ISSN 1870-4069

Implementación de un algoritmo de aprendizaje por refuerzo en ambientes virtuales, orientado a robótica móvil

Sergio Isahí Garrido Castañeda[1], Gabriel Sepúlveda Cervantes[1], Eduardo Vega Alvarado[1], Edgar Alfredo Portilla Flores[1], Miguel Ángel Garrido Castañeda[2]

[1] Instituto Politécnico Nacional, Centro de Innovación y Desarrollo Tecnológico en Cómputo, México

[2] Instituto Politécnico Nacional, Unidad Profesional Interdisciplinaria en Tecnologías Avanzadas, México

sgarridoc2000@alumno.ipn.mx, {gsepulvedac, evega, aportilla}ipn.mx, kurogarri@gmail.com

Resumen. En este trabajo se presenta un sistema de entrenamiento para robots móviles utilizando aprendizaje de máquina (*Machine Learning*) implementado en un ambiente virtual. En los últimos años, el uso de técnicas de inteligencia artificial para resolver problemas del mundo real ha ido en aumento. Las aplicaciones de estas herramientas abarcan una amplia gama de disciplinas, destacando la solución de problemas de clasificación y/u optimización. En el ámbito de la robótica móvil, esto ha abierto el horizonte a nuevos paradigmas para el control de las variables que rigen estos sistemas, siendo el aprendizaje por refuerzo uno de los modelos de aprendizaje de máquina que puede adaptarse mejor a este fin. Por otro lado, el desarrollo de ambientes virtuales en donde se modelan las características físicas de un agente sujeto a un proceso de aprendizaje permite tanto el entrenamiento del agente como la transferencia del entrenamiento al mundo real, teniendo como beneficios la simulación del comportamiento del agente en diversos ambientes, el entrenamiento evitando desgaste físico, y la realización de pruebas sin riesgos relacionados a daños del hardware o incluso a los seres vivos. Asimismo, el desempeño de los equipos de cómputo actuales permite realizar el entrenamiento de manera eficiente en periodos de tiempo cortos. Los resultados obtenidos muestran la flexibilidad del esquema propuesto, el cual permite repetir exitosamente el aprendizaje modificando las condiciones y las actividades a realizar por el agente.

Palabras clave: Aprendizaje por refuerzo, ambiente virtual, agente, robótica móvil.

Implementation of a Reinforcement Learning Algorithm in Virtual Environments, Oriented to Mobile Robotics

Abstract. In this work, a training system for mobile robots using machine learning implemented on a virtual environment is presented. In recent years, the

use of AI techniques for solving real world problems has increased. The application of these tools covers a wide variety of areas, highlighting the solution of classification and/or optimization problems. In the area of mobile robotics, this development has opened new paradigms to control the variables that govern these systems, and the reinforcement learning is one of the models of machine learning that can be best adapted for this task. Additionally, the development of virtual environments where the physical characteristics of an agent in a learning process are modeled, allows the training of the agent and, simultaneously, the transference of the training to the real world, having as benefits the simulation of the agent behavior in diverse environments, the training without physical wear of the system, and the test of the system avoiding damage to its hardware or even to the living beings related to its operation. The training can be carried out efficiently in a short time period because of the performance of current computing equipment. The results of this development show its flexibility to successfully repeat the learning with the capability to modify the conditions and the activities executed by the agent.

Keywords: Reinforcement learning, virtual environment, agent, mobile robotics.

1. Introducción

El aprendizaje automático o de máquina (*Machine Learning*) describe las técnicas con las cuales un sistema aprende y generaliza su comportamiento con base en datos. Puede clasificarse en tres categorías: supervisado, no supervisado y por refuerzo. En la primera se aprende mediante el uso de un supervisor externo que ya cuenta con el conocimiento de lo que se desea hacer, mientras que en la segunda no hay tal apoyo. Por su parte, en el aprendizaje por refuerzo existen dos elementos principales: un agente y un ambiente.

Partiendo de la premisa de que el agente está inmerso en el ambiente y tiene la capacidad de comprender su entorno, el aprendizaje se da mediante un sistema que cuantifica que tan bien el agente realiza su función; el aprender a caminar, percatarse que el fuego quema, etc., son ejemplos de esto. Así, se pueden modelar fielmente sistemas en los que se requiera realizar un entrenamiento que, después de un número determinado de eventos**,** traiga como resultado un aprendizaje.

Fue a finales de la década de los 80s 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], e incluso 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.

Dentro de los sistemas de aprendizaje por refuerzo que cruzan la barrera del mundo digital al mundo real, se han hecho avances en el ámbito de la robótica, donde se han implementado dos variantes. En la primera, el ambiente del aprendizaje es el mundo real: cada uno de los eventos en los que el agente realiza una decisión que, con base a la política establecida desencadena una acción, se lleva a cabo en la realidad. Se han

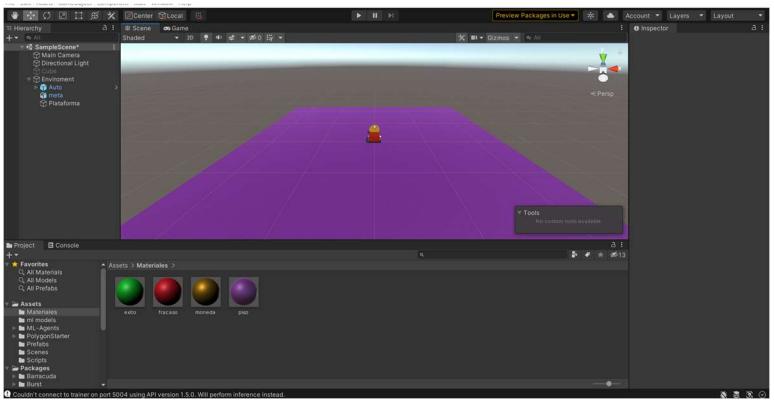

Fig. 1. Interfaz gráfica de Unity 3D.

realizado estudios de esta variante en sistemas robóticos donde el agente aprende a caminar y se observa su progresión al finalizar cada evento de aprendizaje [6], en sistemas orientados a la capacidad de adaptación a cambios de su estructura de locomoción [7], o sistemas de robótica de manipuladores para el lanzamiento de objetos considerando su estructura física y peso [8].

La segunda variante modela computacionalmente el mundo real en el que el agente está inmerso, para simular cada una de las iteraciones del algoritmo de aprendizaje y enviar la información del entrenamiento al agente físico, sin que éste deba realizar las acciones en donde la recompensa no fue maximizada. Ejemplo de esto es el aprendizaje por refuerzo orientado a sistemas, en robots terrestres o aéreos no comunicados entre sí, capaces de esquivar obstáculos en su entorno [9], y su evolución a sistemas capaces de detectar seres vivos y predecir sus movimientos, para una adecuada convivencia entre ellos y el ser humano en ambientes diversos [10, 11].

Existen diversos enfoques para analizar sistemas robóticos colaborativos, con al menos dos individuos entrenados para cumplir una tarea de forma cooperativa. En uno de ellos, los agentes trabajan de manera síncrona para compenetrarse y seguir una estrategia que maximice la recompensa obtenida, cumpliendo la tarea para la que fueron entrenados. En contraste, existe el modelo en el que varios agentes, a partir de estrategias individuales simples y sin alguna sincronía entre sí, pueden unificar su información y comportamiento individual con el fin de formular una política de alto nivel que permita la detección automática de cambios en el entorno, y a su vez una acción de respuesta para adaptarse efectivamente a la situación actual y a posibles cambios futuros.

Esto ha servido para plantear entornos donde un conjunto de robots, separados en equipos, compiten en la realización de una tarea. En [12] se propone que, con base en una serie de estrategias analizadas por cada uno de los individuos de un equipo buscando siempre la maximización de la recompensa, se agrupen con el fin de colaborar efectivamente en contra de sus adversarios.

Un caso particular del aprendizaje por refuerzo es en el que varios agentes interactúan en el mismo ambiente (*Multi-Agent Reinforcement Learning*) siguiendo la misma pauta de buscar maximizar la recompensa en todo momento, independientemente de la política a seguir en el proceso de entrenamiento [13]. En este

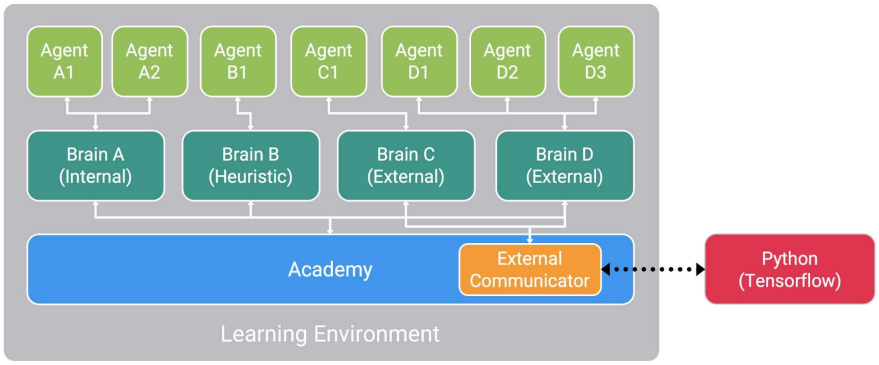

Fig. 2. Flujo de trabajo en un entrenamiento con ML-Agents.

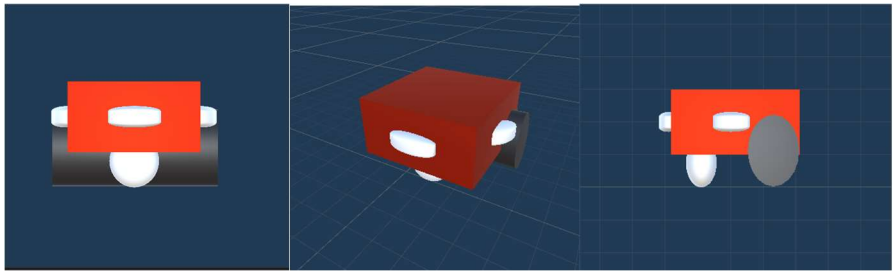

Fig. 3. Vista frontal, de perspectiva y lateral del modelo 3D del agente.

trabajo se presenta una implementación de aprendizaje por refuerzo enfocado a un agente con las características de un vehículo terrestre con ruedas de tracción diferencial, la metodología para la realización de su entrenamiento y a su vez, las acciones que puede realizar para su entrenamiento en un ambiente virtual que permite simular el sistema en su totalidad.

La organización del documento es la siguiente: en la Sección 2 se describen de manera general los parámetros del aprendizaje por refuerzo, su terminología y notación; en la Sección 3 se muestran las herramientas computacionales utilizadas para la implementación del entrenamiento del agente, del ambiente virtual y de la integración de ambos; en la Sección 4 se presenta el caso de estudio a resolver y el desarrollo de la solución; en la Sección 5 se analizan los resultados obtenidos y su validación; finalmente, la Sección 6 aborda las conclusiones y los trabajos a futuro.

2. Parámetros del aprendizaje por refuerzo

Como se mencionó anteriormente, en el aprendizaje por refuerzo se plantea cuál debe ser el comportamiento de un agente dentro de un ambiente, con el objetivo de maximizar una recompensa. 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 lograr optimizarlas. Para

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

2.1. Procesos de decisión de Markov

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 $t = 0,1,2, ...$, 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):

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

2.2. Retorno

Dado que el objetivo de un agente es maximizar las recompensas acumulativas, 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,

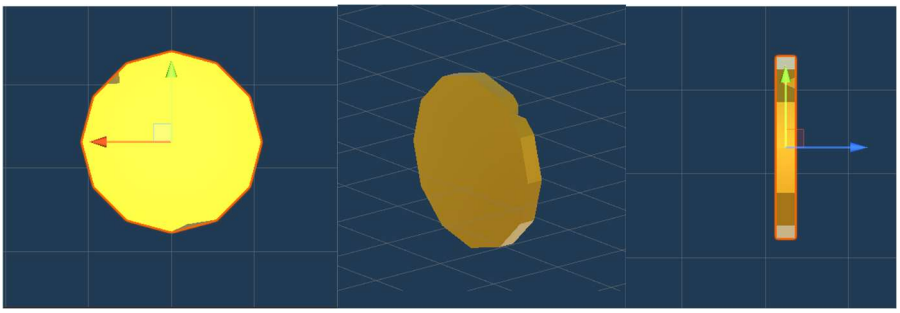

Fig. 4. Vista frontal, de perspectiva y lateral del modelo 3D del agente.

Fig. 5. Ambiente virtual para el entrenamiento.

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):

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \\ = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{3}$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \\ = R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} +) \\ = R_{t+1} + \gamma G_{t+1}\,. \tag{4}$$

3. Herramientas

Existen diversas plataformas para llevar el modelado de un sistema de aprendizaje automático al plano computacional. Una de ellas es TensorFlow, la cual cuenta con herramientas y bibliotecas que permiten crear y desplegar fácilmente aplicaciones basadas en diversos modelos de aprendizaje [14]; lo anterior, mediante la utilización de un solo grafo de flujo de datos que representa todos los cálculos y el estado del algoritmo de aprendizaje automático, lo que incluye también a las operaciones matemáticas individuales, los parámetros del sistema y sus reglas de actualización [15].

Unity 3D (*Unity Technologies*), cuya interfaz se muestra en la Figura 1, 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 [16]. Para el desarrollo de IA existe la interfaz de programación de aplicaciones ML-Agents, para utilizar entornos virtuales realizados tanto en Unity 3D como en ambientes de entrenamiento para sistemas de aprendizaje automático desarrollados previamente [17].

4. Caso de estudio

Un problema común en robótica móvil es el seguimiento o búsqueda de objetos. Esta tarea en primera instancia puede parecer simple, pero su trasfondo tiene una cantidad considerable de parámetros que se deben de tomar en cuenta si se busca modelar su comportamiento con técnicas convencionales, ya que por lo regular estos enfoques no generalizan su comportamiento en todos los casos con base a sus entradas.

4.1. Descripción

En este trabajo se busca implementar un algoritmo basado en aprendizaje por refuerzo mediante PPO (*Proximal Policity Optimization*) para solucionar el caso de estudio, haciendo uso de un ambiente virtual en el que se modelen los componentes requeridos para realizar el aprendizaje de la toma de decisiones del agente. El agente por entrenar es el modelo de un vehículo terrestre de tracción diferencial, el cual tiene como tarea alcanzar una meta en forma del modelo de una moneda.

4.2. Implementación

Para la implementar la solución al caso de estudio seleccionado se hizo uso de Unity 3D con el módulo ML-Agents incorporado, el cual a su vez utiliza en sus capas internas la biblioteca TensorFlow. En Unity 3D se modelan los componentes del ambiente y mediante código en C# se crean las instancias necesarias tales como observaciones, acciones, asignaciones de recompensas y características de los episodios de cada una de las iteraciones del entrenamiento. ML-Agents crea una comunicación entre el ambiente virtual con las configuraciones de hiper-parámetros y TensorFlow, que se encarga de la parte de entrenamiento y aprendizaje.

Así, el ambiente proporciona los datos para el entrenamiento y ML-Agents hace un puente de comunicación, adecuando dichos datos para que TensorFlow los utilice en sus cálculos, cuyo resultado se regresa al ambiente para la adquisición de nuevos datos (ver Figura 2).

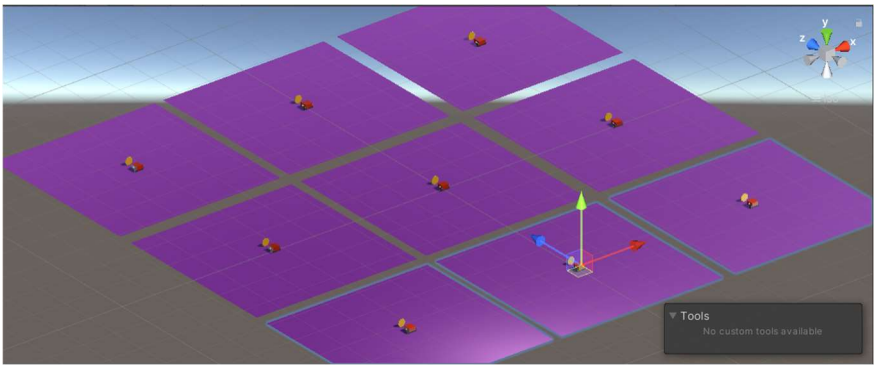

Fig. 6. Configuración para la paralelización del entrenamiento.

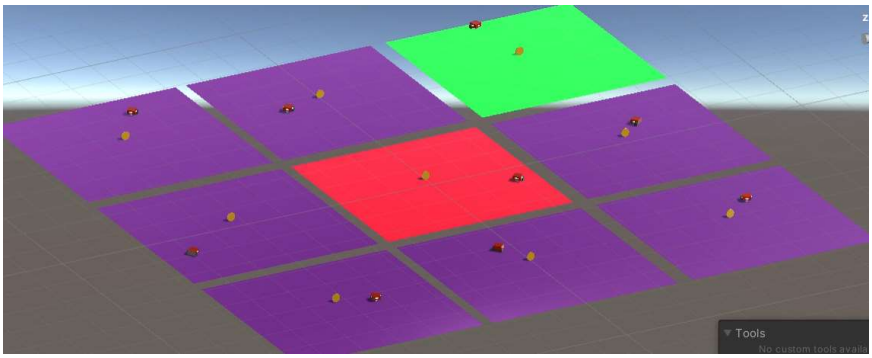

Fig. 7. Entrenamiento en ejecución.

```
2021-04-09 08:45:01 INFO [stats.py:180] agent_config. Step: 12000. Time Elapsed: 27.074 s. Mean Reward: -0.519. Std of Reward: 0.588. Training.
2021-04-09 08:45:45 INFO [stats.py:180] agent_config. Step: 24000. Time Elapsed: 71.402 s. Mean Reward: -0.421. Std of Reward: 0.682. Training.
2021-04-09 08:46:29 INFO [stats.py:180] agent_config. Step: 36000. Time Elapsed: 115.077 s. Mean Reward: -0.059. Std of Reward: 0.900. Training.
2021-04-09 08:47:13 INFO [stats.py:180] agent_config. Step: 48000. Time Elapsed: 158.518 s. Mean Reward: 0.490. Std of Reward: 1.003. Training.
2021-04-09 08:47:57 INFO [stats.py:180] agent_config. Step: 60000. Time Elapsed: 202.715 s. Mean Reward: 0.768. Std of Reward: 0.762. Training.
2021-04-09 08:48:41 INFO [stats.py:180] agent_config. Step: 72000. Time Elapsed: 246.966 s. Mean Reward: 0.926. Std of Reward: 0.516. Training.
2021-04-09 08:49:26 INFO [stats.py:180] agent_config. Step: 84000. Time Elapsed: 291.547 s. Mean Reward: 0.934. Std of Reward: 0.350. Training.
2021-04-09 08:50:11 INFO [stats.py:180] agent_config. Step: 96000. Time Elapsed: 336.699 s. Mean Reward: 0.944. Std of Reward: 0.222. Training.
2021-04-09 08:50:57 INFO [stats.py:180] agent_config. Step: 108000. Time Elapsed: 382.522 s. Mean Reward: 0.936. Std of Reward: 0.174. Training.
2021-04-09 08:51:43 INFO [stats.py:180] agent_config. Step: 120000. Time Elapsed: 428.809 s. Mean Reward: 0.945. Std of Reward: 0.076. Training.
2021-04-09 08:52:30 INFO [stats.py:180] agent_config. Step: 132000. Time Elapsed: 475.757 s. Mean Reward: 0.942. Std of Reward: 0.108. Training.
2021-04-09 08:53:18 INFO [stats.py:180] agent_config. Step: 144000. Time Elapsed: 523.795 s. Mean Reward: 0.939. Std of Reward: 0.101. Training.
2021-04-09 08:54:05 INFO [stats.py:180] agent_config. Step: 156000. Time Elapsed: 571.414 s. Mean Reward: 0.944. Std of Reward: 0.081. Training.
2021-04-09 08:54:53 INFO [stats.py:180] agent_config. Step: 168000. Time Elapsed: 619.096 s. Mean Reward: 0.946. Std of Reward: 0.080. Training.
2021-04-09 08:55:42 INFO [stats.py:180] agent_config. Step: 180000. Time Elapsed: 668.234 s. Mean Reward: 0.949. Std of Reward: 0.055. Training.
2021-04-09 08:56:31 INFO [stats.py:180] agent_config. Step: 192000. Time Elapsed: 716.577 s. Mean Reward: 0.952. Std of Reward: 0.028. Training.
2021-04-09 08:57:19 INFO [stats.py:180] agent_config. Step: 204000. Time Elapsed: 765.226 s. Mean Reward: 0.949. Std of Reward: 0.100. Training.
2021-04-09 08:58:08 INFO [stats.py:180] agent_config. Step: 216000. Time Elapsed: 814.469 s. Mean Reward: 0.901. Std of Reward: 0.319. Training.
2021-04-09 08:58:58 INFO [stats.py:180] agent_config. Step: 228000. Time Elapsed: 863.884 s. Mean Reward: 0.934. Std of Reward: 0.206. Training.
2021-04-09 08:59:47 INFO [stats.py:180] agent_config. Step: 240000. Time Elapsed: 913.316 s. Mean Reward: 0.951. Std of Reward: 0.108. Training.
2021-04-09 09:00:38 INFO [stats.py:180] agent_config. Step: 252000. Time Elapsed: 963.661 s. Mean Reward: 0.954. Std of Reward: 0.064. Training.
2021-04-09 09:01:28 INFO [stats.py:180] agent_config. Step: 264000. Time Elapsed: 1013.534 s. Mean Reward: 0.958. Std of Reward: 0.018. Training.
2021-04-09 09:02:18 INFO [stats.py:180] agent_config. Step: 276000. Time Elapsed: 1063.522 s. Mean Reward: 0.958. Std of Reward: 0.044. Training.
```

Fig. 8. Información del entrenamiento.

4.3. Ambiente virtual

El ambiente se compone de una sola escena, la cual es el entorno del modelo de aprendizaje. Para el diseño de este entorno se utilizaron tres componentes: una plataforma que delimita el espacio del ambiente, un vehículo con ruedas (ver Figura 3) que fungirá como el agente, y una moneda que será el objetivo por alcanzar. El agente

tiene la forma de un robot móvil de tracción diferencial; en adición al diseño se consideró un componente *RigidBody*, el cual permite que se le apliquen fuerzas, y un componente *BoxCollider*, con el cual es posible detectar si colisionó con otro objeto.

Referente al diseño de la meta, ésta tiene forma de moneda (ver Figura 4), y al igual que el modelo del agente, cuenta con los componentes *RigidBody* y *BoxCollider*.

Se utilizaron cinco parámetros de observación y dos acciones como componentes para la implementación del algoritmo PPO. Las cinco observaciones se relacionan con la distancia del agente a la meta y su posición sobre el plano *XZ*, mientras que las acciones corresponden a la transformación de la posición del agente sobre ese mismo plano. A su vez, en cada episodio del entrenamiento el agente tendrá una posición aleatoria sobre el plano, mientras que la meta tendrá una posición fija en el centro de la plataforma. En la Figura 5 se muestra el diseño final del ambiente de entrenamiento.

5. Recompensas

Para este entrenamiento se estableció que el agente obtiene una recompensa igual a 1 cuando logra hacer contacto con la meta; adicionalmente, con esta acción se termina el episodio en ejecución, mientras que en cada estado de tiempo en el que el agente realiza un movimiento sobre el plano, si la distancia a la meta es menor que en el estado anterior, se le premia con una recompensa de 0.01. Con respecto a las recompensas negativas, estas se obtienen si la distancia del agente a la meta es mayor que en el estado anterior, teniendo un valor de -0.01. El episodio se considera como fracaso cuando el agente abandona la plataforma, y finaliza para iniciar el siguiente.

6. Entrenamiento

Con el fin de llevar a cabo el entrenamiento en un tiempo menor, se colocaron en la escena nueve ambientes con las mismas características y parámetros tal como se muestra en la Figura 6; esto se debe a que cada uno de los episodios que son llevados a cabo durante el entrenamiento aporta al resultado final.

Una vez comenzado el entrenamiento se tienen indicadores visuales sobre la plataforma, que informan si en el episodio anterior el agente alcanzó la meta o salió de la plataforma. Un episodio en curso se muestra en color morado y un episodio exitoso se marca con color verde, mientras que un fracaso se indica con color rojo, como se observa en la Figura 7.

Durante el entrenamiento es posible visualizar por consola (ver Figura 8) el tiempo transcurrido y la ganancia promedio obtenida en cada conjunto de episodios realizados. Dado que la recompensa acumulada máxima es aproximadamente 1, se tiene que el entrenamiento se está desarrollando correctamente si la recompensa promedio tiende a esa cifra con el paso de los conjuntos de episodios.

7. Resultados

Para efectos de prueba del sistema propuesto se ejecutaron 400,000 episodios; de las observaciones realizadas se aprecia que con el entrenamiento el agente se comportó

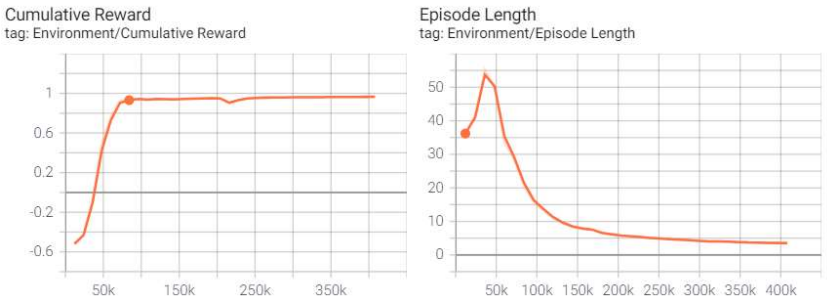

Fig. 9. Gráficas de recompensa acumulada y de longitud de episodio.

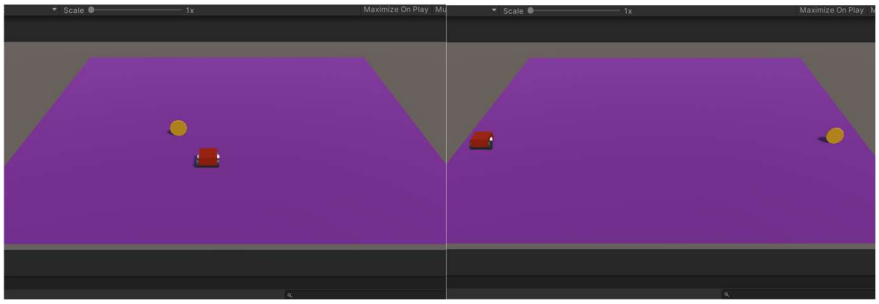

Fig. 10. Posiciones aleatorias del agente y la meta.

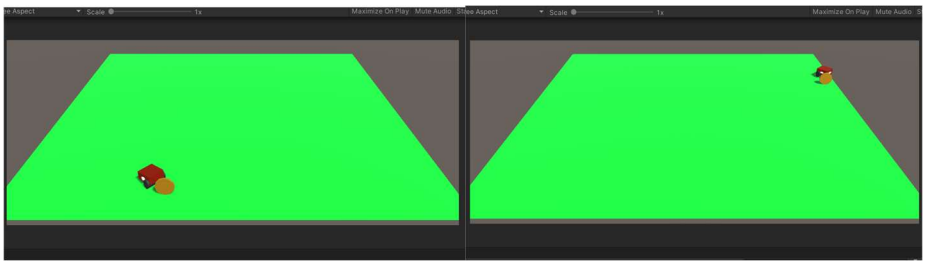

Fig. 11. Validación del entrenamiento con posición de meta aleatoria.

adecuadamente en el ambiente en el que su posición inicial es aleatoria y la posición de la meta a alcanzar es fija. De ahí se infiere que el agente aprendió correctamente ya que, en los casos de prueba, alcanzó el estado en el que obtiene una recompensa máxima.

Mediante la herramienta TensorBoard se obtuvieron las gráficas que se muestran en la Figura 9:

- Ganancia acumulada durante cada uno de los episodios, la cual debe de tender a uno en el caso de que el agente realice bien su labor.
- Longitud de episodio, la cual debe indicar que conforme avanzan los episodios de entrenamiento en los que la recompensa promedio comienza a tender a 1, la longitud debe ir en decremento.

Para la validación del entrenamiento se realizó el cambio de la posición de la moneda, para de determinar si el agente generaliza bien los datos obtenidos de las

observaciones en estados que no se encontraban presentes durante el entrenamiento, por lo que ahora tanto la posición del agente como la de la meta son completamente aleatorias (ver Figura 10).

Como se observa en la Figura 11, en este caso el agente también realiza correctamente la labor para la cual fue entrenado.

8. Conclusiones y trabajo a futuro

Cada modelo de aprendizaje automático cuenta con características que lo hacen adecuado para la solución de grupos de problemas, ya sean de clasificación u optimización, existiendo diversas herramientas para su desarrollo e implementación. En el caso del aprendizaje por refuerzo es necesario modelar un ambiente con una cantidad finita de características, donde el uso de motores de videojuegos como Unity 3D proporciona funcionalidades que flexibilizan y facilitan las tareas de diseño y establecimiento de características, permitiendo la creación de ambientes que pueden ser utilizados para el entrenamiento de agentes que busquen la maximización de una recompensa.

Por otro lado, bibliotecas como Tensor Flow hacen factible la tarea de programar y establecer los hiper-parámetros tanto del ambiente como de los agentes, con el fin de realizar su entrenamiento. Gracias a la unión de estos dos entes es posible realizar de forma sumamente gráfica la implementación de nuevas ideas de modelos de aprendizaje.

Como trabajo a futuro, se propone realizar la implementación del algoritmo de aprendizaje por refuerzo en un agente en el mundo real, en forma de un robot móvil con características similares a las del ambiente virtual, contando con sensores de proximidad y conectividad WiFi para el envío y recepción de datos, con el fin de que el entrenamiento en el mundo virtual se vea reflejado directamente en el mundo real.

De igual manera, se pretende ampliar dicho modelo para incluir características relacionadas con aspectos tales como la cinemática y dinámica del móvil.

Así mismo, se contempla la implementación de sistemas multi-agente que compartan el mismo ambiente virtual, con el fin de generar políticas que permitan el aprendizaje de los agentes para que puedan trabajar en conjunto en la realización de tareas específicas, en un esquema de trabajo colaborativo.

Referencias

1. Kaelbling, L., Littman, M., Moore, A.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research, vol. 4 (1996) doi: 10.1613/jair.301
2. Machado, M., Bellmare, M., Talvitie, E., Veness, J.: Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. Journal of Artificial Itelligence Research, vol. 61 (2017) doi: 10.1613/jair.5699
3. Mnlh, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I.: Playing atari with deep reinforcement learning. Deepmind Technologies (2013) doi: 10.48550/arXiv.1312.5602
4. Li, L., Chu, W., Langford, J., Schapire, R.: A contextual-bandit approach to personalized news article recommendation. Yahoo!, Labs- Dept of Computer Science, Princeton University, pp. 661–670 (2012) doi: 10.1145/1772690.1772758

5. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Juang, A.: Mastering the game of Go without human knowledge. Nature 550 (2017) doi:10.1038/nature24270
6. Haarjona, T., Ha, S., Zhou, A., Tan, J., Tuker, G.: Learning to walk via deep reinforcement Learning. google brain, Berkeley Artificial Intelligence Research (2019) doi: 10.48550/arXiv.1812.11103
7. Chen, Y., Liu, M., Everett, M., How, J.: Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In: Proceedings of IEEE International Conference on Robotics and Automation (2017) doi: 10.1109/ICRA.2017.7989037
8. Ha, S., Kim, J., Yamaane K.: Automated deep reinforcement learning environment for hardware of a modular legged robot. In Proceedings of 15th International Conference on Ubiquitous Robots (UR), pp. 348–354 (2018) doi:10.1109/URAI.2018.8442201
9. Zen, A., Song, S., Lee, J., Rodriguez, A.: Tossingbot learning to throw arbitrary objects with residual physics. IEEE Transactions on Robotics, (2020) doi: 10.1109/TRO.2020.2988642
10. Chen, M., Everett, M., Liu, J.: How socially aware motion planning with deep reinforcement learning. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (2017) doi: 10.1109/IROS.2017.8202312
11. Everett, M., Chen, Y., How, J.: Collision avoidance in pedestrian-rich environments with deep reinforcement learning. In IEEE Access, vol. 9, pp. 10357–10377 (2020) doi:10.1109/ACCESS.2021.3050338
12. Hoang, T., Xiao, Y., Sivaakumar, K., Howl, J. P.: Near-optimal adversarial policy switching for decentralized asynchronous multi-agent systems. In: Proceedings of IEEE International Conference on Robotics and Automation, pp. 6373–6380 (2018) doi:10.1109/ICRA.2018.8460485
13. Busoniu, L., Babuska, R., de Schutter B.: Multi-agent reinforcement learning: An overview. In: Srinivasan D., Jain L.C. Innovations in Multi-Agent Systems and Applications – 1, Studies in Computational Intelligence, Springer, vol. 310, pp. 183–221 (2010) doi:10.1007/978-3-642-14435-6_7
14. Tensor Flow Official website, https://www.tensorflow.org/
15. Abadi, M., Barhman, P., Chen, J., Chen, Z., Davis, A., Dean, J.: tensorflow a system for large-scale machine Learning. In: Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (2016)
16. Juliani, A., Berges, V., Teng, E., Cohen, A.: Unity a general platform for intelligent agents (2018) doi: 10.48550/arXiv.1809.02627
17. Sepúlveda, G., Vega, E., Portilla E.: Machine learning para robots, del entrenamiento virtual a la tarea real. Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI, vol. 7 (2019) doi:10.29057/icbi.v7iEspecial.4785