Diseño e implementación de un sistema inalámbrico de comunicación entre un ambiente virtual y un agente en el mundo real

Sergio Isahí Garrido Castañeda
Instituto Politécnico Nacional
CIDETEC
CDMX, México
sgarridoc2000@alumno.ipn.mx

Gabriel Sepúlveda Cervantes
lnstituto Politécnico Nacional
CIDETEC
CDMX, México
gsepulveda@ipn.mx

Eduardo Vega Alvarado
Instituto Politécnico Nacional
CIDETEC
CDMX, México
evega@ipn.mx

***Resumen*— En el aprendizaje automático de máquina, los sistemas son capaces de aprender mediante datos. Una variante de esta disciplina es el aprendizaje por refuerzo, en la cual un agente aprende mediante la observación del ambiente en el que se encuentra y la asignación de una recompensa por cada acción que realiza. En el caso de que el entrenamiento haya sido realizado en un ambiente virtual y el resultado de este requiera ser llevado a un agente en el mundo real, es necesario un sistema de comunicación que pueda recabar las observaciones necesarias del ambiente, que serán computadas para asignar una acción a realizar por el agente con base a las políticas establecidas durante el entrenamiento. En el presente trabajo, se explica el desarrollo de un sistema de comunicación inalámbrica entre un ambiente virtual y un vehículo terrestre de tracción diferencial el cual es la representación de un agente en el mundo real. El sistema se implementó físicamente, y los resultados obtenidos mostraron su viabilidad y eficiencia.**

Palabras Clave— Aprendizaje por refuerzo, ambiente virtual, comunicación inalámbrica, vehículo terrestre.

***Abstract*— In machine learning, systems are capable of learning through data. A variant of this discipline is reinforcement learning, in which an agent learns by observing the environment in which he is found and assigned a reward for each action he takes. If the training is carried out in a virtual environment and its result requires to be taken to an agent in the real world, a communication system is necessary to collect the observations of the environment, which will be computed for assigning an action to be performed by the agent based on the policies established during the training. In this article, it is explained the development of a wireless communication system between a virtual environment and a differential traction land vehicle, which is the representation of an agent in the real world. The system was implemented physically, and the results shows its viability and efficiency.**

Keywords— Reinforcement learning, virtual environment, wireless communication, land vehicle.

I. Introducción

Por su definición, el aprendizaje automático de máquina (*machine learning*) es el área de la ingeniería que describe las técnicas mediante las cuales un sistema es capaz de aprender y generalizar algún comportamiento deseado haciendo uso de datos. El aprendizaje por refuerzo es el paradigma del aprendizaje automático en el cual un agente qué tiene la capacidad de realizar un conjunto finito de acciones y de comprender el entorno en el que está inmerso, aprende mediante un proceso que cuantifica si la tarea se está realizando de la manera deseada, mediante la asignación de recompensas a cada una de las acciones ejecutadas por el agente; el aprender a caminar, andar en bicicleta, percatarse que el fuego quema, etcétera, son ejemplos claros de esto.

Fue a finales de la década de 1980 que el aprendizaje por refuerzo comenzó a atraer el interés de la comunidad científica [1]. Sus aplicaciones en el ámbito de la investigación son variadas y van desde sistemas capaces de obtener puntajes jamás alcanzados por humanos en videojuegos [2] [3], algoritmos de sugerencia de publicidad con base al comportamiento de usuarios en la web [4], hasta la creación de una inteligencia artificial capaz de vencer al campeón mundial de Go después de solamente unas pocas horas de entrenamiento [5]. Si bien estos ejemplos pudieran no parecer de alto impacto, demuestran los alcances de esta modalidad de la mano de unidades y herramientas computacionales cada vez más potentes.

Por lo anterior, es posible afirmar que la flexibilidad del aprendizaje por refuerzo recae principalmente en la posibilidad de que, en el sistema a diseñar, se definan cada una de las partes que lo componen, siendo la robótica móvil un campo con gran afinidad a la aplicación de algoritmos relacionados a este paradigma de aprendizaje, debido a la flexibilidad intrínseca que se presenta en el diseño de estos sistemas en cuanto a hardware y software. Así mismo, uno de los componentes esenciales para esta integración es un subsistema de comunicación que permita trasladar el resultado del entrenamiento realizado por una herramienta de cómputo a un agente en el mundo real.

El objetivo del presente trabajo es explicar la implementación de un sistema compuesto por un ambiente virtual, un vehículo terrestre en el mundo real y un sistema de comunicación inalámbrica que permite el transporte de la información recabada por el vehículo terrestre al ambiente virtual. Este sistema constituye la infraestructura para la aplicación del resultado en el mundo real de un entrenamiento realizado en el ambiente virtual; lo anterior es posible debido a que los componentes inherentes al aprendizaje por refuerzo están completamente definidos en dicho sistema.

II. Parámetros de aprendizaje por refuerzo

Como se mencionó anteriormente, en el aprendizaje por refuerzo el agente, sin importar su naturaleza, se enfrenta al problema de decidir las acciones a realizar, por lo que se estudian las repercusiones de dichas acciones y se proporciona al agente el aprendizaje para optimizarlas, mediante la maximización de una recompensa. Para la

implementación del aprendizaje por refuerzo se proponen los parámetros de modelado que se detallan a continuación.

A. Procesos de Decisión de Marcov

Los procesos de decisión de Markov (MDP) son una abstracción de los componentes de la toma de decisiones. Partiendo de la existencia de un agente interactuando en un ambiente, las interacciones se dan en forma de acciones derivadas de la toma de decisiones del agente hechas de manera secuencial en cada paso de tiempo. El agente obtiene una representación del estado del ambiente y, de acuerdo con ella, selecciona y realiza una acción dentro de un conjunto finito. La acción cambia el estado del ambiente en el siguiente paso de tiempo y el agente recibe una recompensa de acuerdo con que tan buena fue la decisión tomada. Una trayectoria es la secuencia de la toma de decisión para pasar al estado siguiente con la recompensa correspondiente al estado anterior. El agente trata de maximizar tanto la recompensa del estado siguiente como la recompensa acumulada a lo largo de la trayectoria.

En un MDP se denomina S al conjunto finito de estados, mientras que A es el conjunto finito de acciones que el agente puede seleccionar, que traen como consecuencia que obtenga una recompensa perteneciente al conjunto R. El proceso anterior se ejecuta en un estado de tiempo del conjunto t = 0,1,2, ..., T, en el cual el agente recibe al estado $S_t \in S$ y ejecuta una acción $A_t \in A$, obteniendo así el par (S_t, A_t) y la recompensa $R_{t+1} \in R$; posteriormente se pasa al tiempo $t + 1$, donde el ambiente pasa al estado $S_{t+1} \in S$. Lo anterior se resume en la expresión (1):

$$f(S_t, A_t) = R_{t+1} \tag{1}$$

B. Retorno

Dado que el objetivo de un agente es maximizar la recompensa acumulada, todas sus decisiones se toman orientadas a este fin. Las recompensas se calculan mediante el retorno esperado en un plazo de tiempo determinado, donde se define a G como el retorno en el momento t, tal como se expresa en la Ec. (2):

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + R_T \tag{2}$$

Un episodio es una porción de la trayectoria, siempre y cuando tenga un paso de tiempo final, que termina en el tiempo T y el siguiente estado pasa a ser uno con condiciones iguales a las de $t = 0$ o algún otro dentro de un conjunto de estados posibles. En particular, el episodio siguiente es independiente del estado final del episodio que le antecedió. Las tareas que no tienen definido un tiempo final T son tareas continuas en las que $T = \infty$, y en consecuencia $G = \infty$. Para solucionar este problema es necesario hacer que el agente considere el retorno esperado, dándole un peso mayor a las recompensas inmediatas obtenidas en cada paso de tiempo, pero con un descuento. La tasa de descuento γ puede tomar un valor entre 0 y 1, y es el mecanismo mediante el cual se descontarán las recompensas futuras y a su vez se determinará el valor presente de dichas recompensas, como se indica en la Ec. (3). Se puede observar que, al introducir la tasa de descuento, el agente pasa en automático a considerar las recompensas inmediatas para decidir las acciones que realizará, mientras que las recompensas futuras tienden a tener un menor peso mientras mayor sea el paso del tiempo, lo cual se expresa en la Ec. (4),

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \\ &= \textstyle\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \end{aligned} \tag{3}$$

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned} \tag{4}$$

III. VEHÍCULO TERRESTRE

El sistema desarrollado tiene como objetivo final establecer una comunicación entre un agente en el mundo real y un ambiente virtual. Esto permite definir restricciones convenientes al diseño de dicho agente en forma de características, las cuales a su vez pueden aplicarse a los componentes restantes del sistema.

A. Características del vehiculo terrestre

Las características del vehículo terrestre son las funcionalidades que darán forma al MDP del sistema por lo que, con base a la definición de éste, se tiene un conjunto finito de estados S y un conjunto finito de acciones A. El conjunto S está ligado a todo aquello que el robot puede percibir de su ambiente, por lo que en el agente se ve reflejado como sensores, mientras que el conjunto de acciones A esta ligado a su locomoción.

Para las observaciones se estableció que el vehículo terrestre pudiese medir la distancia a la que se encuentran los obstáculos a su alrededor y, a su vez, tuviera la funcionalidad de adquirir imágenes de lo que tiene frente a él. En correspondencia, las acciones que el vehículo terrestre tiene la capacidad de realizar se son cuatro: moverse hacia adelante, moverse hacia atrás, rotar a la derecha y rotar a la izquierda. El ultimo requerimiento a cumplir es que sea posible enviar de manera inalámbrica los datos recopilados por los sensores y a su vez recibir información para su interpretación en forma de las acciones que debe realizar.

B. Diseño

Para la locomoción del vehículo terrestre se seleccionó una configuración con ruedas de tracción diferencial, la cual requiere de dos motores colocados de manera colineal en cada uno de los extremos traseros de una plataforma y de una rueda de giro sin fin en la parte delantera. Cada uno de los motores es capaz de girar independientemente, siendo la combinación del giro de ambos lo que permite la variación de movimiento del vehículo terrestre.

Para el censado del ambiente fueron utilizados tres sensores ultrasónicos colocados en el frente del vehículo, uno en el centro y los restantes en cada extremo a un ángulo de 60°. Para la adquisición de imagen se colocó una cámara debajo del sensor central, mientras que para la comunicación inalámbrica la tecnología seleccionada fue *WiFi*, debido a su fácil implementación y a su velocidad de transferencia de datos.

Los componentes electrónicos utilizados se muestran en la Tabla 1.

Tabla 1. Componentes del vehículo terrestre.

Elemento	Cantidad
Microcontrolador ESP32	1
Microcontrolador ESP32-Cam	1
Sensor ultrasónico HC-SR04	4
Pantalla OLED 0.96 pul.	1
Driver MX1508	1
Convertidor MT3608	1
Batería 5000mA 5V	1

El funcionamiento de la electrónica se basa en primera instancia en un microcontrolador ESP32, el cual tiene la tarea de obtener las distancias medidas por los tres sensores ultrasónicos HC-SR04 y a su vez, gestionar la activación de cada una de las entradas del *driver* que controla tanto el sentido de giro como la velocidad de cada uno de los motores. Este microcontrolador también gestiona el envío de la información de conexión y lecturas de sensores a la pantalla OLED mediante el protocolo I2C. Para la obtención de la imagen frente al vehículo se hace uso de una cámara OV7670 conectada a un microcontrolador ESP32 CAM.

Con respecto a la alimentación eléctrica del móvil, ésta es suministrada por una batería de 5v, 5000 mA. Dicha tensión alimenta directamente a los sensores ultrasónicos y a los microcontroladores ESP32 y ESP32 CAM. Un *driver* MX1508 es utilizado para el control de giro y velocidad de los motores de CD, mientras que el voltaje de alimentación de dichos motores es suministrado por un convertidor de voltaje *step up* MT3608 calibrado a 6v y también conectado a la batería.

C. *Construcción.*

Para el chasis del modelo de vehículo terrestre construido se tomaron como base algunos componentes de un kit educativo de la marca *RoboRobo*: la plataforma base, postes para la unión con el segundo nivel y las piezas correspondientes a este mismo, tornillería y los propios motores de corriente directa con sus respectivas ruedas. En el caso de las piezas restantes requeridas, conformadas en su totalidad por soportes para componentes electrónicos tales como sensores, tarjetas electrónicas y batería, fueron modeladas a medida en el software CAD *OnShape* y posteriormente fueron impresas en material *PLA* y montadas en el chasis previamente descrito.

Figura 1. Vehículo terrestre.

Las dimensiones finales del vehículo son 14.4 cm de largo por 15 cm de ancho ya considerando la longitud extra por concepto de los sensores montados en la parte frontal. El diseño final se muestra en la Figura 1.

IV. Comunicación

Como consecuencia del establecimiento de *WiFi* como la tecnología para el envío y recepción de datos entre el agente y el ambiente virtual, el protocolo de comunicación a utilizar para ese fin debía cumplir dos características principales: mantener una comunicación persistente entre los dispositivos conectados a una red local, y el envío y recepción de datos de manera rápida. Si bien en la actualidad existen diversos protocolos que permiten realizar estas dos funciones, *WebSocket*s es el que se ajusta mejor a las características del sistema, por lo que fue seleccionado para la implementación.

A. *WebSockets*

Los *WebSocket*s permiten una comunicación bidireccional y *FullDuplex* entre dispositivos conectados entre sí. La conexión comienza con una solicitud del cliente, la cual el servidor responde mediante un evento denominado *handshake*, en el cual el servidor acepta la solicitud y posterior a ello, la comunicación permanece abierta. Es mediante eventos que los datos son enviados y recibidos por ambas partes; tanto el servidor como el cliente esperan en todo momento la activación de alguno de los eventos y actúan de acuerdo a cuál fue el que se activó.

B. *Servidor y clientes WebSocket*

En el protocolo *WebSocket* existen diversos eventos que pueden ser escuchados por el servidor siendo cinco los principales para establecer una comunicación sencilla:

- Evento *Listening*: Establece la dirección y puerto en el cual la conexión se llevará a cabo.
- Evento *Connection*: Petición de conexión de un cliente.
- Evento *Message*: Indica que se ha recibido un mensaje.
- Evento *Close*: Uno de los clientes ha cerrado la conexión.
- Evento *Error*: Se activa cuando se detecta un error en la comunicación entre el servidor y el cliente.

En el servidor *WebSocket* es donde se lleva a cabo el transporte y direccionamiento de la información. Este se encuentra alojado en el *localhost* del equipo que concentra al ambiente virtual en un puerto especifico.

En el caso de los clientes *WebSocket*, al igual que en el servidor, escuchan eventos y dependiendo del mismo realizan distintas acciones. En este caso solo se hace uso de los eventos *Connection* y *Message*, lo cual representa que la conexión se realiza al servidor mediante su dirección IP y

puerto, y en el tiempo subsecuente se mantendrán a la espera de algún mensaje para realizar las acciones correspondientes.

C. *Funciones WebSocket del sistema*

Las acciones que pueden realizar tanto el servidor como los clientes *WebSocket* son las siguientes:

- *Send*: Envía información ya sea en forma de una cadena de texto o un arreglo de bytes. En el caso de que un cliente haga uso de esta función, la información es escuchada por el servidor y puede o no ser contestada a los demás clientes.
- *Close*: Se encarga de cerrar la comunicación.

Para el caso del sistema diseñado, se cuenta con un servidor y tres clientes, los cuales corresponden al ambiente virtual, al microcontrolador ESP32 encargado de recopilar la lectura de los sensores ultrasónicos y al microcontrolador ESP32 encargado de obtener la imagen de lo que se encuentra frente del vehículo terrestre. El cliente montado en el ambiente virtual escucha los mensajes del servidor, los cuales vienen de parte de dos clientes restantes y es capaz de recibir información tanto en cadena de caracteres (correspondientes a las lecturas de sensores) como en arreglos de bytes (información de la imagen de la cámara).

El cliente *WebSocket* embebido en el microcontrolador ESP32 escucha mensajes del servidor únicamente en cadenas de caracteres, y con base en la cadena recibida realiza una de las acciones planteadas en el MDP:

- Adelante: Los dos motores del vehículo terrestre giran a la misma velocidad y en el mismo sentido, hacia adelante.
- Atrás. Los dos motores giran a la misma velocidad en el mismo sentido, hacia atrás
- Derecha: Los motores giran en diferente sentido, el derecho gira hacia adelante y el motor izquierdo gira hacia atrás a una velocidad menor.
- Izquierda: Los motores giran en diferente sentido, el de la izquierda gira hacia adelante y el motor derecho hacia atrás a una velocidad menor.

El cliente embebido en el microcontrolador ESP32 CAM, no recibe alguna clase de información.

V. Ambiente Virtual

El ambiente virtual es el medio por el cual se realizará el futuro entrenamiento del agente; en este se deben modelar cada uno de los componentes del MDP del sistema, así como al agente. Posterior a la realización del entrenamiento, este ambiente será el encargado de definir las acciones que debe ejecutar el agente en cada estado de tiempo, con base en los datos provenientes del mundo real. Por lo anterior, en esta versión preliminar del ambiente virtual se definió que debe ser posible almacenar la información recibida debido a que en su conjunto, son las entradas del cómputo que establecerá que acción le corresponde al agente realizar. A su vez, la información debe ser mostrada mediante una interfaz gráfica de usuario.

VI. Herramientas

Las herramientas utilizadas en el desarrollo de este sistema de comunicación corresponden en su totalidad a la generación del ambiente virtual (en este punto interfaz gráfica de usuario), la implementación del servidor *WebSocket* y el Entorno Integrado de Desarrollo para la programación de microcontroladores; dichas herramientas se detallan a continuación.

A. *Unity 3D*

Unity 3D (*Unity Technologies*) es una herramienta capaz de crear entornos virtuales que reflejen las condiciones exactas del ambiente de entrenamiento, por medio de un motor de desarrollo de plataformas interactivas. Para la simulación de fenómenos físicos cuenta con los motores *Nvidia PhysX* o *Havock Physics*, además de tener la ventaja de ser flexible a la adecuación de motores de otros grupos de desarrolladores, tales como *Bullets* y *MuJoCo* [6]. Adicionalmente, para el desarrollo de aplicaciones de Inteligencia Artificial se cuenta con *ML-Agents*, una interfaz de programación de aplicaciones que permite utilizar entornos virtuales realizados tanto en *Unity 3D* como en ambientes de entrenamiento para sistemas de aprendizaje automático desarrollados previamente [7].

B. *Node JS*

Node JS es un entorno basado en el lenguaje de programación *Java Script*, capaz de crear aplicaciones del lado del servidor que permiten gestionar múltiples conexiones al mismo tiempo, por lo que se ha posicionado como la herramienta principal para el desarrollo de aplicaciones web en tiempo real. Su principal característica es que las entradas y salidas de datos son asíncronos con una arquitectura orientada exclusivamente a eventos.

C. *PlatformIO*

La extensión del entorno integrado de desarrollo *VS Code,* que permite la programación de microcontroladores de diversas plataformas (incluida la familia ESP32), incluye diferentes herramientas de uso simple para el desarrollo de proyectos, además de un gestor de bibliotecas y soporte para diferentes *framewoks* en distintos lenguajes de programación.

VII. Integración

Dentro de una red local, a cada dispositivo conectado a ella se le asigna una dirección IP que funge como su identificación; dicha dirección puede ser estática o dinámica. El equipo de cómputo que ejecuta algún programa y está conectado a una red local, puede ser llamado *localhost*. Por lo anterior, los microcontroladores ESP32 y ESP CAM poseen una dirección IP; el equipo de computo en el que se ejecuta el Servidor *WebSocket* y el ambiente virtual posee también una dirección IP, pero a su vez esta dirección puede adquirir el nombre *localhost*.

La integración de cada uno de los elementos del sistema se traduce en que los clientes (ambiente virtual y microcontroladores) tengan como parámetro de conexión la dirección IP del servidor así como el puerto, mientras que, del lado del servidor la dirección IP de cada uno de los clientes permite conocer el origen y destino de los datos.

Para la ejecución del sistema, es necesario primero lanzar al servidor, para posteriormente lanzar a los clientes. El servidor no cuenta con interfaz gráfica, por lo que únicamente puede visualizarse por mensajes de consola que indican el estatus del mismo, así como los clientes que han logrado establecer una conexión con él. El ambiente virtual cuenta con una interfaz gráfica de usuario la cual muestra los datos de las distancias medidas por los sensores ultrasónicos del vehículo terrestre, así como la imagen proporcionada por la cámara.

Finalmente, el protocolo *WebSocket* permite reconexiones de los clientes en el caso de que alguno de ellos haya perdido la conexión por lo que, si el ambiente virtual se está ejecutando y el vehículo terrestre es desconectado, puede volver a recibir la información una vez que la reconexión se haya producido.

VIII. Resultados.

Para la ejecución del sistema de comunicación propuesto, es necesario en primera instancia inicializar el servidor para que así los clientes puedan establecer la conexión con él. Si bien el orden en el que la conexión se realiza no es relevante, se estableció que el vehículo terrestre fuera el primero en conectarse, mientras que el ambiente virtual fuera el último; ello debido a que el ambiente requiere ser ejecutado desde Unity 3D para la inicialización del cliente *WebSocket*. La lista de clientes conectados y el estado del servidor puede observarse por consola, tal como se muestra en la Figura 2.

```
PS C:\Users\Estudiantes Maestria\Desktop\websocket-server> node .
Servidor inicializado:
Puerto: 8080
Cliente conectado: Robot_1
Cliente conectado: Camara_1
Cliente conectado: Unity
```

Figura 2. Consola del Servidor.

Una vez que los clientes se encuentran conectados al servidor, es posible observar en el ambiente virtual los cambios en las mediciones de los sensores del vehículo terrestre, así como la imagen de la cámara montada. Es importante recalcar que cualquier fuente de datos que se encuentra dentro del ambiente virtual puede ser utilizada para realizar procesos dentro del mismo, por lo que tanto los datos de imagen como las distancias recabadas por los sensores son accesibles para utilizarse como observaciones dentro del resultado de algún entrenamiento previamente realizado. El diseño de la interfaz gráfica puede observarse en la Figura 3.

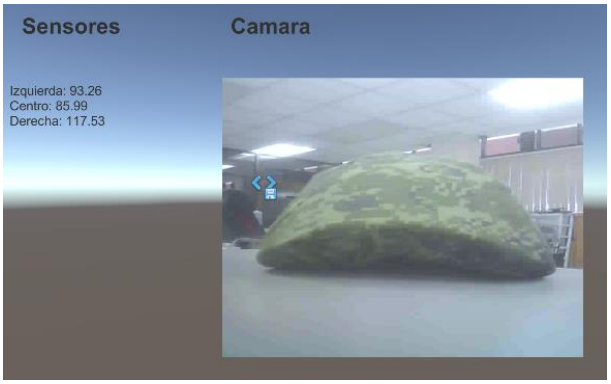

Figura 3. Interfaz de usuario.

El sistema permite la reconexión de clientes que se hayan desconectado del servidor, siendo el ambiente virtual el único que tiene la capacidad de cerrar la conexión, esto mediante la suspensión de la ejecución en Unity 3D (ver Figura 4). Finalmente, referente a la distancia efectiva del sistema, 20 metros fue la mayor distancia alcanzada por el vehículo en un ambiente cerrado sin muros sin que alguno de los clientes (ESP32 y ESP32 CAM) perdiera la conexión con el servidor.

```
PS C:\Users\Estudiantes Maestria\Desktop\websocket-server> node .
Servidor inicializado:
Puerto: 8080
Cliente conectado: Robot_1
Cliente conectado: Camara_1
Cliente conectado: Unity
Conexion cerrada por: Unity
```

Figura 4. Conexión cerrada.

IX. Conclusiones y trabajo a futuro

Al finalizar el desarrollo de este proyecto, se concluye que para el diseño y puesta en marcha de un sistema de comunicación entre un ambiente virtual y un agente en el mundo real por medio de *WiFi* y *WebSockets*, es necesario definir al conjunto finito de observaciones *S* y al conjunto finito de acciones *A,* ya que esto permite conocer los parámetros a tomar en cuenta en la construcción del agente en el mundo real, el tipo de datos que se enviarán y recibirán, así como las características del servidor y del ambiente virtual.

Como trabajo a futuro considera escalar el sistema de comunicación con el fin de obtener datos de más agentes, tanto de vehículos terrestres como aéreos; con el fin de realizar su entrenamiento en un ambiente virtual (diseñado en Unity 3D) para una tarea colaborativa específica y que posteriormente el resultado del entrenamiento sea probado en el mundo real.

X. Referencias

[1] L. Kaelbling, M. Littman, A. Moore, “Reinforcement Learning: A survey”, Journal of Artificial Intelligence Research 4, 1996. https://doi.org/10.1613/jair.301

[2] M. Machado, M. Bellmare, E. Talvitie, J. Veness, “Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents”, 2017. arXiv:1709.06009v2

[3] V. Mnlh, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou,” Playing Atari with Deep Reinforcement Learning”, Deepmind Technologies, 2013 rXiv:1312.5602v1

[4] L. Li, W. Chu, J. Langford, R. Schapire, “A Contextual-Bandit Approach to Personalized News Article Recommendation”, Yahoo! Labs- Dept of Computer Science Princeton University 2012. arXiv:1003.0146v2

[5] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Juang, “Mastering the game of Go without human knowledge” Nature 550, 2017. https://doi.org/10.1038/nature24270

[6] A. Juliani, V. Berges, E. Teng, A. Cohen, J. Harper, ”Unity: A General Platform for Intelligent Agents”, 2018. arXiv:1809.02627v2

[7] G, Sepúlveda, E. Vega, E. Portilla “Machine Learning para Robots, del Entrenamiento Virtual a la Tarea Real”, Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI, Publicación semestral Pädi Vol. 7 2019. https://doi.org/10.29057/icbi.v7iEspecial.4785