Navegación Autónoma y Evasión de Obstáculos en UAV usando Aprendizaje por Refuerzo

Diego Felipe Ruiz Barreto

María Carolina Bravo Navarro

Universidad Santo Tomás

Facultad de Ingeniería Electrónica

Bogotá D.C

2019

Navegación Autónoma y Evasión de Obstáculos en UAV usando Aprendizaje por Refuerzo

Diego Felipe Ruiz Barreto

María Carolina Bravo Navarro

Proyecto de grado presentado como requisito parcial para optar al título de:

Ingeniería Electrónica

Director:

Juan Manuel Calderón Chávez

Codirector:

Carolina Higuera Arias

Universidad Santo Tomás

Facultad de Ingeniería Electrónica

Bogotá D.C

2019

Nota de aceptación:

_____

_____

_____

_____

_____

_____

_____

_____

Firma del director

_____

Firma del codirector

_____

Firma del jurado

_____

Firma del jurado

Bogotá, 3 de octubre del 2019

Agradecimientos

Agradecemos a Dios por habernos regalado la motivación y los recursos para llevar a cabo este proyecto, a nuestros padres y hermanos por todo su apoyo, a todos nuestros familiares y amigos cercanos. Agradecemos de forma especial al Ingeniero Juan Manuel Calderón y a la Ingeniera Carolina Higuera Arias, por haber aceptado ser nuestros mentores y habernos guiado durante todo este proceso formativo.

RESUMEN

La navegación autónoma en entornos no estructurados es una de las tareas más desafiantes para los vehículos aéreos no tripulados (UAV). Para enfrentar esta clase de retos es necesario el uso de algoritmos complejos de control y de aprendizaje que colaboren en procesos de adaptación del vehículo no tripulado a los continuos cambios del ambiente en el que navega. Uno de los campos de la inteligencia artificial más prometedores en tareas de aprendizaje no supervisado es el aprendizaje por refuerzo. Este trabajo propone el uso de Q-Learning en tiempo real, para generar el sistema de aprendizaje de navegación de un UAV. Dado que el proceso de aprendizaje toma bastante tiempo y los UAVs tienen una autonomía de vuelo bastante limitada por la capacidad de la batería. Para esto se propone el uso de un ambiente de simulación que permita la evolución del sistema de aprendizaje independientemente de las limitaciones de autonomía de un robot real, en donde el robot aprenderá a navegar autónomamente y evadir obstáculos.

Se presenta una propuesta usando aprendizaje por refuerzo más específicamente la técnica de Q-Learning para resolver el problema de la navegación autónoma en un UAV, en la solución propuesta se establecen los estados del agente teniendo en cuenta varios factores, como es la lectura de los sensores en la detección de obstáculos, la distancia que hay entre el agente y el objetivo, y la dirección hacia donde se encuentra el GOAL. Para esta solución son establecidas 8 acciones que son los movimientos que el UAV puede realizar, una política de premios y castigos para evaluar el aprendizaje del agente, Se implementa la técnica propuesta en el software matemático MATLAB y el entorno de simulación virtual V-REP, para acercarse lo más posible a un escenario real como lo es el rescate de personas en desastres naturales, en la entrega de paquetes, en la agricultura de precisión, entre otras aplicaciones.

CONTENIDO

Pág.

Lista de Figuras

1. INTRODUCCIÓN

En la actualidad, los vehículos aéreos no tripulados (UAV) son utilizados en un gran número de tareas, como se muestra en artículos como *"Robust Real-time Vision-based Aircraft Tracking From Unmanned Aerial Vehicle*" [1] resaltando, desde el seguimiento de objetos hasta el rescate de seres vivos en desastres naturales, esto se debe a su versatilidad y a las posibilidades de solucionar problemas que han encontrado los investigadores a través de ellos. Los estudios que se han realizado en relación con la navegación autónoma de UAV son muy limitados y en la mayoría, se asume previo conocimiento del ambiente donde se pondrá en marcha el UAV, alejándose del objetivo principal de este trabajo. Artículos como "*Autonomous Navigation of UAV by Using Real-Time Model-Based Reinforcement Learning*" [2] dirigido por Nursultan Imanberdiyev, Changhong Fu, Erdal Kayacan, I-Ming Chen quienes muestran el desarrollo de algoritmos con el uso de aprendizaje por refuerzo (RL) de forma paralela (TEXPLORE), donde en base al entorno proponen una acción por cada estado del ambiente sin tener en cuenta decisiones previas, pero conservando los datos de la batería, la posición y la recompensa, logrando obtener un sistema con mayor velocidad de respuesta. Sin embargo otras investigaciones como "*Autonomous UAV Navigation Using Reinforcement Learning*" [3], dirigido por Huy X, et al, proponen un algoritmo de *Reinforcement Learning* (RL) basado en aproximación de funciones para tratar con el gran número de representación de estados, permitiendo de esta manera obtener un tiempo de convergencia menor, logrando la navegación autónoma en ambientes desconocidos del agente, con el objetivo de encontrar a una persona desaparecida, material importante para el desarrollo de este proyecto.

Para lograr usar los UAV en tareas civiles tales como en tareas de rescate es importante contar con un algoritmo que le permita aprender a moverse libremente sin importar la poca información que se tenga del medio donde se encuentra el agente, de tal modo que pueda llegar hasta su objetivo en el menor tiempo posible, evitando los obstáculos que haya en el lugar. Es en este punto donde entra a jugar un papel importante el aprendizaje de refuerzo, inspirado en la psicología conductista, donde el agente aprende de sus experiencias positivas y negativas (dadas en términos de recompensas numéricas), las cuales obtiene por las acciones que lleva a cabo en el entorno. En el desarrollo de este trabajo, se muestra el enfoque que se usó para solucionar el problema en la sección de marco teórico, el diseño del algoritmo de aprendizaje mostrado en la sección de ejecución del proyecto, la implementación del algoritmo en un entorno de simulación encontrado en la ejecución del proyecto y finalmente las conclusiones.

Este proyecto fue posible a través del apoyo y asesoría del Grupo de Investigación y Desarrollo en Robótica de la Universidad Santo Tomas (G.E.D.). El grupo se enfoca en tres ramas principales de investigación que son denominadas como: Visión Artificial, Robótica y sistemas autónomos o inteligentes, dentro de las ramas de investigación se han desarrollado varios proyectos investigativos con un enfoque

en la robótica cooperativa, robots humanoides, Inteligencia artificial (I.A) y robótica educativa [4], [5], este último se ha enfocado en motivar a estudiantes de colegio a escoger carreras afines a la ingeniería y a las matemáticas. El grupo se ha destacado al participar activamente en el evento “RoboCup”, en donde se construyó un equipo robótico de futbol para participar en la liga “Small Size” del evento [6], [7], [8], [9], además en el desarrollo de estrategias de lógica difusa y control para reducir la fuerza de impacto al realizar saltos o presentar caídas en robots humanoides [10], [11], [12], [13]. Hoy en día se desarrolla proyectos investigativos que van enfocados en el aprendizaje autónomo para robots tal como se evidencia en el desarrollo de este proyecto.

2. PLANTEAMIENTO DEL PROBLEMA

En la actualidad, los vehículos aéreos no tripulados (UAV), por sus siglas en inglés "Unmanned Aerial Vehicle" son utilizados en un gran número de tareas, como la preservación de especies, mitigar los desastres naturales como los incendios, búsqueda y rescate de personas, investigaciones biológicas, en la agricultura en Colombia como la implementación de nuevas estrategias para el cuidado del campo, como se muestra en el artículo del periódico el Espectador, titulado "Drones granjeros para el campo Colombiano", donde resaltan la forma en la que los drones pueden ayudar a identificar las áreas de un terreno que requieran una atención especial, entre otras aplicaciones. Todo esto está atrayendo la atención de grupos de investigadores que buscan desarrollar los aspectos más relevantes de su manipulación, como lo es la navegación autónoma de UAVs. Este auge está permitiendo ampliar las aplicaciones que hoy en día estos presentan y mejorando su eficiencia, como se muestra en el artículo "*Autonomous UAV Navigation Using Reinforcement Learning*" [3], donde se resalta el uso de los UAV para el rescate de seres vivos en desastres naturales y en investigaciones topográficas a través de las fotografías que se toman con ayuda de los agentes. Teniendo en cuenta el panorama expuesto anteriormente, es posible considerar el aspecto común que tienen todas estas aplicaciones, es decir que son desarrolladas en un entorno desconocido, por lo tanto, es necesario que exista un aprendizaje del entorno para cumplir con su objetivo.

Se puede tener conocimiento del ambiente usando aplicaciones como lo son *Google maps* o un *Global Positioning System* (GPS), pero estos métodos no aseguran que el entorno esté actualizado, por lo tanto, es necesario que exista una navegación inteligente a través del UAV, resaltando de esta manera la importancia del desarrollo de una navegación inteligente para realizar estas tareas, a través de la implementación de un algoritmo, asegurando de esta manera una trayectoria eficiente, garantizando que el agente llegue hasta su objetivo en el menor número de pasos realizados, sin chocar con obstáculos. Sin embargo, en la mayoría de los estudios realizados de la navegación autónoma se tiene en cuenta algunas condiciones iniciales como el conocimiento previo del lugar donde desempeñará la tarea el agente, es decir, se obtiene inicialmente un mapa del ambiente, construyendo la planificación del movimiento del robot de manera más sencilla y haciéndolo un problema con soluciones ya propuestas. En [2] aplican el aprendizaje por refuerzo, considerándolo como un método que le permite a un agente aprender acciones correctas con poco o sin conocimiento previo sobre su entorno, para permitir la navegación autónoma del agente estableciendo un entorno cerrado con ciertas especificaciones donde actuará el UAV, al determinar el número de estados posibles le proporciona información a este, facilitando la llegada hasta su objetivo. Por otra parte, en la referencia [3] proponen un aprendizaje basado en el algoritmo RL, en el cual se tienen en cuenta una arquitectura paralela, es decir visualizan varias condiciones del agente que permiten su navegación, como el nivel de su

batería, las recompensas que ha obtenido y la posición global, sin embargo aunque esta técnica funciona de manera sencilla, puede reducir su eficiencia al encontrarse con obstáculos complejos, como los que se pueden presentar en una ciudad. No obstante, aunque en los estudios nombrados se resuelve parcialmente el problema de la navegación autónoma de UAV en ambientes desconocidos, en la mayoría de ellos se presenta un mapa del entorno donde desempeñará la tarea el agente alejándose de la finalidad del proyecto, que es permitir que el UAV logre llegar hasta su objetivo, sin el conocimiento previo del ambiente, solo con la información adquirida a través de sensores, porque al tomar la información de estos, permiten que el algoritmo sea más eficiente y logre hallar la ruta hasta el punto final que se le ha definido.

3. ESTADO DEL ARTE

Para el desarrollo de este proyecto es importante tener en cuenta el estado del arte proporcionado por las bases de datos más relevantes en la ingeniería, como *IEEE Xplore, Scopus* y *ScienceDirect*, logrando visualizar la forma en la que otros autores intentan solucionar el problema. A continuación, serán descritos algunos artículos relacionados con la navegación autónoma de vehículos aéreos no tripulados (UAV), más específicamente con la técnica de *Machine Learning, Q-learning.*

En [3], logran entrenar a un agente para que llegue hasta su objetivo en el menor número de iteraciones posibles en un ambiente desconocido, utilizando el método de *Q- learning* de aprendizaje por refuerzo, pero restringiendo el ambiente donde éste desarrolla la tarea, en otros artículos como en [19] se describe el uso de un método de aprendizaje por refuerzo (RL) para lograr un control de alto nivel de la misión del UAV en un ambiente desconocido, logrando que el agente aprenda un buen comportamiento en algunas iteraciones en tiempo real, a través de la limitación del espacio donde se llevaron a cabo las simulaciones, estos experimentos mostraron la validez de un método de aprendizaje por refuerzo para la navegación autónoma de UAV. Este artículo [19], si bien, presenta el aprendizaje autónomo del agente, limita los estados que este puede experimentar en el ambiente donde se encuentra, alejándose de la realidad donde habrá muchas variables que no pueden ser ignoradas para obtener el mejor desempeño del UAV.

Las herramientas que facilitan las aplicaciones de los métodos que ofrece el aprendizaje por refuerzo para obtener los estados son varias, desde cámaras, como se muestra en [20], a través de las cuales el agente puede detectar el objetivo con procesamiento de imágenes y aplicar premios y castigos al UAV según la acción que tome en cierto estado. Otra herramienta es el buscador de rango láser, donde, como se muestra en [20] logran la estimación del movimiento del UAV a bordo en tiempo real, utilizando un filtro Kalman (que fusiona la velocidad y la aceleración que presenta el agente), permitiendo la navegación autónoma del UAV en un ambiente de prueba a pequeña escala. Otra herramienta que ha sido usada, son las señales de GPS, como en [2] a través de las cuales se puede derivar un ángulo de orientación y posición actual y de destino, permitiéndole al agente cumplir con la misión establecida de forma más eficiente. En este trabajo, se modela el problema de la navegación autónoma de vehículos aéreos no tripulados como un problema de control discreto, aplicando una técnica más avanzada del aprendizaje por refuerzo (RL) que es llamada *Deep Reinforcement Learning* (DRL) que se sale de la temática de la técnica estudiada para el desarrollo de este proyecto.

Por otra parte, para mejorar el rendimiento del algoritmo, en estudios como [21] presentan un entorno de simulación para el diseño y la evaluación de estabilizadores del agente y los controladores para el seguimiento de sus trayectorias. La altura a la que se desplaza el agente se establece, reduciendo la

complejidad de la navegación a dos dimensiones. Con la reducción de dimensionalidad de la navegación se intenta enfocar toda su atención en la evasión de los obstáculos que se encuentran en el lugar donde desarrolla la misión.

Otros artículos como [24], se enfocan en la aplicación de un aprendizaje por refuerzo, centrado en un modelo de *"Basal ganglia network*", donde tienen en consideración la corteza orbitofrontal y la toma de decisiones es crítica debido a su responsabilidad para representar la recompensa del agente, siendo fundamental en la aplicación del aprendizaje autónomo. En este estudio, se propone un sistema de memoria de acciones para que el UAV guarde estas durante el proceso de aprendizaje. Cabe destacar, que esta propuesta es muy compleja ya que no usan *Q-learning* para el aprendizaje, sino un modelo de *"Basal ganglia network*", donde las simulaciones muestran un comportamiento eficaz. Para llevar a cabo el proyecto en un UAV real sería necesaria una plataforma con alta capacidad de cómputo. En [25], exponen la solución para la evasión de obstáculos de un UAV, aplicando un modelo de aprendizaje por refuerzo rápido, a través de un algoritmo de convolución profunda de redes neuronales (CNN), para la toma de imágenes, donde se intenta lograr una aproximación a la visión del ser humano. Para la eficiencia en el control de espacios o estados y de acción se usa una red neuronal RBF, para probar la solución propuesta se realiza el procesamiento de imágenes donde se intenta aplicar visión artificial para la detección y evasión de obstáculos; este enfoque de solución se aleja del propuesto en este trabajo, debido a que, en este proyecto se busca realizar la evasión de obstáculos por medio de las señales que generan los sensores, sin requerir el procesamiento de imágenes adicional.

En [26], proponen un algoritmo nuevo de aprendizaje, enfocado a resolver los problemas de maniobrabilidad, prevención de colisiones e intercambio de información en la planificación de rutas para UAV, este proyecto se diferencia del trabajo desarrollado en este documento porque su enfoque es hacia la navegación de un conjunto de drones, teniendo en cuenta variables diferentes, que para la navegación autónoma de un solo agente no se deben tener presente, como la comunicación eficiente entre los agentes.

Si se tiene en cuenta la cantidad de trabajos realizados respecto al tema en Colombia, son resaltados algunos como [27] , en el que se desarrolla una comparación de estrategias de navegación utilizando teoría de multiagentes para el barrido de zonas posiblemente minadas y [28] donde se hace uso de técnicas de aprendizaje por refuerzo (*Q-Learning*), con el objetivo de entrenar un grupo de robots para generar comportamientos de enjambre, siendo este trabajo el más similar desarrollado por estudiantes de la Universidad Santo Tomás al propuesto en este proyecto, donde estudian los diferentes métodos para lograr una navegación de multiagentes en una zona de difícil acceso.

De esta manera, es posible afirmar que el aprendizaje por refuerzo, como es descrito en [29], ofrece a la robótica un conjunto de herramientas para el diseño de comportamientos difíciles de plantear y la robótica proporciona inspiración, impacto

y validación para el desarrollo en esta área. Trabajando en conjunto, es posible desarrollar algoritmos que permitan obtener buenos resultados para el problema que se está resolviendo, de esta manera, se destaca la importancia del desarrollo de métodos basados en aprendizaje por refuerzo. Logrando como en [30], el desarrollo teórico y práctico para obtener un producto de bajo costo enfocado al mercado del entretenimiento, con la particularidad de ser estable, robusto y fácil de usar, a través de sistemas de navegación que combinan sensores inerciales de bajo costo, técnicas de visión por computador y modelos de aerodinámica.

En el caso especial de navegación en sistemas multi-agente basado en UAVs, en [31] un sistema de formación basado en el seguimiento de posición y velocidad de un líder es desarrollado haciendo uso de leyes de lógica difusa. En [32] por ejemplo el uso de un algoritmo de consenso es mostrado con el propósito de conocer la ubicación de una víctima en un entorno conocido. En [33] un algoritmo multi-quadricoptero es simulado con el propósito de llevar a cabo labores de navegación e identificación de víctimas en una zona de desastre, haciendo uso de funciones de potencial y consenso, las cuales generan fuerzas de repulsión ante posibles choques o atracción hacia un punto de interés (una víctima) y la identificación o clasificación de un punto de interés como víctima respectivamente. Por otro lado, en [34] el mismo propósito es considerado, pero contemplando también el efecto de exploración de una zona desconocida la cual es alcanzada haciendo uso de celdas de voronoi heterogéneas, esto último debido al hecho que robots con diferentes capacidades y velocidades son tenidos en cuenta trabajando en base a una función de costos. Cabe destacar que los enfoques anteriores consideran la conexión de agentes como enjambres o sub-enjambres, sin embargo, en [35] un algoritmo que permite a un agente comunicarse con otros o explorar por su cuenta es desarrollado inspirado en comportamientos de hormiga optimizando de esta forma la cantidad de área explorada y luego compartida entre los agentes que están en rango de comunicación. En esta misma línea de la bio-inspiración para algoritmos de control para sistema multi-robot, en [36] es presentado un sistema de navegación en entornos con múltiples obstáculos manteniendo la unión del enjambre basado en comportamientos básicos de abejas.

Finalmente, en [37] se encuentra un estudio semejante, pero alejado al objetivo principal de este trabajo donde proponen la solución para el procesamiento de datos y la difusión de estos en un enjambre de drones en un centro urbano, utilizando técnicas de aprendizaje para equilibrar de manera adaptativa la velocidad de transmisión entre los agentes, donde el enjambre se adapta mediante la ejecución de transiciones de estado mientras están conectados, también se tiene en cuenta el equilibrio en el almacenamiento de datos y la asignación de energía que se le da a cada dron, entre los datos que estos pueden tomar se encuentra el ruido del ambiente, entre otros. Realizando un contraste con la idea a desarrollar en este trabajo, la referencia encontrada logra obtener la comunicación eficiente entre los

agentes sin garantizar a través de un algoritmo la navegación autónoma de los mismos.

4. JUSTIFICACIÓN

Las aplicaciones de vehículos aéreos no tripulados (UAVs), ha permitido facilitar las diferentes tareas que el ser humano realiza, disminuyendo el tiempo de su ejecución y mejorando la eficiencia con la que actúan, como por ejemplo el seguimiento de los cultivos, logrando obtener mejores productos de las cosechas como se muestra en el documento “Vehículos Aéreos no tripulados, Drones” [14].

La gama de aplicaciones de los UAVs ha sido ampliada en los últimos años [14] gracias a las mejoras hechas en el campo de la estabilidad y herramientas de simulación y diseño. Algunas de las aplicaciones más importantes de los UAVs se encuentran en el área de la agricultura, donde en el caso del territorio colombiano, estos dispositivos podrían generar un impacto directo transformando los métodos de trabajo en el campo, trayendo beneficios como fumigaciones más eficientes, atracción a los jóvenes en este sector por la tecnología que se aplica, ahorro de costes, aumento de la productividad, todo esto, por medio de la recolección de datos que permitirán una mejor toma de decisiones.

La navegación autónoma de un UAV es un reto sobre cómo desarrollar un algoritmo que le permita al agente aprender a moverse con libertad y de manera inteligente en un ambiente desconocido, brindándole a los usuarios un dispositivo capaz de llevar a cabo tareas complejas como las anteriormente nombradas, sin necesidad de conocer el entorno a priori. Sin embargo, contextualizando esto en Colombia es posible afirmar como describen en [14], que el interés en el desarrollo tecnológico con los UAVs es mínimo.

Adicionalmente, estos dispositivos generan un impacto directo en el contexto de la ciudad, porque a través de ellos se facilita y se hacen más eficientes tareas como el “*delivery”* de última milla logrando optimizar el tiempo y reduciendo costos de la entrega de paquetes, como lo muestra la Escuela de Negocios de la innovación y los Emprendedores en la nota titulada “¿Cómo aprovechar los drones en la industria logística?” [15]. Otra aplicación se encuentra en el periodismo, logrando captar mejores ángulos que otras herramientas no logran y visualizando de manera más completa el ambiente del que se está informando, como lo presenta Josep Lluís Micó en la nota “Drones (para) reporteros” del periódico español “La vanguardia” [16]. También se podría enfocar su uso en tareas que son muy peligrosas para los seres humanos, como la limpieza de zonas afectadas por materiales nocivos para la salud, como se resalta en [17]. Finalmente, la aplicación más inmediata y común en el mercado es para el entretenimiento y el deporte, donde los UAVs logran captar imágenes de muy buena calidad. Resaltando de esta manera, que con la implementación de la navegación autónoma se hace más accesible el uso de UAVs para personas con poco conocimiento sobre sus maniobras, aumentando de esta manera la seguridad al momento de manipularlo.

El desarrollo tecnológico con los UAVs, permiten dentro de la Universidad Santo

Tomás trabajar en conjunto con otras facultades, desarrollando proyectos interdisciplinarios, como el estudio de las zonas rurales con la ingeniería civil, detección de efectos negativos o positivos que afecten al medio ambiente, adicionalmente permite incentivar a los jóvenes a estudiar carreras enfocadas en ingeniería y matemáticas.

5. OBJETIVOS

5.1. Objetivo general

Diseñar un sistema de navegación autónoma para un UAV en entorno de simulación, utilizando técnicas de aprendizaje por refuerzo como *Q-Learning*.

5.2. Objetivos específicos

- Definir los estados y acciones de la matriz Q del algoritmo *de Q-Learning* para facilitar el aprendizaje en la navegación del UAV.
- Establecer políticas de premios y castigos para la generación de la matriz R en el algoritmo de *Q-Learning.*
- Simular el algoritmo de aprendizaje para permitir la evolución de este, independientemente de las limitaciones de autonomía de un robot real.
- Evaluar el rendimiento del sistema de aprendizaje acorde con la variación de la cantidad de acciones y estados posibles.

6. MARCO TEÓRICO

El aprendizaje por refuerzo se encuentra relacionado con el concepto de aprendizaje natural, este último es asociado comúnmente con la exploración del entorno en el que se encuentra, aprendiendo según las experiencias que se obtengan en el ambiente. Llevando este concepto a robots móviles, es importante resaltar la presencia de sensores, cámaras u otras herramientas que permitan obtener información del medio donde se está moviendo, obteniendo una relación causa y efecto para alcanzar el objetivo deseado. El aprendizaje por refuerzo, de esta manera, se considera como la acción de aprender a actuar de acuerdo con la experiencia con el medio ambiente, permitiendo maximizar una señal de recompensa numérica que representa la tarea, es decir, no se le asignan las acciones que debe llevar a cabo, sino que de acuerdo con la exploración del entorno que haga el agente descubre cuál es la mejor decisión entre las opciones que tiene. Estas dos características, la búsqueda a prueba y error y la recompensa obtenida, son propias del aprendizaje por refuerzo. Para conocer más a fondo el aprendizaje por refuerzo es necesario definir conceptos importantes que hacen parte de este, como se muestra en la sección 6.1.

6.1. Conceptos importantes

- Acción (a)

Considerada como todos los movimientos posibles que el agente puede realizar.

- Estado (s)

Es un conjunto de observaciones que representan la interacción con el entorno donde se mueve el agente.

- Recompensa (R)

Es entendida como la devolución inmediata que genera el entorno donde se encuentra el agente, para evaluar la última acción de este, es decir, la recompensa es la representación numérica de la tarea que se desea que el agente aprenda a realizar.

- Política (π)

La política, es el método que permite definir la forma en la que se comporta el agente en un momento determinado, teniendo presente un mapeo de los estados percibidos en el entorno para conocer las acciones que se deben realizar sobre éste, de forma coloquial, la política corresponde a las reglas establecidas que el agente debe seguir o a la estrategia que emplea el agente para determinar la siguiente acción en función del estado actual.

- Valor Q

El valor Q, es la utilidad que recibe el agente desde el estado 's', con la acción 'a', comportándose óptimamente según la política 'π' en adelante.

- Función de recompensa

En la función de recompensa, se define la meta del problema de aprendizaje por refuerzo, aquí se establecen cuáles son los eventos buenos y malos para el agente (de forma inmediata), a través de una representación numérica de la tarea que se desea que el agente aprenda a realizar. El objetivo principal de este último es maximizar la recompensa total que recibe a lo largo de su recorrido. Debe ser inalterable por el agente, pero si la base para modificar la política del algoritmo.

- Función de valor

La función de valor define la utilidad de las acciones que son buenas para el agente, pero en un largo plazo, es decir, los valores que se tienen en esta función indican la conveniencia en un tiempo prolongado de los estados, después de tener en cuenta cuales de éstos probablemente seguirán y las recompensas que se generen de ellos.

- Factor de descuento (γ)

El factor de descuento permite determinar en el algoritmo la importancia que se le da a las recompensas futuras, se encuentra definido en un rango entre 0 y 1, cumpliéndose que, si el valor es más cercano a 0 el agente no se esforzará por alcanzar una alta recompensa, mientras que si el valor de 'γ' es más cercano a 1, hará que el agente busque obtener una alta recompensa.

- Tasa de aprendizaje (α)

La tasa de aprendizaje, es un valor que determina en qué medida la información recién adquirida por el agente actualiza su conocimiento de la tarea que realiza y del entorno con la nueva información, se encuentra definido en un rango entre 0 y 1, cumpliéndose que, si $\alpha = 1$ el agente solo tomará la información más reciente ignorando el conocimiento previo para explorar más opciones, mientras que, si $\alpha = 0$ el agente no aprende, solo usa el conocimiento previo adquirido a través de la experiencia.

- Exploración

La exploración en este ámbito es considerada como la selección de acciones aleatorias según las políticas de exploración. El fin, es incrementar el conocimiento del agente sobre sus acciones y el entorno por medio de la información de la recompensa.

- Explotación

La explotación se da cuando el agente prefiere realizar las acciones que ha tomado en el pasado y ha encontrado que son efectivas dado que entregan mayor utilidad a largo plazo.

Ninguna de las dos, exploración o explotación puede perseguirse de manera exclusiva, consiste más bien en un trabajo en conjunto para equilibrar los estados que el agente ha explorado, con el uso de las acciones que anteriormente le han funcionado bien. Algunas de las políticas de exploración más conocidas son *E-greedy y Softmax*, en la primera de ellas, la mayor parte del tiempo se selecciona la acción que da el mayor valor estimado, pero con la probabilidad Q es seleccionada una acción aleatoriamente, mientras que en Softmax, la probabilidad de selección de cada acción depende de su valor estimado.

Teniendo presente estos elementos del aprendizaje por refuerzo, como lo son la política, la función de recompensa, la función de valor, la acción, el estado, la recompensa, el valor Q, la función de valor, el factor de descuento, la tasa de aprendizaje, la exploración y la explotación, se puede diferenciar de otras técnicas por el aprendizaje selectivo que realiza el agente, considerándose de esta manera una técnica completa, aunque el tiempo que se requiera para el entrenamiento sea extenso en la mayoría de los casos, dependiendo de los objetivos que sean establecidos.

Otro aspecto importante para entender el aprendizaje por refuerzo es su historia, que es dividida en tres hilos principales, como es explicada en [18], aspectos que fueron estudiados de manera independiente antes de conocer el concepto de aprendizaje por refuerzo moderno. El primero de los aspectos se refiere al aprendizaje que se da a través del ensayo y del error, inspirado en la psicología de aprendizaje animal, donde se pueden obtener varias respuestas de una acción determinada y escoger la que más se acerque a la voluntad del animal en esa situación, este hilo condujo a la reactivación del aprendizaje por refuerzo alrededor de los años ochenta. El segundo aspecto, se enfocó más en solucionar el problema del control óptimo, término que empezó a usarse a finales de la década de los años cincuenta donde se abordó el problema de diseñar un controlador para minimizar una medida del comportamiento de un sistema dinámico en el tiempo, este enfoque utiliza principalmente los conceptos del estado en un sistema dinámico y la ecuación de Bellman, elementos esenciales que aportan a la teoría y algoritmos del aprendizaje por refuerzo moderno, el último hilo hace referencia a los métodos de diferencia temporal, donde se tiene en cuenta la diferencia entre las estimaciones que se realizaron en la misma cantidad, implementando un “reforzador secundario”, por medio del cual se tiene un estímulo que se encuentra conectado con el “reforzador primario”, logrando obtener respuestas más precisas debido a que existe mayor información sobre la cual se pueden tomar decisiones, este último aspecto se ha desarrollado menos que los dos hilos anteriormente nombrados pero juega

un papel igualmente importante por su novedad. Finalmente, en 1989 surge *Q-Learning* como la unión del hilo de control óptimo con el método de diferencias temporales, un trabajo desarrollado por Chris Watkins quien más adelante aportó al campo de las redes neuronales. Otro de los investigadores sobresalientes fue Gerry *Tesauro's* quien aplicando los conceptos al juego conocido como *"backgammon*" logró captar la atención hacia el aprendizaje por refuerzo, otros de los investigadores sobresalientes en el área son nombrados en los antecedentes, donde se destacan algunos de los trabajos que han tenido buenos resultados empleando las técnicas de aprendizaje por refuerzo o técnicas similares.

Entre los algoritmos más destacados de aprendizaje por refuerzo se encuentra *Q-Learning* y Sarsa. Para esto, es importante tener claro el concepto de aprendizaje de diferencia temporal, llamado así por su nombre en inglés "*Temporal-Difference Learning*" (TD), como se muestra a continuación:

El aprendizaje de diferencia temporal es considerado el aporte más novedoso para el aprendizaje por refuerzo. Nace como la combinación entre las ideas de Monte Carlo y las ideas de programación dinámica. Es así, que a través de los métodos TD, es posible que el agente logre aprender de su entorno sin conocer un modelo de su dinámica. La principal diferencia entre los tres métodos se presenta en las aproximaciones que realiza cada uno para resolver el problema de predicción. En el algoritmo 1 se presenta de forma detallada el procedimiento para el desarrollo del aprendizaje de diferencia temporal.

Inicializar Q(s) arbitrariamente, π es la política para poder evaluar el estado actual

Repetir (Para cada episodio)

Inicializar estado, s

Repetir (Para cada paso del episodio)

$a \leftarrow$ acción dada por la política π para el estado s

Aplicar la acción a; observar recompensa r y estado siguiente $s_{(t+1)}$

$$Q(s) \leftarrow Q(s) + \alpha[r + \gamma * Q(s_{(t+1)}) - Q(s)]$$

$s \leftarrow s_{(t+1)}$

Hasta que s sea el estado objetivo o "*Goal"*

Nota:

$s =$ Estado actual

$s_{(t+1)} =$ Estado siguiente

$a =$ Acción actual

$\alpha =$ Tasa de aprendizaje

$\gamma =$ Factor de descuento

Algoritmo 1. Forma general del algoritmo de aprendizaje de diferencia temporal.

En el algoritmo 1 se observa que se realiza una inicialización de $Q(s)$ de forma arbitraria, además de tener en cuenta que π es la política para evaluar, lo anterior se repite para cada episodio. Luego, es inicializada la variable s, que corresponde a los estados del algoritmo, esto se repite para cada episodio teniendo en cuenta las recompensas obtenidas y el comportamiento del agente en el siguiente estado. De esta manera y teniendo en cuenta estudios previos que llevan a cabo una comparación entre los métodos TD y Monte Carlo, se logra observar la principal ventaja del primero sobre el segundo, donde este no requiere de un modelo del entorno, permitiendo tener mayor aplicabilidad a situaciones reales, además de que estos métodos se implementan de forma natural, siendo totalmente incrementales, aprendiendo así, de cada transición que realizan independientemente de las acciones posteriores que se tomen.

6.2. *Q-Learning*

Teniendo en cuenta lo descrito anteriormente, es posible definir una función $Q\ (s, a)$ para el algoritmo de Q-Learning como se muestra en la ecuación (1).

$$Q\ (s_t, a_t) = max\ (R_{t+1}) \tag{1}$$

Donde se representa la recompensa máxima futura que se recibe cuando se toma una acción a en un estado s. Es importante resaltar que el objetivo de la ecuación (1) es obtener el mejor puntaje al final del entrenamiento del agente, haciendo uso de una política óptima, por medio de la cual se escoge la acción que maximiza la recompensa en el estado s, en la sección 8.3 "Desarrollo del algoritmo de *Q-Learning*" se observan más detalles al respecto.

Se debe tener en cuenta, que la función $Q\ (s, a)$ puede ser expresada en términos del valor Q del siguiente estado (s_{t+1}) ó (s') como se muestra en la ecuación (2), en el desarrollo de este documento se representará el siguiente estado como (s_{t+1}).

$$Q^{(t+1)}(s_t, a_t) \leftarrow (1 - \alpha) * Q\ (s_t, a_t) + \alpha * (r_t + \ \gamma * \ maxQ\ (s_{t+1}, a)) \tag{2}$$

Donde $Q\ (s_t, a_t)$ es el valor anterior de Q, (α) es la tasa de aprendizaje, (r_t) la recompensa, (γ) es el factor de descuento, y $maxQ\ (s_{t+1}, a)$ permite estimar el valor futuro óptimo, cada uno de estos valores se encuentran mejor definidos en la sección de 8.3 "Desarrollo del algoritmo de *Q-Learning*" con un enfoque hacia los objetivos del proyecto. Cabe resaltar que la ecuación (2), donde se expresa Q en términos del siguiente estado, es conocida como la ecuación de Bellman, siendo la base principal de *Q-Learning*. En el algoritmo 2 se muestra el procedimiento para el desarrollo de este.

Inicializar $\mathrm{Q}(s, \mathrm{a})$ arbitrariamente

Repetir (Para cada episodio):

Inicializar estado,s

Repetir (para cada paso del episodio):

Elija a en s , usando la política derivada de Q

Tome la acción a, obtener recompensa r y estado futuro s_{t+1}

$$Q\ (s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s_{t+1}, a_{t+1}) - Q(s,a)]$$

$s \leftarrow s_{t+1}$

Hasta que s sea el estado objetivo o "*Goal"*

Nota:

$s =$ Estado actual

$s_{t+1} =$ Estado siguiente

$a =$ Acción actual

$\alpha =$ Tasa de aprendizaje

$\gamma =$ Factor de descuento

Algoritmo 2. Forma general del algoritmo de Q-Learning.

Cada uno de los parámetros definidos en el procedimiento del algoritmo de *Q-Learning*, son desarrollados de forma más completa en la sección 8, donde se lleva a cabo el diseño y ejecución del proyecto.

6.3. Sarsa

Este algoritmo fue explorado por primera vez en 1994 por Rummery y Niranjan, el nombre lo adoptó en 1996 Rich Sutton*, Sarsa*, es llamado así por "*state-action-reward-state-action*", como es descrito en [18]. Para aplicar este algoritmo, se debe tener en cuenta inicialmente aprender la función de valor de acción en lugar de la función de valor estado, donde se debe estimar el valor de $Q = (s, a)$, donde $'\pi'$ corresponde a la política actual que se está usando, 's' los estados y 'a', a las

acciones del agente. Matemáticamente, expresado como se muestra en la ecuación (3).

$$Q(s_t, a_t)) \leftarrow Q(s_t, a_t) + \alpha\,[r_{t+1} + \gamma Q(s_{t+1,} a_{t+1}) - Q(s_t, a_t)] \quad (3)$$

Donde a_{t+1} es la acción que se realiza en el siguiente estado, bajo la política actual. En el algoritmo 3, se muestra de forma general el procedimiento para el desarrollo del algoritmo de control *Sarsa.*

Inicializar $Q(s, a)$ arbitrariamente

Repetir (Para cada episodio)

Inicializar estado, s

Escoger la acción a del estado s, usando la política derivada de Q (e.g.,ε-greedy)

Repetir (Para cada paso del episodio):

Aplicar la acción a, observar la recompensa r y el estado siguiente s_{t+1}

Escoger la acción a del estado s, usando la política derivada de Q (e.g.,ε-greedy)

$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma * Q(s_{t+1}, a_{t+1}) - Q(s, a)]$

$s \leftarrow s_{t+1}, a \leftarrow a_{t+1}$

Hasta que s sea el estado objetivo o "*Goal"*

Nota:

$s =$ Estado actual

$s_{t+1} =$ Estado siguiente

$a =$ Acción actual

$\alpha =$ Tasa de aprendizaje

$\gamma =$ Factor de descuento

Algoritmo 3. Forma general del algoritmo de control Sarsa.

Con el procedimiento descrito en el algoritmo 3, es posible notar que en *Sarsa*, la función principal que permite actualizar el valor $Q,$ depende del estado actual del agente (s), de la acción que el agente elige (a), de la recompensa (r) que el agente obtiene al elegir la acción, del estado (s_{t+1}) que el agente ingresa después de tomar esa acción y finalmente de la siguiente acción (a_{t+1}) que el agente escoge en su nuevo estado. Teniendo en cuenta lo anterior, se puede llevar a cabo una comparación entre *Q-Learning* y *Sarsa*, observando que en el primer algoritmo se actualiza una estimación de la función de valor 'acción-estado' óptima, basada en

la recompensa máxima de las acciones disponibles, mientras que *Sarsa* aprende los valores de Q asociados a tomar la política que sigue.

7. TRABAJO PRELIMINAR

En esta sección se muestra el trabajo que ha sido desarrollado previamente a este proyecto, publicado con el nombre "*Autonomous Navigation for Exploration of Unknown Environments and Collision Avoidance in Mobile Robots Using Reinforcement Learning*" [43], donde se propuso el uso de aprendizaje por refuerzo para proporcionar habilidades de navegación y adaptación en un ambiente desconocido para un robot móvil, llevando a cabo el uso de la información del sensor, realizando de esta manera el proceso de navegación y exploración de este. Se propone el uso del algoritmo de aprendizaje Q, sin embargo, los estados son establecidos teniendo en cuenta la información proveniente de los sensores. Se propone una política de recompensas que se centra en alejar al robot de los obstáculos y adicionalmente se genera una política de exploración, permitiendo esta última facilitar el proceso de adaptación del robot. El sistema completo, es evaluado utilizando el software de simulación V-REP, en los resultados se observa que el robot aprende a navegar, evitando obstáculos y demostrando su habilidad para explorar ambientes desconocidos. A continuación, se expone el enfoque propuesto en este artículo, los experimentos, los resultados y las respectivas conclusiones.

7.1. Enfoque propuesto

El objetivo principal de esta investigación, como se muestra en [43], fue generar un sistema de aprendizaje que le permitiera a un robot aprender a navegar de forma autónoma, teniendo en cuenta que el sistema de navegación no depende de la ubicación espacial del agente, permitiendo de esta manera aprender a navegar en un entorno desconocido y facilitando el proceso de exploración y adaptación de los nuevos ambientes a los que se podría enfrentar. Es importante resaltar, que el proceso de aprendizaje del agente se basa principalmente es la transición de este a través de una serie de estados gracias a la selección de acciones, estas últimas llevan al agente a realizar cambios. La selección de la acción se puede hacer de forma aleatoria o determinista, según el enfoque dado al comportamiento del robot, una vez que la acción se toma y el robot actualiza su estado, se evalúa el siguiente estado, generando un valor de recompensa y castigo de acuerdo con la conveniencia del nuevo estado. Cabe mencionar, que las recompensas generan cambios en la matriz Q que determinan la calidad de la acción realizada en un estado anterior, siendo de esta manera, la base del conocimiento adquirido y quien determinará a largo plazo el comportamiento del robot. A continuación, se definen los aspectos más importantes para el desarrollo del algoritmo.

7.1.1. Estados

Los estados en el artículo [43], fueron definidos a través del número de sensores con los que cuenta el agente y sus respectivos niveles de detección. Dado que en el sistema los estados son discretos, la información que proviene de estos debe ser discretizada para obtener un número finito de estados. De esta manera, la dimensionalidad de los estados del sistema puede ser representado a través de la expresión (4), tomada de [43].

$$SD = (Dm)^{n_s} \quad (4)$$

Donde Dm es la cantidad de niveles generados por la medición de la distancia del sensor y n_s corresponde al número de sensores del robot. Se debe tener en cuenta que la cantidad de estados crece exponencialmente de acuerdo con la relación que existe entre los estados discretos y el número de sensores del agente.

7.1.2. Acciones

Las acciones son definidas como todos los movimientos que pueden ser realizados por el robot, como por ejemplo ir hacia delante, girar a la izquierda o a la derecha. Se debe resaltar, que, en el caso de los robots omnidireccionales, las acciones son discretizadas. Los estados y las acciones del sistema definen el tamaño de la matriz Q, por lo que es necesario mantener un bajo dimensionamiento de estos para simplificar el proceso de aprendizaje.

7.1.3. Enfoque de aprendizaje por refuerzo

En el desarrollo de este artículo, fue usada la ecuación clásica de *Q-Learning*. Sin embargo, fue modificada la forma en la que se actualizan los valores Q, teniendo en cuenta que los estados se definen por la medida de las distancias de los sensores, siendo complejo saber cuál sería el nuevo estado del agente antes de ejecutar la acción. De esta manera, la actualización de los valores de la matriz Q se realizan después de la ejecución de la acción, logrando que el valor para actualizar es el valor del estado anterior $Q(s_{t-1}, a_{t-1})$, basado en el nuevo estado $Q(s_t, a)$ y la política de recompensas como se indica en la ecuación (5).

$$Q(s_{t-1}, a_{t-1}) \leftarrow (1-\alpha)Q(s_{t-1}, a_{t-1}) + \alpha(R + \gamma max Q(s_t, a)) \quad (5)$$

Donde, $Q(s_{t-1}, a_{t-1})$ es el valor del estado anterior en la acción seleccionada, $'\alpha'$ es la tasa de aprendizaje, $'R'$ es la política de recompensa y $'\gamma'$ es el factor de descuento. La ecuación anterior, expresa, que la actualización de estos valores se realiza hacia atrás en el tiempo, siendo una forma similar de cómo se aprende en la vida real, teniendo en cuenta que solo se conoce la eficiencia de las acciones después de haberlas ejecutado y evaluado el nuevo estado al que condujo esa

acción, llamándola “experiencia”, los parámetros involucrados en la actualización de los valores Q son explicados en las secciones 7.1.4, 7.1.5, 7.1.6 y 7.1.7.

7.1.4. Política de recompensa

La versión clásica del aprendizaje por refuerzo utiliza una recompensa asociada a cada estado, el objetivo es llegar al estado en que recibe una recompensa máxima. El sistema propuesto en este trabajo preliminar no tiene los estados asociados con la ubicación del agente, es decir, que no hay una ruta exacta que lleve al agente a alcanzar el estado ideal, siendo de esta manera el objetivo principal, mantener al robot lo más alejado posible de los obstáculos al navegar libremente. Dado a que los estados se encuentran determinados por la distancia media de los sensores y la transición de estado se basa en el cambio de esta distancia, la política de recompensa se basa en el tiempo derivado de las mediciones de los sensores y la transición de los estados, como se indica en la expresión (6).

$$R_{(t-1)} = \sum_{i=1}^{n} a_i \frac{dm_i}{dt} + b_i m_{i(t)} \tag{6}$$

Donde se observa que la ecuación de recompensa se encuentra representada por la suma de factores dinámicos y estáticos de las mediciones de los sensores del robot, siendo $a_i \frac{dm_i}{dt}$ la parte dinámica y $b_i m_{i(t)}$ la parte estática, donde n es el número de sensores utilizados por el robot. Es importante resaltar, que la parte dinámica de esta expresión se refiere a una recompensa que depende del tiempo derivado de la medición del sensor y que el factor tiempo, viene dado por la transición de estado, como se muestra en $\frac{dm_i}{dt}$ siendo m_i la medida de la $i-th$ sensor. La recompensa dinámica se multiplica por un factor a_i que puede variar según la posición del sensor, por ejemplo, un sensor ubicado en la parte frontal del robot, podría tener valores mayores a los sensores ubicados en los laterales del mismo, de esta manera, la derivada será positiva cuando el robot se aleje de los obstáculos, convirtiéndose el valor en una recompensa, de lo contrario cuando el agente se acerca a los obstáculos, el valor de la derivada será negativa, convirtiéndose este valor en un castigo en el sistema de aprendizaje.

Para simplificar los cálculos de la recompensa, se puede asumir que la transición del estado se realiza en un régimen de tiempo constante, permitiendo convertir el cálculo de la derivada en una simple diferencia de los valores de los estados, como se muestra en la ecuación (7).

$$\frac{dm_i}{dt} = m_{i(t)} - m_{i(t-1)} \tag{7}$$

Donde $m_{i(t)}$es la medida $i-ésima$ del sensor en el estado actual y $m_{i(t-1)}$es la medida $i-th$ del sensor en el estado previo. La parte estática de la ecuación de recompensa se refiere a la medida del sensor en el estado actual del agente $m_{i(t)}$, para posiciones alejadas de obstáculos, este valor es una recompensa y es un castigo en posiciones cercanas a los mismos, la parte estática se multiplica por el valor b_i, que varía de acuerdo con las lecturas del sensor, dando valores mayores a los sensores frontales del robot.

7.1.5. Tasa de aprendizaje

La tasa de aprendizaje, denotada con la letra (α), indica la tasa de cambio para las actualizaciones de los valores Q en el proceso de aprendizaje, idealmente, se espera que (α) comience con valores cercanos a 1, disminuyendo con el tiempo a valores cercanos a 0. Se debe tener en cuenta, que el decremento de alfa se encuentra relacionado con el número de veces que el valor de $Q(s_{t-1}, a_{t-1})$ ha sido actualizado, se espera que el valor de la tasa de aprendizaje en un tiempo infinito tenderá a algún valor cercano a cero, la expresión para alfa se muestra en la ecuación *(8).*

$$\alpha = \frac{1-\alpha_\infty}{1-e^{(\alpha_s n_{(s,a)}\alpha_{50})}} + \alpha_\infty \qquad (8)$$

Donde, α_∞ es el valor esperado de α cuando el tiempo tiende a infinito $(\alpha(t \to \infty))$, α_s indica la tasa de disminución esperada de α, α_{50} es el número de actualizaciones de $Q(s_{t-1}, a_{t-1})$ requeridas por alfa para alcanzar el 50 % de su valor inicial y finalmente $n_{(s,a)}$ es el número de veces que el valor de $Q(s_{t-1}, a_{t-1})$ ha sido actualizado, en la figura 1 tomada de [43],se muestra el valor de alfa a lo largo del tiempo con parámetros de α_∞= 0.1, $\alpha_{50} = 5$ y diferentes valores de $\alpha_s = 0.5, 0.8, 1.5$.

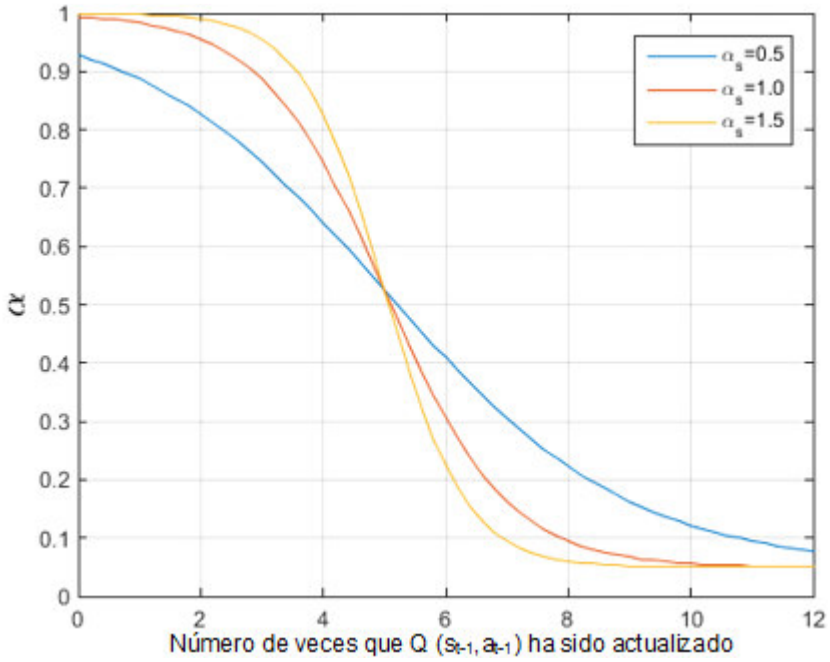

Figura 1. Disminución de la tasa de aprendizaje a través del tiempo.

Fuente: [43]

7.1.6. Factor de descuento

El factor de descuento, representado con (γ), determina el impacto de la recompensa del estado actual para el planteamiento propuesto. Sin embargo, no es primordial para el presente caso, porque la parte estática de la política de recompensas se tiene en cuenta indirectamente esta recompensa, por esta razón se le asigna un valor constante de 0.5 en la mayoría de los casos.

7.1.7. Política de exploración

La política de exploración le da al agente la oportunidad de elegir entre la mejor acción encontrada hasta ese momento o probar con otra acción, a estas selecciones se les da el nombre de explotación y exploración respectivamente. El objetivo de esta política es proporcionar al agente la posibilidad de explorar cuando inicia el proceso de aprendizaje, pero como el agente evoluciona, terminará eligiendo la acción con el mejor Q, es decir, explotación, en la ecuación (9) es descrita la política de exploración para el algoritmo desarrollado.

$$a_t = max(Q(s_{(t)}, a) + \frac{k_a}{n_{(q)} + 1}) \tag{9}$$

Donde, a_t es la acción a seleccionar, $Q(s_{(t)}, a)$ es el valor Q de todas las acciones posibles del estado actual, $n_{(q)}$ es el vector con el número de veces que cada acción posible del estado actual ha sido históricamente seleccionado y k_a es un factor que ayuda a alternar entre la acción con el mejor valor de Q y las demás acciones. Por lo general, k_a debe ser mayor que Q; sin embargo, para la propuesta de este algoritmo, se cumple que $k_a \gg max(R)$, donde la recompensa máxima se obtiene evaluando todas las recompensas, como se muestra en la expresión (6) de la sección 7.1.4. El objetivo de la ecuación (9), es alternar la selección de la acción, entre la mejor acción y las otras posibles acciones del estado actual, inicialmente los valores de Q son cero y k_a comienza dividido por 1, dado que $n_{(q)}$ también es cero para las otras acciones, iniciar el algoritmo de esta manera, permite que cualquier de las posibles acciones sea seleccionada, sin embargo, a medida de que transcurre el tiempo, $n_{(q)}$ crece de acuerdo con el número de veces que ha seleccionado una acción en particular. Cuando $n_{(q)}$ crece, el factor k_a tiende a cero y la política comienza a seleccionar la acción óptima con mayor frecuencia.

7.2. Experimentos y resultados

Se evalúa el sistema de aprendizaje de refuerzo propuesto a través de la implementación de un experimento, este, se utiliza para verificar que el agente aprenda a navegar de forma autónoma sin chocar con los obstáculos a su alrededor, además de analizar los parámetros más importantes del sistema propuesto. El experimento se basa en la puesta en marcha de un robot terrestre con tracción diferencial en un entorno desconocido, el agente utilizado en este experimento es el modelo virtual de un "*Pioneer 3-DX"* y el entorno virtual desconocido es una oficina. El sistema de simulación utilizado es V-REP *(Virtual Robot Experimentation Platform).* Este sistema, permite el uso del modelo de robots virtuales y la construcción de una variedad de escenarios. El escenario y el robot utilizados en las pruebas se muestran en las figuras 2 y 3 respectivamente, tomadas de la referencia [43].

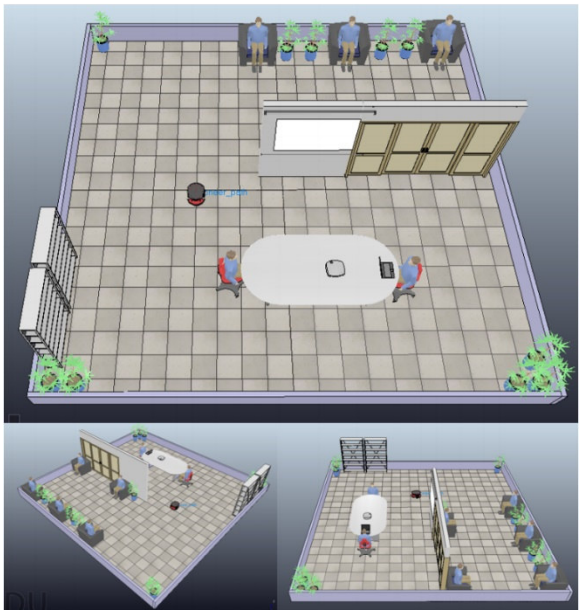

Figura 2. Ambiente de simulación desconocido para el robot

Fuente: [43]

- Robot

El robot utilizado en este experimento es un *Pioneer 3-DX*, el cual tiene tracción diferencial y tres rangos de distancia. Los sensores fueron ubicados, uno en la parte frontal y los otros dos a los lados, como se muestra en la figura 3. El robot se controla configurando la velocidad de las ruedas hacia la izquierda y hacia la derecha.

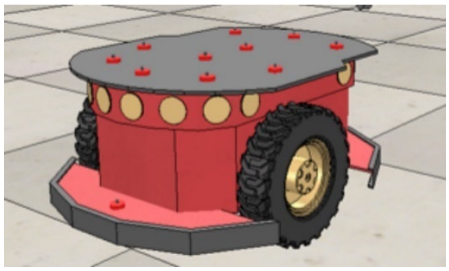

Figura 3. Robot Pioneer 3DX

Fuente: [43]

- Escenario

El escenario, consiste en una oficina con un tamaño de 10x10 metros, con sofás, mesas de reuniones, ordenadores, estanterías, puertas corredizas, muros, plantas y personas. Todos los objetos en el escenario mantienen una relación real entre ellos y el robot.

- Configuración del experimento

Para llevar a cabo el desarrollo del experimento, fueron definidos algunos parámetros importantes, como la velocidad máxima del robot (*maxVel*), siendo esta de $0.5\ m/s$, además de esto, se debe tener en cuenta que el robot tiene 3 sensores (uno ubicado en la parte delantera y dos en los laterales). Para llevar a cabo la discretización de estos, se establecen 5 niveles de distancia, donde el nivel máximo $(L4)$ corresponde a distancias superiores de 4 metros y el mínimo $(L0)$ para distancias menores a 1 metro. Dado que el agente cuenta con 3 sensores y la discretización de estos es de 5 niveles, según la expresión (4) de la sección 7.1.1 el número de estados del sistema es de 125. Los movimientos del robot fueron limitados a tres, donde el agente puede avanzar, girar a la izquierda o girar a la derecha.

Teniendo en cuenta la definición del número de acciones, los niveles de discretización de los sensores y el número de estos, se establece la dimensión de la matriz Q, siendo una matriz de $125x3$. Los coeficientes de la política de recompensa fueron definidos teniendo en cuenta la posición del sensor, de la siguiente manera: $a_{1,3} = 50,\ a_2 = 100,\ b_{1,3} = 5,\ b_2 = 20$, donde los sensores 1 y 3 se encuentran ubicados en los laterales del robot y el sensor 2 se encuentra ubicado en la parte frontal. Los parámetros de la tasa de aprendizaje se definieron como $\alpha_\infty = 0.05,\ \alpha_s = 1\ y\ \alpha_{50} = 8$, el factor de descuento $(\gamma) = 0.5$, finalmente el valor inicial para la política de exploración $k_a = 800$.

- Resultados experimentales

El experimento consistía en permitir que el agente navegara libremente a través del escenario, a medida que el robot interactúa con el entorno el algoritmo se ejecuta, la interacción del algoritmo y el robot son descritos en los siguientes pasos.

- Paso 1: El robot toma la información de los sensores e identifica el estado actual.
- Paso 2: Basado en la política de exploración, el robot decide qué acción realizar y la ejecuta durante un periodo de tiempo establecido.
- Paso 3: El agente toma información de los sensores e identifica el nuevo estado
- Paso 4: Los valores de la matriz *Q* se actualizan en base a la política de recompensa, el estado actual y el estado anterior. Los estados se actualizan y se repite el paso 2.

A la ejecución de los pasos anteriores se le llama episodio, cabe resaltar que, para los fines de este experimento, fueron ejecutados alrededor de 10000 episodios, permitiéndole al agente ajustar sus parámetros y que permitiera navegar sin estrellarse con los obstáculos. Dentro del proceso de aprendizaje algunos factores importantes de análisis fueron el número de choques que tuvo el agente, el camino seguido por el mismo durante el proceso de aprendizaje, la convergencia de los valores $Q(s, a)$ y la política de exploración, como se muestra a continuación:

- Ruta

Durante el proceso de aprendizaje, el robot navega explorando diferentes lugares del entorno, al mismo tiempo el agente mantiene una distancia prudencial con los obstáculos, evitando chocar con los mismos. El camino descrito por el robot es evidencia de las habilidades de exploración y de evasión de obstáculos desarrolladas por el mismo, en la figura 4 se muestra la ruta de navegación del robot a través de 10,000 episodios.

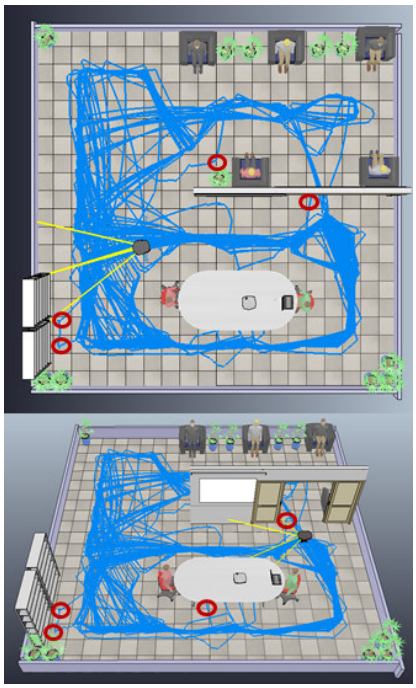

Figura 4. Ruta generada por el robot a lo largo del tiempo en el proceso de aprendizaje

Fuente: [43]

En la figura 4, se muestra el camino seguido por el robot del experimento, donde la línea azul corresponde al camino que este tomaba y los círculos rojos a los lugares donde el agente chocaba con más frecuencia.

- Choques

El incidente de choque es definido como el evento que sucede cuando el robot choca con un objeto de la escena o cuando este pasa demasiado cerca a los objetivos y paredes como se muestra en la figura 4 tomada de [43], la cuantificación de los choques del agente es representada en la gráfica de la figura 5.

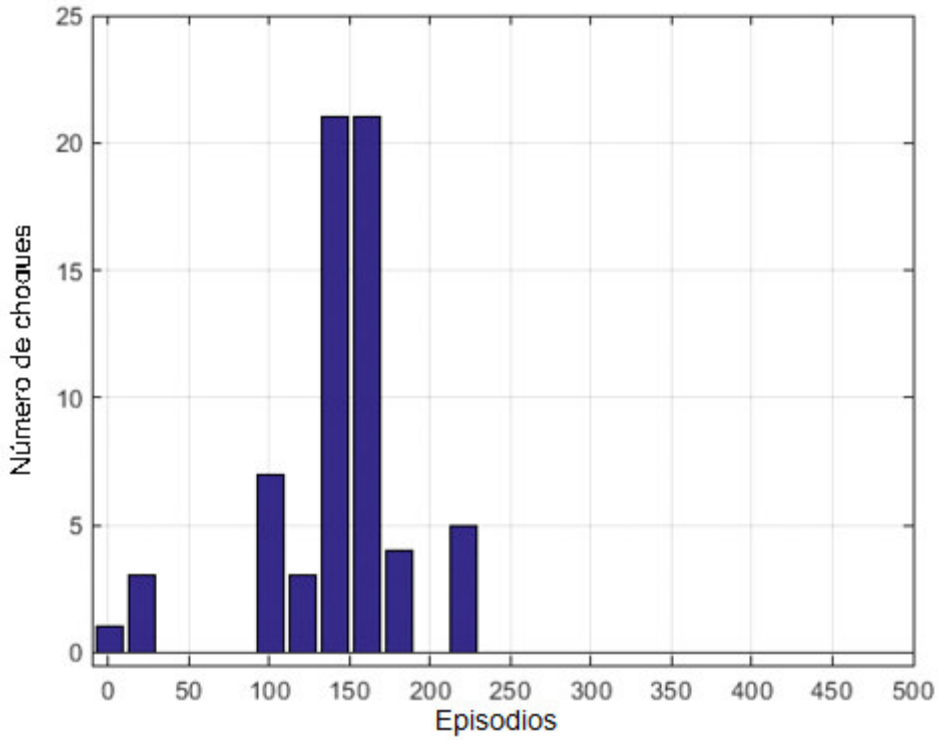

Figura 5. Representación de los choques del agente a través de los episodios de aprendizaje.

Fuente: [43]

En la figura 5, se muestra el número de choques dividido por partes de 20 episodios a través de los primeros 500 episodios. Es posible notar, que después del 230 episodio, el sistema no registró más incidentes de colisión, interpretando de esta manera que el robot aprendió a navegar en el entorno que fue puesto y utilizó el resto de los episodios para ajustar los nuevos valores de Q, para lograr un comportamiento de navegación óptimo.

- Convergencia de los valores de Q

Es importante reconocer, que el proceso de aprendizaje se da por las actualizaciones de la matriz Q a través del tiempo, en párrafos anteriores, se observó que el robot aprendió a navegar, evadiendo obstáculos, en el episodio 230, sin embargo, el agente continuó su proceso de aprendizaje y búsqueda de mejores comportamientos. Los valores de la matriz Q alcanzaron un estado estable, donde las actualizaciones no generan grandes cambios, como se muestra en la figura 6.

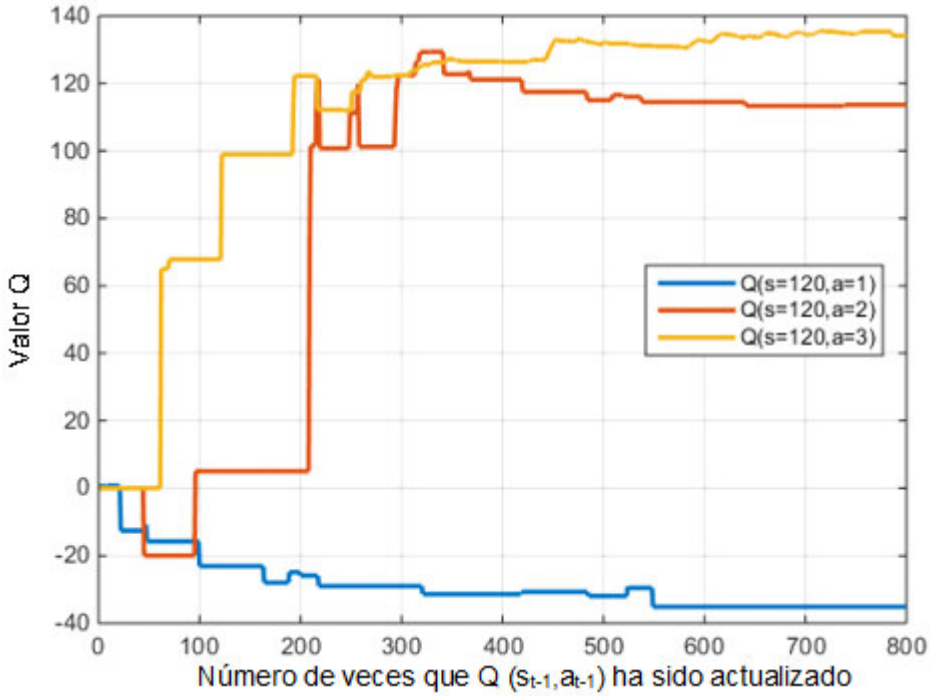

Figura 6. Evolución del aprendizaje del agente a través de los valores de Q

Fuente: [43]

La figura 6 se muestra que después del episodio 700, los valores de $Q\ (s = 120)$ se mantienen prácticamente constantes, es decir que el sistema converge y el agente encontró el mejor grupo de acciones para navegar y evitar los obstáculos. La línea azul, representa $(Q = 120$ y $a = 1)$, la línea naranja es $(Q = 120$ y $a = 2)$ y la línea amarilla es $(Q = 120$ y $a = 3)$.

- Política de exploración

La política de exploración le da al robot la posibilidad de alternar entre dos tipos de comportamientos (explorar o explotar), como se muestra en la selección de la acción, permitiendo de esta manera que se realice la selección del comportamiento, durante este experimento, el robot utilizó la ecuación descrita en la sección 7.1.7. para seleccionar acciones. En la figura 7 *se* muestran los tres posibles valores de selección para las acciones del estado.

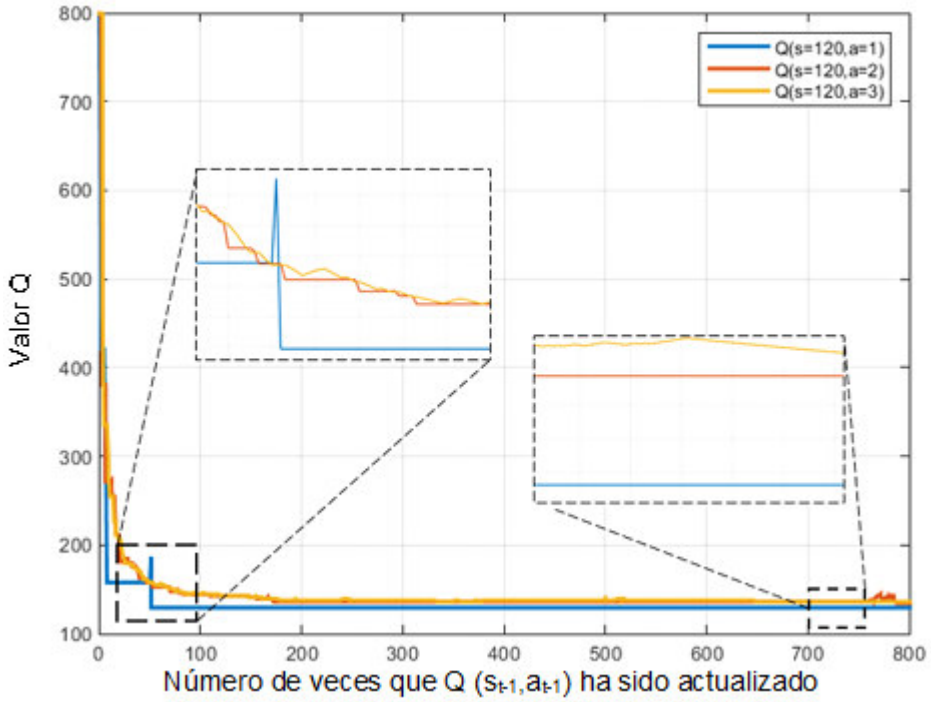

Figura 7. Políticas de explorar y explotar a través del proceso de aprendizaje

Fuente: [43]

Como fue descrito anteriormente, se observa que los valores de selección disminuyen con el tiempo, dando la oportunidad de alternar las acciones. La figura 7 de [43] muestra la forma como se alternan los valores de selección alrededor del episodio 50, esta variación de valores le da al sistema la oportunidad de alternar la selección de acción y ejecutar un comportamiento de exploración. Se observa, además, que alrededor del episodio 720, los valores se mantienen constantes, haciendo siempre la misma selección de acción, generando de esta manera una explotación del comportamiento e indicando que el sistema converge y que el agente encuentra la acción óptima.

8. DISEÑO Y EJECUCIÓN DEL PROYECTO

8.1. HERRAMIENTAS USADAS

8.1.1. AMBIENTE DE SIMULACIÓN VIRTUAL (V-REP)

Existen varios simuladores de robots como lo son V-REP, GAZEBO, ARGOS, entre otros. Sin embargo, para este proyecto se usó V-REP por las ventajas que representa frente a los otros simuladores, en la tabla 1 se puede evidenciar las ventajas (sombreado de color verde) y desventajas (sombreado de color rojo) que tienen estos entornos virtuales, esta comparación se llevó a cabo con la ayuda de la referencia [30].

V-REP	**GAZEBO**	**ARGoS**
Disponible para MacOS, Linux y Windows. Los paquetes binarios están disponibles para todas las plataformas.	Disponible para MacOS, Linux y Windows. Un paquete binario sólo está disponible para Linux Debian. Gazebo se instala a través de la línea de comandos utilizando administradores de paquetes de terceros en otros sistemas.	Disponible para MacOS y Linux. Los paquetes binarios están disponibles para Linux. En MacOS, ARGoS se instala a través de la línea de comandos utilizando un administrador de paquetes de terceros.
CAPACIDADES INCORPORADAS		
Los motores de física predeterminados incluyen: Bullet 2.78, Bullet 2.83, ODE, Vortex y Newton.	Solo el motor de física ODE está disponible por defecto. Sin embargo, es posible construir Gazebo desde la fuente con un motor de física diferente.	De forma predeterminada, están disponibles motores de física 2D y 3D personalizados con capacidades muy limitadas.
Incluye un código y un editor de escenas.	Incluye un código y un editor de escenas.	Incluye un editor de guiones Lua pero ningún editor de escenas
Las mallas pueden ser manipuladas (por ejemplo, cortadas) por robots en tiempo real.	No hay manipulación de malla disponible.	No hay manipulación de malla disponible.
Los objetos de la escena pueden ser completamente interactuados con (por ejemplo, movidos o agregados) por el usuario durante la simulación. El mundo vuelve a su estado original cuando se reinicia la simulación.	Los objetos de la escena pueden ser completamente interactuados con (por ejemplo, movidos o agregados) por el usuario durante la simulación. El mundo no vuelve a su estado original cuando se reinicia la simulación.	Los objetos de la escena pueden ser movidos por el usuario durante la simulación.
Las salidas incluyen video, gráficos de datos personalizados y archivos de texto.	Las salidas incluyen archivos de registro de simulación, marcos de video como imágenes y archivos de texto.	Las salidas incluyen cuadros de video como imágenes y archivos de texto.
Incluye sistemas de partículas.	No hay sistemas de partículas disponibles.	No hay sistemas de partículas disponibles.

ROBOT Y OTROS MODELOS		
Proporciona una gran variedad de robots, incluyendo robots bipedales, hexápodos, con ruedas, voladores y con forma de serpiente. También proporciona una gran cantidad de actuadores y sensores de robot.	Una biblioteca menos diversa de robots predeterminados, que en su mayoría incluye robots con ruedas y voladores. Los modelos de robots de terceros están disponibles, pero su documentación suele ser deficiente.	A bastante pequeña biblioteca de robots, solamente incluyendo el e-puck , ojo-bot ,Kilobot , marXbot, y Spiri robots.
MÉTODOS DE PROGRAMACIÓN		
Una escena se guarda en un formato especial V-REP. Por lo tanto, toda la edición de escenas debe realizarse utilizando la interfaz V-REP.	Una escena se guarda como un archivo XML. Esto hace posible, por ejemplo, crear un guión de bash que cambie la escena y luego ejecute una simulación.	Una escena se guarda como un archivo XML. Esto hace posible, por ejemplo, crear un guión de bash que cambie la escena y luego ejecute una simulación.
Hay varias opciones para la funcionalidad de programación de archivos, incluidos scripts adjuntos a robots, complementos, nodos ROS o programas separados que se conectan a V-REP a través de RemoteAPI.	La funcionalidad se puede programar como complementos de C ++ compilados o mediante programas ROS . La falta de secuencias de comandos hace que sea difícil ejecutar pruebas rápidas con soluciones ad-hoc.	Los robots se pueden programar a través de scripts Lua o en C ++.
INTERFAZ DE USUARIO (UI)		
No se experimentaron problemas de congelación con la interfaz	La interfaz se congeló varias veces y el programa, y algunas veces la computadora, tuvo que reiniciarse. Esto ocurrió, por ejemplo, al editar modelos de robot, iniciar o detener la simulación, y en otros casos.	No se experimentaron problemas de congelación con la interfaz.
Toda la funcionalidad es bastante intuitiva y sigue las convenciones generales conocidas de aplicaciones similares.	La usabilidad de la interfaz de usuario es relativamente baja. Por ejemplo, la barra de herramientas de la aplicación superior a veces desaparece, no es posible copiar y pegar varios objetos, o guardar una escena en el mismo archivo después de realizar cambios en ella.	La interfaz de usuario es muy limitada, pero toda la funcionalidad es bastante intuitiva y sigue las convenciones generales conocidas de aplicaciones similares.

Tabla 1. Comparación del software que podrían usarse para las simulaciones de las escenas del agente. [38]

Teniendo en cuenta la tabla 1, se puede evidenciar que el entorno de simulación virtual V-REP nos permite tener varias ventajas, principalmente por su interfaz de usuario ya que es muy intuitiva lo cual facilita su uso, en los métodos de programación permite usar el lenguaje de programación LUA dentro del simulador, o da la opción de usar una API remota el cual permite controlar la simulación desde una aplicación externa o algún hardware externo. La simulación se puede llamar desde un programa externo escrito en lenguajes como C/C++, JAVA, Python,

MATLAB u Octave y en LUA, V-REP tiene su versión gratuita para estudiantes la cual fue usada para este proyecto.

Los resultados de este proyecto son una base para resolver problemas de la navegación autónoma de un UAV en ambientes de desastres naturales, como la búsqueda y rescate de víctimas o el conteo de estas. Es por ello por lo que, para el desarrollo de este, se simulará un ambiente de un incendio forestal donde el UAV debe lograr llegar a ciertos objetivos que se fijarán durante el desarrollo del algoritmo de aprendizaje, el ambiente propuesto y desarrollado en V-REP se muestra en la figura 8.

Figura 8. Ambiente en V-REP sobre el que se realizan las pruebas del algoritmo.

Fuente: Autores

8.1.2. SOFTWARE MATEMÁTICO

Para llevar a cabo el desarrollo del algoritmo para este proyecto, fue escogido el software matemático Matlab, conocido en todo el mundo como un sistema de cómputo numérico, amigable para el desarrollador, con un entorno de desarrollo integrado, conocido por sus siglas (IDE) y en el que se lleva a cabo el despliegue de un algoritmo en un lenguaje llamado, lenguaje M. Matlab, puede encontrarse en diferentes plataformas, algunas de ellas son Windows, Linux y MAC. En la Universidad Santo Tomás se cuenta con la licencia de este software, siendo esta una de la razones por las que fue escogida la plataforma para el desarrollo del

algoritmo de la navegación autónoma de un UAV, además de ser Matlab reconocido a nivel mundial por los manejos y análisis que se le hacen a matrices, la representación de datos que se puede realizar en esta plataforma y la fácil comunicación con programas de otros lenguajes, siendo aspectos importantes, debido a que se encuentran fuertemente relacionados con el proyecto que se busca implementar.

Matlab, como software de desarrollo cuenta con muchas herramientas que facilitan el trabajo investigativo, combinando un entorno de escritorio adaptado para lograr un análisis iterativo de datos, incluyendo además un editor para realizar scripts que se combinan con el código que se está desarrollando. Por ser un software tan bien acogido entre la comunidad académica, investigativa y empresarial, cuenta con un banco significativo de información, donde pueden ser resueltas las dudas y hacer posible el mejoramiento de los proyectos que se desarrollan bajo esta plataforma. Además, Matlab se encuentra a la par de temas relevantes en la actualidad, ofreciendo en lo que respecta a *Machine Learning* modelos y aplicaciones para clasificación y regresión donde se obtienen resultados de forma rápida, así como la posibilidad de llevar a cabo una comparación de modelos basados en métricas y emplear optimización de hiper parámetros integrada en las funciones de entrenamiento de modelos para el código propio, ajustando los parámetros de una manera personalizada según los resultados que deseen ser obtenidos.

Otra de las ventajas de usar Matlab para el desarrollo de este proyecto, consiste en que la implementación de los modelos de *Machine Learning* puede ser desarrollada donde se desee, es decir, incluir código C/C++, sistemas de IT empresariales o hasta la misma nube, permitiendo generar un código C autónomo a partir del código desarrollado en Matlab para crear modelos implementables con una velocidad de predicción de alto rendimiento y poca necesidad de memoria, permitiendo de esta manera ampliar las aplicaciones que se pueden obtener al finalizar este proyecto.

8.2. COMUNICACIÓN ENTRE V-REP Y MATLAB

Para lograr la comunicación entre V-REP y Matlab, es importante tener en cuenta que se realizará una única vez a menos de que existan actualizaciones para alguno de los dos software, además, que V-REP ofrece una API remota para controlar la simulación desde una aplicación externa. De esta manera, el primer paso para realizar una comunicación exitosa entre estos dos software consiste en la importación de algunas funciones, para esto, es necesario dirigirse al directorio donde está instalado V-REP, siguiendo la ruta que se muestra en la figura 9.

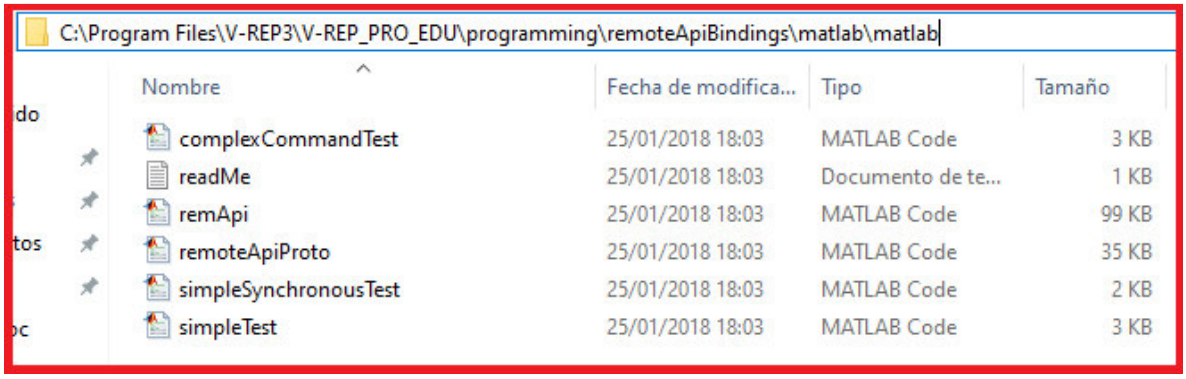

Figura 9. Ruta de los archivos para la comunicación Matlab-VREP.

Fuente: Autores

Al estar ubicados en ese punto, se deben copiar todos los archivos terminados en (.m) para luego ser pegados en el directorio donde se tiene el proyecto de Matlab. Siguiente a esto, es necesario regresar a la ruta inicial y seleccionar la carpeta de 32 bit o 64 bit, según corresponda con el sistema operativo donde se requiera la instalación, como se muestra en la figura 10.

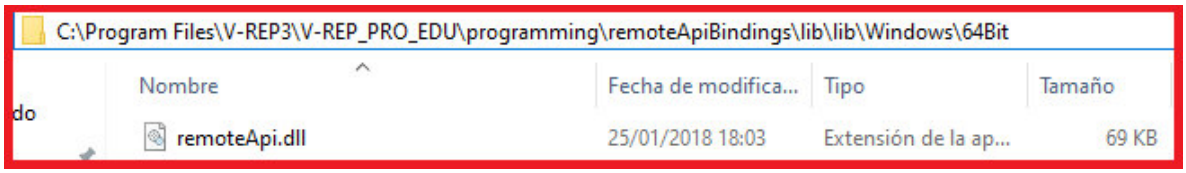

Figura 10. Ruta de la librería a usar para la comunicación entre Matlab y V-REP.

Fuente: Autores

Se encuentra un archivo (.dll) el cual es la librería con la cual se puede realizar la conexión desde este Sistema Operativo de Windows, en el caso de que sea un Sistema Operativo de Linux se encontrará un archivo (.so), de la misma manera que en el paso anterior, se procede a escoger el que corresponda al sistema operativo donde se requiera la información y este archivo hará parte de los ficheros de la carpeta del proyecto de Matlab junto con los demás archivos copiados anteriormente. Para verificar que los pasos se hayan realizado de forma correcta, se abre el software matemático Matlab ubicándose en la carpeta del proyecto y en la ventana izquierda del mismo, donde se encuentra el nombre "*Current Folder*" en donde deben estar todos los archivos que acabaron de ser copiados, como se muestra en la figura 11.

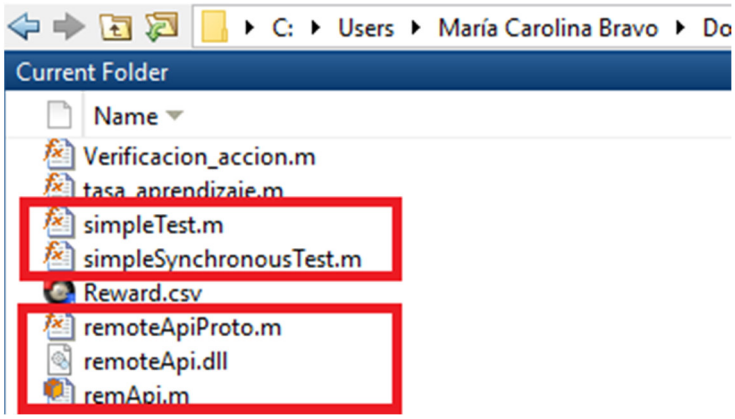

Figura 11. Archivos copiados en la carpeta del proyecto de Matlab para la conexión entre el software matemático y el simulador V-REP.

Fuente: Autores

Como segundo paso, se lleva a cabo la apertura del software V-REP, en el que se debe preparar una escena según los requerimientos del proyecto, luego debe ser abierto el editor de scripts (botón en la barra lateral izquierda), al pulsar este botón, se abrirá un panel con todos los scripts actuales que hay en la escena, se debe tener en cuenta que el script principal, el que se encarga de gestionar todos los cálculos sólo deberá ser modificado en casos especiales donde se requiera una configuración totalmente personalizada. Para crear un nuevo *script*, se debe pulsar el botón *"Insert new script*", de esta manera aparecerá un desplegable donde se puede llevar a cabo la selección del tipo de script que se desea realizar, para el caso de la conexión entre V-REP y Matlab, se escoge "*Child script (threaded)",* este aparece en color azul y cabe destacar que no se encuentra asociado a ningún objeto. Seguido a esto, se abre el desplegable y se escoge un objeto de la escena, es importante destacar que el objeto que se escoja, sea un objeto que no se vaya a destruir durante la simulación, como es el caso de "*DefaultCamera*" que es la cámara que permite ver y moverse por la escena de la simulación, debido a que este script habilitará la conexión entre el simulador (V-REP) y Matlab.

Luego de haber asociado el script, este, debe ser modificado, añadiendo la instrucción *"simExtRemoteApiStart (19999)"* al inicio, permitiendo de esta manera crear un puerto desde el que Matlab podrá llevar a cabo la conexión con la simulación, como se muestra en la figura 12.

```
function sysCall_init()
    simRemoteApi.start(19999)
    smokeDrawingObjType=sim.drawing_discpc
        sim.drawing_facingcamera+sim.drawi
    smokeCont=sim.addDrawingObject(smokeDr
    oh=sim.getObjectAssociatedWithScript(s
    s=sim.getObjectSizeFactor(oh)
```

Figura 12. Configuración para la conexión entre V-REP y Matlab.

Fuente: Autores

Como paso final, se abre el software matemático Matlab y en el archivo principal (.m) del proyecto es copiado y pegado el siguiente *script:*

```
vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
if (clientID > -1)
    disp('Connected')
```

Figura 13. Configuración para la conexión entre V-REP y Matlab.

Fuente: Autores

En la figura 13, se muestra la configuración entre V-REP y Matlab, donde la primera línea crea el objeto V-REP y además carga la librería con todas sus funciones, la tercera línea permite inicializar la simulación llamando la función *simxStart*, teniendo en cuenta 6 parámetros, el primero de ellos, la dirección IP a la que debe conectarse, es decir la dirección IP de V-REP, por defecto se tiene el valor 127.0.0.1 como se muestra en la figura 13, seguido a este valor, se tiene el puerto al que debe conectarse, es importante resaltar, que este número debe coincidir con el número del puerto que se creó con la instrucción *"simExtRemoteApiStart()"* del *script* del simulador, el tercer parámetro debe encontrarse en el valor de "*True*" para que Matlab intente conectarse con el simulador y logre una conexión exitosa o se termine el tiempo de espera, el siguiente valor en *"true"* permite que Matlab no intente reconectarse con el simulador y se presenta una interrupción en la comunicación de los dos programas, el siguiente valor, corresponde al tiempo de espera (milisegundos) en el que Matlab intentará conectarse con V-REP y finalmente se tiene el valor de la frecuencia de envío de datos que es un valor estándar (5). En la última línea del *script* de la figura 13, se tiene una interrupción de Matlab y V-REP, siendo necesaria, para que no haya problemas al conectarse nuevamente a ese mismo puerto. Para comprobar que la conexión ha sido exitosa, al correr el *script*, en la variable *clientID* tiene un valor diferente de (-1) de lo contrario, la conexión ha fallado y se deberán revisar todos los pasos anteriores.

8.3. DESARROLLO DEL ALGORITMO DE Q-LEARNING

8.3.1. DETERMINACIÓN DE ESTADOS Y ACCIONES

Para llevar a cabo la determinación de estados y acciones se tuvieron en cuenta tres aspectos principales, como se muestra a continuación:

1. Sensores

Para la detección de los obstáculos en la escena de la simulación, fueron establecidos cuatro sensores, uno por cada dirección (Norte, sur, derecha, izquierda) cada uno, con un rango de detección de dos metros. Para su análisis se discretiza esta medición en tres niveles {0, 1, 2} como se muestra en la figura 14.

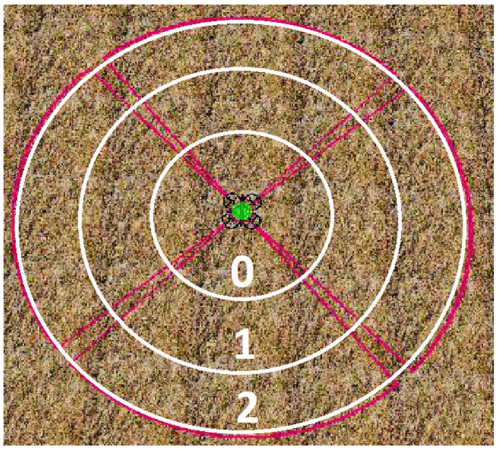

Figura 14. Niveles de los sensores del agente

Fuente: Autores

Los sensores que se escogieron en la escena son sensores de proximidad tipo disco, encontrados en el software de simulación como *disk-type*. Se usó este tipo de sensor, teniendo en cuenta los tipos de movimientos o acciones que puede realizar el agente, considerándolo el más adecuado para la aplicación desarrollada. El número de sensores ($S = 4$) y el rango que maneja cada uno ($RS = 3$), siendo 3 el número de rangos que tiene el sensor es importante para entender la ecuación al final de este ítem que permite calcular el número de estados del algoritmo.

2. Distancia

Otro aspecto importante para la determinación de los estados en el algoritmo es la distancia inicial (*distancia_max*) del agente con respecto al objetivo, ya que esta medida es la base para ver el avance del aprendizaje del agente. Esta medida se discretiza en tres umbrales {0, 1, 2}, siendo el número 2 la distancia más lejana que existe entre el agente y el objetivo, 1 la distancia media entre estos dos y 0 la distancia mínima que puede presentarse. Se debe tener en cuenta el número de

umbrales de distancia para el cálculo de estado ($RD = 3$). En la figura 15 se muestra gráficamente la explicación.

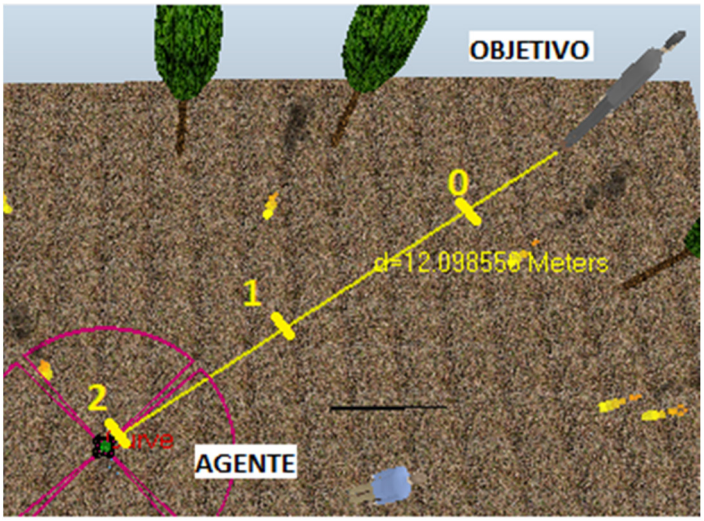

Figura 15. Niveles de distancia del agente.

Fuente: Autores

3. Direcciones

Para permitir que el algoritmo se pueda adaptar a cualquier ambiente donde se desee entrenar el agente, es necesario tener en cuenta la dirección hacia dónde se encuentra el objetivo, para ello, se tomaron las 4 direcciones principales (Norte, Sur, Este y Oeste) como se muestra en la figura 16 siendo el dron el punto (0,0) del plano.

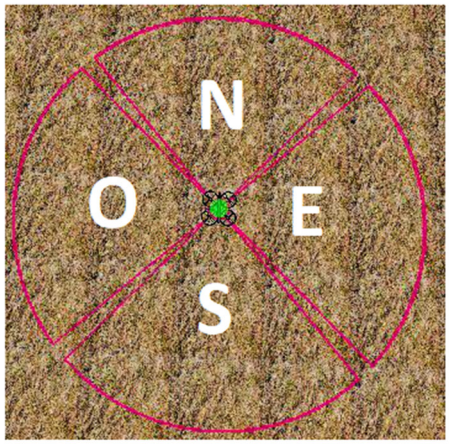

Figura 16. Direcciones del agente

Fuente: Autores

Para el cálculo final de estados se debe tener en cuenta el número de direcciones ($ND = 4$) usadas.

4. Determinación del número de estados:

Teniendo en cuenta los factores expuestos anteriormente es posible llevar a cabo la determinación del número de estados que serán usados para el desarrollo del algoritmo, como se muestra a continuación.

- Variables para tener en cuenta:

1. $S = 4$ (Número de sensores usados)
2. $RS = 3$ (Número de umbrales por sensor)
3. $RD = 3$ (Número de umbrales de distancia entre el agente y el objetivo)
4. $ND = 4$ (Número de direcciones que puede tomar el dron)
5. num_e= 972 (Número de estados a usar)

- Cálculo del número de estados:

$$num\_e = RS^S * RD * ND \quad (10)$$

Reemplazando en la ecuación (10), se tiene que:

$$num\_e = 3^4 * 3 * 4$$

$$num\_e = 972$$

Con la ecuación (10), es definido el número de estados usados para el desarrollo del algoritmo, resultando un valor de 972 estados. Se debe resaltar que el número de estados se definió teniendo en cuenta los aspectos principales que afectan el agente y su llegada al objetivo, siendo necesario analizar muy bien el número de factores alrededor del mismo.

5. Determinación del número de acciones:

Para determinar los movimientos del agente, se tuvieron en cuenta 8 acciones (*num_a*) definidas como (Norte, Sur, Oeste, Este, Noroeste, Noreste, Sureste y Suroeste), estas pueden visualizarse de forma más clara en la figura 17.

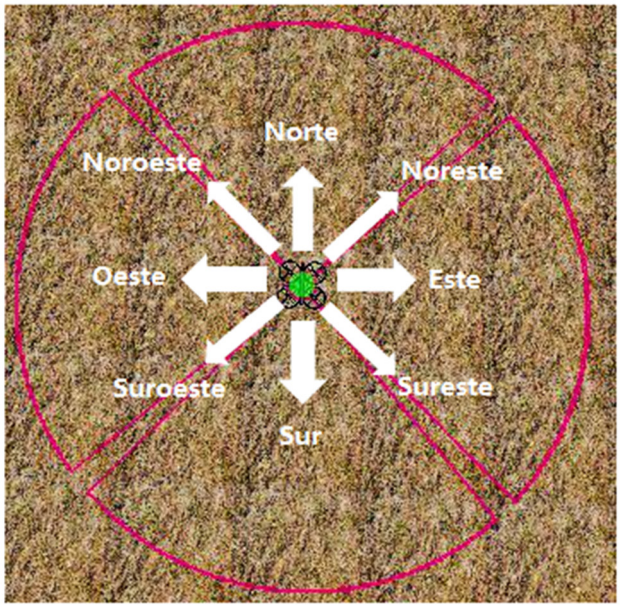

Figura 17. Acciones que puede tomar el agente.

Fuente: Autores

8.3.2. FACTOR DE DESCUENTO

Teniendo en cuenta el algoritmo *de Q-Learning* se define el factor de descuento (γ) como un valor entre el rango de 0 y 1, si el valor es más cercano a "0" el agente no se esfuerza tanto por alcanzar una alta recompensa en su entrenamiento, mientras que, si el valor es muy cercano a "1", el dron se esforzará por hallar una alta recompensa. Teniendo presente lo anterior el valor definido para Gamma (γ) en el algoritmo desarrollado es de 0.95, permitiendo al agente obtener altas recompensas.

8.3.3. TASA DE APRENDIZAJE

La tasa de aprendizaje (α) es un valor que determina en qué medida la información recién adquirida por el agente, actualiza su conocimiento de la tarea que realiza y del entorno con la nueva información, se encuentra definido en un rango entre 0 y 1, donde:

- Si $\alpha \approx 0$ el agente no aprende, solo usa el conocimiento previo adquirido a través de la experiencia.
- Si $\alpha \approx 1$ el agente solo toma la información más reciente ignorando el conocimiento previo para explorar más opciones.

Para determinar (α), se puede establecer un valor fijo o un valor calculado según la experiencia que ha tenido el agente, para este algoritmo la tasa de aprendizaje será

un valor calculado teniendo en cuenta el número de veces que ha llegado a una cierta posición de la matriz Q, donde esta posición, se encuentra definida por el estado en que se encuentra el agente y la acción que toma llevándolo a un nuevo estado. Las variables para tener en cuenta son:

$$Nveces = Número\ de\ veces\ que\ el\ agente\ ha\ estado\ en\ una\ posición\ de\ la\ matriz\ Q$$

$$Iteraciones = Número\ total\ de\ iteraciones\ que\ tendrá\ el\ agente\ durante\ todo\ el\ entrenamiento$$

De esta manera, se expresa la tasa de aprendizaje como se muestra en la expresión (11).

$$\alpha = 0.90 - (Nveces * (100/Iteraciones)) \quad (11)$$

La ecuación (11), describe la forma en la que se lleva a cabo el cálculo de la tasa de aprendizaje, es decir (α), factor importante en el algoritmo para determinar la relevancia de la información adquirida por el UAV durante el entrenamiento.

8.3.4. CÁLCULO DEL ESTADO DEL UAV

El algoritmo de *Q-Learning* se basa en el análisis del entorno por parte del agente (UAV), a este aspecto se llama estado, a cada estado se le asignan acciones, estas son los movimientos que el agente puede realizar. Para calcular el estado, el UAV toma factores del entorno, estos elementos dependen del problema que se quiere solucionar a través del aprendizaje por refuerzo, para el desarrollo de este proyecto se tuvieron en cuenta los siguientes factores:

- **Detección de obstáculos por parte de los sensores:**

El agente (UAV) tiene cuatro sensores de proximidad para la detección de obstáculos, los valores dados por los sensores están discretizados en 3 niveles como se muestra en la Figura 14, definidos de la siguiente manera:

1. Sensor Frontal (FS)
2. Sensor Posterior (BS)
3. Sensor lado Izquierdo (LS)
4. Sensor lado Derecho (RS)

En la figura 18 se muestran gráficamente.

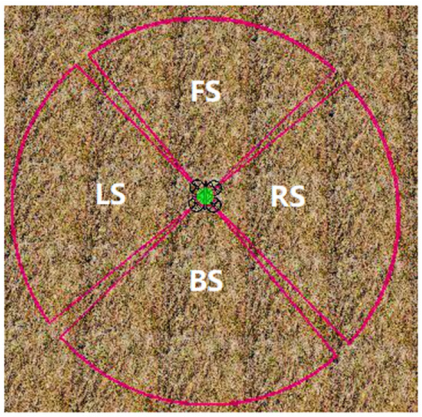

Figura 18. Definición de los cuatro sensores de proximidad del agente

Fuente: Autores

- **Distancia del agente con respecto al objetivo:**

El agente toma la distancia inicial que tiene con respecto al objetivo, en base a esta medida el valor de distancia se discretiza en 3 niveles como lo muestra la Figura 15.

- **Orientación del agente con respecto al objetivo:**

Un factor muy importante es la orientación del UAV con respecto al objetivo, ya que permite saber la dirección hacia donde éste deberá dirigirse y tomar la acción más eficiente. Para la orientación se tomaron en cuenta 4 direcciones como se muestra en la Figura 16, para calcular la dirección se tuvo en cuenta el ángulo con respecto al objetivo y las coordenadas (X, Y) tanto del agente como el del objetivo, expresado gráficamente como se muestra en la figura 19.

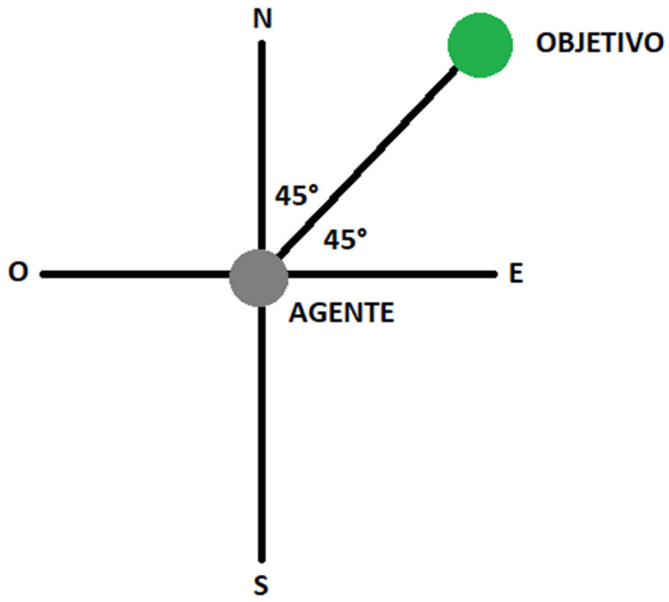

Figura 19. Orientación del agente con respecto al objetivo

Fuente: Autores

- **Calcular estado:**

Teniendo en cuenta los factores mencionados anteriormente, el cálculo del estado se realizó de la siguiente manera:

1. **Sensores de proximidad:** Al tener en cuenta que los sensores tienen 3 niveles de detección {0, 1, 2}, se debe realizar el cálculo en base 3 y sumar $(+1)$ ya que el valor de 0 no permite calcular el estado correctamente y también para conocer cuántas combinaciones de sensado se tienen para los 4 sensores y sus niveles de detección, teniendo en cuenta lo anterior, el cálculo se hace teniendo en cuenta la ecuación (12).

$$Sensores = ((FS * 1 + BS * 3 + LS * 9 + RS * 27)) \quad (12)$$

2. **Distancia:** La distancia está discretizada en 3 niveles {0, 1, 2}, para calcular el estado se debe sumar $(+1)$ al valor discreto ya que el valor de 0 no permite calcular el estado de forma correcta como se muestra a continuación:

distancia_max = distancia inicial del agente con respecto al objetivo.

Donde,

Distancia = distancia del agente con respecto al objetivo, este valor se toma luego de realizar una acción.

Valor de distancia discretizado = $(((distancia)/distancia\_max) * 2)$, este valor es aproximado al valor entero mínimo más cercano, el comando "*floor()"* de Matlab permite realizar dicha aproximación.

$$Distancia = (valor\ de\ distancia\ discretizado)$$

3. **Orientación:** La orientación, está definida por 4 direcciones $(num\_orientaciones)$ como los puntos cardinales siendo [Norte = 1, Sur = 2, Oeste = 3, Este = 4], que permiten saber la posición del agente con respecto al objetivo, para calcularla se deben tener en cuenta dos aspectos: El ángulo del UAV con respecto al objetivo y las coordenadas (X, Y) del agente y el objetivo, procediendo de la siguiente manera:

- Se toman las coordenadas (X, Y) del UAV y del objetivo, situando al agente como punto $(0,0)$, así:

$$Xobj = (dron\ (x) * (-1) + objetivo\ (x)), nueva\ coordenada\ X\ del\ objetivo$$

$$Yobj = (dron\ (y) * (-1) + objetivo(y)), nueva\ coordenada\ Y\ del\ objetivo$$

- Se obtiene el cuadrante en el que se encuentra el objetivo, teniendo en cuenta los valores mencionados anteriormente $(Xobj\ y\ Yobj\ )$, los cuales según su valor ubican al objetivo en un cuadrante del plano cartesiano, como se muestra en la figura 20.

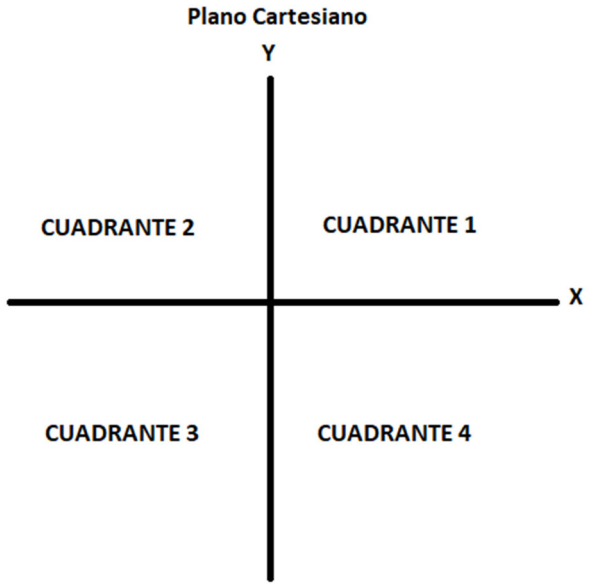

Figura 20. Orientación en cuadrantes para calcular el estado del agente

Fuente: Autores

- Se realiza el cálculo del ángulo que existe entre el agente con respecto al objetivo, así:

1. Inicialmente se debe asegurar que los valores $(Xobj\ y\ Yobj\ )$ sean positivos, es decir que se deben ubicar en el cuadrante 1.
2. Seguido a esto, se calcula la distancia o tangente con las coordenadas $(Xobj, Yobj\ )$, haciendo uso del teorema de Pitágoras, como se indica en la ecuación *(13).*

$$Distancia\ = \sqrt{Xobj^2 + Yobj^2} \tag{13}$$

3. El ángulo es obtenido usando la ley del seno con los datos obtenidos anteriormente, como se muestra en la expresión (14).

$$Angulo = \ sin^{-1}(Yobj/Distancia) \tag{14}$$

Se debe tener en cuenta que el resultado de "*Ángulo*" en la expresión (14) es expresado en grados.

Teniendo en cuenta el ángulo se verifica si el valor es mayor o menor a 45°, como se muestra en la figura 12, infiriendo la lógica que se muestra en la descripción 1.

Si,

$Valor\ del\ ángulo = [0°,\ 45°]$, entonces

$$ubicación = 1$$

Si,

$Valor\ del\ ángulo = [46°,\ 90°]$, entonces

$$ubicación = 2$$

Descripción 1. Ubicación del agente a través del valor del ángulo.

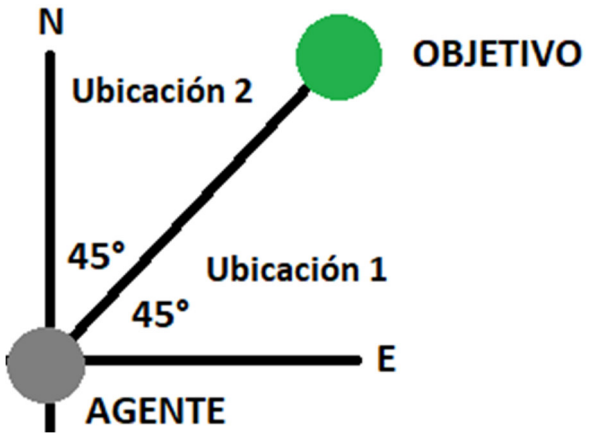

Figura 21. Cálculo del ángulo para obtener el estado del agente

Fuente: Autores

Para obtener la orientación se toman los valores del cuadrante y la ubicación del agente, teniendo en cuenta la siguiente lógica:

- Si el cuadrante es igual a 1 o 4 y el valor de ubicación es 1, la orientación es igual 4, esto significa que el objetivo se encuentra en dirección "ESTE"

$$Sí, cuadrante = 1 \text{ ó } 4 \; y \; ubicación = 1, entonces$$

$$orientación = 4$$

$$Dirección \; del \; objetivo = Este$$

- Si el cuadrante es igual a 1 o 2 y el valor de ubicación es 2, la orientación es igual 1, esto significa que el objetivo se encuentra en dirección NORTE.

$$Sí, cuadrante = 1 \text{ ó } 2 \; y \; ubicación = 2, entonces$$

$$orientación = 2$$

$$Dirección \; del \; objetivo = Norte$$

- Si el cuadrante es igual a 2 o 3 y el valor de ubicación es 1, la orientación es igual 3, esto significa que el objetivo se encuentra en dirección OESTE

$$Sí, cuadrante = 2 \text{ ó } 3 \; y \; ubicación = 1, entonces$$

$$orientación = 3$$

$$Dirección \; del \; objetivo = Oeste$$

- Si el cuadrante es igual a 3 o 4 y el valor de ubicación es 2, la orientación es igual 2, esto significa que el objetivo se encuentra en dirección SUR.

$$Sí, cuadrante = 3 \text{ ó } 4 \; y \; ubicación = 2, entonces$$

$$orientación = 2$$

$$Dirección \; del \; objetivo = Sur$$

Teniendo en cuenta los factores descritos anteriormente, el estado del agente se calcula usando la ecuación (15).

$$Estado = Sensores + (RS^S * Distancia) + ((num\_e/num\_orientaciones) * (Orientacion - 1)) \quad (15)$$

donde,

$RS = 3$ (Número de umbrales por sensor)

$S = 4$ (Número de sensores)

$num\_e = (Número \; de \; estados)$, en este caso 972.

$num\_orientaciones = (Número \; de \; orientaciones)$, en este caso 4.

Reemplazando en la ecuación (15), se tiene que:

$$Estado = Sensores + (3^4 * Distancia) + ((972/4) * (Orientacion - 1))$$

8.3.5. POLÍTICA DE EXPLORACIÓN

Para el desarrollo del algoritmo de *Q-Learning* el método de exploración más usado es el método *Simple Greedy* (ε), donde se define un valor fijo para "ε" entre 0 y 1 para determinar las condiciones que le permitan al agente tener en cuenta o no la experiencia adquirida para elegir la acción que éste realizará.

Para la aplicación del método en el algoritmo de aprendizaje por refuerzo de este proyecto, se estableció un valor inicial para la toma de decisiones $\varepsilon = 0.25$. Para tomar en cuenta o no la experiencia adquirida en la matriz *Q* se generó un valor aleatorio *(rand)* entre 0 y 1, donde se cumplen las siguientes condiciones:

- Sí $rand < \varepsilon$, la acción que realizará el agente será definido de forma aleatoria sin tomar en cuenta lo adquirido a través de la experiencia.
- Sí $rand > \varepsilon$, la acción que realizará el agente solo toma la información más reciente ignorando el conocimiento previo para explorar más opciones.

El valor de "ε" disminuye teniendo en cuenta el número de veces que ha llegado al estado evaluado, la tasa de disminución es de 0.005, hasta llegar a un punto en el

que el valor de "ε" sea igual o menor a 0 $(\varepsilon \leq 0)$ y permitiendo que la toma de decisiones sea solo teniendo en cuenta la experiencia adquirida en la matriz *Q*. En la figura 22 se expresa gráficamente la política de exploración para el algoritmo de este proyecto.

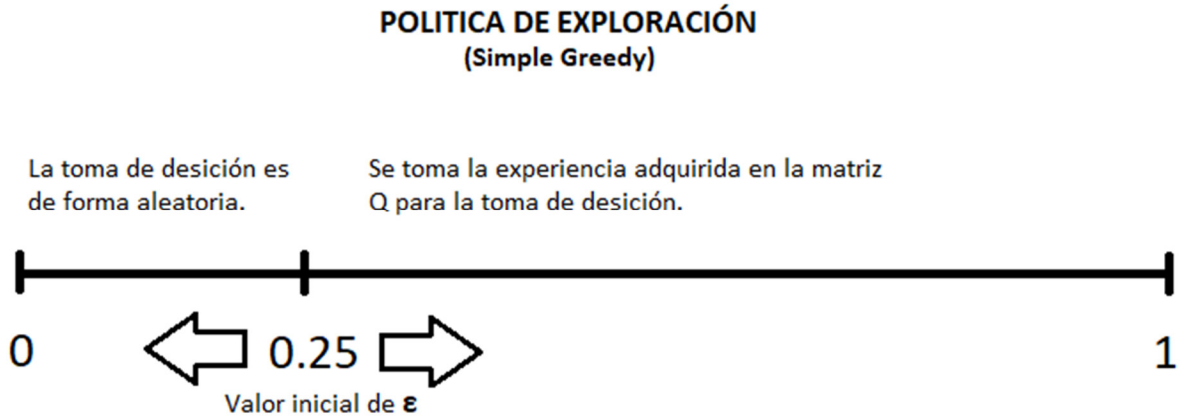

Figura 22. Política de exploración en el algoritmo

Fuente: Autores

8.3.6. FUNCIÓN DE RECOMPENSA

En el aprendizaje por refuerzo es importante el factor de recompensa porque permite castigar o premiar las acciones realizadas en un estado por parte del agente, verificando de esta manera su progreso para llegar al objetivo o "*Goal".*

Teniendo en cuenta los objetivos propuestos para el desarrollo del proyecto, es importante mencionar que el uso de la matriz *R* en el algoritmo no fue eficaz, debido a que solo es una guía con la que el agente será recompensado durante su entrenamiento, lo que implica que cada una de sus casillas deben ser completadas manualmente con los valores correspondientes según el par estado acción que se esté evaluando. Teniendo presente las consideraciones anteriores, se opta por el uso de una ecuación que tiene en cuenta el entorno del UAV para poder recompensarlo, los factores más relevantes para determinar la recompensa según el estado del agente luego de haber realizado una acción se muestran a continuación:

1. Sensores de proximidad:

Para dar recompensa a la lectura de los sensores se tienen en cuenta los valores obtenidos en el estado anterior y el estado actual, verificando su progreso. Para ello se tiene en cuenta una parte estática y otra dinámica, estas están definidas como se muestra en la descripción 2.

Parte estática (Est)

Definida como,

$Est = \sum_{i=1}^{N} \quad (\mathrm{Sensor(i)_t} - 2)$

Nota:

$Est \ = \ Parte\ estática$

$N \ = \ Número\ de\ sensores$

$t \ = \ Estado\ actual$

Descripción 2. Definición de la parte estática en la ecuación de la lectura de los sensores.

Como se observa en la descripción 2, el valor de (Est) viene dado por la sumatoria del progreso de cada sensor con respecto al valor ideal en este caso 2, ya que este valor indica que no tiene obstáculo cerca el agente.

Para la parte dinámica $(Dina)$ el valor viene dado por la sumatoria del progreso de cada sensor con respecto a los valores sensados del estado anterior $(t-1)$, como se muestra en la descripción 3.

Parte dinámica $(Dina)$

Definida como,

Dina= $\sum_{i=1}^{N} \quad (Sensor(i)_t - Sensor(i)_{t-1})$,

Nota:

$Dina \ = \ Parte\ dinámica$

$N \ = \ Número\ de\ sensores$

$t \ = \ Estado\ actual$

$t-1 \ = \ Estado\ anterior$

Descripción 3. Definición de la parte dinámica en la ecuación de la lectura de los sensores.

Teniendo en cuenta estos dos factores, se puede obtener la recompensa respectiva a la lectura de los sensores, como se muestra en la ecuación (16).

$$Rsensores = Est + Dina \qquad (16)$$

Donde,

$$Rsensores = Recompensa\ de\ lectura\ de\ sensores$$

$$Est = Parte\ estática$$

$$Dina = Parte\ dinámica$$

2. Aproximación al objetivo:

El avance del agente hacia el objetivo es recompensado teniendo en cuenta la distancia no discretizada del estado actual (*t*) y el estado anterior (*t-1*), tomando la diferencia para verificar su progreso, como se muestra en la ecuación (17).

$$diferencia\_distancia = distancia_{t-1} - distancia_t \quad (17)$$

Donde,

$$distancia = distancia\ entre\ el\ agente\ y\ el\ objetivo\ no\ discretizado.$$

$$t = Estado\ actual$$

$$t - 1 = Estado\ anterior$$

3. Avance en umbrales de distancia:

Los umbrales de distancia son importantes porque permiten verificar la forma en la que avanza el agente hacia el objetivo, en este caso el objetivo es la navegación, por tanto es importante saber el progreso en la distancia del UAV, para ello se toman los valores de distancia ya discretizados para premiar su progreso, de esta manera, el valor discretizado de la distancia que está entre el agente y el "*Goal"* es el valor del umbral con el rango de [0, 1, 2], donde:

$$valor\_umbral = 0, \quad muy\ cerca\ al\ GOAL$$

$$valor\_umbral = 1, \quad cerca\ al\ GOAL$$

$$valor\_umbral = 2, \quad lejos\ del\ GOAL$$

Luego de tener claros los factores para generar la recompensa del agente, se evalúan los valores obtenidos para darle una recompensa, esto se realiza de la siguiente manera:

1. Sensores de proximidad:

La evaluación de la recompensa de los sensores se realiza teniendo en cuenta la lógica que se muestra en la descripción 4.

Sí $Rsensores > 0$, entonces

La lectura de los sensores tuvo progreso, así

$$Valor\ de\ recompensa = 100$$

Sí $Rsensores = 0$, entonces

La lectura de los sensores no tuvo progreso, fue estático, así

$$Valor\ de\ recompensa = -50$$

Si $Rsensores < 0$, entonces

La lectura de los sensores no tuvo progreso, así

$$Valor\ de\ recompensa = -200$$

Descripción 4. Definición de la parte dinámica en la ecuación de la lectura de los sensores.

2. Aproximación al objetivo:

La evaluación de la recompensa en la aproximación hacia el objetivo se realiza teniendo en cuenta la lógica que se muestra en la descripción 5.

Si,

$diferencia\_distancia > 0$, *entonces*

El agente se acercó al objetivo, así

$$valor\ de\ recompensa = 100$$

Si,

$diferencia\_distancia = 0$, *entonces*

El agente no tuvo avance hacia el objetivo, pero tampoco retrocedió, así

$$valor\ de\ recompensa = -50$$

Si,

$diferencia\_distancia < 0$ *entonces,*

El agente retrocedió con respecto al objetivo, así

$$valor\ de\ recompensa = -200$$

Descripción 5. Recompensa del agente con respecto a la cercanía del objetivo.

3. Avance en umbrales de distancia:

La evaluación de la recompensa en el avance por los umbrales de distancia hacia el objetivo se realiza teniendo en cuenta la lógica que se muestra en la descripción 6.

Si,

$valor\_umbral > 0$,

El agente se encuentra muy cerca al objetivo, así

$$recompensa = 100$$

Si,

$diferencia\_distancia = 0$,

El agente se encuentra a mitad de camino al objetivo, así

$$recompensa = -50$$

Si,

$diferencia\_distancia < 0$,

El agente se encuentra lejos al objetivo

$$recompensa = -200$$

Descripción 6. Recompensa del agente con respecto a los umbrales de distancia.

Teniendo en cuenta los valores asignados para recompensar cada factor, se calcula la recompensa que tiene el agente en base al estado que se llega por la acción tomada, esta recompensa se calcula como se muestra en la ecuación (18).

$$Recompensa = (Rsensores * 0.33 + diferencia\_distancia * 0.33 + valor\_umbral * 0.33) \quad (18)$$

Como se puede evidenciar en la ecuación (18), se asignó un peso del 33% de la recompensa total para cada factor evaluado, ya que si se coloca más peso hacia algún factor durante el entrenamiento el agente no tendrá en cuenta el factor que tenga menos peso en la ecuación. Un ejemplo claro es disminuir ese porcentaje al factor de lectura de sensores, el cual es muy importante para que el agente aprenda esquivar obstáculos mientras navega.

8.3.7. ENTRENAMIENTO DEL AGENTE

El entrenamiento es muy importante al implementar un algoritmo de aprendizaje por refuerzo, ya que es el modo que tiene el agente para aprender en base a su experiencia que se ve reflejada en la matriz Q.

- Entrenamiento del agente en V-REP

Para el entrenamiento del agente en el ambiente de V-REP, fueron identificadas algunas complicaciones, como la velocidad del ambiente de simulación, ya que entre más objetos tenga este, la conexión con el simulador se hará más lenta, lo que hace que los movimientos del UAV se vean en un tiempo posterior al que realmente es. Luego de haber identificado esta situación, se procedió a agregarle un retardo al movimiento del agente para que este pudiera ser completado oportunamente y se detectara de forma correcta el estado en el que se encontraba.

Para llevar a cabo el entrenamiento del agente fueron realizados ambientes de prueba en el software de simulación V-REP, la escena principal se muestra en la figura 23, donde se evidencia un ambiente de catástrofe natural, un incendio forestal, donde el agente es el encargado de monitorear el área para encontrar sobrevivientes y de esta manera permitir que los equipos de rescate lleguen hasta ellos, el ambiente tiene una dimensión de $(25m * 25m)$, tiene 9 personas atrapadas en la catástrofe y también tiene obstáculos que en este caso son árboles.

Figura 23. Escena en V-REP de catástrofe natural para el entrenamiento del agente

Fuente: Autores

El UAV que fue usado para las pruebas se muestra en la figura 24, siendo la única opción por el tipo de versión del software que se usó, la versión educativa de V-REP. En el ambiente de prueba, el agente puede interactuar de manera directa con los objetos que la conforman, como es el caso de los árboles, el fuego y las personas, permitiendo un acercamiento a situaciones reales.

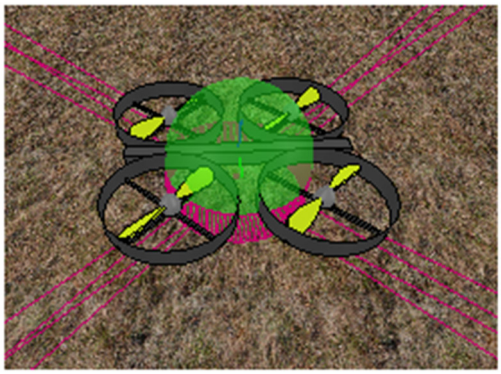

Figura 24. UAV de V-REP usado en la simulación de la escena.

Fuente: Autores

➢ *UAV*

El UAV utilizado es este proyecto, es un *Quadricopter*, el cual tiene 4 sensores, estos fueron ubicados en cada una de las orientaciones donde se puede situar el objetivo, es decir: Norte, Sur, Este y Oeste, esto se puede evidenciar en la figura 16. El *Quadricopter* se controla moviendo su *target*, como se muestra en la figura 24 (esfera de color verde). Al mover las coordenadas X, Y y Z del *target* permite al UAV moverse hacia las coordenadas que se le indican.

➢ Sensores

El tipo de sensores usados en el UAV son de tipo de proximidad, su rango de detección en ángulos es de 90°, mientras su rango en distancia es de 2 m, la forma del sensor es de disco ya que las acciones que el UAV puede realizar solo se dan en el plano X, Y, por lo tanto, no es necesario sensar el eje Z, como se muestra en la figura 14. Se usaron 4 sensores para tener un rango de visión de 360°.

➢ Objetivo o "*Goal"*

El objetivo usado en el ambiente de V-REP fue una persona, en el entorno se conoce con el nombre de "*Bill*", en la figura 23 se evidencia el uso de más de 7 personas para el entrenamiento del UAV, buscando que el ambiente fuera lo más real posible, simulando de esta manera una catástrofe natural o un ejercicio de entrega de mensajería.

- Entrenamiento en el software matemático

Para la implementación del algoritmo de aprendizaje por refuerzo propuesto, se debe tener en cuenta algunas variables las cuales durante el entrenamiento del agente cambian su valor, mientras otras se definen con un valor fijo para todo el entrenamiento, por lo tanto, se deben inicializar estas variables, las cuales son las siguientes:

1. Número total de estados = 972
2. Número total de acciones = 8
3. Número de iteraciones por episodio = 10000
4. Número de episodios = 300
5. Factor de descuento (γ) = 0.95
6. Tasa de aprendizaje (α) = 0.9
7. La Matriz Q se inicializa con valores de cero.
8. La Matriz N se inicializa con valores de cero (Matriz que guarda el número de veces que ha visitado cada par (estado, acción))
9. La Matriz C se inicializa con valores de cero (Matriz que guarda el número de veces que ha chocado durante el entrenamiento en cada par (estado, acción))

Teniendo las variables inicializadas se puede realizar la implementación del algoritmo de aprendizaje por refuerzo propuesto en este proyecto, como se muestra en el algoritmo 2.

Teniendo en cuenta las complicaciones obtenidas por tiempo y variables externas como la inercia en el movimiento del UAV en el ambiente en V-REP, fue adicionado al código implementado el manejo de estos, de la siguiente manera:

- **Inercia del UAV:** Durante el entrenamiento el agente necesita haber completado el movimiento para poder detectar el nuevo estado, por lo tanto, se colocó un retardo para asegurar que el agente completó la acción. Es importante destacar, que este retardo varía según el tipo de ambiente y la máquina que se usa para realizar la simulación.
- **Choques:** Al usar un simulador para el entrenamiento, se espera que el agente durante su aprendizaje choque con los obstáculos que encuentra, pero esto en el simulador hace que el UAV se desestabilice, obligando de esta manera a detener la simulación. Para evitar esto, se verifica la distancia que tiene el agente con respecto al objetivo antes y después de realizar una acción, se determinó que por cada acción el agente se mueve 0.2 m en el simulador. Por tanto, se estableció que, si la diferencia de las dos distancias leídas es igual o mayor a 3 m, se detiene automáticamente la simulación en V-REP y en Matlab.
- **UAV caído:** Durante el entrenamiento se espera que el UAV se choque, pero algunas veces el UAV no se desestabiliza, sino, que queda en el suelo impidiendo seguir entrenando, por lo tanto se manejó este problema verificando el valor en el eje Z del agente, ya que este siempre es de 1 m, pero si el valor

leído es menor a 0.3 m en el programa, el UAV está en el suelo, por lo tanto se detiene automáticamente la simulación en V-REP y en Matlab.

- **Salida del ambiente**: Durante el entrenamiento es posible que el dron tome decisiones erróneas las cuales lo lleven a salirse del ambiente, pero esto no es favorable, ya que lo que se busca es que durante el entrenamiento se explore el mayor número de estados posibles y al no estar dentro del ambiente propuesto no encontrará obstáculos. Para manejar este problema se verifica por cada acción tomada, el valor de las coordenadas X y Y, verificando que no supere los valores de las dimensiones del ambiente las cuales son $25mx25m$.

Para que el agente pudiera lograr visitar más estados se determinó que cuando llega el agente al *Goal*, automáticamente se establece un nuevo objetivo en el mapa permitiendo que interactúe con los obstáculos para que no llegue a chocar y pueda lograr un aprendizaje más rápido.

Un ejemplo del entrenamiento del agente con el algoritmo propuesto se muestra a continuación, en este, el agente en 5 intentos pretende llegar al mismo objetivo, donde se ve el progreso a medida que este avanza.

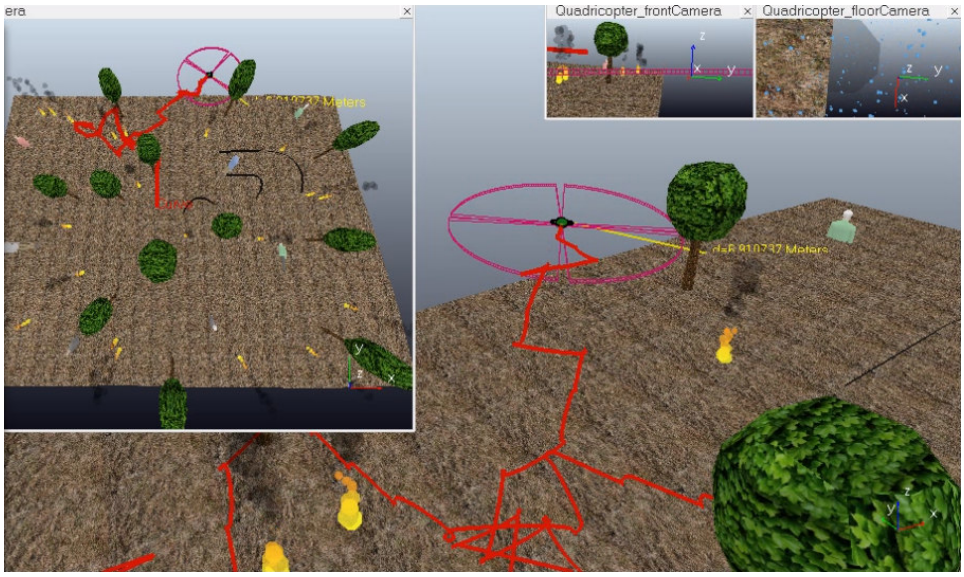

Figura 25. Intento 1 del agente para llegar al objetivo.

Fuente: Autores

En la figura 25, se muestra como el agente pretende llegar al objetivo, los movimientos del UAV están marcados por la línea de color rojo.

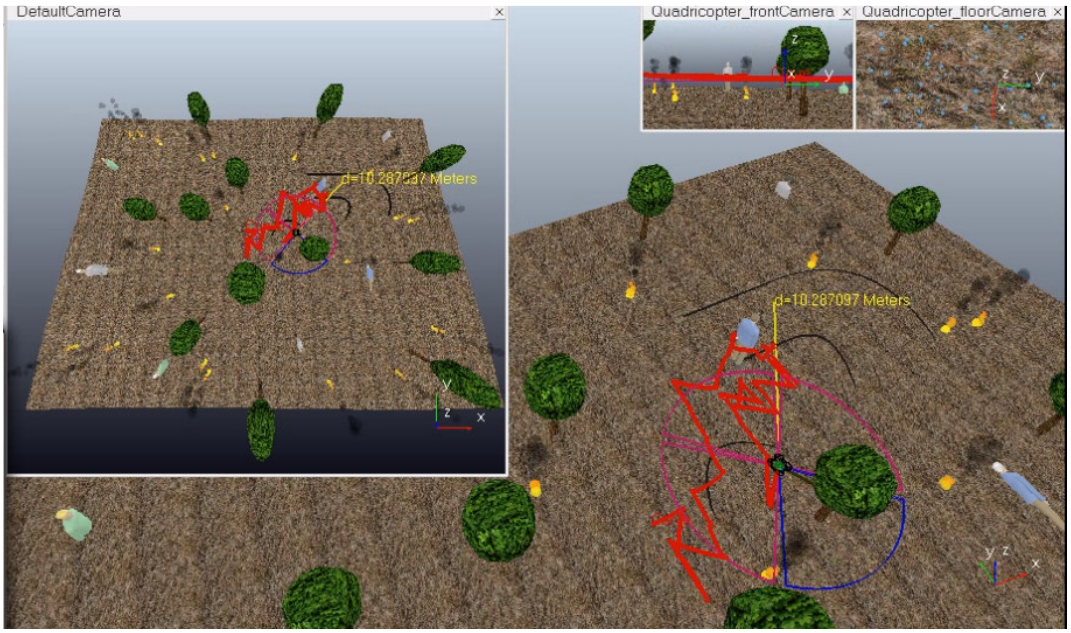

Figura 26. Intento 2 del agente para llegar al objetivo.

Fuente: Autores

El agente se acerca al objetivo, esquivando los obstáculos que se encuentra en el camino, como se muestra en la figura 26.

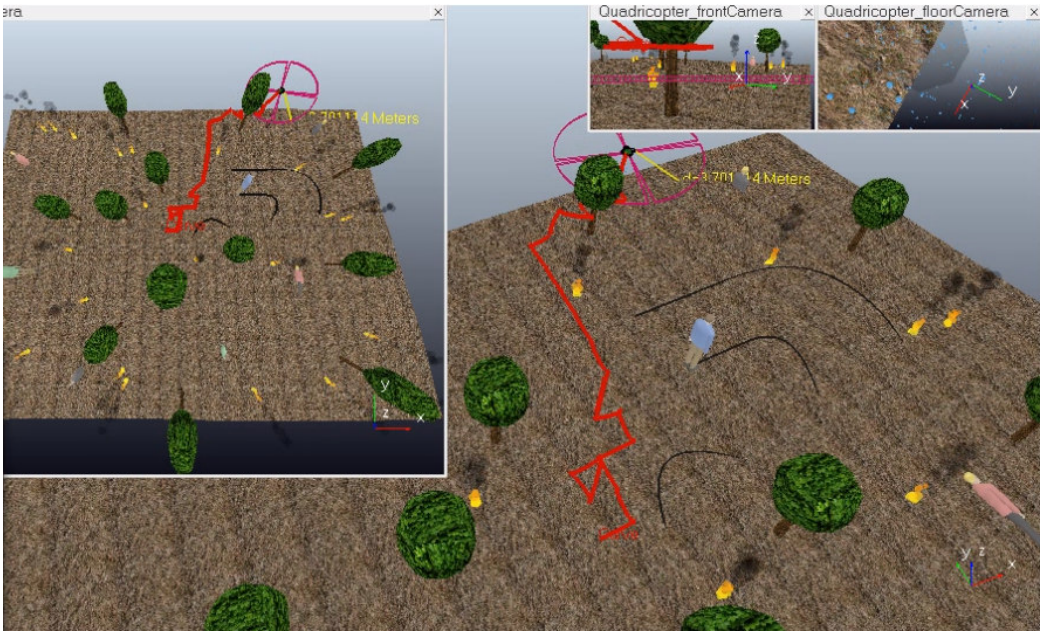

Figura 27. Intento 3 del agente para llegar al objetivo.

Fuente: Autores

El agente se acerca al objetivo, como se muestra en la figura 27, pero choca con un obstáculo (árbol), la ruta que ha tomado el UAV están marcados de color rojo.

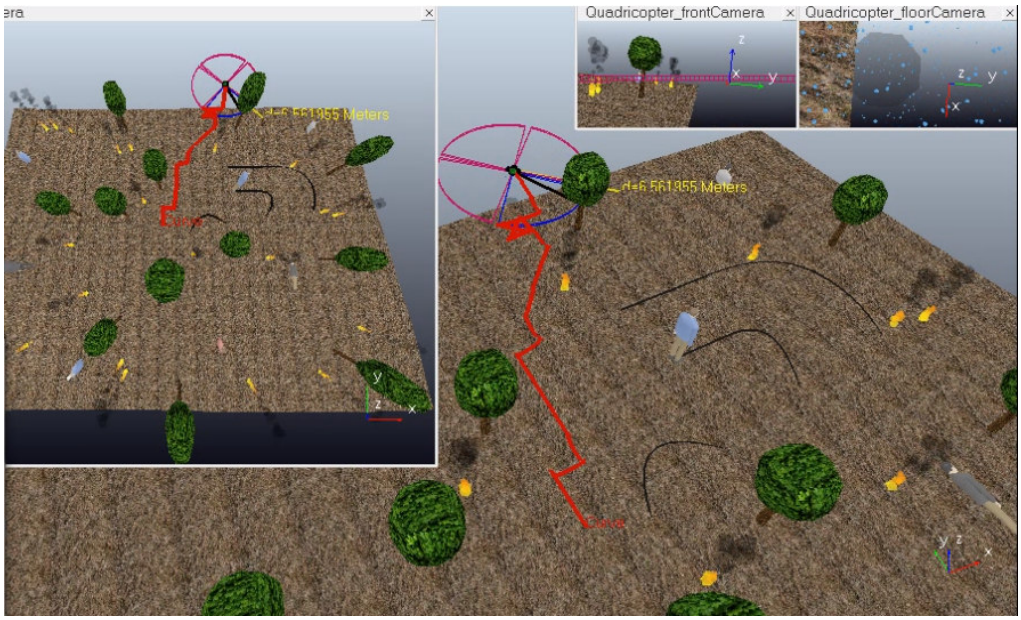

Figura 28. Intento 4 del agente para llegar al objetivo.

Fuente: Autores

El agente se encuentra a una distancia muy corta del objetivo, como se muestra en la figura 28, aprendiendo a esquivar el obstáculo que anteriormente hizo que chocara. La ruta que ha tomado el UAV está marcada de color rojo.

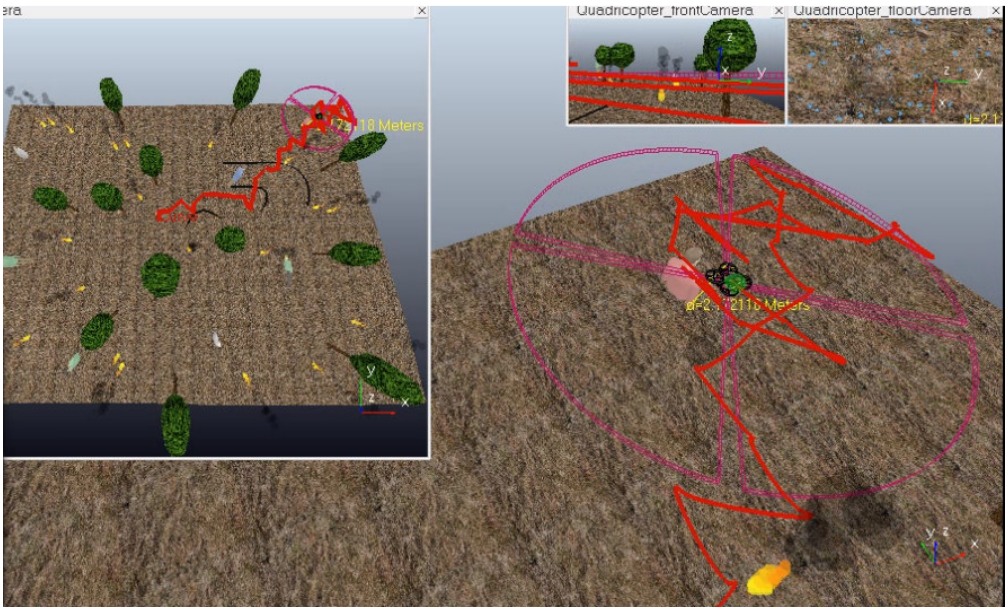

Figura 29. Intento 5 del agente para llegar al objetivo.

Fuente: Autores

El agente llega al objetivo, la ruta completa que tomó el UAV hasta este se muestra de color rojo en la figura 29.

Como se pudo evidenciar el algoritmo de aprendizaje por refuerzo implementado, permite al agente mejorar cada intento que tiene para llegar al objetivo y así lograr visitar los estados necesarios que lo lleven al objetivo o "*Goal".*

9. RESULTADOS DEL PROYECTO

En esta sección se muestran los resultados obtenidos con la implementación del proyecto teniendo en cuenta las especificaciones que se definieron en el desarrollo del algoritmo de *Q-Learning*, en la sección 8.3.

9.1. Pruebas experimentales

Para comprobar experimentalmente el funcionamiento del algoritmo de aprendizaje por refuerzo, fue necesario realizar el ejercicio de explotación, el cual consiste en tomar el valor máximo de las acciones que corresponda al estado actual del agente, obteniendo la decisión óptima a ejecutar. Para que este ejercicio sea lo más cercano a la realidad posible, se recreó un mapa virtual en el simulador V-REP, al cual el UAV es expuesto.
Teniendo en cuenta que una de las aplicaciones del proyecto es el rescate de personas, se desarrolló un ambiente de prueba que representa una catástrofe natural, más específicamente un incendio forestal, como se muestra en la figura 16, la descripción de este problema se desarrolla a continuación:

- Situación de prueba para el algoritmo

Se detecta una catástrofe natural, específicamente un incendio forestal, en el que hay 7 personas atrapadas. Los rescatistas no pueden acceder con facilidad debido a la gravedad del incendio. De esta manera, se opta por el uso de un UAV, que les permita a las víctimas guiarse por el camino menos peligroso hasta un lugar seguro. Para llevar a cabo el rescate, el UAV es llevado hasta un punto central del lugar, permitiendo que el rescate se lleve a cabo en el menor tiempo posible.

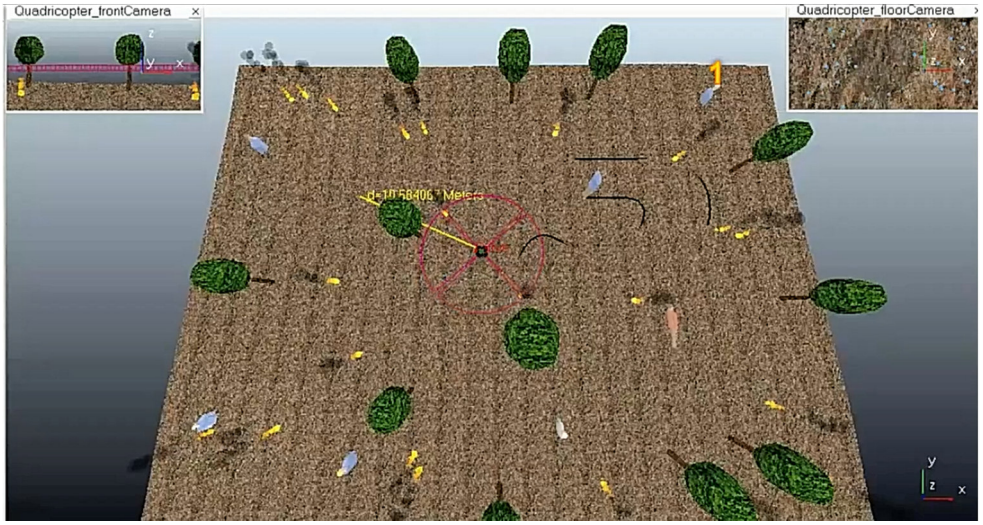

Figura 30. Ambiente de incendio forestal en V-REP con la ubicación del UAV y la víctima número 1.

Fuente: Autores

En la figura 30, se observa el UAV en un punto central del ambiente de V-REP, siendo un lugar adecuado para encontrar con mayor facilidad a las personas. La víctima número 1 es señalada en el ambiente de prueba.

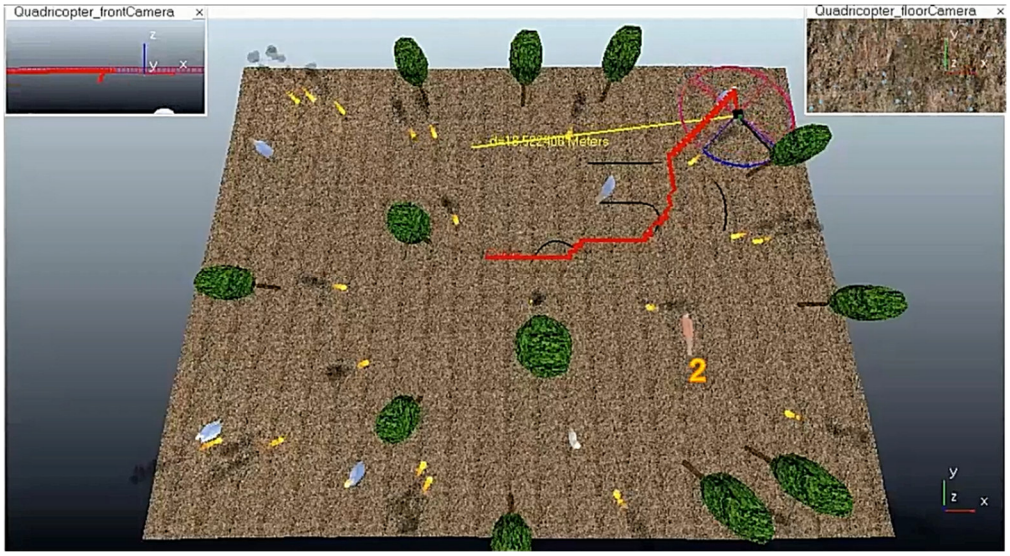

Figura 31. Ambiente de incendio forestal en V-REP con la ubicación del UAV y la víctima número 2.

Fuente: Autores

En la figura 31, es posible observar el camino que toma el agente para llegar hasta la víctima número 1, logrando guiarla hasta un lugar seguro. En la figura 31 es señalada la ubicación de la víctima número 2, es decir el siguiente objetivo del UAV.

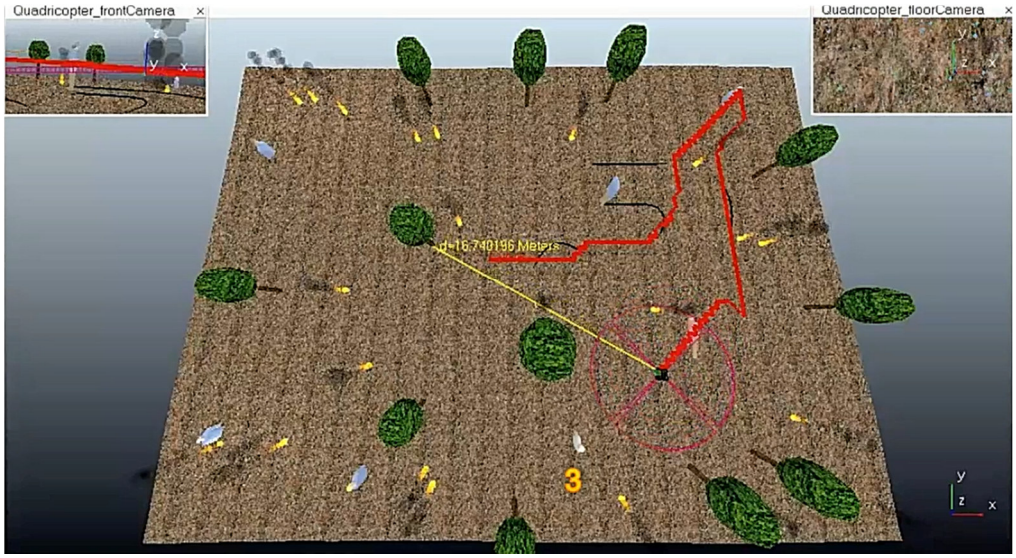

Figura 32. Ambiente de incendio forestal en V-REP con la ubicación del UAV y la víctima número 3.

Fuente: Autores

En la figura 32, se muestra la ubicación del agente, donde este logra hallar a la víctima número dos e inicia el camino para llegar hasta la tercera víctima del incendio forestal.

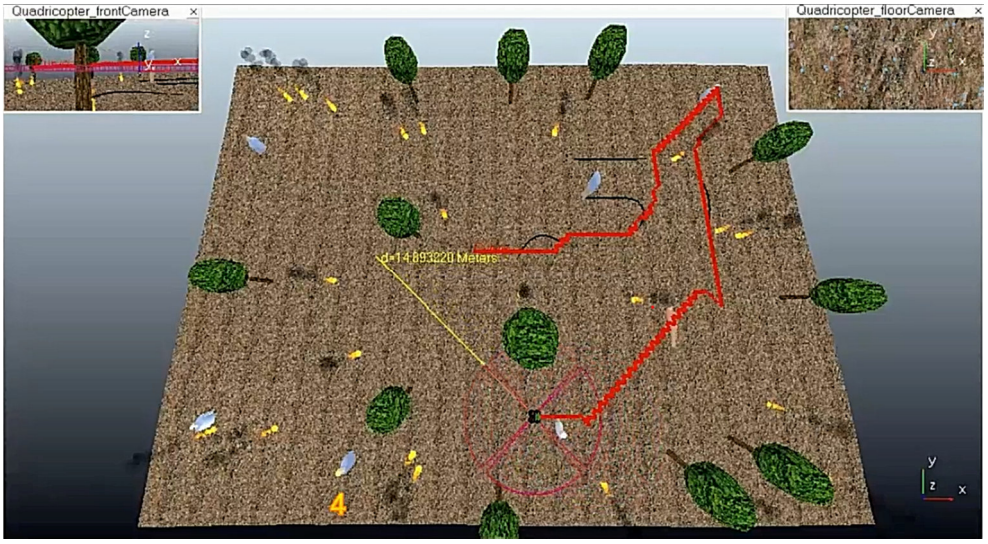

Figura 33. Ambiente de incendio forestal en V-REP con la ubicación del UAV y la víctima número 4.

Fuente: Autores

En la figura 33 se observa como el UAV realiza el hallazgo de la persona número 3 e identifica la ubicación de la cuarta víctima para poder rescatarla.

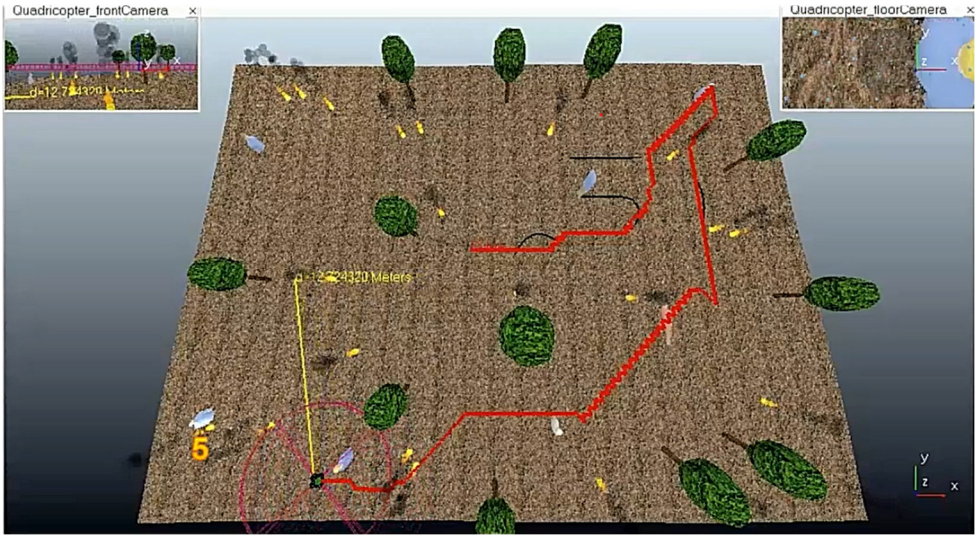

Figura 34. Ambiente de incendio forestal en V-REP con la ubicación del UAV y la víctima número 5.

Fuente: Autores

En la figura 34 se muestra la ruta que el UAV (línea de color rojo) ha seguido hasta el objetivo 4.

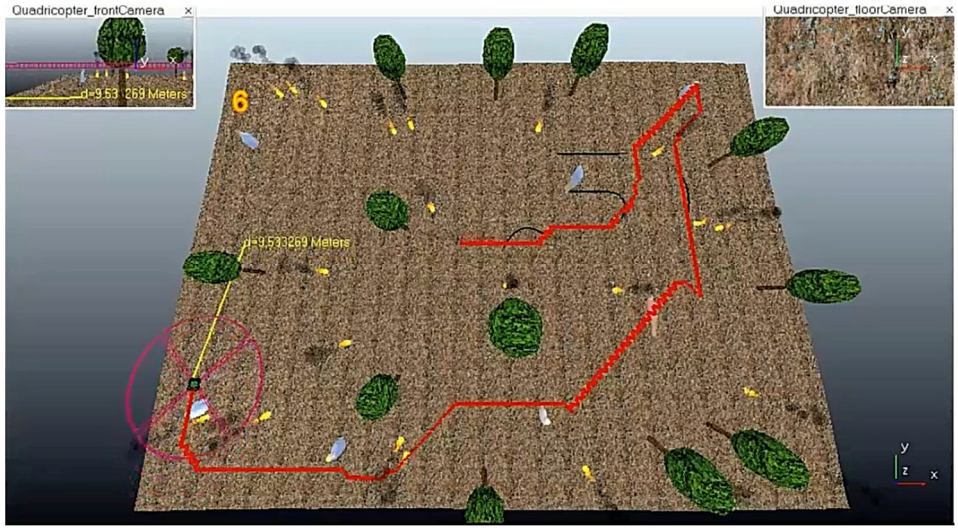

Figura 35. Ambiente de incendio forestal en V-REP con la ubicación del UAV y la víctima número 6.

Fuente: Autores

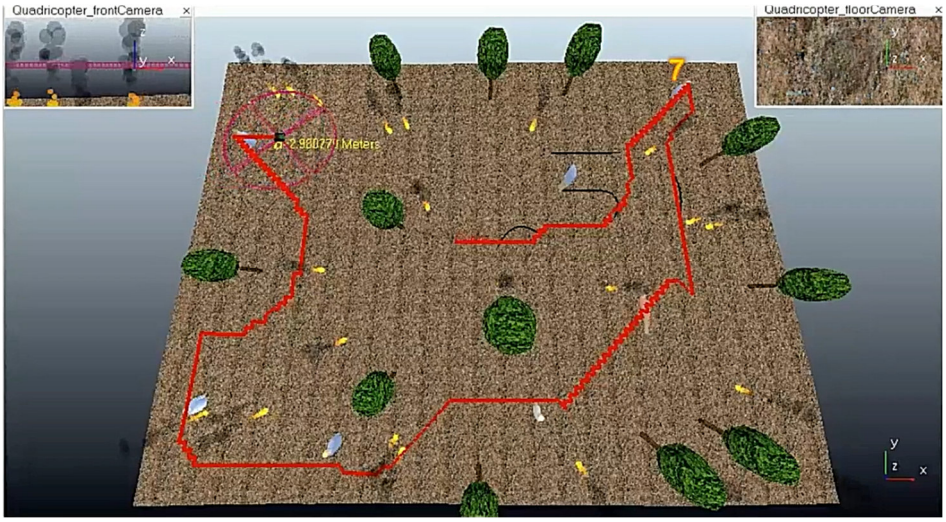

Figura 36. Ambiente de incendio forestal en V-REP con la ubicación del UAV y la víctima número 7.

Fuente: Autores

En las figuras 35, 36 y 37 se observa la ubicación del agente en puntos clave para el rescate de las víctimas 5, 6 y 7, exponiendo con la línea de color rojo el camino que ha recorrido para finalizar la misión de rescate de manera exitosa.

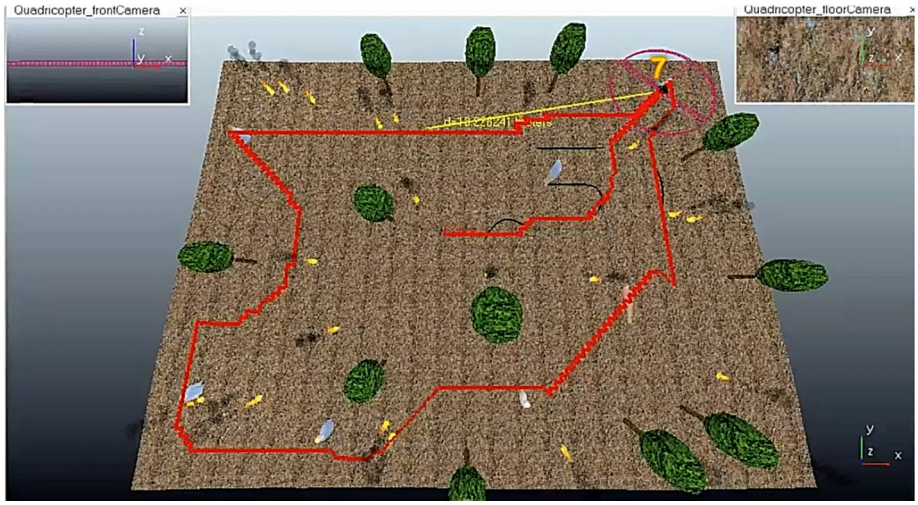

Figura 37. Ambiente de incendio forestal en V-REP con la ubicación del UAV y la finalización del rescate.

Fuente: Autores

En la Figura 37 se ve reflejado el recorrido completo (línea roja) que realizó el agente para lograr finalizar la misión de rescate de las 7 personas en el incendio forestal, observándose que los resultados obtenidos son satisfactorios. Es importante destacar que cada una de las víctimas se encontraban ubicadas en puntos diferentes, representando para el agente un reto distinto debido a que los obstáculos

que debía esquivar presentaban dificultad mayor o menor para llegar hasta cada una de las víctimas.

Los resultados presentados en este proyecto permiten ampliar las aplicaciones que pueden ser mejoradas a través del uso del algoritmo propuesto, como es el caso del servicio de entrega de paquetes con drones, evitando retardos en la entrega de estos porque se estaría esquivando el tráfico de la ciudad donde se lleve a cabo la implementación del proyecto. En la agricultura de precisión se identifica una aplicación valiosa para el algoritmo propuesto en este proyecto, debido a que el uso de un UAV en la fumigación de cultivos permitiría una menor exposición del agricultor a químicos que afecten negativamente su salud, logrando preservar el cultivo y mejorando la efectividad de los procesos en el campo.

9.2. Análisis de los datos obtenidos

- Choques

Un factor muy importante en la verificación del aprendizaje automático es el análisis de las decisiones mal tomadas por el agente durante el entrenamiento y cómo este factor disminuye con el tiempo, demostrando que hubo aprendizaje por refuerzo y cómo a través de la experiencia redujo esa toma decisiones que al realizar una acción le impide llegar al objetivo o *Goal.*

Llevando esta idea al proyecto, el factor a ser analizado son los choques que presentó el UAV durante su entrenamiento con el algoritmo de aprendizaje por refuerzo propuesto, ya que nos permite mostrar cómo a través del tiempo el UAV aprendió a no chocarse.

Para realizar el respectivo análisis se usó la matriz C, la cual se obtuvo durante el entrenamiento donde se almacenan los choques de cada par $(estado, acción)$ a través del tiempo, teniendo esto en cuenta se generó la figurea 38 en donde se muestran los choques obtenidos durante el periodo de entrenamiento.

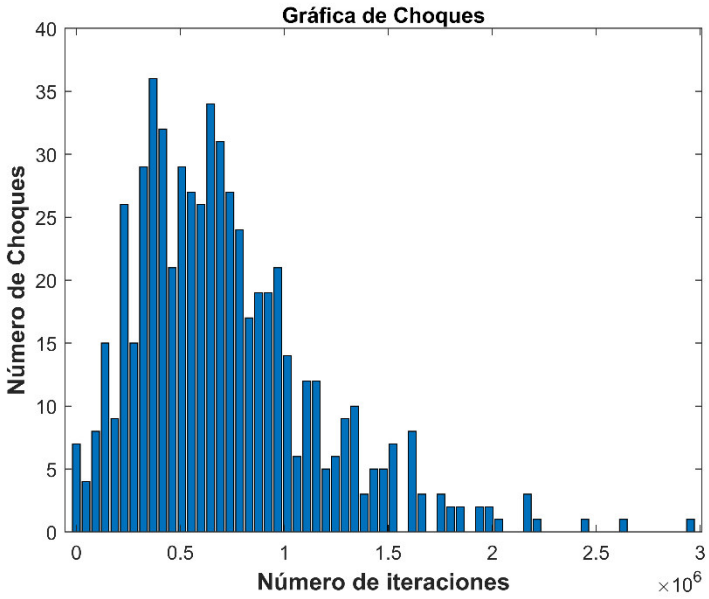

Figura 38. Representación de los choques del agente durante el entrenamiento del mismo Fuente: Autores

Teniendo en cuenta la figura 38, se observa que, al iniciar el entrenamiento, el agente presenta un aumento en el número de choques con los obstáculos, debido a que se encuentra explorando, sin embargo, a medida que aumenta el número de iteraciones, el número de choques disminuyen, facilitando de esta manera al agente, esquivar los obstáculos y demostrando que sí está aprendiendo por refuerzo en base a su experiencia.

- CONVERGENCIA DE LOS VALORES DE Q EN EL ENTRENAMIENTO DEL UAV

Una forma de verificar el aprendizaje del agente durante el entrenamiento es el análisis de los valores de la matriz Q, ya que estos datos representan la experiencia obtenida por el UAV y permiten conocer la manera como el agente actúa en los diferentes estados que visita. Teniendo en cuenta lo anterior, se llevó a cabo el análisis de tres estados, así:

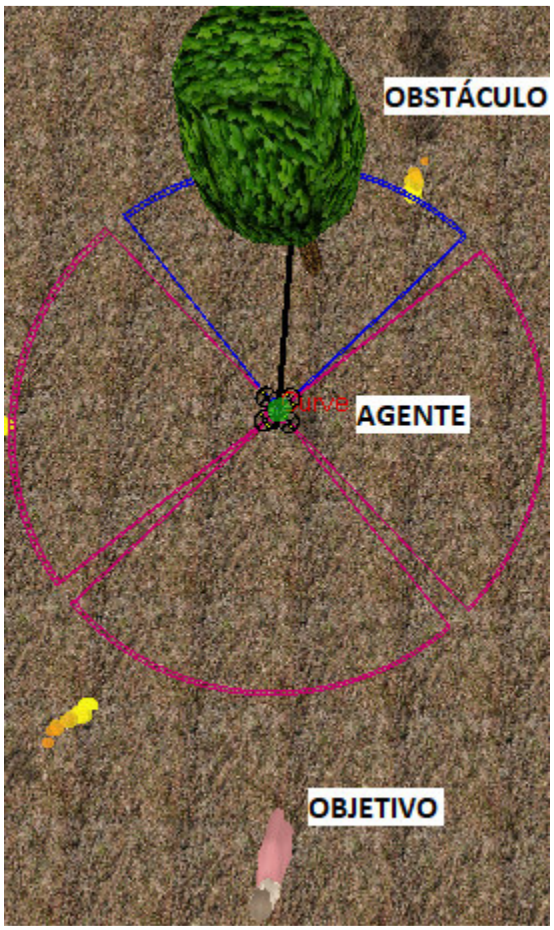

Figura 39. Situación en la que el objetivo se encuentra en la orientación 2. Fuente: Autores

Según la situación mostrada en la Figura 39, el objetivo se encuentra en orientación SUR y el UAV detecta un obstáculo con el sensor frontal (FS), el algoritmo interpreta esta situación con el estado 484, este valor se obtiene al seguir paso a paso la sección 8.3.5. En la figura 40 se muestra la representación de los valores Q de este estado para todas las acciones posibles, de la siguiente manera:

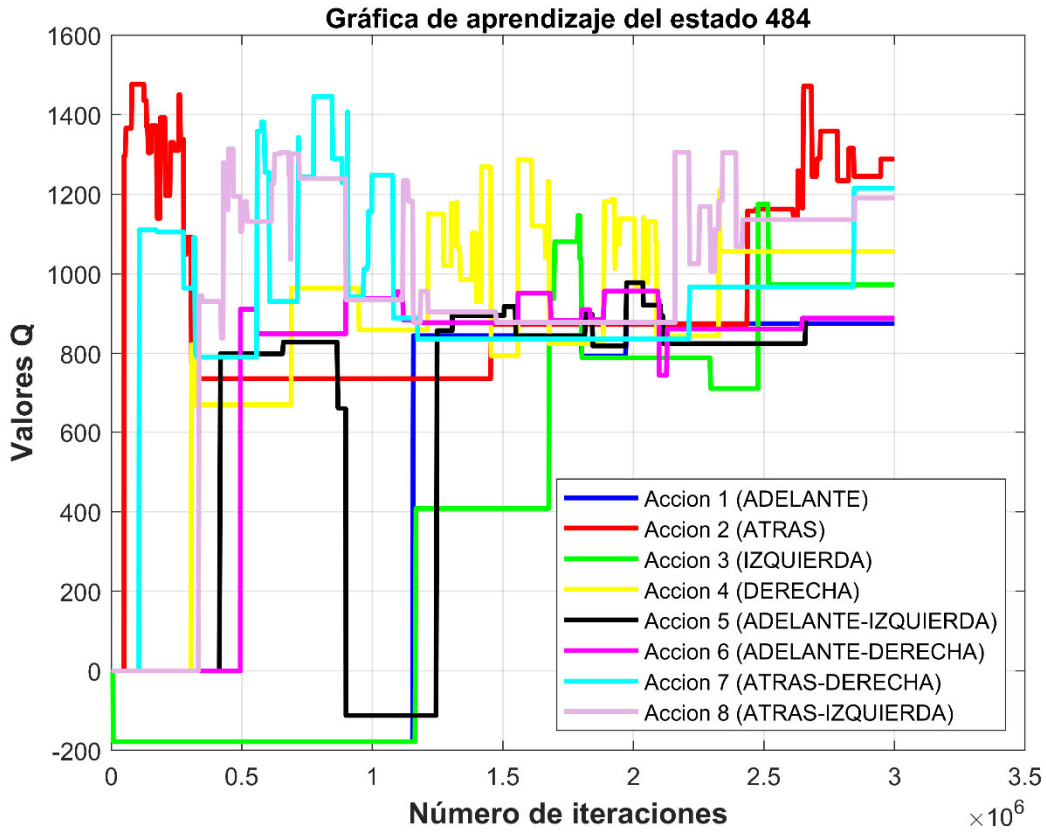

Figura 40. Valores Q de todas las posibles acciones en el estado 484. Fuente: Autores

En la figura 40 se muestra que las acciones que finalmente predominan en el estado 484 sobre el agente son:

- Acción 2 (ATRÁS), de color rojo.
- Acción 7 (ATRÁS-DERECHA), de color *cyan.*
- Acción 8 (ATRÁS-IZQUIERDA), de color malva.

Por otra parte, las acciones que tienen menor probabilidad en ser escogidas por el agente son:

- Acción 1 (ARRIBA), de color azul.
- Acción 5 (ARRIBA-IZQUIERDA), de color negro.
- Acción 6 (ARRIBA-DERECHA), de color púrpura.

Teniendo en cuenta el anterior análisis, es posible afirmar, que durante el entrenamiento el UAV aprendió a esquivar y llegar hasta su objetivo, debido que, para el estado analizado la opción óptima para el agente es irse en dirección hacia "atrás", sin descartar otras acciones, como el movimiento hacia "atrás en diagonal".

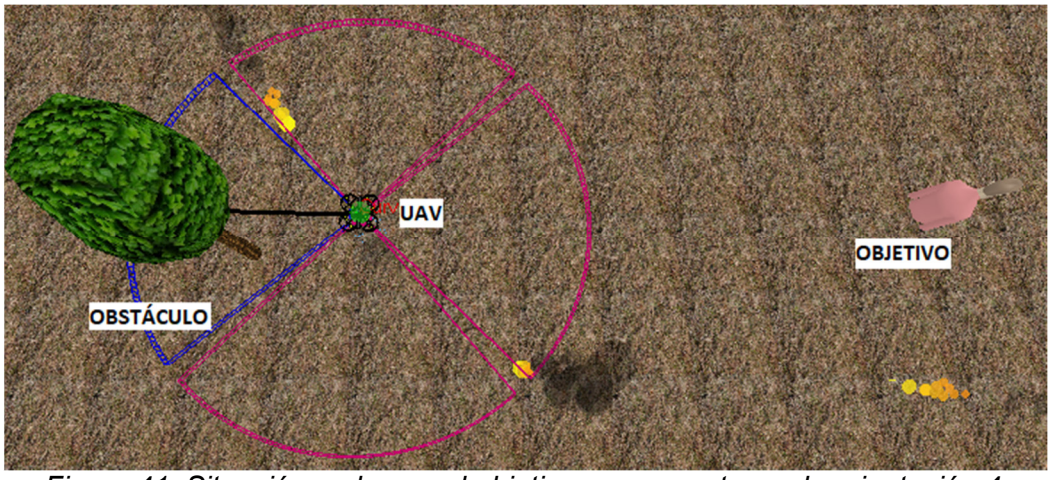

Figura 41. Situación en la que el objetivo se encuentra en la orientación 4. Fuente: Autores

El segundo estado a analizar es el número 954, representado en la figura 41. En este estado, el objetivo se encuentra ubicado hacia la derecha del agente y este a su vez, halla un obstáculo hacia la izquierda. Teniendo presente esta situación, los valores de la matriz Q para las posibles acciones se muestran en la figura 42.

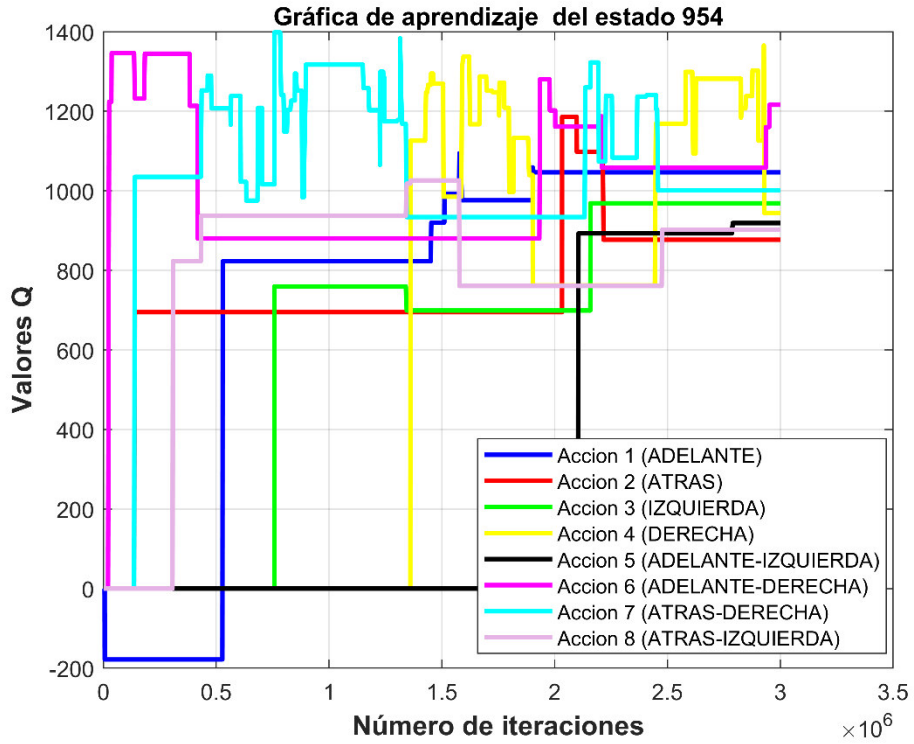

Figura 42. Valores Q de todas las posibles acciones en el estado 954. Fuente: Autores

En la figura 42, se muestra que las acciones que finalmente predominan en el estado 954 son:

- Acción 7 (ATRÁS-DERECHA), de color cyan.
- Acción 4 (DERECHA), de color amarillo.
- Acción 6 (ARRIBA-DERECHA), de color púrpura.

Por otra parte, las acciones que tienen menor probabilidad en ser escogidas por el agente son:

- Acción 2 (ATRÁS), de color rojo.
- Acción 5 (ADELANTE-IZQUIERDA), de color negro.
- Acción 8 (ATRÁS-IZQUIERDA), de color rosado.

Teniendo en cuenta el anterior análisis, es posible afirmar, que durante el entrenamiento el UAV aprendió a esquivar y llegar hasta su objetivo, debido que, para el estado analizado la opción óptima para el agente es irse en dirección hacia la "derecha" hacia donde está el objetivo, teniendo presente acciones similares que lo acercan hasta este y evitando aquellas que le permiten tener algún contacto con el obstáculo más cercano, como la acción "adelante-izquierda".

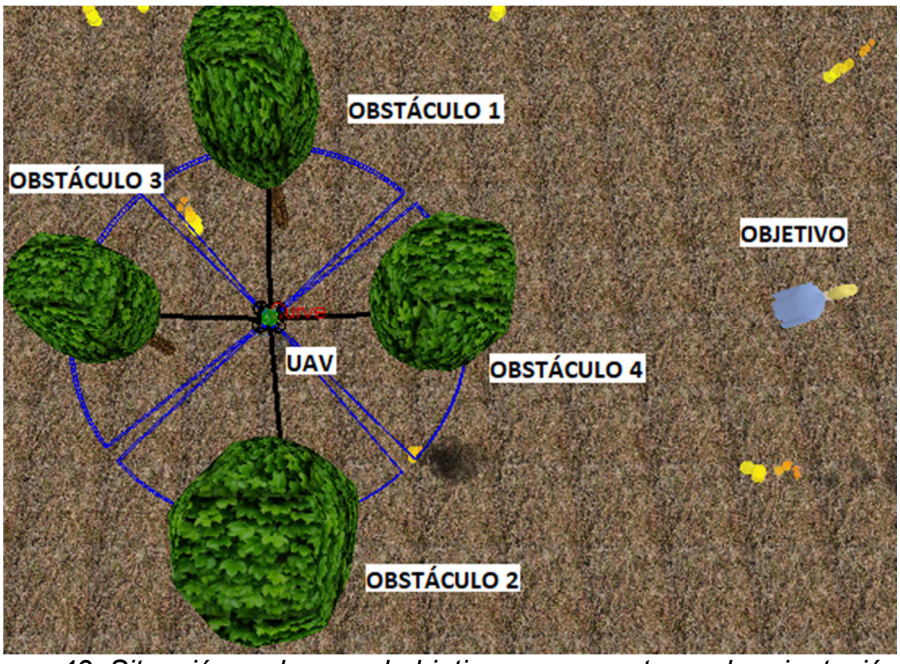

Figura 43. Situación en la que el objetivo se encuentra en la orientación 4. Fuente: Autores

Finalmente, se llevó a cabo el análisis de una última situación, el estado 810 como se muestra en la figura 43, donde el objetivo se encuentra ubicado hacia la derecha del agente y los sensores del UAV detectan obstáculos en cada una de las

direcciones.

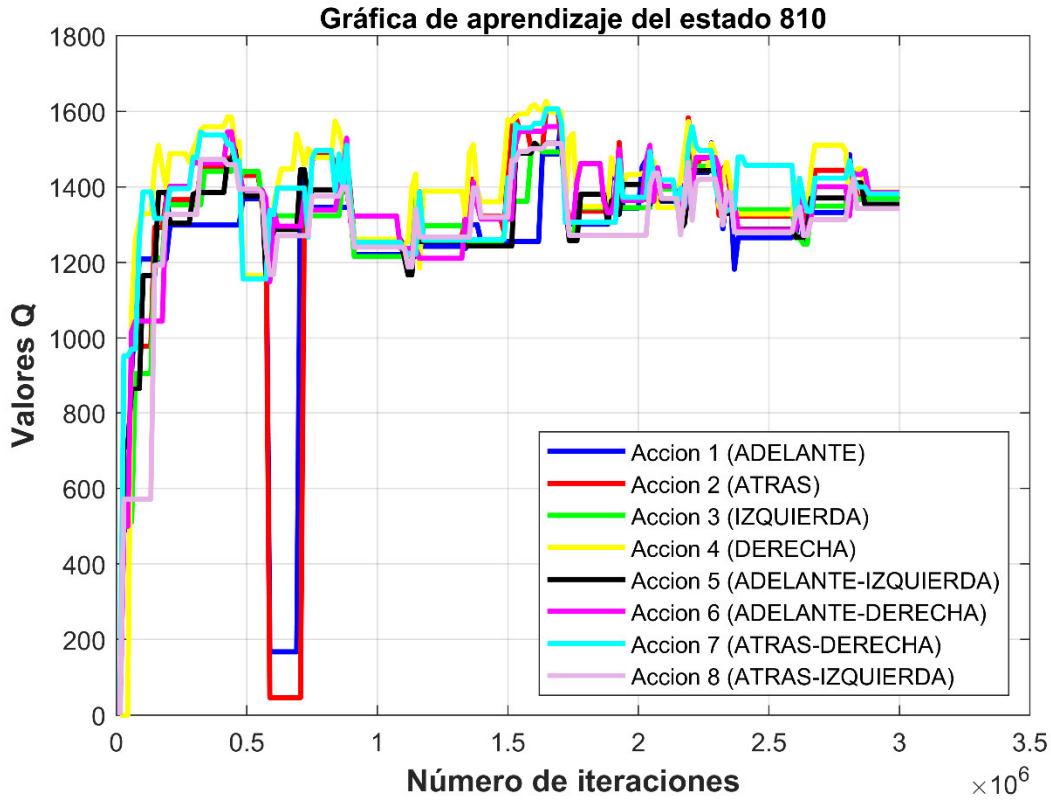

Figura 44. Valores Q de todas las posibles acciones en el estado 810. Fuente: Autores

Se observa en la figura 44, que cualquier acción tomada por el UAV tiene valores similares en Q, esto se debe a el estado donde se encuentra el agente, debido a que se halla en medio de obstáculos, siendo mucho más complejo tomar decisiones acertadas para llegar hasta el objetivo.

Para llevar a cabo el análisis del rendimiento del sistema de aprendizaje acorde con la variación de la cantidad de acciones y estados posibles se llevó a cabo la representación de los choques en 3 situaciones adicionales a la configuración del algoritmo propuesto, como se muestra en las figuras 45,46,47.

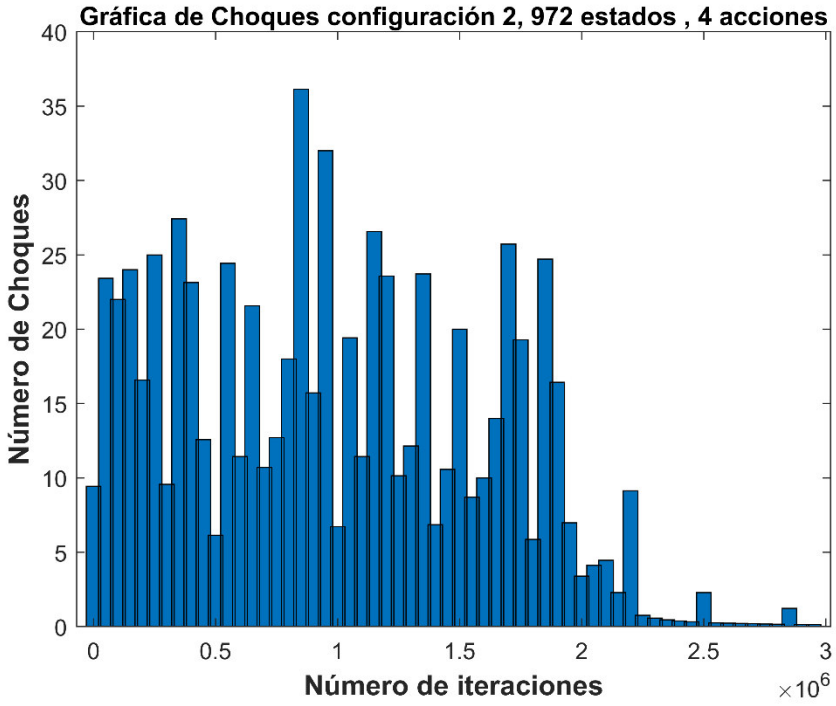

Figura 45. Representación de los choques del agente durante el entrenamiento de la configuración 2 del algoritmo.
Fuente: Autores

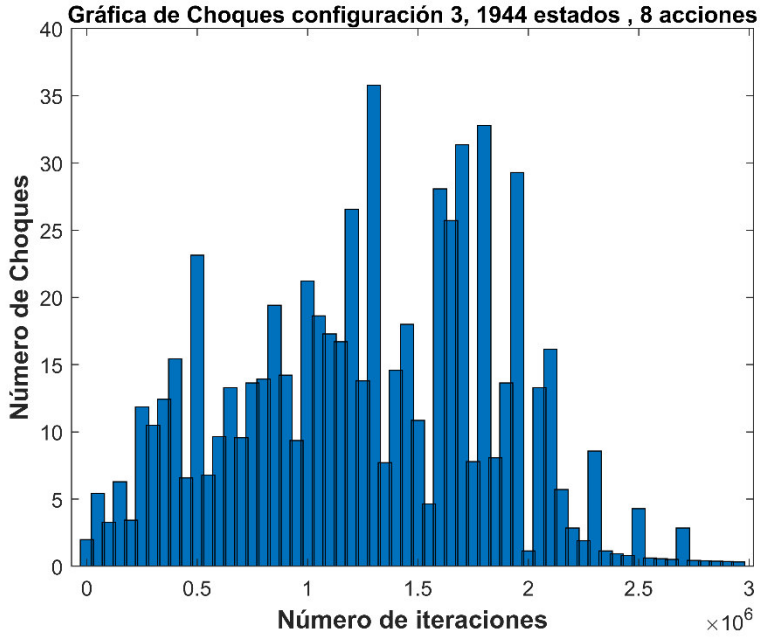

Figura 46. Representación de los choques del agente durante el entrenamiento de la configuración 3 del algoritmo.
Fuente: Autores

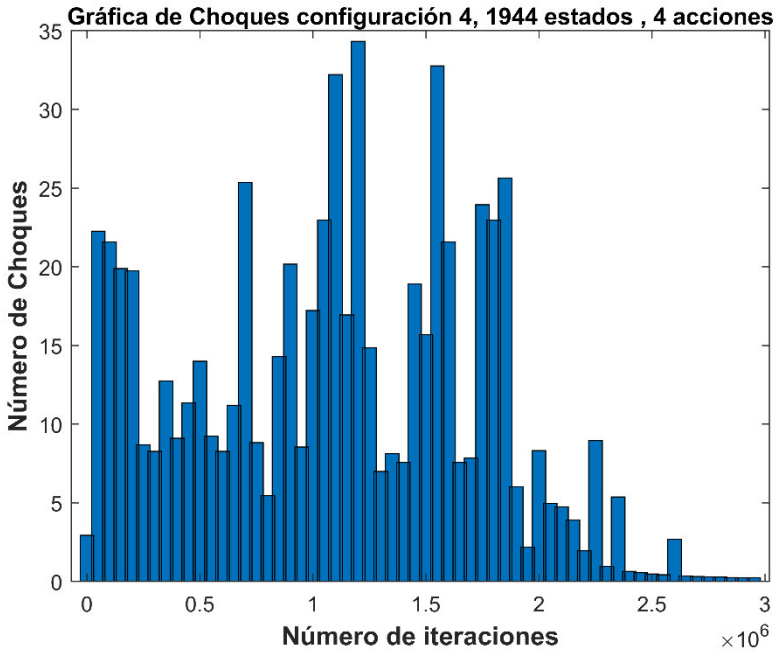

Figura 47. Representación de los choques del agente durante el entrenamiento de la configuración 4 del algoritmo.
Fuente: Autores

En la configuración 2, donde los estados son los mismos a la configuración propuesta y lo único que cambia es el número de acciones las cuales disminuyen a 4, se evidencia en la figura 45, cómo los valores de choques a través del tiempo demuestran el aprendizaje que tuvo el agente al esquivar un obstáculo en el mismo periodo de tiempo de entrenamiento que la configuración propuesta, ya que estos valores tienden a disminuir al final del entrenamiento.

En las configuraciones 3 y 4, donde se aumenta el número de estados a 1944, siendo el doble de estados de la configuración propuesta, ya que las direcciones a donde puede estar el *GOAL* aumenta de 4 a 8 direcciones, tiene 4 acciones en la configuración 3 y 8 acciones en la configuración 4.
En la figura 46 y en la figura 47, se puede evidenciar que el sistema aprende a esquivar obstáculos en el mismo periodo de tiempo de entrenamiento que la configuración propuesta, pero estos valores de choques tienden a disminuir más lento al final del entrenamiento, a comparación de la configuración 1 (propuesta) y la configuración 2.
Además de la comparación de las gráficas de los choques del agente en las cuatro configuraciones del entrenamiento, se obtuvo la representación en porcentaje de los estados no visitados en cada una de las configuraciones propuestas, como se muestra en la figura 48.

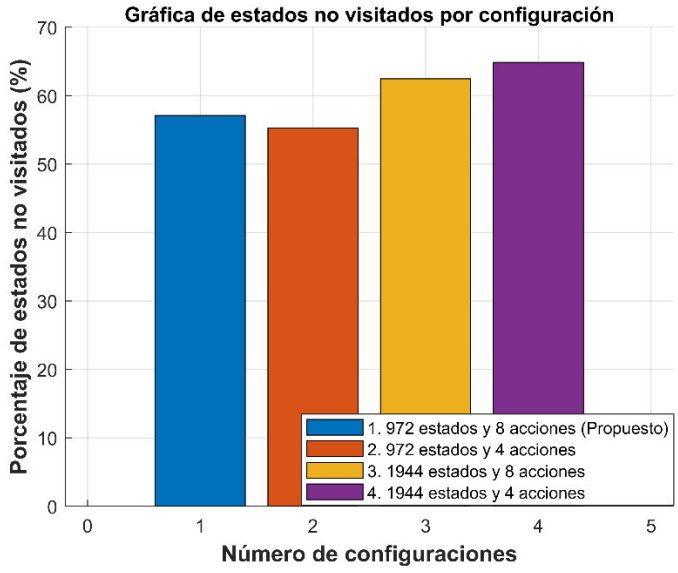

Figura 48. Representación en porcentaje del número de estados no visitados en las 4 configuraciones del algoritmo.
Fuente: Autores

Teniendo en cuenta la variación del número de estados se evidencia, que al cambiar este valor en la configuración del algoritmo, el porcentaje de estados no visitados en el mismo rango de tiempo supera el 50% para todas las configuraciones, lo que nos indica que la mayoría de estados no fueron visitados, esto ocurre por la "maldición de la dimensionalidad" la cual indica que el número de estados se incrementa exponencialmente a medida que se incrementa la complejidad del sistema, además no todos los estados serán visitados dado que el sistema aprende y deja muchos estados sin explorar, pero son necesarios tenerlos en cuenta para la implementación del algoritmo, sin importar la variación de estados y acciones, como se muestra en la figura 48.

10. IMPACTO SOCIAL

El desarrollo y el uso de vehículos aéreos no tripulados (UAV), ha sido posible gracias al sin número de proyecciones de las aplicaciones que se le puede dar a estos en diferentes áreas, como es el caso de la agricultura en Colombia, que con el crecimiento poblacional y el cambio climático, la producción de alimentos exige ser más eficiente y sostenible, surgiendo de esta manera el concepto de agricultura de precisión, en la que se combinan esfuerzos entre ciencias agrícolas, ingeniería, robótica y sistemas de información, por medio de las cuales se busca dar respuesta a este desafío. Los vehículos aéreos no tripulados, llamados más comúnmente como drones, facilitan cumplir con los objetivos expuestos anteriormente, siendo importante la navegación autónoma de los mismos, por medio de la cual se lograría inspeccionar la tierra de forma más eficiente, evaluando de esta manera la salud de los cultivos y analizar variables del suelo como la humedad y la distribución de la maleza, facilitando en un gran porcentaje la toma de decisiones para el agricultor. La combinación de la visión por computadora y la inteligencia artificial para discriminar cuales son cultivos y cuales maleza permite abrir la posibilidad de implementar sistemas de rociadores autónomos, golpeando a los cultivos con fertilizantes, resaltando la precisión y la eficiencia en porcentaje que este sistema puede tener.

El desarrollo de la navegación autónoma y evasión de obstáculos en un UAV impactaría de forma positiva a la comunidad tomasina, permitiendo ampliar los proyectos que podrían realizarse en el área de inteligencia artificial, debido a que se establecen las bases para la implementación del proyecto en un UAV real y traería consecuencias positivas, como un dron “lazarillo” que guíe a las personas dentro de las instalaciones de la universidad, facilitando su movilidad dentro de esta. En casos de desastres naturales como terremotos o incendios, el desarrollo de este algoritmo permitiría el rescate de personas, facilitando el trabajo de los órganos de rescate y minimizando el impacto negativo del desastre hacia la comunidad universitaria.

11. CONCLUSIONES

En el desarrollo de este trabajo, se llevó a cabo el diseño de un sistema de navegación autónomo para un UAV en los entornos de simulación V-REP y Matlab, haciendo uso de técnicas de aprendizaje por refuerzo como *Q-Learning*. Para lograr cumplir con los objetivos de este proyecto, fue importante un análisis de la situación a la cual se enfrentó el agente, permitiendo de esta manera definir los estados y acciones propios de la matriz Q, así como una política de premios y castigos, factores importantes para posibilitar el entrenamiento del UAV. Luego de haber hallado los valores adecuados para el aprendizaje del agente, se llevó a cabo la simulación del algoritmo, permitiendo analizar la evolución de este al tener en cuenta el ambiente de catástrofe natural diseñado en V-REP y la configuración establecida en todo el algoritmo. En las primeras simulaciones, se observa como el dron se encuentra desorientado y presenta mayor dificultad para esquivar los obstáculos y llegar hasta el objetivo, con el avance del entrenamiento se obtienen mejoras con la toma de decisiones gracias a la exploración que ha realizado el mismo y los valores guardados en la matriz Q.

La solución planteada en este proyecto se configuró para incluir varios objetivos en el ambiente de simulación, a los cuales se logró demostrar que el agente llegó con satisfacción, permitiendo de esta manera evidenciar que el UAV aprende a esquivar los obstáculos del medio donde se encuentra y a tomar decisiones inteligentes teniendo en cuenta el estado donde se halla y el modelamiento de la función de recompensa del algoritmo. Es importante destacar, que, en la solución propuesta, el robot no posee información sobre la ubicación espacial.

En los análisis de los estados, teniendo en cuenta las gráficas obtenidas de los valores de la matriz Q (Figura 40, 42 y 44), se observa cómo dependiendo de la situación en la que se encuentra el agente (ubicación de los obstáculos y ubicación del objetivo), este, prioriza ciertas acciones que le permitirá llegar hasta el objetivo que se le fue establecido, demostrando su aprendizaje por medio de las decisiones acertadas que tomó durante el entrenamiento. Otro punto importante, es el análisis realizado en la figura 38, de la representación de los choques del UAV con las iteraciones realizadas, donde se percibe una disminución de estos a medida que el agente avanza en su entrenamiento, pudiendo afirmar de esta manera, que el agente aprende a esquivar los obstáculos del ambiente donde se llevan a cabo las pruebas.

Teniendo en cuenta los experimentos anteriores y los resultados obtenidos a través del desarrollo de este proyecto, se demuestra que el agente aprende a navegar de forma autónoma en ambientes desconocidos con el uso de la información adquirida a través de los sensores y que por medio de la transición de la exploración y la explotación se logra generar habilidades de adaptación del robot, en general los resultados son satisfactorios debido a que el sistema converge.

12. RECOMENDACIONES

Para futuros proyectos que tengan como base el algoritmo de aprendizaje por refuerzo propuesto en este documento, se recomienda que al realizar el entrenamiento del agente si es también en una ambiente virtual que simula un escenario de la vida real, este sea en una máquina con características que permita hacer rápida la ejecución en la conexión entre el software matemático y el ambiente virtual, ya que según el tipo de escenario puede afectar el tiempo esperado para que el agente termine el entrenamiento.

El uso de sensores de proximidad realizado en este proyecto puede llegar a ser mejorado sumando el uso de visión artificial para asegurar una detección de obstáculos mucho mejor, logrando un entrenamiento mucho más rápido y además logrando ser aplicado a muchos problemas como lo es en la verificación del estado de cultivos para una cosecha óptima o la fumigación de cultivos ilícitos.

En la implementación del algoritmo a un UAV real se debe tener en cuenta el control que tiene el dron para moverse, ya que en la toma de decisiones para realizar una acción lo más posible es que se deba colocar un retardo para asegurar que el UAV terminó la acción por completo y lograr que al calcular el nuevo estado, este sea el correcto.

13. TRABAJOS RELACIONADOS

Este proyecto, se encuentra directamente relacionado con el artículo *"Autonomous Navigation for Exploration of Unknown Environments and Collision Avoidance in Mobile Robots Using Reinforcement Learning" [43]* el cual fue el trabajo preliminar para la finalización de este trabajo.

Autonomous Navigation for Exploration of Unknown Environments and Collision Avoidance in Mobile Robots Using Reinforcement Learning

G. A. Cardona*, C. Bravo†, W. Quesada†, D. Ruiz†, M. Obeng‡, X. Wu‡, and J. M. Calderon†‡
*Department of Electrical and Electronic Engineering Universidad Nacional de Colombia, Bogota, Colombia.
gacardonac@unal.edu.co
†Department of Electronic Engineering, Universidad Santo Tomás Bogota, Colombia
{mariabravon, wilsonquesada, diegoruizb, juancalderon} @usantotomas.edu.co
‡Computer Science & Engineering Department, Bethune-Cookman University Daytona Beach, FL, USA
{wux, obengm, calderonj} @cookman.edu

***Abstract*—This paper proposes the use of reinforcement learning to provide navigation and adaptation skills to a mobile robot in unknown environments. The use of sensor information is developed instead of the spatial location of the robot with the aim to perform the navigation and exploration process. The use of the Q-learning algorithm is proposed, however, the states are established based on the information coming from the sensors. A reward policy is applied to focus on guide the robot away from obstacles and allowing the exploration of unknown environments. Additionally, an exploration policy is generated which collaborates in the adaptation process of the robot. The complete system is evaluated using a robot simulation environment known as V-Rep. The results show that the robot learns to navigate avoiding to collide with obstacles, besides presenting skills to explore unknown environments.**

I. Introduction

In recent years, the collaboration of mobile robots in many fields as manufacturing process [1], home assistant [2], search and rescue operations of victims trapped in a disaster zone [3], the elderly care [4], among others have grown significantly thanks to the development of new technologies and theories applied to robotics. As a matter of fact, these facilitate that mobile robots have benefits and advantages performing tasks that are challenging for humans, such as repetitive actions which can be harmful, transport of heavy elements, performing exploration and identification victims in a disaster zone avoiding to put in risk additional lives in addition to a possible quicker execution and so on. Taking into account the foregoing mentioned about some tasks that mobile robots can perform, it is worth to mention, that for the success of this tasks some skills are necessary. Skills such as exploration and mapping of its environment including their own localization and the stuff around them, this task is usually called as SLAM; The exploration consists of the algorithm used to offer the ability to navigate in a scenario to the robot. Some works as [5] performed a fuzzy logic system in which a robot swarm navigates through a place based on the information from leader, usually known as leader formation control. On the other hand, artificial potential functions is another approach to navigate in a non-convex space, working as magnetic fields with an attraction to the target place and repulsion effect to avoid collisions between robots and the robots with obstacles as is depicted in [6], [7].

In the same way, in [8], [9] is shown how to use the concept of Voronoi Tessellation and artificial potential functions finding how to explore and navigate in an unknown scenarios, and performing the path planning, in contrast to the aim of this paper which use the Reinforcement learning approach in order to teach a robot to navigate autonomously in an totally unknown environment. The development of this kind of skills will help the robot to adapt and to fulfill tasks more complex than a simple navigation. The adaptation skills and self-learning will help robots in the daily interaction of the man and the machine. Most of the applications in which robots are expected to be used require that the robot has adaptive and learning capabilities. Reinforcement learning is a research area that will helps robots learn autonomously and adapt to new environments and situations. This paper focuses on teaching a robot to navigate through the use of reinforcement learning. The system that is proposed is not based on the use of a map or on the previously known environment. The main objective is to provide a learning system to the robot, allowing him to navigate in a totally new environment. It also seeks to adapt the robot to a new environment by learning to evade obstacles and explore the new terrain. The development of this type of skills will help the robot to adapt and to fulfill tasks much more complex than a simple navigation. The adaptation skills and self-learning will help robots in the daily interaction of the man and the machine in the future.

The rest of the document is organized as follows: Section II addresses the Related Work, in the context of Q-Learning applied to robotics, Section III explains the proposed approach, where the states and the laws are defined, Section IV presents

978-1-7281-0137-8/19/$31.00 ©2019 IEEE

Figura 49. Artículo presentado en la conferencia "Southeastcon 2019"
Fuente: [43]

Este artículo obtuvo comentarios satisfactorios dados por los jurados de la conferencia.

14. BIBLIOGRAFÍA

[1] C. Fu, A. Carrio, M.A. Olivares-Mendez, R. Suarez-Fernandez y P. Campoy, “Robust real-time vision-based aircraft tracking from Unmanned Aerial Vehicles”. IEEE International Conference on Robotics and Automation (ICRA), 24 September 2014.

[2] C.Wang, J. Wang, X. Zhang and X. Zhang, “Autonomous Navigation of UAV in large-Scale unknown complex environment with deep reinforcement learning”, Tsinghua University, China.

[3] H.X. Pham, Hung. M. La, D. Feil-Seifer, L.V. Nguyen, “Autonomous UAV Navigation Using Reinforcement Learning”, 2018.

[4] Baez, H., Perez, K., Rojas, E., Rodríguez, S., Lopez, J., Quintero, C., & Calderón, J. M. (2013, November). Application of an educational strategy based on a soccer robotic platform. In 2013 16th International Conference on Advanced Robotics (ICAR) (pp. 1-6). IEEE

[5] Calderon, J. M., Rojas, E. R., Rodriguez, S., Baez, H. R., & Lopez, J. A. (2012, July). A Robot soccer team as a strategy to develop educational iniciatives. In Latin American and Caribbean Conference for Engineering and Technology, Panama City, Panama.

[6] Rodrıguez, S., Rojas, E., Pérez, K., Quintero, C., Pena, O., Reyes, A., & Calderón, J. (2013). STOx’s 2013 Team Description Paper.

[7] Rodrıguez, S., Rojas, E., Pérez, K., López, J., Quintero, C., Calderón, J. M., & Pena, O. (2014). STOx’s 2015 Extended Team Description Paper. Joao Pessoa, Brazil.

[8] Rodríguez, S., Rojas, E., Pérez, K., López, J., Quintero, C., & Calderón, J. (2014, July). Fast path planning algorithm for the robocup small size league. In Robot Soccer World Cup (pp. 407-418). Springer, Cham.

[9] Quintero, C., Rodríguez, S., Pérez, K., López, J., Rojas, E., & Calderón, J. (2014, July). Learning soccer drills for the small size league of robocup. In Robot Soccer World Cup (pp. 395-406). Springer, Cham.

[10] Cardona, G. A., Moreno, W., Weitzenfeld, A., & Calderon, J. M. (2016, March). Reduction of impact force in falling robots using variable stiffness. In SoutheastCon 2016 (pp. 1-6). IEEE.

[11] Elibol, E., Calderon, J., Llofriu, M., Quintero, C., Moreno, W., & Weitzenfeld, A. (2015, July). Power usage reduction of humanoid standing process using q-learning. In Robot Soccer World Cup (pp. 251-263). Springer, Cham.

[12] Elibol, E., Calderon, J., & Weitzenfeld, A. (2013, June). Optimizing energy usage through variable joint stiffness control during humanoid robot walking. In Robot Soccer World Cup (pp. 492-503). Springer, Berlin, Heidelberg.

[13] Calderon, J. M., Elibol, E., Moreno, W., & Weitzenfeld, A. (2013). Current usage reduction through stiffness control in humanoid robot. In 8th Workshop on Humanoid Soccer Robots, IEEE-RAS International Conference on Humanoid Robots.

[14] G. Sánchez, M. Mulero, E. Saumeth, "Vehículos Aéreos no tripulados, Drones", Mayo 2013. Available: https://www.infodefensa.com/wp-content/uploads/Vehiculos_aereos_no_tripulados_en_Latam.pdf

[15] C.Juan, "¿Cómo aprovechar los drones en la industria logística?", Escuela de negocios de la innovación y los emprendedores, España, 2016. Available: https://www.iebschool.com/blog/drones-en-la-industria-logistica/

[16] J. Lluís Micó, "Drones (para) reporteros", La vanguardia, España, 2018. Available: https://www.lavanguardia.com/tecnologia/20180225/441003672054/dron-periodismo-reporteros-crisis-universidad.html

[17] P. Fruticola, "Aplicación de productos químicos con drones en la agricultura", agritotal.com, Argentina, 2017. Available: https://www.agritotal.com/nota/32047-aplicacion-de-productos-quimicos-con-drones-en-la-agricultura/

[18] R. S. Sutton and A. G. Barto, Reinforcement learning: An Introduction. Cambridge, MA: MIT press, 1998.

[19] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A Survey," International Journal of Robotics Research, vol. 32, no. 11, pp. 1238–1274, 2013

[20] N. Imanberdiyev, C. Fu, E. Kayacan and M. Chen, "Autonomous Navigation of UAV by Using Real-Time Model-Based Reinforcement Learning", School of Mechanical and Aerospace Engineering 2ST Engineering, 2016.

[21] T. Sugimoto and M. Gouko, "Acquisition of hovering by actual UAV using reinforcement learning", Tohoku Gakuin University, Japan, 2016.

[22] J.Q. Cui, S. Lai, X. Dong, P. Liu, B.M. Chen, T.H. Lee, "Autonomous Navigation of UAV in forest", IEEE International Conference on Unmanned Aircraft Systems (ICUAS) 26 June 2014.

[23] S. R. B. dos Santos, C. L. Nascimento, and S. N. Givigi, "Design of attitude and path tracking controllers for quad-rotor robots using reinforcement learning," in Aerospace Conference, 2012.

[24] Y. Zeng, G. Wang and B. Xu, "A Basal Ganglia Network Centric Reinforcement Learning Model and Its Application in Unmanned Aerial Vehicle". IEEE Transactions

on cognitive and developmental systems, June 2016.

[25] Z. Ma, C. Wang, Y. Niu, Z, Wang and L. Shen, “A saliency-based reinforcement learning approach for a UAV to avoid flying obstacles” Robotics and Autonomous Systems, Volume 100, February 2018, Pages 108-118.

[26] B. Zhang, W. Liub, Z. Maoa, J. Liud, L. Shenc, “Cooperative and Geometric Learning Algorithm (CGLA) for path planning of UAVs with limited information”, Automatica Volume 50, Issue 3, March 2014, Pages 809-820

[27] C.A. Ibagué Ramos, J.S. Parra Torres, Comparación de estrategias de navegación utilizando teoría de multiagentes para el barrido de zonas posiblemente minadas, Universidad Santo Tomás, 2017.

[28] W. Quesada, “Generación de comportamientos de enjambre en robots móviles a través del uso del aprendizaje por refuerzo”, Universidad Santo Tomás, 2019.

[29] J. Kober, J. A. Bagnell and J. Peters, "Reinforcement learning in robotics: A survey," The International Journal of Robotics Research, vol. 32, (11), pp. 1238-1274, 2013. Available: https://doi.org/10.1177/0278364913495721. DOI: 10.1177/0278364913495721.

[30] P. Bristeau et al, "The Navigation and Control technology inside the AR. Drone micro UAV," IFAC Proceedings Volumes, vol. 44, (1), pp. 1477-1484, 2011. Available: https://www.sciencedirect.com/science/article/pii/S1474667016438188.

[31] DOI: 10.3182/20110828-6-IT-1002.02327

[32] Quesada, W. O., Rodriguez, J. I., Murillo, J. C., Cardona, G. A., Yanguas-Rojas, D., Jaimes, L. G., & Calderón, J. M. (2018, June). Leader-Follower Formation for UAV Robot Swarm Based on Fuzzy Logic Theory. In International Conference on Artificial Intelligence and Soft Computing (pp. 740-751). Springer, Cham.

[33] León, J., Cardona, G. A., Jaimes, L. G., Calderón, J. M., & Rodriguez, P. O. (2018, June). Rendezvous Consensus Algorithm Applied to the Location of Possible Victims in Disaster Zones. In International Conference on Artificial Intelligence and Soft Computing (pp. 700-710). Springer, Cham.

[34] Cardona, G.A.; Calderon, J.M. Robot Swarm Navigation and Victim Detection Using Rendezvous Consensus in Search and Rescue Operations. Appl. Sci. 2019, 9, 1702.

[35] Yanguas-Rojas, David, et al. Victims search, identification, and evacuation with heterogeneous robot networks for search and rescue. En 2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC). IEEE, 2017. p. 1-6.

[36] Cardona, G.A., Yanguas-Rojas, D., Arevalo-Castiblanco, M.F. & Mojica-Nava, E. 2019, "Ant-based multi-robot exploration in non-convex space without global-connectivity constraints", 2019 18th European Control Conference, ECC 2019, pp.

2065.

[37] León, J., Cardona, G. A., Botello, A., & Calderón, J. M. (2016, December). Robot swarms theory applicable to seek and rescue operation. In International Conference on Intelligent Systems Design and Applications (pp. 1061-1070). Springer, Cham.

[38] Di Wu, Member, IEEE, Dmitri I. Arkhipov, Minyoung Kim, Carolyn L. Talcott, Amelia C. Regan, Member, IEEE, Julie A. McCann, Member, IEEE and Nalini Venkatasubramanian, Senior Member, IEEE, “ADDSEN: Adaptive Data Processing and Dissemination for Drone Swarms in Urban Sensing”, 2017.

[39] Transductor, (4 Julio, 2017), Interacción entre V-REP y Matlab. [Online], Available: https://robologs.net/2017/07/04/interaccion-entre-v-rep-y-matlab/

[40] Adaptive Computation and Machine Learning, Thomas Dietterich, series editor, Christopher Bishop, David Heckerman, Michael Jordan, and Michael Kearns, associate editors.

[41] Pitonakova, L, Giuliani, M., Pipe, A, Winfield, A. (2018) Feature and performance comparison of the V-REP, Gazebo and ARGoS robot simulators. Proceedings of the 19th Towards Autonomous Robotic Systems Conference (TAROS 2018), Lecture Notes in Computer Science, vol 10965, Springer, 357-368. http://lenkaspace.net/tutorials/programming/robotSimulatorsComparison

[42] R. Kurozumi, S. Fujisawa, T. Yamamoto and Y. Suita.” Path planning for mobile robots using an improved reinforcement learning scheme,”Proceedings of the 41st SICE Annual Conference. SICE 2002., 2002, pp.2178-2183 vol.4.

[43] Cardona, G. A., Bravo, C., Quesada, W., Ruiz, D., Obeng, M., Wu, X., and Calderon, J. M. "Autonomous Navigation for Exploration of Unknown Environments and Collision Avoidance in Mobile Robots Using Reinforcement Learning.” In SoutheastCon 2019, pp. 1-6. IEEE, 2019.