormente, entre mayor es su valor las futuras recompensas toman más importancia en el entrenamiento [46].

5. Calcular los parámetros del crítico, usando la función de perdida (L), que reduce progresivamente el error entre el comportamiento actual y el

comportamiento deseado [62]. Donde M es el tamaño del buffer de experiencia

$$L = \frac{1}{M}\sum_{i=1}^{M}(y_i - Q)^2 \tag{12}$$

6. Se aplica el factor de suavizado a los nuevos parametros del crítico:

$$\theta_{Q'} = \tau\theta_Q + (1-\tau)\theta_{Q'} \tag{13}$$

- Donde τ es el factor de suavizado configurado anteriormente, que asegura una variación pequeña entre cada episodio, lo cual optimiza encontrar parametros adecuados para el comportamiento.

7. Actualizar el valor de épsilon para el nuevo episodio de entrenamiento usando la ecuación descrita en el primer paso.

6.4.4 Resultados

En el entrenamiento, se coloca un valor que detiene el entrenamiento al alcanzar un valor promedio de 6500 en los últimos 10 episodios, el cual se alcanza cuando en la exploración del entorno, el agente encuentra un comportamiento que le permite llegar a la posición final. El alcanzar dicho estado terminal le permite asociar la experiencia obtenida para repetir el proceso en futuros episodios.

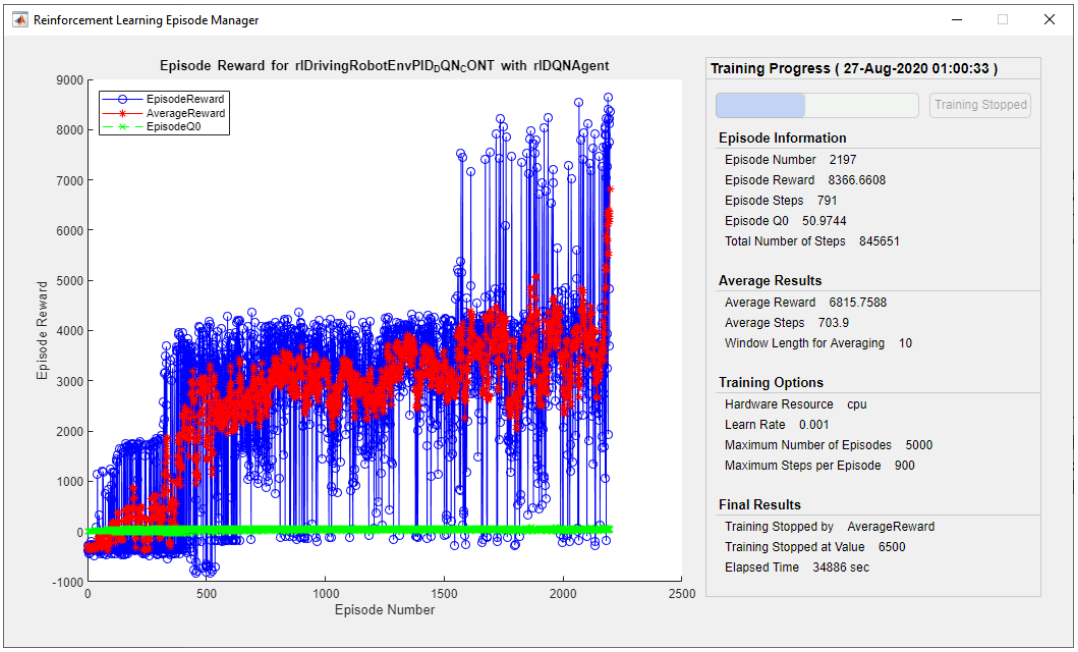


Figura 72. Ventana entrenamiento DQN

En la Figura 72, se evidencia como los valores de la recompensa aumentan progresivamente hasta el punto en que el entrenamiento es finalizado, ya que se determina que el agente ha parametrizado el comportamiento necesario para mantenerse en una distancia prudente de los bordes del camino y ha logrado alcanzar el estado terminal para la posición final.

6.4.5 Validación

La validación del entrenamiento se realiza directamente sobre Simulink, donde se pueden visualizar todas las señales del entrenamiento para el agente entrenado. A continuación, se muestra la trayectoria seguida por el vehículo:

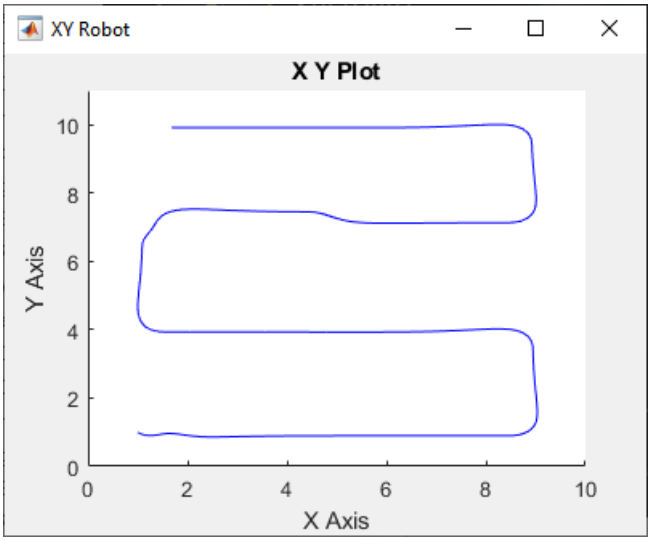


Figura 73. Trayectoria del vehículo

A continuación, se revisa el comportamiento de la señal de referencia en orientación establecidas por el agente luego de pasar por un filtro pasa bajas, con el objetivo de suavizar el comportamiento del robot. Se establecieron como referencias 4 orientaciones correspondientes a los puntos cardinales, debido a que permitió una parametrización más estable en comparación al agente DDPG, donde el rango abierto de orientaciones le confirió señales de referencia que generaban oscilaciones y sobre impulsos no deseados por parte del controlador PID.

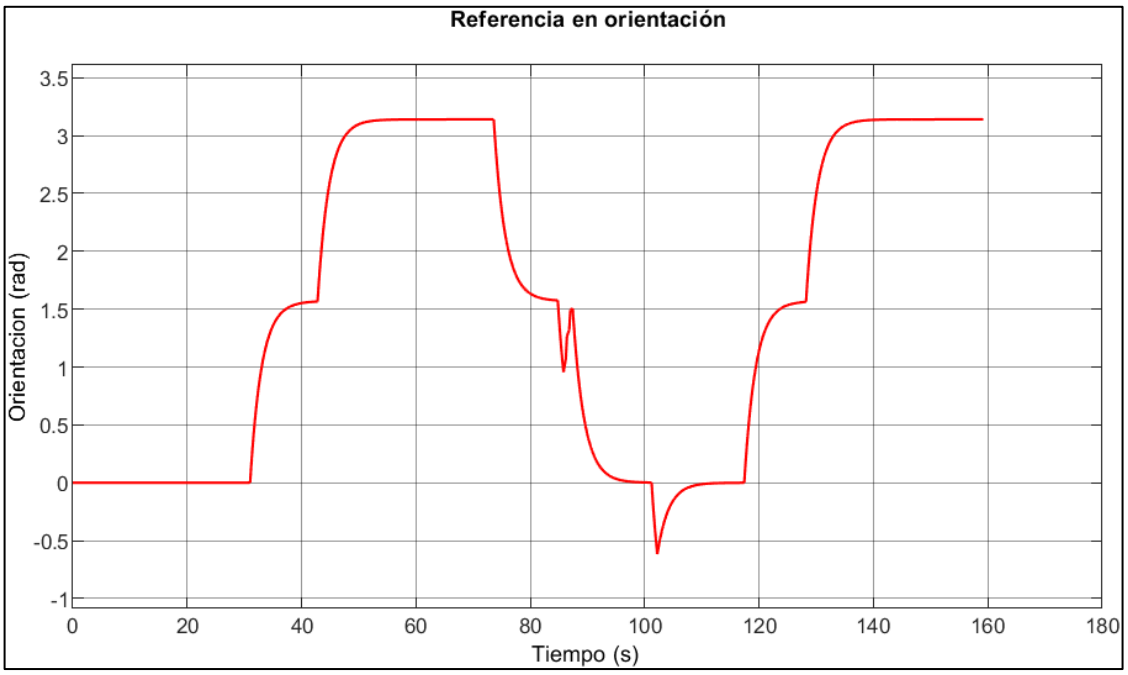


Figura 74. Referencia en orientación por DQN

La señal de referencia establecida en la Figura 74, debía realizar seguimiento de una orientación deseada, según la posición del robot y la dirección en que se debía desplazar. A continuación, se evalúa como fue el seguimiento:

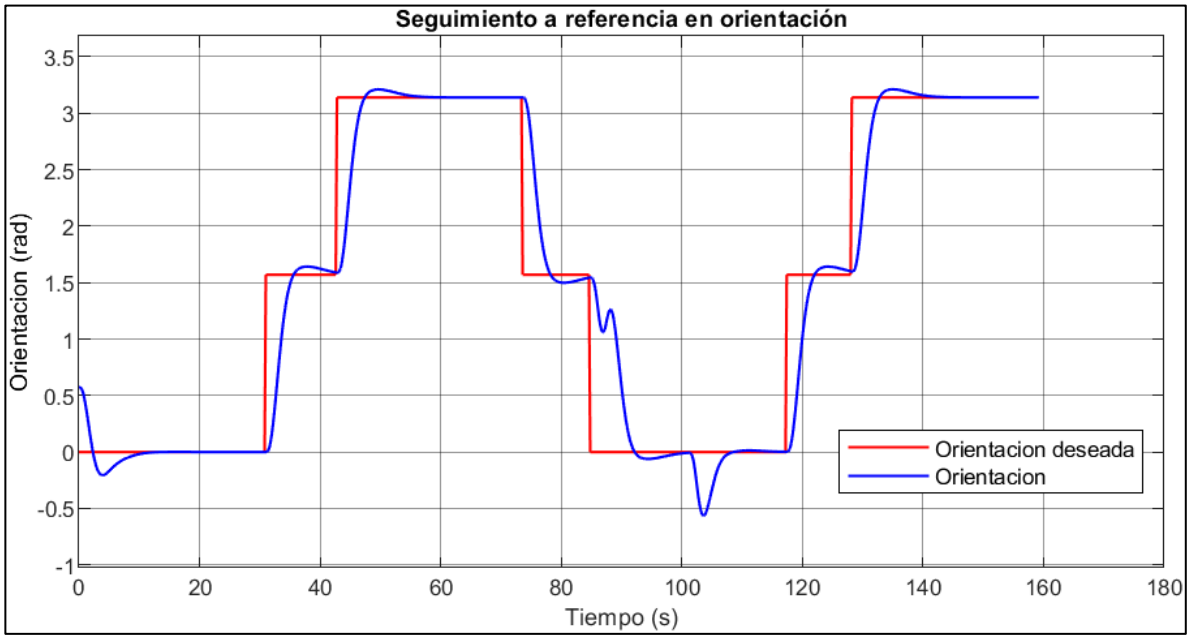


Figura 75. Seguimiento a la orientación deseada - DQN

Como en este caso lo que interesa analizar es el comportamiento del controlador PID que se encuentra en la plataforma móvil y la señal de control generada por el mismo, vale la pena hacer la revisión de la señal de referencia entregada por agente entrenado (rojo) y la señal de orientación real otorgada por el controlador PID como se muestra en la Figura 76.

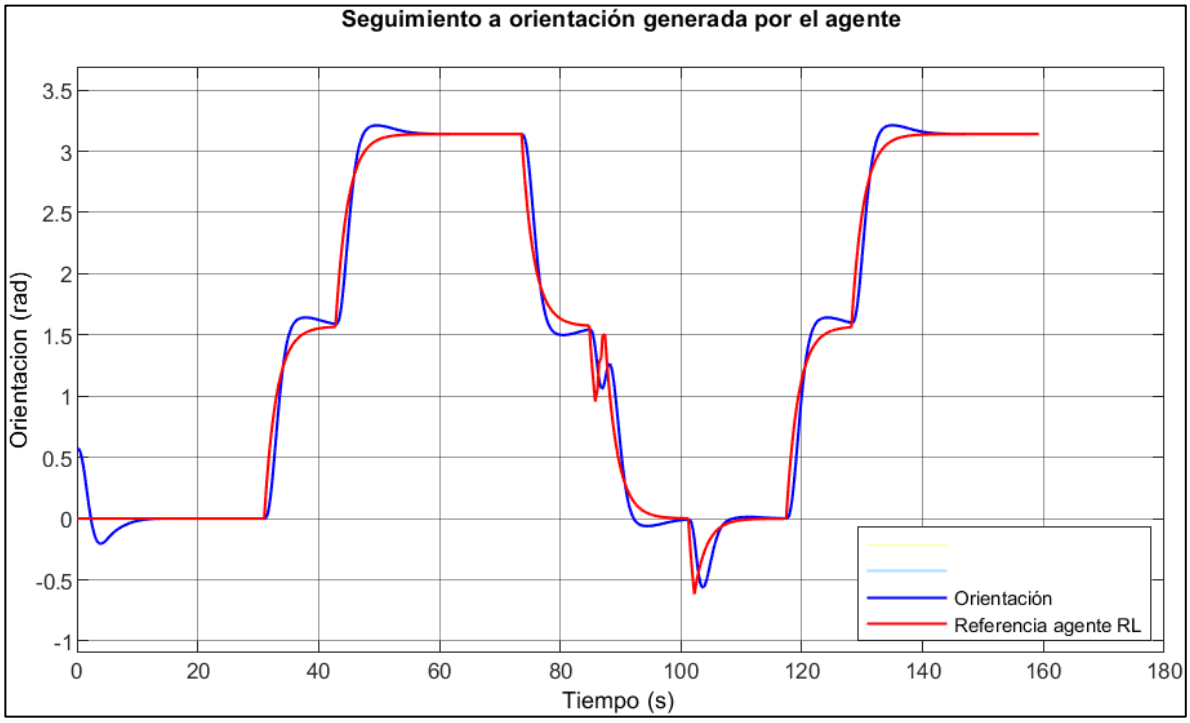


Figura 76. Seguimiento a la referencia generara por el agente - DQN

Como se evidencia en la Figura 76, la orientación real del robot y la cual se maneja por parte del controlador PID, efectivamente realiza el seguimiento a la referencia asignada por el agente entrenado, sin realizar ningún tipo de comportamiento no deseado como oscilaciones o sobre impulsos notables.

En cuanto al suavizado del comportamiento durante el desplazamiento a lo largo del camino, se revisa directamente el desplazamiento horizontal y vertical del vehículo (Figura 77) para validar que no se presentan algún desplazamiento brusco, complementando lo evidenciado a la Figura 75 y en la Figura 76 .

Se evidencia entonces en la Figura 77, que el comportamiento tanto horizontalmente (“x” en color azul) como verticalmente (“y” en color morado) no presentan variaciones de gran intensidad y que todo el desplazamiento se está llevando a cabo de manera suave.

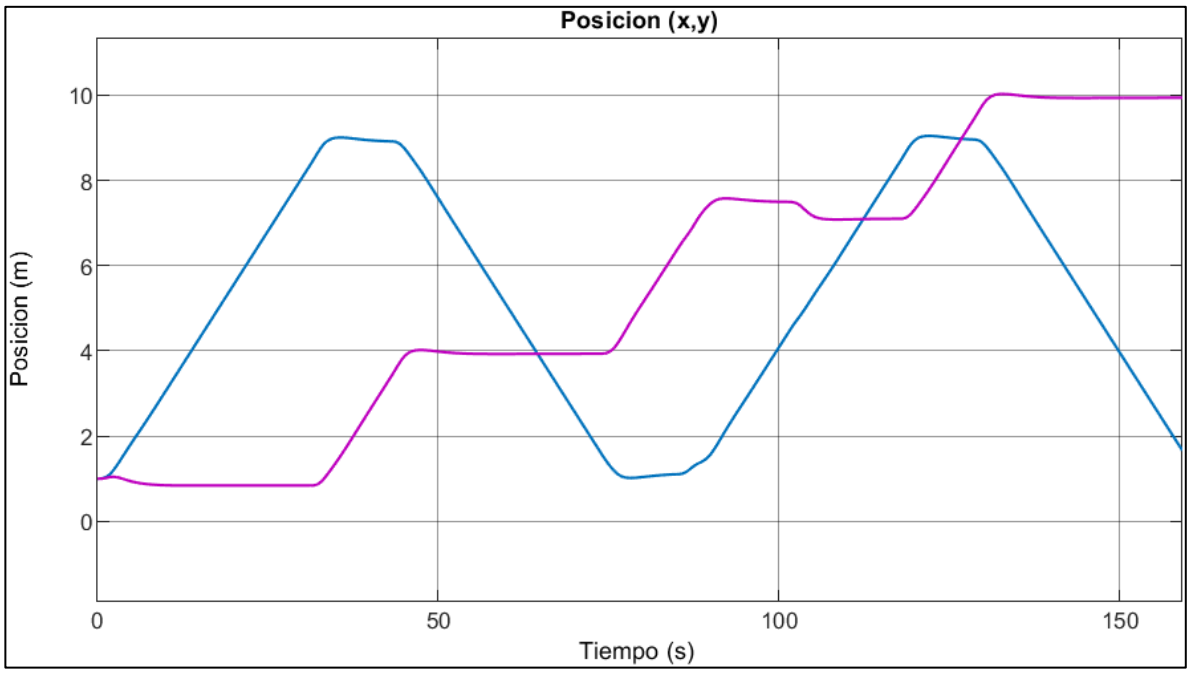


Figura 77. Comportamiento del robot en x,y

Efectivamente, se logró una parametrización en el comportamiento de un agente DQN que se encarga de brindar referencias a un controlador PID tanto en orientación como en velocidad. El agente permite que el robot se desplace de una manera suave y segura alrededor de un camino establecido hasta una posición final evitando cualquier tipo de colisión a lo largo del mismo.

6.5. Simulación agente DQN usando modelo dinámico PIONEER 3DX sobre ROS y GAZEBO

Para la validación se realiza una prueba de las señales de control generadas por el agente entrenado. Para estas prueba se hace uso del entorno de simulación Gazebo, debido a que permite asociar el modelo de la plataforma Pioneer 3DX, asociar un controlador diferencial, crear el entorno en donde se va a desplazar, asociar la sensórica y extraer información sobre la odometría durante la simulación [32]. Una vez se valida el funcionamiento dentro de la simulación solo hace falta la conexión al hardware para la experimentación.

El manejo de ROS y Gazebo se realizó desde el toolbox de ROS para Matlab y un subsistema Linux para Windows asociado a Ubuntu 18.04 LTS, la distribución utilizada fue ROS Melodic y la versión Gazebo 9.14

6.5.1 Implementación del Pioneer 3DX y creación del entorno

Como punto de partida, se debe realizar la instalación del modelo de la plataforma Pioneer 3DX el cual fue extraído de un repositorio que descansa sobre GitHub [67]. Este modelo tiene asociada toda la dinámica del robot, su controlador diferencial en términos de velocidad lineal y angular, así como un sensor laser 2D. La instalación y validación del modelo, se muestra en la Figura 78.

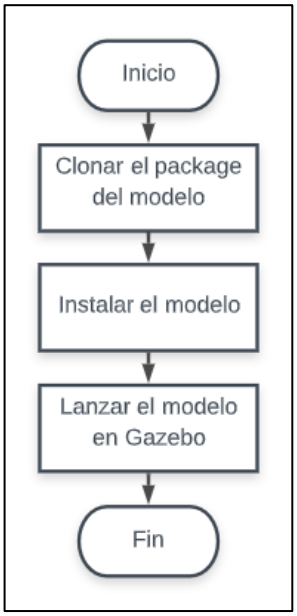


Figura 78. Instalación y validación modelo Pioneer 3DX

Para comprobar que el modelo fue debidamente instalado, se realiza la ejecución del modelo directamente en gazebo (Figura 79).

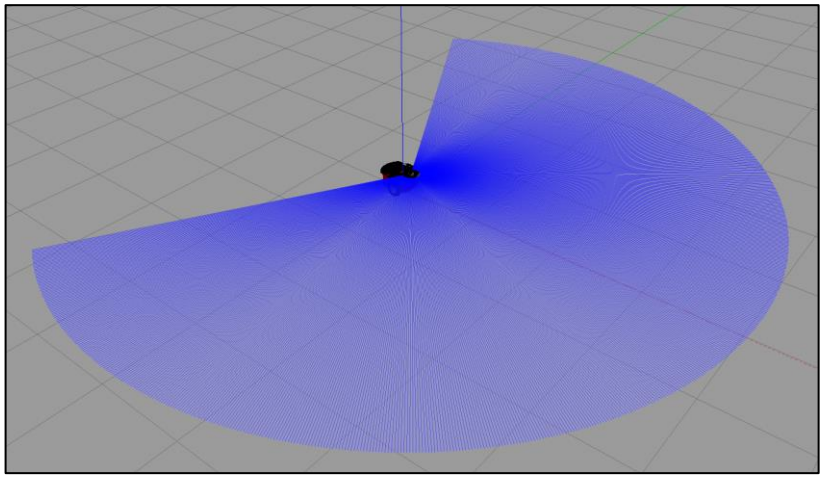


Figura 79. Modelo Pioneer 3DX en Gazebo

En segunda instancia se debe crear el entorno por el que se desplaza la plataforma, para esto se debe ingresar al editor de construcción, donde se pueden añadir los elementos necesarios, en este caso se usan paredes de 20 centímetros con una altura de un metro para crear el mismo espacio generado para el entrenamiento. El procedimiento se ilustra en la Figura 80 y el resultado en la Figura 81.

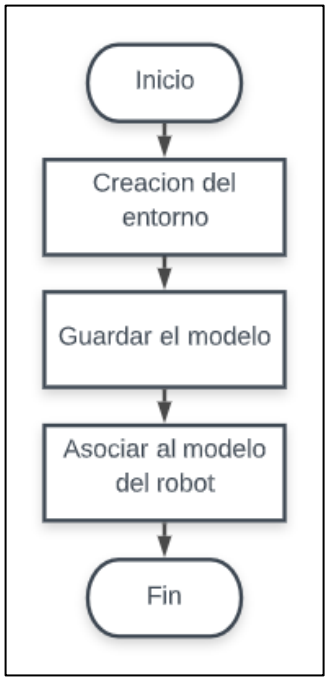


Figura 80. Proceso creación del entorno en Gazebo

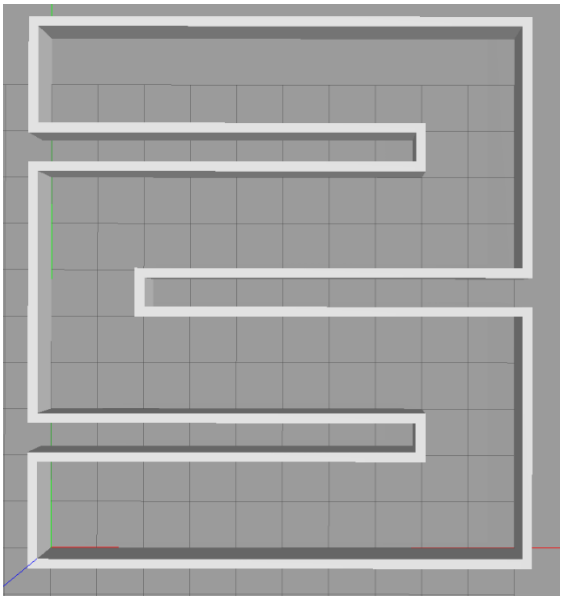


Figura 81. Entorno creado en Gazebo

6.5.2 Creación red ROS y envío señales de control

Una vez el entorno y el modelo de la plataforma están debidamente configurados y validados dentro de Gazebo, se continua con la creación de la red en ROS haciendo uso de un script en Matlab, el cual la configura debidamente y se encarga de hacer envío tanto de la velocidad angular como la velocidad lineal reales reflejadas en el robot (Figura 82).

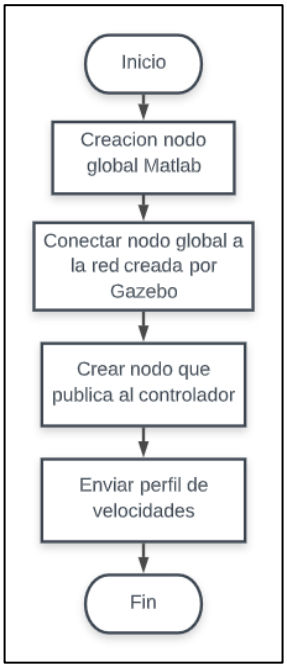


Figura 82. Creación de la red ROS y envío señales de control

La velocidad lineal que será enviada es constante como se determinó para el entrenamiento DQN y la velocidad angular se extrae después de derivar la posición angular obtenida de la dinámica del robot. El perfil de velocidad completo a ser enviado se muestra en la Figura 83.

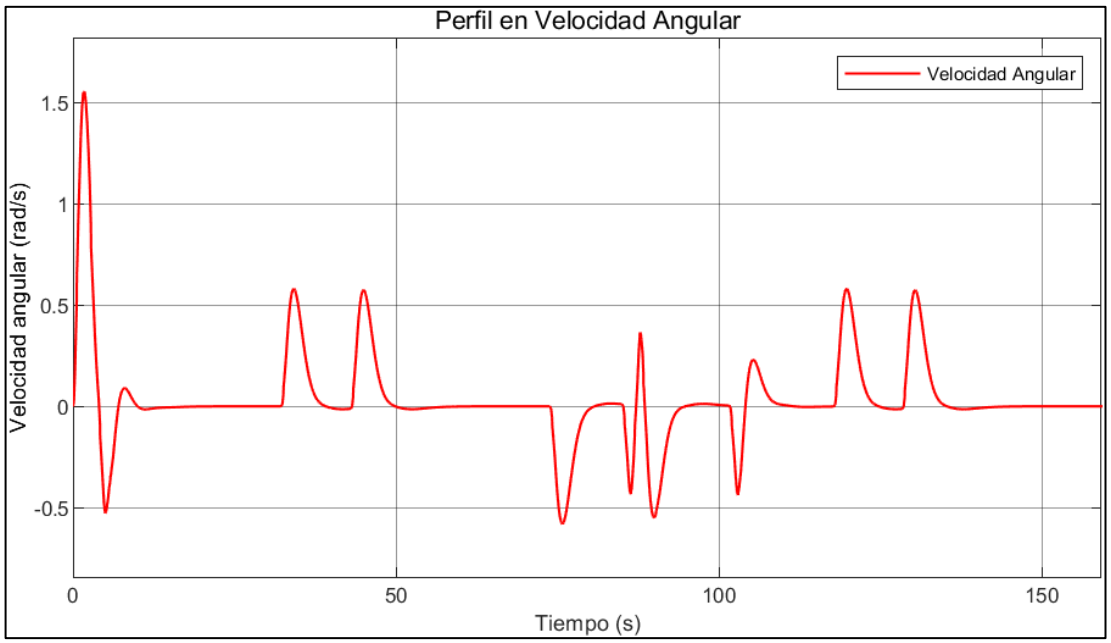


Figura 83. Perfil de velocidades angulares a enviar

6.5.3 Validación trayectoria generada por el agente y ejecutada en Gazebo

Una vez el perfil de velocidades fue enviado al modelo en Gazebo desde su posición inicial (Figura 84), se valida la trayectoria ejecutada por la plataforma leyendo la odometría que fue generada durante todo el desplazamiento. La trayectoria seguida se muestra en la Figura 85 en comparación a la trayectoria generada en Matlab.

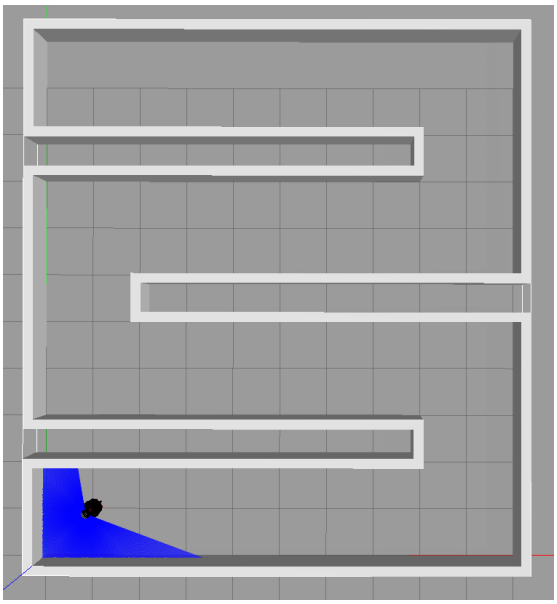


Figura 84. Pioneer 3DX en posición inicial

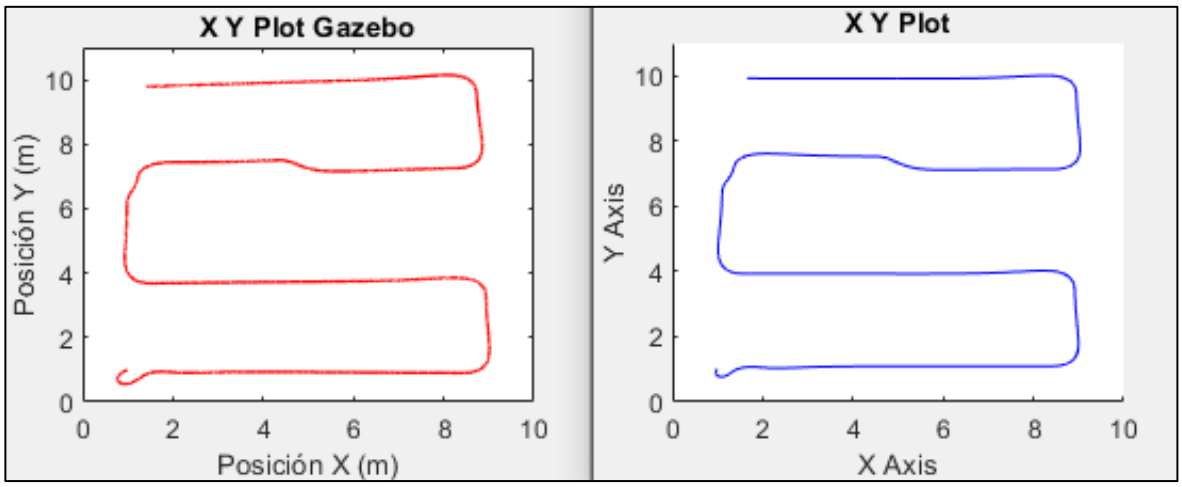


Figura 85. Trayectoria ejecutada en Gazebo vs Simulink

En cuanto a la velocidad angular que experimento la plataforma durante la trayectoria, se obtuvieron los valores reflejados en la c, muy similares al perfil de velocidad establecido por el agente (Figura 83).

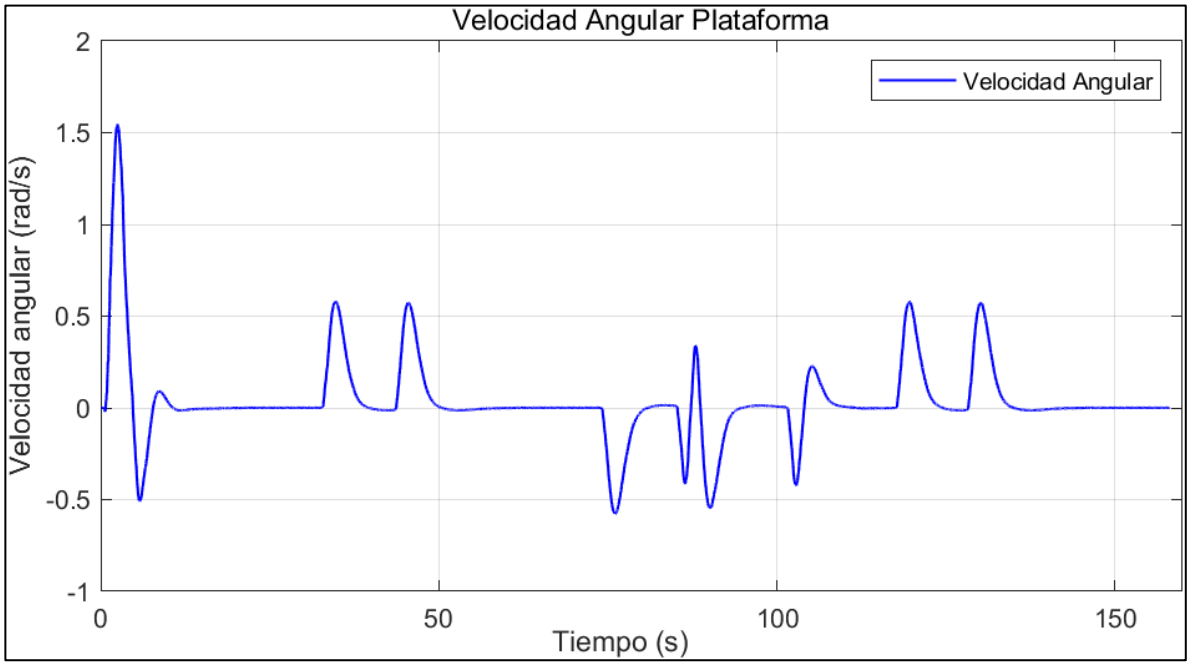


Figura 86. Velocidad angular durante la trayectoria

Como se evidencia en la Figura 85 y Figura 86 la implementación de las señales de control generadas por el agente DQN entrenado fue exitosa, adicionalmente a que realiza el seguimiento de las velocidades establecidas y la navegación en el entorno creado.

7. CONCLUSIONES

Se desarrolló un algoritmo de aprendizaje por refuerzo que permite la navegación de una plataforma en configuración diferencial sobre un entorno desconocido, a partir de una serie de recompensas que limitan su comportamiento para mantener su centro de masa a una distancia mínima de 40 centímetros a cualquier obstáculo y un conjunto de valores en referencia para controlar la posición angular. La definición de las recompensas presentó cierto nivel de complejidad y consumo de tiempo (20 horas aproximadamente para casa ensayo), ya que debido a la dinámica asociada del controlador PID a la plataforma se debe realizar una cantidad de pruebas de ensayo y error para ajustar las funciones con sus respectivos valores numéricos.

Se implementaron dos técnicas sobre el modelo de una plataforma diferencial, que involucran el uso de Deep-Learning con el objetivo de realizar la evasión de obstáculos. Las dos técnicas cumplieron con el objetivo general de efectuar la navegación con evasión de obstáculos, pero la técnica DQN presentó un mejor comportamiento en cuanto al error en seguimiento y la estabilidad en las señales de control, por otra parte, la técnica DDPG, aunque permitía una mayor flexibilidad en las señales de control, presentaba señales de control abruptas que fueron difíciles de manejar atenuar o mitigar mediante las recompensas.

Se realizó la implementación de las señales de control generadas por el agente DQN haciendo uso ROS, permitiendo publicar las señales a un modelo de simulación creado en Gazebo que despliega la plataforma Pioneer 3DX con objetivo de validar tanto la forma de controlar la plataforma como el comportamiento establecido por el agente entrenado. En este modelo se permitió la verificación del seguimiento efectivo a referencias (velocidad angular) y la navegación de la plataforma en un entorno desconocido y realizar un acercamiento a una posible experimentación física sobre la plataforma Pioneer 3DX.

Para la continuación del desarrollo realizado se sugiere realizar la implementación física en la plataforma diferencial Pioneer 3DX de las señales de control establecidas, para tener certeza de la completa validación. Adicionalmente se sugiere migrar todo el proceso de entrenamiento a otra plataforma que permita hacer el despliegue del algoritmo y el agente en tiempo real para la navegación como Open AI para integración con ROS, lo cual permitirá asociar sensórica y odometría del robot para analizar su comportamiento en entornos reales. Una vez este proceso pueda ser implementado sobre la plataforma de prueba se puede repetir para el CERES-Agrobot

8. BIBLIOGRAFÍA

[1] D. Matus, «Digital Trends,» 11 01 2018. [En línea]. Available: https://es.digitaltrends.com/autos/autos-autonomos-ces-2018/. [Último acceso: 25 06 2019].

[2] Syngenta, «Syngenta Colombia,» 12 01 2018. [En línea]. Available: http://www.syngenta.com.co/news/noticias/robots-en-el-campo. [Último acceso: 25 05 2019].

[3] P. Long , L. Wenxiu y P. Jia, «Deep-Learned Collision Avoidance Policy for Distributed Multi-Agent Navigation,» *IEEE Robotics and Automation Letters,* vol. 2, nº 2, pp. 656-663, 2017.

[4] A. Barrientos y J. del Cerro, «Interempresas,» 24 02 2016. [En línea]. Available: http://www.interempresas.net/Horticola/Articulos/151745-El-uso-de-robots-en-tareas-agricolas.html. [Último acceso: 26 06 2019].

[5] L. Cancar , D. Sanz , J. D. Hernandez Vega y J. del Cerro, «Precision Humidity and Temperature Measuring in Farming Using Newer Ground Mobile Robots,» de *ROBOT2013: First Iberian Robotics Conference: Advances in Robotics*, Springer International Publishing, 2013, pp. 443-456.

[6] L. Tai, S. Li y M. Liu, «A deep-network solution towards model-less obstacle avoidance,» de *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, IEEE, 2016, pp. 2759-2764.

[7] P. Robher, «Autonomous Driving: How Much Autonomy Driving Does Stand,» *ATZelectronics worldwide,* vol. 2, pp. 26-29, 2015.

[8] J. Ibañez-Guzman, C. Laugier, J.-D. Yoder y S. Thrun, «Autonomous Driving: Context and State-of-the-Art,» de *Handbook of Intelligent Vehicles*, Londres, Springer, 2012, pp. 1271-1310.

[9] R. Álvarez, «Zataka,» 7 11 2017. [En línea]. Available: https://www.xataka.com/automovil/el-coche-autonomo-de-google-waymo-se-vuelve-completamente-autonomo-y-por-primera-vez-sale-a-la-calle-sin-conductor. [Último acceso: 22 01 2019].

[10] W. Xinglong, Y. Cao, Z. Zhao y Y. Yan, «Deep Reinforcement Learning Based Collision Avoidance Algorithm for Differential Drive Robot,» de *Intelligent Robotics and Applications*, Newcastle, Springer, 2018, pp. 186-198.

[11] J. C. Lopez, «Xataka,» 09 03 2018. [En línea]. Available: https://www.xataka.com/automovil/donde-esta-realmente-el-coche-autonomo-a-dia-de-hoy-y-que-han-prometido-las-marcas-en-el-salon-de-ginebra-para-el-futuro. [Último acceso: 22 01 2019].

[12] B. Paden , M. Cap, S. Z. Yong, D. Yershov y E. Frazzoli, «A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles,» *IEEE Transactions on Intelligent Vehicles,* vol. 1, nº 1, pp. 33-35, 2016.

[13] J.-R. Xue, J.-W. Fang y Z. Pu, «A Survey of Scene Understanding by Event Reasoning in Autonomous Driving,» *International Journal of Automation and Computing,* pp. 249-266, 18 04 2018.

[14] Q. Tran y J. Firl, «Modelling of traffic situations at urban intersections with probabilistic non-parametric regression,» de *2013 IEEE Intelligent Vehicles Symposium (IV)*, Gold Coast, IEEE, 2013, pp. 334-339.

[15] M. Alexis, «The Atlantic,» 15 05 2014. [En línea]. Available: https://www.theatlantic.com/technology/archive/2014/05/all-the-world-a-track-the-trick-that-makes-googles-self-driving-cars-work/370871/. [Último acceso: 14 04 2019].

[16] D. N. Maryam Abadi y M. H. Khooban, «Design of optimal Mamdani-type fuzzy controller for nonholonomic wheeled mobile robots,» *Journal of King Saud University - Engineering Sciences,* vol. 27, nº 1, pp. 92-100, 2015.

[17] P. S, «Sensor Fusion for Mobile Robot Navigation: Fuzzy Associative Memory,» *Procedia Engineering,* vol. 41, pp. 251-256, 2012.

[18] M. N. Al-Din y K. Tahboub, «A Neuro-Fuzzy Reasoning System for Mobile Robot Navigation,» *Jordan Journal of Mechanical and Industrial Engineering,* vol. 3, nº 1, pp. 77-88, 2009.

[19] A. Alireza, M. H. Khooban y D. N. Maryam Abadi, «Teaching-learning-based optimal interval type-2 fuzzy PID controller design: A nonholonomic wheeled mobile robots,» *Robotica,* vol. 31, nº 7, pp. 1059-1071, 2013.

[20] S. Hu, C. Cao y J. Pan, «Deep-learned pedestrian avoidance policy for robot navigation,» de *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Macau, IEEE, 2017, pp. 338-343.

[21] F. Aznar, M. Pujol y R. Rizo, «Obtaining fault tolerance avoidance behavior using deep reinforcement learning,» *Neurocomputing,* vol. 345, pp. 77-91, 2019.

[22] P. K. Mohanty, A. K. Sah, V. Kumar y S. Kundu, «Application of Deep Q-Learning for Wheel Mobile Robot Navigation,» de *2017 3rd International Conference on Computational Intelligence and Networks (CINE)*, Odisha, IEEE, 2017, pp. 88-93.

[23] A. V. Ngo, N. H. Viet, S. Lee y T. Chung, «Obstacle Avoidance Path Planning for Mobile Robot Based on Ant-Q Reinforcement Learning Algorithm,» de *Advances in Neural Networks - ISNN 2007*, Nanjing, Springer, 2007, pp. 704-713.

[24] L. Huang, H. Qu, M. Fu y W. Deng, «Reinforcement Learning for Mobile Robot Obstacle Avoidance Under Dynamic Environments,» de *PRICAI 2018: Trends in Artificial Intelligence*, Nanjing, Springer, 2018, pp. 441-453.

[25] Precision Agriculture, «Precision Agriculture,» 27 07 2018. [En línea]. Available: https://precisionagricultu.re/es/el-impacto-de-la-agricultura-de-precision/#:~:text=La%20sostenibilidad%20a%20trav%C3%A9s%20de,monitoreo%20remoto%20de%20la%20granja.. [Último acceso: 13 09 2020].

[26] Bloomberg, «Agro Negocios,» 27 03 2019. [En línea]. Available: https://www.agronegocios.co/tecnologia/indice-de-confianza-en-tractores-autonomos-aun-no-es-tan-alto-como-se-esperaba-2844642. [Último acceso: 26 06 2019].

[27] CONtexto ganadero, «Contexto Ganadero,» 24 07 2018. [En línea]. Available: https://www.contextoganadero.com/agricultura/futuro-del-agro-en-colombia-esta-en-la-agricultura-de-precision-andres-sanchez. [Último acceso: 12 09 2020].

[28] H. Valdés Conroy, «Sostenibilidad,» 06 04 2017. [En línea]. Available: https://blogs.iadb.org/sostenibilidad/es/agricultura-de-precision-una-posible-respuesta-al-cambio-climatico-y-a-la-seguridad-alimentaria-pero-es-asequible-para-todos-2/. [Último acceso: 12 09 2018].

[29] S. Fratini, «Agronegocios e industria de alimentos,» 31 08 2019. [En línea]. Available: https://agronegocios.uniandes.edu.co/2019/08/31/agricultura-de-precision-el-futuro-del-agro-colombiano/. [Último acceso: 12 09 2020].

[30] Skymind, «AI Wiki,» 2019. [En línea]. Available: https://pathmind.com/wiki/deep-reinforcement-learning#define. [Último acceso: 2019 06 20].

[31] D. Ortego Delgado, «Open Webinars,» 21 09 2017. [En línea]. Available: https://openwebinars.net/blog/que-es-ros/. [Último acceso: 25 06 2019].

[32] Gazebo, «Gazebo,» 30 01 2019. [En línea]. Available: http://gazebosim.org/. [Último acceso: 25 06 2019].

[33] S. Ray, «A Quick Review of Machine Learning Algorithms,» de *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, Faridabad, IEEE, 2019, pp. 35-39.

[34] R. Sutton y A. G. Barto, Reinforcement Learning, Cambridge: MIT Press, 2018, pp. 1-13.

[35] Diatech, «Inteligencia Artificial,» [En línea]. Available: https://inteligenciaartificial.io/machine-learning/. [Último acceso: 2020 02 15].

[36] Tech Talks, «Raona,» 28 04 2017. [En línea]. Available: https://www.raona.com/machine-learning-tipos-machine-learning/. [Último acceso: 16 02 2020].

[37] J. Luna Gonzalez, «Medium,» 8 02 2018. [En línea]. Available: https://medium.com/soldai/tipos-de-aprendizaje-autom%C3%A1tico-6413e3c615e2. [Último acceso: 15 02 2020].

[38] Mathworks, «Mathworks,» [En línea]. Available: https://www.mathworks.com/content/dam/mathworks/ebook/gated/reinforcement-learning-ebook-part1.pdf. [Último acceso: 17 02 2020].

[39] E. Alpaydin, «Introduction to Machine Learning,» de *Reinforcement Learning*, MIT Press, 2010, pp. 447-474.

[40] M. Merino, «Xataka,» 17 01 2019. [En línea]. Available: https://www.xataka.com/inteligencia-artificial/conceptos-inteligencia-artificial-que-aprendizaje-refuerzo. [Último acceso: 22 02 2020].

[41] I. Prado, «Medium,» [En línea]. Available: https://medium.com/@YoSoyIsma/introducci%C3%B3n-y-conceptos-clave-de-aprendizaje-reforzado-reinforcement-learning-8769b3f6d0e8. [Último acceso: 2020 02 23].

[42] M'Saad y M. Farza, «Output feedback control of a class of nonlinear systems,» de *CCCA12*, Marsella, 2012.

[43] MathWorks, «MathWorks,» [En línea]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/what-is-reinforcement-learning.html. [Último acceso: 20 03 2020].

[44] «Quora,» [En línea]. Available: https://www.quora.com/What-is-an-easy-way-to-explain-online-vs-offline-machine-learning-to-a-machine-learning-novice. [Último acceso: 02 03 2020].

[45] MathWorks, «MathWorks,» [En línea]. Available: https://la.mathworks.com/help/reinforcement-learning/ug/train-q-learning-agent-to-solve-basic-grid-world.html. [Último acceso: 05 09 2020].

[46] R. López, «Rúben López Wordpress,» 12 05 2015. [En línea]. Available: https://rubenlopezg.wordpress.com/2015/05/12/q-learning-aprendizaje-automatico-por-refuerzo/. [Último acceso: 02 04 2020].

[47] A. Parkinson, «Medium,» 2 12 2019. [En línea]. Available: https://medium.com/analytics-vidhya/the-epsilon-greedy-algorithm-for-reinforcement-learning-5fe6f96dc870. [Último acceso: 03 27 2020].

[48] MathWorks, «MathWorks,» [En línea]. Available: https://www.mathworks.com/help/reinforcement-learning/ref/rlqagentoptions.html. [Último acceso: 27 03 2020].

[49] MathWorks, «MathWorks,» [En línea]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/q-agents.html. [Último acceso: 02 04 2020].

[50] Y. Wang, J. Tong , T.-Y. Song y W. Zhang-Hong, «Unmanned Surface Vehicle Course Tracking Control Based on Neural Network and Deep Deterministic

Policy Gradient Algorithm,» de *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*, Kobe, 2018.

[51] Y. Liu, W. Zhang, F. Chen y J. Li, «Path planning based on improved Deep Deterministic Policy Gradient algorithm,» de *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chengdu, 2019.

[52] C. Yoon, «Towards Data Science,» 05 02 2019. [En línea]. Available: https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f. [Último acceso: 05 05 2020].

[53] K. Vijay y J. Tsitsiklis, «Actor-Critic Algorithms,» *Siam Journal on Control & Optimization,* vol. 42, nº 4, pp. 1143-1166, 2003.

[54] MathWorks, «MathWorks,» [En línea]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/create-policy-and-value-function-representations.html. [Último acceso: 10 05 2020].

[55] P. Yue, J. Xin, H. Zhao, D. Liu, M. Shan y J. Zhang, «Experimental Research on Deep Reinforcement Learning in Autonomous navigation of Mobile Robot,» de *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, Xian, IEEE, 2019, pp. 1612-1616.

[56] S.-H. Han, H.-J. Choi, P. Benz y J. Loaciga, «Sensor-Based Mobile Robot Navigation via Deep Reinforcement Learning,» de *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Shanghai, IEEE, 2018, pp. 147-154.

[57] P. Wang, H. Li y C.-Y. Chan, «Continuous Control for Automated Lane Change Behavior Based on Deep Deterministic Policy Gradient Algorithm,» de *2019 IEEE Intelligent Vehicles Symposium (IV)*, Paris, 2019.

[58] MathWorks, «MathWorks,» [En línea]. Available: https://www.mathworks.com/help/reinforcement-learning/ref/rlddpgagentoptions.html. [Último acceso: 25 05 2020].

[59] «StackExchange,» [En línea]. Available: https://ai.stackexchange.com/questions/11640/how-large-should-the-replay-buffer-be. [Último acceso: 27 05 2020].

[60] T. Haarnoja, A. Zhou, P. Abbel y S. Levine, «Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor,» de *ICML 2018, Computer Science, Mathematics*, 2018.

[61] MathWorks, «MathWorks,» [En línea]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html. [Último acceso: 02 06 2020].

[62] J. Brownlee, «Machine Learning Mastery,» 28 01 20198. [En línea]. Available: https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/. [Último acceso: 23 05 2020].

[63] J. Hui, «Medium,» 12 09 2018. [En línea]. Available: https://medium.com/@jonathan_hui/rl-policy-gradients-explained-9b13b688b146. [Último acceso: 02 06 2020].

[64] G. Zuo, T. Du y J. Lu, « Double DQN Method For Object Detection,» de *Chinese Automation Congress (CAC)*, Jinan, IEEE, 2017, pp. 6727-6732.

[65] MathWorks, «MathWorks,» [En línea]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/dqn-agents.html. [Último acceso: 17 06 2020].

[66] S. Zhou , X. Liu, X. Yingfu y J. Guo, «A Deep Q-network (DQN) Based Path Planning Method for Mobile Robots,» de *International Conference on Information and Automation*, Wuyishan, IEEE, 2018, pp. 366-371.

[67] «GitHub,» 31 05 2019. [En línea]. Available: https://github.com/mario-serna/pioneer_p3dx_model. [Último acceso: 09 09 2020].

[68] National Instruments, «National Instruments,» 14 05 2020. [En línea]. Available: https://www.ni.com/es-co/innovations/white-papers/17/what-is-hardware-in-the-loop-.html. [Último acceso: 07 08 2020].

[69] MathWorks, «MathWorks,» [En línea]. Available: https://www.mathworks.com/help/reinforcement-learning/ug/create-simulink-environment-and-train-agent.html. [Último acceso: 19 04 2020].