GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

EVALUACION DEL RENDIMIENTO DE UN AGENTE ENTRENADO MEDIANTE APRENDIZAJE POR REFUERZO PROFUNDO EN LA TRANSFERENCIA DE EXPERIENCIA ENTRE ENTORNOS VIRTUALES Y REALES

Autor: Félix García Fernández

Director: Lucía Güitta López

Co-Director: Álvaro Jesús López López

Madrid, Julio 2024

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Evaluación del rendimiento de un agente entrenado mediante aprendizaje por refuerzo profundo en la transferencia de experiencia entre entornos virtuales y reales

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2023/24 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Félix García Fernández Fecha: 08/ 07/2024

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Lucía Güitta López Fecha: 09/ 07/2024

Álvaro Jesús López López

Firmado digitalmente por Álvaro Jesús López López
Fecha: 2024.07.09 10:34:55 +02'00'

Fdo.: Álvaro Jesús López López Fecha: 09/ 07/2024

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

EVALUACION DEL RENDIMIENTO DE UN AGENTE ENTRENADO MEDIANTE APRENDIZAJE POR REFUERZO PROFUNDO EN LA TRANSFERENCIA DE EXPERIENCIA ENTRE ENTORNOS VIRTUALES Y REALES

Autor: Félix García Fernández

Director: Lucía Güitta López

Co-Director: Álvaro Jesús López López

Madrid, Julio 2024

EVALUACION DEL RENDIMIENTO DE UN AGENTE ENTRENADO MEDIANTE APRENDIZAJE POR REFUERZO PROFUNDO EN LA TRANSFERENCIA DE EXPERIENCIA ENTRE ENTORNOS VIRTUALES Y REALES

Autor: García Fernández, Felix

Director: Güitta López, Lucía

Co-Director: López López, Álvaro Jesús

Entidad colaboradora: ETS de Ingeniería, Universidad Pontificia Comillas

Palabras clave

Aprendizaje por Refuerzo Profundo, Transferencia Sim2Real, Entrenamiento de Brazo Robótico

Resumen

Utilizando el aprendizaje por refuerzo profundo, se entrenó un brazo robótico UR3e para aproximarse a un objetivo a partir de imágenes del entorno. Se evaluó su rendimiento en ambos entornos, destacando la importancia de técnicas como la adición de ruido gaussiano para mejorar la adaptación al entorno real. Los resultados muestran una mayor precisión en el entorno virtual y subrayan la necesidad de mejorar las técnicas de transferencia para el entorno real.

Introducción

El presente Trabajo Fin de Grado titulado "Evaluación del Rendimiento de un Agente Entrenado Mediante Aprendizaje por Refuerzo Profundo en la Transferencia de Experiencia entre Entornos Virtuales y Reales" tiene como objetivo principal analizar la capacidad de un agente robótico, entrenado en un entorno virtual, para transferir sus conocimientos a un entorno real. Este estudio se enmarca en el contexto de la Industria 4.0, donde la integración de tecnologías avanzadas como la inteligencia artificial y la robótica se utiliza para optimizar la eficiencia y la productividad.

El proyecto se enfoca en la utilización del aprendizaje por refuerzo profundo (Deep Reinforcement Learning, DRL) [1] para entrenar un agente en un entorno virtual, utilizando el brazo robótico UR3e de Universal Robots [2]. Este brazo robótico debe ser capaz de, a partir de una imagen del entorno, aproximarse a un cubo rojo. Posteriormente, se evalúa la

capacidad del agente entrenado en el entorno virtual para aplicar sus conocimientos en un entorno físico (sim2real) [3], similar al virtual, analizando las diferencias en rendimiento y comportamiento.

Metodología

Para alcanzar estos objetivos, el trabajo se divide en varias fases:

1. **Virtualización del Robot UR3e**: Creación de un modelo virtual del brazo robótico y su entorno de trabajo utilizando la biblioteca MuJoCo [4].
2. **Entrenamiento del Agente en el Entorno Virtual**: Implementación de algoritmos de DRL para entrenar al agente en la realización de tareas específicas en el entorno simulado.
3. **Transferencia al Entorno Real**: Diseño de un entorno físico que replica las condiciones del entorno virtual y aplicación de técnicas de transferencia de conocimiento para facilitar la adaptación del agente al mundo real.
4. **Evaluación del Rendimiento en el Entorno Real**: Realización de pruebas para comparar el desempeño del agente en ambos entornos y analizar las discrepancias.

A la hora de entrenar los modelos en el entorno virtual, se hicieron cuatro distinciones. En la simulación con la sombra activada o desactivada. Y con ruido gaussiano o sin él. Lo de la sombra es para analizar si la sombra del entorno real influye o no. Y lo del ruido gaussiano es para facilitar al agente la adaptación al entorno real. Ya que como se ve en las siguientes figuras, el histograma de los pixeles del entorno virtual sin ruido es demasiado diferente del de el entorno real. El del entorno virtual sin ruido es demasiado artificial, los son muy uniformes y demasiado parecidos.

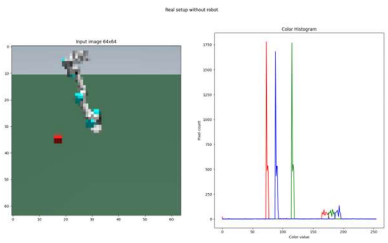


Imagen e histograma del entorno virtual sin ruido

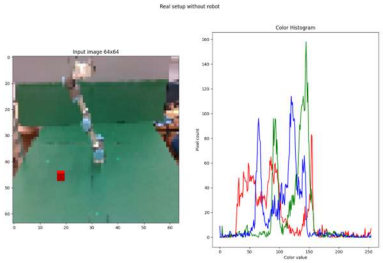


Imagen e histograma del entorno real

En cambio, como se ve en la siguiente figura, al añadir ruido gaussiano, el histograma se asemeja mucho más al de la realidad. Esto en simulación nos dará peores resultados que los de sin ruido, pero a la hora de volcarlo a la realidad, tendrá un mejor desempeño.

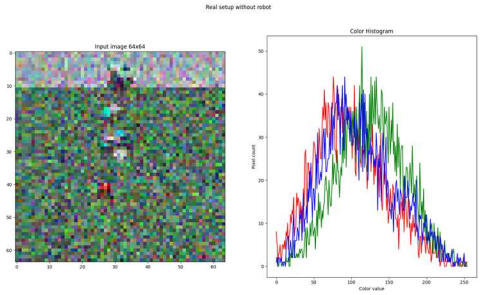
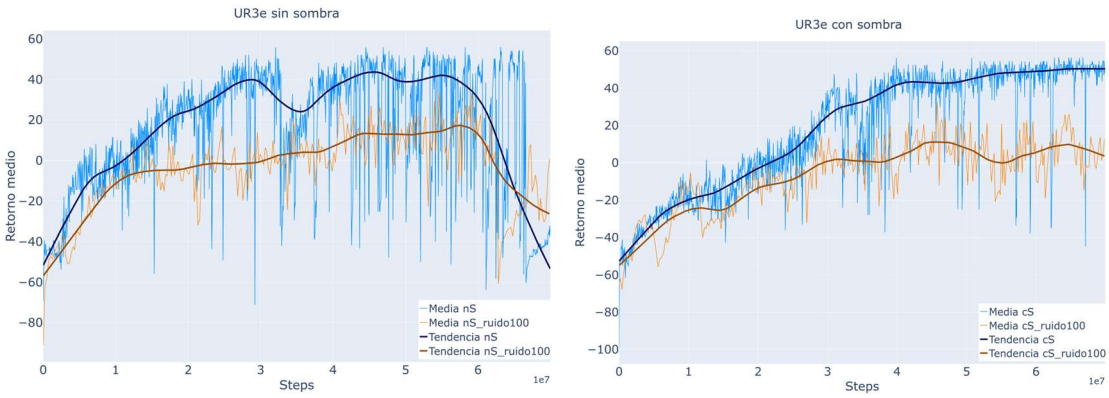


Imagen e histograma del entorno virtual con ruido

Resultados

Los resultados obtenidos muestran un rendimiento significativo del agente en el entorno virtual, con un porcentaje de éxito medio del 95% aproximadamente para los modelos sin ruido y un 85% aproximadamente en los modelos con ruido. En cuanto al retorno medio del agente, para los modelos sin ruido obtenemos un valor de 56, mientras que en el de con ruido obtenemos un valor de 33. Se observa que las diferencias entre con sombra y con sombra con influyen notablemente en los valores obtenidos.



Entrenamiento en simulación sin sombra y con sombra

En el entorno real, al hacer el Zero-Shot [5] con la técnica de Domain Randomization (DR) [6], se evidenció que los modelos entrenados sin ruido no se desempeñaron bien debido a

las diferencias de las imágenes entre los entornos. El histograma de colores de la imagen sintética sin ruido tiene una desviación mucho menor que la imagen real. Sin embargo, los modelos entrenados con ruido gaussiano mostraron una mejora, asemejándose más su histograma al del entorno real, llegando a un porcentaje de éxito en su desempeño del 30%. Se logró una comunicación eficiente entre el agente y los componentes del sistema en el entorno real.

Modelo	Precisión entorno virtual	Precisión entorno real
Sin Sombra con ruido	90%	30%
Con Sombra con ruido	83%	20%

Conclusiones

Las conclusiones destacan la efectividad de la virtualización del robot y su entorno, y la mejora en los resultados al entrenar con ruido gaussiano. No obstante, se identifica la necesidad de mejorar las técnicas de transferencia y adaptación continua para lograr una mayor precisión y eficiencia en el entorno real.

Bibliografía

[1] K. Arulkumaran, M. P. Deisenroth, M. Brundage, y A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process Mag, vol. 34, no. 6*, pp. 26–38, 2017, doi: 10.1109/MSP.2017.2743240.

[2] Universal Robots, "Robot UR3," disponible en: https://www.universal-robots.com/es/productos/robot-ur3/, último acceso: 14/06/2024. (Página web).

[3] E. Salvato, G. Fenu, E. Medvet, y F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access, vol. 9*, pp. 153171-153187, 2021.

[4] E. Todorov, T. Erez, y Y. Tassa, "MuJoCo: A physics engine for model-based control," en IEEE International Conference on Intelligent Robots and Systems, del

7 al 12 de octubre de 2012, Algarve, Portugal., pp. 5026–5033, doi: 10.1109/IROS.2012.6386109.

[5] Z. Chen, Y. Deng, Y. Li, y Q. Gu, "Understanding Transferable Representation Learning and Zero-shot Transfer in CLIP," arXiv preprint arXiv:2310.00927, 2023. (Preprint).

[6] J. Tobin et al., "Domain randomization for transferring deep neural networks from simulation to the real world," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), del 24 al 28 de September de 2017, Vancouver, British Columbia, Canada. pp. 23-30, IEEE.

EVALUATION OF THE PERFORMANCE OF AN AGENT TRAINED THROUGH DEEP REINFORCEMENT LEARNING IN THE TRANSFER OF EXPERIENCE BETWEEN VIRTUAL AND REAL ENVIRONMENTS

Author: García Fernández, Felix

Director: Güitta López, Lucía

Co-Director: López López, Álvaro Jesús

Collaborating Entity: School of Engineering, Universidad Pontificia Comillas

Key Word's

Deep Reinforcement Learning, Sim2Real Transfer, Robotic Arm Training

Abstract

Using deep reinforcement learning, a UR3e robotic arm was trained to approach a target based on images of the environment. The performance was evaluated in both virtual and real environments, highlighting the importance of techniques such as adding Gaussian noise to improve adaptation to the real environment. The results show higher accuracy in the virtual environment and emphasize the need to improve transfer techniques for the real environment.

Introduction

The present Bachelor's Thesis titled "Performance Evaluation of an Agent Trained Using Deep Reinforcement Learning in the Transfer of Experience between Virtual and Real Environments" aims to analyze the ability of a robotic agent, trained in a virtual environment, to transfer its knowledge to a real environment. This study is set in the context of Industry 4.0, where the integration of advanced technologies such as artificial intelligence and robotics is used to optimize efficiency and productivity.

The project focuses on the use of deep reinforcement learning (DRL) [2] to train an agent in a virtual environment, using the UR3e robotic arm from Universal Robots [1]. This robotic arm must be able to approach a red cube based on an image of the environment. Subsequently, the ability of the agent trained in the virtual environment to apply its knowledge in a physical environment (sim2real) [9] similar to the virtual one is evaluated, analyzing the differences in performance and behavior.

Methodology

To achieve these objectives, the work is divided into several phases:

1. **Virtualization of the UR3e Robot**: Creation of a virtual model of the robotic arm and its working environment using the MuJoCo [14] library.
2. **Training the Agent in the Virtual Environment**: Implementation of DRL algorithms to train the agent to perform specific tasks in the simulated environment.
3. **Transfer to the Real Environment**: Design of a physical environment that replicates the conditions of the virtual environment and application of knowledge transfer techniques to facilitate the adaptation of the agent to the real world.
4. **Performance Evaluation in the Real Environment**: Conducting tests to compare the agent's performance in both environments and analyze the discrepancies.

When training the models in the virtual environment, four distinctions were made: simulation with the shadow activated or deactivated, and with or without Gaussian noise. The shadow aspect is to analyze whether the shadow of the real environment influences or not. The Gaussian noise is intended to facilitate the agent's adaptation to the real environment. As seen in the following figures, the histogram of the pixels in the virtual environment without noise is too different from that of the real environment. The histogram in the virtual environment without noise is too artificial, with very uniform and similar colors.

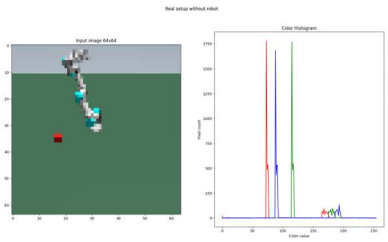


Image and histogram of the virtual environment without noise

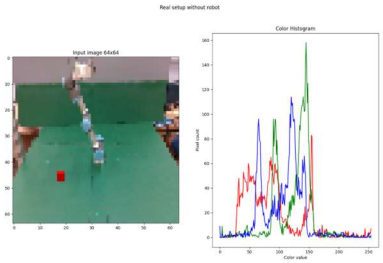


Image and histogram of the real environment

However, as seen in the following figure, adding Gaussian noise makes the histogram much more like the real one. This will give us worse results in the simulation than without noise, but when transferred to reality, it will perform better.

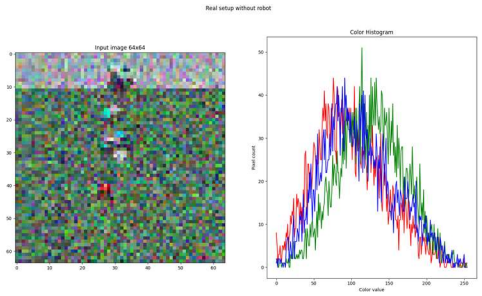
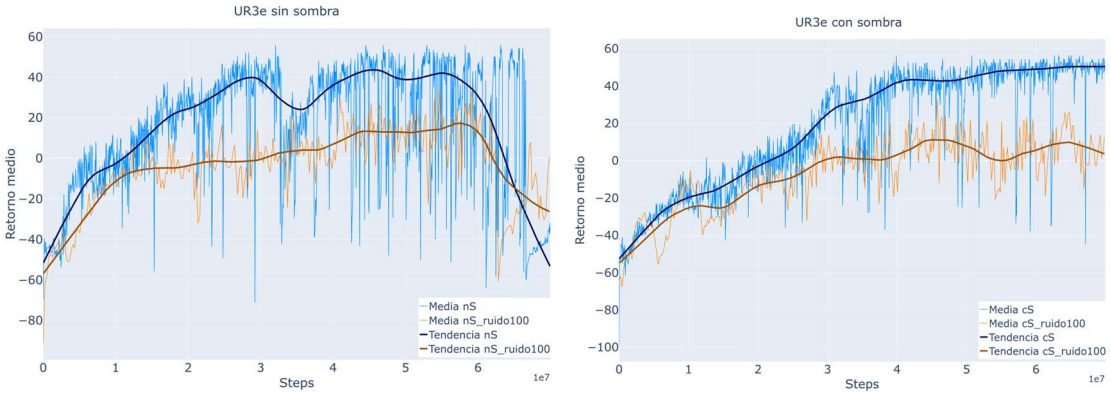


Image and histogram of the virtual environment with noise

Results

The results obtained show a significant performance of the agent in the virtual environment, with an average success rate of approximately 95% for the models without noise and approximately 85% for the models with noise. Regarding the agent's average return, for the models without noise, we obtain a value of 56, while for the ones with noise, we obtain a value of 33. It is observed that the differences between with shadow and without shadow do not notably influence the values obtained.



Training in simulation without shadow and with shadow

In the real environment, when doing the Zero-Shot [19] with the Domain Randomization (DR)[18] technique, it was evidenced that the models trained without noise did not perform well due to the differences in images between the environments. The color histogram of the synthetic image without noise has much less deviation than the real image. However, the models trained with Gaussian noise showed an improvement, with their histogram resembling that of the real environment more closely, achieving a success rate in their performance of 30%. Efficient communication between the agent and the system components in the real environment was achieved.

Model	Virtual Environment Precision	Real Environment Precision
Without Shadow with noise	90%	30%
With Shadow with noise	83%	20%

Conclusions

The conclusions highlight the effectiveness of the virtualization of the robot and its environment, and the improvement in results when training with Gaussian noise. However, the need to improve transfer techniques and continuous adaptation to achieve greater accuracy and efficiency in the real environment is identified.

Bibliography

[1] K. Arulkumaran, M. P. Deisenroth, M. Brundage, y A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process Mag, vol. 34, no. 6*, pp. 26–38, 2017, doi: 10.1109/MSP.2017.2743240.

[2] Universal Robots, "Robot UR3," disponible en: https://www.universal-robots.com/es/productos/robot-ur3/, último acceso: 14/06/2024. (Página web).

[3] E. Salvato, G. Fenu, E. Medvet, y F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access, vol. 9*, pp. 153171-153187, 2021.

[4] E. Todorov, T. Erez, y Y. Tassa, "MuJoCo: A physics engine for model-based control," en IEEE International Conference on Intelligent Robots and Systems, del 7 al 12 de octubre de 2012, Algarve, Portugal., pp. 5026–5033, doi: 10.1109/IROS.2012.6386109.

[5] Z. Chen, Y. Deng, Y. Li, y Q. Gu, "Understanding Transferable Representation Learning and Zero-shot Transfer in CLIP," arXiv preprint arXiv:2310.00927, 2023. (Preprint).

[6] J. Tobin et al., "Domain randomization for transferring deep neural networks from simulation to the real world," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), del 24 al 28 de September de 2017, Vancouver, British Columbia, Canada. pp. 23-30, IEEE.

Índice de la memoria

ÍNDICE DE FIGURAS Y TABLAS

Capítulo 1. INTRODUCCIÓN

La Industria 4.0, conocida también como la Cuarta Revolución Industrial, representa un cambio en la manera en que las empresas operan, integrando en sus procesos tecnologías como la inteligencia artificial, la robótica y el *big data*. Este avance optimiza la eficiencia y la productividad mediante la automatización de máquinas y sistemas inteligentes. En este contexto, la robótica y los sistemas de IA (Inteligencia Artificial) juegan un papel crucial, no solo en la automatización de tareas repetitivas, sino también en la toma de decisiones y en la mejora de los procesos productivos.

En concreto el aprendizaje por refuerzo profundo se ha erigido como un pilar fundamental de esta nueva industria, dándoles a los agentes la posibilidad de aprender y mejorar sus acciones mediante la interacción con su entorno. Esta tecnología facilita a los sistemas de IA a tomar decisiones en entornos cambiantes, mejorando su autonomía y adaptabilidad.

El enfoque de este proyecto radica en la evaluación del rendimiento de un agente entrenado mediante aprendizaje por refuerzo profundo en un entorno virtual y su habilidad para transferir estos conocimientos al entorno real. El objetivo de este agente es la aproximación precisa a una pieza colocada de manera aleatoria en su espacio de trabajo. Para esto, se utilizará el brazo robótico UR3e de Universal Robots [1], tanto en el entorno virtual como en el real.

Este proyecto, se estructura en diversas fases. Comienza con la virtualización del brazo robótico y de su zona de trabajo. En este entorno virtualizado se entrenará el agente, que se someterá a un algoritmo de aprendizaje automático, para poder cumplir el objetivo. Se compararán diferentes estrategias para maximizar la eficiencia y adaptabilidad del agente.

A continuación, se transfieren los conocimientos de este agente en el entorno virtual, al entorno real. Para ello, se diseña un entorno físico lo más parecido posible al entorno virtual donde se ha entrenado. Se analizarán las diferencias entre ambas simulaciones, mirando las

diferencias de precisión y de comportamiento entre el entorno virtual y el físico. Para minimizar estas diferencias y maximizar la eficiencia de transferencia de experiencia, se implementarán estrategias de *sim-to-real.*

Este análisis va más allá de la evaluación del comportamiento del agente en ambos entornos, busca entender le eficiencia y las dificultades en la transferencia de conocimiento entre el entorno virtual y el real. La investigación se posiciona como una contribución valiosa a la robótica autónoma, profundizando en la adaptabilidad y viabilidad del aprendizaje por refuerzo profundo en la industria.

Capítulo 2. ESTADO DE LA CUESTIÓN

2.1. APRENDIZAJE POR REFUERZO PROFUNDO EN ROBÓTICA

El aprendizaje por refuerzo profundo (DRL por sus siglas en inglés Deep Reinforcement Learning) ha desempeñado un papel fundamental en la evolución de la robótica autónoma en los últimos años. Esta técnica ha permitido a los agentes robóticos aprender comportamientos complejos y tomar decisiones inteligentes en entornos dinámicos y variados [2].

Uno de los avances más significativos ha sido la aplicación de redes neuronales profundas en el aprendizaje por refuerzo. Esto ha permitido a los agentes aprender directamente a partir de datos sensoriales brutos, eliminando la necesidad de características previamente diseñadas y simplificando en gran medida el proceso de entrenamiento.

Algoritmos como el Deep Q-Network (DQN),[3], el Proximal Policy Optimization (PPO) [4] y el Asynchronous Advantage Actor-Critic (A3C),[5] se han convertido en referencias en este campo. El DQN ha demostrado su eficacia al aprender a jugar juegos de video de manera sobresaliente, mientras que el PPO y el A3C han destacado en la optimización de políticas para entornos robóticos más complejos.

Además, los enfoques de aprendizaje por refuerzo profundo han evolucionado para abordar desafíos específicos en robótica, como la manipulación de objetos, la navegación autónoma [6], la interacción humano-robot y la ejecución de tareas complejas en entornos no estructurados [7].

La capacidad de estos algoritmos para realizar transferencia de conocimiento entre tareas relacionadas también ha sido un área de interés. La habilidad de un agente para aprovechar el conocimiento aprendido en una tarea para mejorar el rendimiento en otro dominio relacionado ha sido un tema de investigación significativo.

Sin embargo, persisten desafíos importantes en este campo. La eficiencia del aprendizaje y la generalización de los modelos a entornos no vistos son aspectos cruciales que siguen siendo objeto de investigación activa. La necesidad de reducir la cantidad de experiencia requeridos para entrenar agentes DRL y mejorar su capacidad para adaptarse a entornos cambiantes y dinámicos sigue siendo un área de enfoque para los investigadores.

2.2. TRANSFERENCIA ENTRE ENTORNOS VIRTUALES Y REALES

La transferencia de conocimiento y habilidades entre entornos virtuales y entornos físicos en robótica ha sido un tema de investigación crítico y en constante evolución. Este campo busca superar la brecha entre las simulaciones virtuales, donde los agentes pueden adquirir experiencia de manera eficiente, y la implementación práctica en entornos reales, donde se espera que estos apliquen su conocimiento adquirido [8].

Las estrategias de sim-to-real (paso de simulación a realidad) han sido una dirección prominente en esta área. Estas estrategias buscan mitigar las discrepancias y diferencias entre los entornos virtuales y físicos para lograr una transferencia efectiva de habilidades y comportamientos aprendidos [9]. Técnicas como el aprendizaje adversario, el aprendizaje por transferencia, y el uso de modelos generativos han sido exploradas para mejorar la adaptabilidad de los agentes a entornos reales.

El aprendizaje adversario, también conocido como aprendizaje generativo adversarial (GANs), utiliza dos redes neuronales que se entrenan simultáneamente: una generadora, que crea datos sintéticos, y una discriminadora, que evalúa la autenticidad de los datos. Estos agentes aprenden a transformar datos de forma eficiente y robusta. Un ejemplo de uso es el transformar datos del entorno real al simulado y viceversa.

El aprendizaje por transferencia se basa en reutilizar un modelo entrenado en una tarea específica para una tarea relacionada. Al transferir el conocimiento adquirido en un entorno virtual, los agentes se pueden adaptar más rápidamente y con mayor eficacia a entornos reales, reduciendo la necesidad de grandes cantidades de datos y tiempo de entrenamiento.

Esta técnica es especialmente útil cuando los datos del entorno real son limitados o costosos de obtener.

Los modelos generativos, se utilizan para a partir de unos datos, generar datos sintéticos muy similares. Estos modelos permiten a los agentes simular y explorar múltiples escenarios. Al generar una variedad de ejemplos realistas, los agentes pueden entrenarse de manera más eficaz sin una gran cantidad de datos del entorno real.

Los modelos de simulación han mejorado en su precisión y fidelidad, buscando replicar de manera más exacta las condiciones del mundo real. Además, se han investigado métodos para adaptar los modelos virtuales, utilizando datos del entorno real para calibrar y mejorar la precisión de la simulación [10].

La optimización de políticas y estrategias de control robustas que puedan generalizar el conocimiento adquirido en simulaciones a entornos desconocidos en el mundo real ha sido otro foco de investigación [11]. Se han empleado enfoques que buscan maximizar la eficiencia de transferencia y minimizar la brecha entre los entornos, permitiendo que los agentes mantengan un rendimiento consistente independientemente de las diferencias entre las simulaciones y la realidad.

A pesar de los avances, la transferencia efectiva de habilidades entre entornos virtuales y reales sigue siendo un desafío significativo. La diversidad y complejidad de los entornos reales, junto con las limitaciones de las simulaciones, hacen que la adaptación de los agentes sea un área de investigación en curso. La generalización robusta y la capacidad de los robots para aprender y adaptarse a nuevas situaciones sin requerir grandes cantidades de datos de entrenamiento son metas continuas en este campo.

2.3. MODELADO DE ROBOTS Y ENTORNOS VIRTUALES

El modelado de robots y entornos virtuales es fundamental en la robótica moderna, especialmente en el contexto del aprendizaje por refuerzo y la transferencia de experiencia entre entornos virtuales y reales. Este enfoque implica la creación de representaciones

virtuales lo más precisas y fieles posibles, manteniendo un equilibrio entre precisión y coste computacional, permitiendo simular con el comportamiento de los sistemas robóticos en un entorno virtual antes de su implementación en el mundo real [12].

En términos de modelado de robots, se ha avanzado significativamente en la creación de modelos de brazos robóticos como el UR3e de Universal Robots [1]. Estos modelos incluyen representaciones de la cinemática, dinámica y comportamiento de los brazos robóticos, lo que permite simular sus movimientos y acciones. Además, se han desarrollado herramientas y software especializados para facilitar este modelado y la simulación precisa de robots en entornos virtuales.

En lo relativo a modelado virtual de entornos, se han desarrollado técnicas de generación de escenarios y ambientes que reproducen fielmente condiciones del mundo real, considerando aspectos como la iluminación, la textura, la física y las interacciones. Herramientas de simulación cada vez más sofisticadas permiten la representación de entornos complejos y no estructurados, lo que resulta muy útil para el entrenamiento y la validación de algoritmos de aprendizaje automático en entornos realistas pero controlados. Cabe destacar que existe un compromiso entre la precisión y la rigurosidad del entorno virtual y el coste computacional que conlleva su simulación, siendo esta relación directamente proporcional.

Al considerar plataformas de simulación para el desarrollo de agentes en entornos virtuales, es esencial evaluar varias opciones, como *Unity*,[13], *MuJoCo*,[14], y *Gazebo*,[15]. Cada una de estas plataformas ofrece distintas ventajas en términos de capacidad de renderizado, motor de física y accesibilidad del código.

Unity [13] es conocido por su buena capacidad de renderizado y su versatilidad en el diseño de entornos visualmente detallados. Sin embargo, su motor de física y que no sea completamente de código abierto pueden ser limitantes dependiendo del tipo de aplicación y las necesidades específicas del proyecto.

MuJoCo [14] por otro lado, se destaca por su potente motor de física, lo que lo hace ideal para simular dinámicas complejas de robots y otros sistemas. Aunque su renderizado no es

tan avanzado como el de Unity, ofrece un equilibrio favorable entre realismo físico y costo computacional, y es muy apreciado en la investigación robótica.

Gazebo [15] siendo una plataforma de simulación de código abierto, es muy accesible y ampliamente utilizada en la comunidad de robótica. Ofrece un buen balance entre realismo y flexibilidad, pero requiere más configuración y optimización para obtener niveles altos de realismo.

Teniendo en cuenta estos factores, se decide utilizar *MuJoCo* [14] para este proyecto. Su superioridad en la simulación física y el equilibrio adecuado entre realismo y eficiencia computacional lo hacen la opción más adecuada para entrenar y evaluar agentes en un entorno virtual antes de transferirlos al mundo real.

2.4. EVALUACIÓN DEL RENDIMIENTO Y TRANSFERENCIA A ENTORNOS FÍSICOS

La evaluación del rendimiento de agentes entrenados con aprendizaje por refuerzo en entornos virtuales y su posterior transferencia a entornos físicos es un área de la robótica y la inteligencia artificial [16]. Este enfoque busca entender cómo los agentes, una vez entrenados en simulaciones virtuales, pueden desempeñarse en entornos del mundo real, lo que es fundamental para aplicaciones prácticas de robótica.

Sin embargo, la transferencia de habilidades y conocimientos adquiridos en entornos virtuales a entornos físicos plantea desafíos significativos. Las diferencias entre la simulación y la realidad, como las variaciones sensoriales, la física del mundo real y la presencia de ruido, pueden afectar el rendimiento de los agentes entrenados en simulaciones virtuales.

Se han propuesto y desarrollado estrategias para abordar este problema, incluyendo técnicas de sim-to-real [17]. Estas estrategias buscan reducir la brecha entre los entornos virtuales y físicos, permitiendo que los agentes mantengan un rendimiento óptimo en el mundo real a pesar de las discrepancias en la simulación. Algunas técnicas utilizadas son: *Domain Randomization* [18], *zero-shot transfer* [19], *Domain Adaptation* [20].

Domain Randomization [18] mejora la robustez y generalización del agente, permitiéndole adaptarse mejor a situaciones inesperadas en el mundo real. Al exponer al modelo a variaciones durante el entrenamiento, como cambios en la iluminación, texturas y colores, se reduce la dependencia de características específicas del entorno simulado. Esto facilita la transferencia del aprendizaje del entorno virtual al real.

Zero-shot transfer [19] es un modo de transferencia del conocimiento que permite implementar lo aprendido en una tarea o dominio para resolver problemas en un dominio completamente nuevo, sin haber sido entrenado específicamente en ese nuevo dominio. Entrenando el modelo en un entorno virtual altamente realista, permite desplegar el modelo directamente en el entorno real y obtener buenos resultados.

Domain Adaptation [20] es una técnica en el aprendizaje automático que se centra en mejorar el rendimiento de un modelo cuando se enfrenta a un dominio de datos diferente del que fue entrenado originalmente. Esta técnica recorta las diferencias entre el dominio de origen (donde se entrenaron los datos) y el dominio objetivo (donde se desplegará el modelo) mediante la adaptación del modelo para que funcione bien en el contexto objetivo.

Capítulo 3. MOTIVACIÓN Y OBJETIVOS

3.1. MOTIVACIÓN

La motivación de este TFG está en tratar uno de los retos más importantes de la robótica y de la inteligencia artificial: la transferencia de conocimientos de forma efectiva de entornos virtuales a entornos físicos. El TFG busca explorar como pueden aplicar, los agentes entrenados en entornos virtuales, sus conocimientos en entornos reales.

La necesidad de esta investigación surge por el aumento de elementos robóticos en la industria, donde es muy importante la capacidad de adaptación y el aprendizaje de forma autónoma. Según va avanzando la tecnología, la búsqueda de robots más autónomos y adaptables en entornos cambiantes se vuelve una tarea fundamental.

La capacidad de transferir los conocimientos aprendidos por un agente en un entorno virtual a uno real facilita y acelera la implementación de sistemas robóticos en la industria. Sin embargo, es todavía objeto de estudio debido a su complejidad en algunos casos.

3.2. OBJETIVOS DEL PROYECTO

- **Evaluar el rendimiento del agente en entornos virtuales**

El objetivo principal es analizar y medir la capacidad del agente entrenado en un entorno virtual de acercarse a la pieza objetivo.

- **Facilitar la Transferencia de Experiencia al Entorno Real**

Buscar estrategias y técnicas que faciliten la transferencia del aprendizaje obtenido por el agente, del entorno virtual al mundo real. Esto requiere diseñar el entorno real donde va a trabajar el agente, lo más parecido al entorno virtual.

- **Optimizar la Adaptabilidad del Agente**

Mejorar la adaptabilidad del agente ante diferencias entre los entornos virtuales y reales. Esto implica explorar estrategias o métodos como sim-to-real, para minimizar la brecha entre simulaciones y ensayos reales.

- **Comprender los Desafíos de la Transferencia de Experiencia**

Investigar los desafíos que surgen en la transferencia de experiencia entre entornos virtuales y reales en aplicaciones robóticas. Esto implica identificar las limitaciones y dificultades en el proceso de transferencia.

Capítulo 4. RECURSOS Y METODOLOGÍA

4.1. RECURSOS A EMPLEAR

4.1.1 RECURSOS TECNOLÓGICOS:

- **Brazo Robótico UR3e**

Acceso al brazo robótico UR3e de Universal Robots para la implementación y pruebas en entornos reales.

- **Plataforma de Simulación Virtual**

Emplear la biblioteca *MuJoCo* para la virtualización precisa del brazo robótico UR3e y la creación de entornos virtuales realistas.

- **Algoritmos de Aprendizaje por Refuerzo Profundo**

Uso de *framework* de aprendizaje automático *PyTorch* para implementar algoritmos de aprendizaje por refuerzo.

- **Hardware Informático**

Potencia de cómputo suficiente para el entrenamiento de modelos de aprendizaje profundo y ejecución de simulaciones complejas. Se empleará un PC con la gráfica de NVIDIA RTX2080Ti.

Cámara de Intel D435 para capturar datos del entorno real para retroalimentar al agente.

4.1.2 INFRAESTRUCTURA Y ESPACIO:

Espacio físico adecuado para la implementación y pruebas del brazo robótico en entornos reales. Se usa el laboratorio de IA de ICAI.

4.2. METODOLOGÍA

4.2.1 VIRTUALIZACIÓN DEL ROBOT UR3E

- Desarrollo del modelo del brazo robótico UR3e para su representación en un entorno virtual, en este caso se usará *Blender*.
- Diseño del escenario virtual que simule las condiciones y tareas a realizar.

4.2.2 ENTRENAMIENTO DEL AGENTE EN EL ENTORNO VIRTUAL

- Implementación de algoritmos de aprendizaje por refuerzo profundo mediante Python para entrenar al agente en el entorno virtual diseñado.
- Evaluación continua del desempeño del agente durante el entrenamiento.

4.2.3 TRANSFERENCIA AL ENTORNO REAL

- Diseño del escenario y condiciones del entorno real para que se asemeje en la medida de lo posible al entorno virtual.
- Aplicación de técnicas de transferencia de conocimiento y adaptación para facilitar la transición del agente al mundo real.

4.2.4 EVALUACIÓN DEL RENDIMIENTO EN EL ENTORNO REAL

- Ejecución de pruebas exhaustivas para evaluar el desempeño del agente en el entorno físico.
- Análisis comparativo entre el rendimiento en los entornos virtual y real para identificar discrepancias y mejoras necesarias.

Capítulo 5. PLAN DE TRABAJO Y CRONOGRAMA

	Septiembre	Octubre	Noviembre	Diciembre	Enero	Febrero	Marzo	Abril	Mayo
Preparación y diseño									
Entrenamiento y evaluación entorno virtual									
Transferencia y preparación entorno real									
Evaluación en entorno real y análisis									
Documentación y Conclusiones									

Tabla 1 Diagrama de Gantt.

5.1.1 PREPARACIÓN Y DISEÑO

Virtualización del Robot UR3e y diseño del entorno virtual.

Elección y aplicación de algoritmos de aprendizaje por refuerzo.

5.1.2 ENTRENAMIENTO Y EVALUACIÓN EN ENTORNO VIRTUAL

Entrenamiento del agente en el entorno virtual.

Evaluación continua y ajustes en el proceso de entrenamiento.

5.1.3 TRANSFERENCIA Y PREPARACIÓN DEL ENTORNO REAL

Diseño y preparación del entorno real basado en la simulación virtual.

Implementación de técnicas de transferencia de experiencia.

5.1.4 EVALUACIÓN EN ENTORNO REAL Y ANÁLISIS

Ejecución de pruebas en el entorno real y recopilación de datos.

Análisis comparativo entre los resultados obtenidos en entornos virtual y real.

5.1.5 DOCUMENTACIÓN Y CONCLUSIONES

Redacción del informe final.

Conclusiones, recomendaciones y posibles mejoras.

Capítulo 6. VIRTUALIZACIÓN DEL UR3E

Para la virtualización del brazo robótico UR3e, utilizamos los archivos .stl proporcionados por Universal Robots, que contienen la información sobre la forma de las distintas partes del robot. Es fundamental interconectar estas partes no solo para construir la estructura del robot, sino también para simular el movimiento de sus componentes y evitar que actúen como elementos fijos en nuestras simulaciones. Para conectar las partes y generar el entorno del modelo, empleamos un archivo .xml. Este tipo de archivo, utilizado por MuJoCo para estructurar entornos, permite seleccionar los grados de libertad de los objetos .stl, añadir nuevos objetos como planos o formas geométricas, y personalizar atributos de los objetos, como la masa o el color.

6.1. ESTRUCTURA BÁSICA DE UN ARCHIVO .XML PARA MUJOCO

MuJoCo ofrece una forma estructurada y precisa para generar modelos complejos mediante una jerarquía anidada de elementos dentro de su archivo .xml. Este archivo relaciona los diferentes cuerpos rígidos declarados en los .stl y declara las reglas del movimiento relativo entre ellos mediante juntas.

Los elementos clave de estos archivos son los siguientes. <compiler> establece las configuraciones para compilar el modelo, incluyendo masa total, unidad angular utilizada y el directorio de mallas (archivos .stl). <option> declara las opciones de simulación, siendo estas, el paso de tiempo y la gravedad.

```
<compiler inertiafromgeom='true' angle="radian" settotalmass="11.1"
meshdir="stl/" />
<option timestep="0.002" gravity="0 0 -9.81"/>
```

Código 1. Configuración XML Mujoco.

<size> especifica los tamaños máximos de elementos como juntas y contactos. <asset> define los activos, incluyendo mallas y materiales utilizados en el modelo. <worldbody>

contiene los elementos físicos de la simulación, tales como geometrías y cuerpos. <body> representa un cuerpo físico con una posición y puede incluir componentes anidados como juntas y geometrías. <geom> son las geometrías asociadas con los cuerpos para la detección de colisiones y visualización. <joint> define las juntas entre cuerpos, especificando el tipo y el eje de rotación o traslación. Finalmente, <actuator> define los actuadores para controlar las juntas, como motores y controladores de posición.

Estos archivos están estructurados en cuatro partes: configuración, declaración de assets, worldbody y actuadores. En la configuración se utilizan las etiquetas <compiler>, <option> y <size> para definir la configuración básica de la simulación en Mujoco. La declaración de assets asigna nombres a las diferentes geometrías en formato .stl para poder relacionarlas posteriormente además de declarar las texturas o materiales utilizados por las distintas geometrías del modelo. La parte más importante, el worldbody, es una etiqueta que envuelve todos los elementos que tendrá nuestra simulación. Cada elemento está formado por al menos una geometría, o varias geometrías interconectadas mediante juntas. A estos elementos se les puede dar atributos como peso, color o inercia. Para mover estas juntas definidas en el worldbody están los actuators, que nos permiten en simulación interactuar con el entorno y mover las diferentes partes declaradas anteriormente. Se ha adjuntado en el Anexo A el código xml completo.

6.2. DESCRIPCIÓN DEL ENTORNO VIRTUAL

El entorno virtual que se utiliza para la simulación está formado por un plano de tierra de color verde sobre el que se sustenta el UR3e. En perpendicular a este plano de tierra y como pared hay otro plano, quedando definido así el entorno de trabajo. Además del UR3e sobre el entorno de trabajo se encuentra el bloque objetivo que va tomando posiciones aleatorias al inicio de cada episodio, estas posiciones están dentro de un rango definido dentro del rango de trabajo del brazo robótico. Este bloque objetivo es un cubo de color rojo al cual el brazo robótico debe aproximarse desde una posición inicial aleatoria.

El entorno virtual que se utiliza para la simulación está formado por el UR3e, que se sitúa sobre un plano de tierra de color verde. Este plano actúa como la base del entorno de trabajo, y está acompañado por un plano perpendicular que funciona como pared, creando así un espacio tridimensional para las operaciones del UR3e. En este entorno de trabajo se encuentra también el cubo de color rojo.

En MuJoCo, existen varias opciones de simulación que se pueden configurar a través de *mjOption* y *mjModel*. Las opciones (*mjOption*) permiten ajustar los parámetros de la simulación, como la precisión de la integración numérica, la gravedad y los límites de velocidad, para asegurar que la simulación sea estable y suficientemente precisa. El compilador (*mjModel*) define las propiedades del modelo físico del robot y el entorno, incluyendo la geometría, las propiedades de contacto, los rangos de movimiento de las articulaciones, las texturas y demás configuraciones.

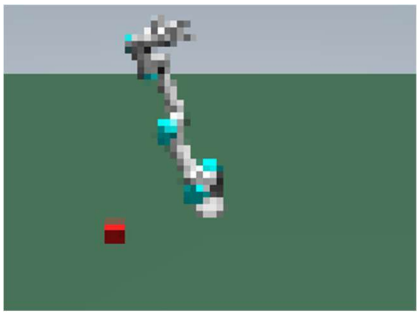


Figura 1. Ejemplo de observación del entorno virtual con el brazo UR3e y el cubo rojo a alcanzar.

Capítulo 7. DISEÑO Y RESULTADOS DEL ENTORNO VIRTUAL

7.1. MDP CONCEPTOS Y DEFINICIONES

El aprendizaje por refuerzo es una rama del aprendizaje automático donde un agente aprende a tomar decisiones a través de la retroalimentación que recibe del entorno en el cual opera. A diferencia de otros tipos de aprendizaje automático, en el aprendizaje por refuerzo, el agente no está supervisado directamente, sino que debe descubrir qué acciones producen la mayor recompensa a través de prueba y error.

En este caso, se utiliza aprendizaje por refuerzo profundo (DRL) debido a que el estado del sistema está caracterizado por una imagen de 64x64 píxeles con los 3 canales de color (RGB). Las redes neuronales profundas son capaces de procesar y extraer características relevantes de estas imágenes complejas, lo que permite al agente interpretar el estado del entorno. La utilización de DRL facilita la gestión de la alta dimensionalidad y complejidad de los datos visuales, mejorando el aprendizaje.

Un problema de aprendizaje por refuerzo puede ser modelado como un problema de decisión Markoviano (MDP por sus siglas en inglés). Sus elementos fundamentales incluyen el estado del entorno, representado por s, que es la percepción que el agente tiene del entorno en un momento dado. La acción, representada por a, que es la decisión que el agente toma en el estado s. El espacio de posibles acciones puede ser discreto o continuo. La recompensa, representada por r, que es el *feedback* que el agente recibe después de tomar una acción por parte del entorno. Esta recompensa puede ser positiva o negativa y ayuda al agente a aprender cuál es la mejor acción que tiene que tomar.

En el problema que aquí se resuelve, el espacio de acciones posibles se ha modelado de forma discreta y se actúa para cada articulación del robot. Está expresado en función del

valor MPI (Máximo Incremento de Posición), que es el ángulo máximo al que puede llegar una articulación menos el ángulo mínimo. Por seguridad, y para que no sea demasiado violento, se aplica un factor del 30% al rango máximo de movimiento. Además, las diferentes acciones se modelan como una escala logarítmica asegurando tener un buen compromiso entre largas distancias y aproximación más precisa. El espacio de acciones está formado por siete acciones discretas que se muestran en la Tabla 1. Posteriormente estos incrementos o decrementos de los ángulos se aplican a los ángulos actuales para obtener los nuevos ángulos objetivo que se envían al robot.

La recompensa con la que se realimenta el modelo es discontinua y se definen dos rangos. El caso que el *gripper* se encuentre a más de 5cm del objetivo o por el contrario que este más cerca. Si está a más de 5cm la recompensa se ha diseñado para darle un feedback negativo al agente, en la forma que se muestra en la Tabla 1. Donde la distancia (*dist*) es la medida relativa entre el *gripper* y el objetivo. En el otro caso la recompensa es directamente 70 y acaba el episodio. Cabe destacar que en la evaluación la distancia de recompensa se incrementa hasta 10cm, siendo así el agente entrenado en circunstancias más complejas que en las que luego se le evalúa.

Cada episodio de entrenamiento consta de 50 pasos, donde un paso se define como cada vez que el agente toma una acción. Sin embargo, el episodio puede terminar antes de llegar a los 50 pasos si el agente logra acercarse a menos de 5cm del objetivo. En este caso, el episodio se considera completo, independientemente del número de pasos transcurridos.

Conjunto de acciones	Distribución de recompensa	Distancia exitosa
0 ± MPI ± MPI/10 ±MPI/100	70 Si se llega al objetivo -(2 x dist) ² Si no se llega	5 cm en entrenamiento 10 cm en evaluación

Tabla 2 Parámetros del MDP.

7.2. ALGORITMO DE APRENDIZAJE

El *Asynchronous Advantage Actor Critic* (A3C) es un algoritmo de aprendizaje por refuerzo profundo que combina las ventajas de los enfoques basados en políticas y en estimación de la función de valor del estado. Es conocido por su eficiencia y capacidad de paralelización.

El nombre A3C proviene de los componentes fundamentales del algoritmo: actor, *critic* y asíncrono. El componente actor es responsable de aprender y ejecutar la política del agente, es decir, las decisiones sobre qué acciones tomar en cada estado del entorno. Por otro lado, el componente *critic* estima la función de valor de estado, que representa el valor esperado de estar en un estado específico. La parte asíncrona del algoritmo se refiere a la utilización de múltiples agentes (procesos) que interactúan con sus propios entornos, instancias independientes, y actualizan una red neuronal global compartida.

El A3C funciona ejecutando múltiples copias del agente en paralelo, cada una interactuando con una copia independiente del entorno. Cada agente aprende su propia política y periódicamente actualiza una red global compartida. La principal ventaja es que la ejecución en paralelo permite un aprendizaje más rápido y con menos varianza.

La arquitectura del modelo está estructurada en varias capas tal y como se representa en la Figura 2. La entrada al modelo es el estado del sistema, la imagen 64x64x3, y la salida las 6 políticas, una para cada articulación, y el valor del estado.

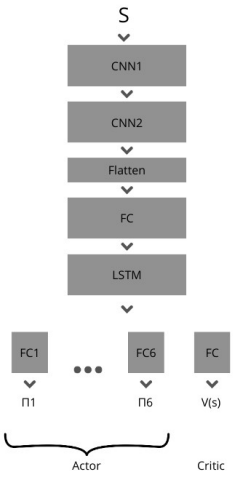


Figura 2. Arquitectura del A3C implementado.

La arquitectura comienza con dos capas convolucionales, CNN1 y CNN2, que se encargan de procesar las imágenes de entrada, siendo estas la parte de "visión" al agente. Después, el tensor resultante se aplana (Flatten) y se pasa a través de una capa *fully connected* (FC). Esta capa se encarga de la reducción de la dimensionalidad y la extracción de características más abstractas de las activaciones procesados por las capas convolucionales.

Estas activaciones resultantes son la entrada de una capa *Long Short-Term Memory* (LSTM). Esta capa tiene la capacidad de manejar secuencias de datos y mantener estados ocultos a lo largo del tiempo. Esto permite al agente aprender dependencias temporales en sus decisiones, capturando cómo las acciones actuales afectan las futuras.

Las siguientes capas *fully connected* (FC1 a FC6) se encargan de generar las políticas de acciones del agente. Cada una de estas capas produce distribuciones de probabilidad sobre las posibles acciones que el agente puede tomar en función de su estado actual. La función *softmax* se aplica a las salidas de estas capas para asegurar que las probabilidades de las acciones sumen uno, proporcionando así una decisión clara y probabilística para cada acción posible.

Finalmente, la última capa *fully connected* (FC) calcula el valor del estado. Este valor es una estimación de la recompensa esperada a largo plazo desde el estado actual del agente. El

valor crítico es esencial en el método A3C, ya que guía al agente hacia acciones que maximicen las recompensas futuras esperadas, ayudando así en la toma de decisiones.

Como vista en conjunto, al agente le entra el estado, que es una imagen 64x64 en color del estado del entorno, y como salida tiene 6 vectores, uno por cada articulación, de 7 elementos, que contiene la probabilidad de que la acción de las siete sea la mejor, más un último valor que es el valor del estado.

7.3. EXPERIMENTOS Y RESULTADOS

Se han entrenado cuatro modelos con diferentes características. Dos de estos modelos incluyen la sombra del robot en la simulación, mientras que los otros dos no la incluyen. Esto se debe a que el siguiente paso es transferirlo a la realidad (sim-to-real), y la iluminación puede influir significativamente en el rendimiento de los modelos.

Para cada uno de estos dos tipos de modelos, se introduce una nueva diferencia: uno se entrena utilizando las imágenes de la simulación sin ningún tipo de procesamiento, mientras que al otro se le añade un ruido gaussiano a las imágenes para reducir la artificialidad inherente a la simulación. Este ruido aumenta la aleatoriedad de los valores de los píxeles, ya que en la simulación los colores de los objetos tienden a ser uniformes, algo que no ocurre en la realidad. De esta manera, se busca que los modelos sean más robustos y capaces de manejar la variabilidad de los entornos reales.

El primer modelo (*Baseline*) es con las sombras deshabilitadas y sin aplicar el ruido gaussiano. Como se puede ver en la Figura 3, la distribución de los tres colores principales tiene muy poca desviación.

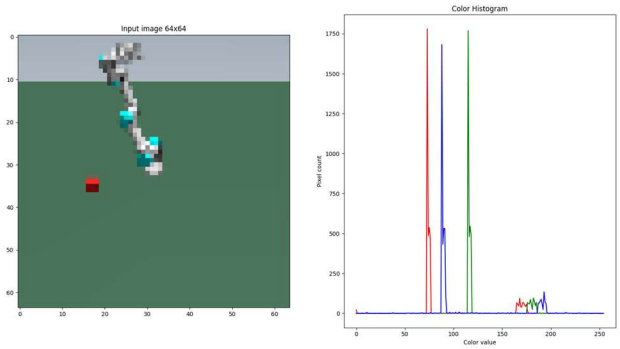


Figura 3. Imagen e histograma del entorno sin sombra ni ruido.

Al segundo modelo se le aplica ruido gaussiano en la imagen y se le mantiene con la sombra deshabilitada. Como se puede ver en la Figura 4, la distribución de los tres colores principales tiene mucha más desviación que la imagen anterior.

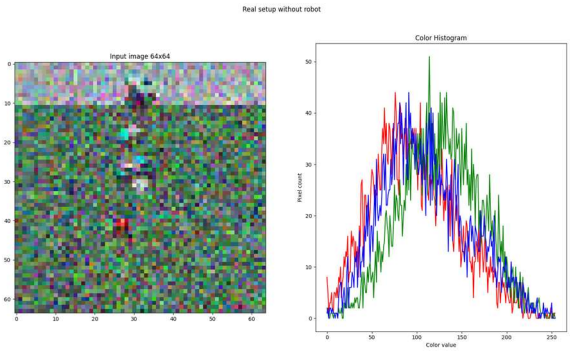


Figura 4. Imagen e histograma del entorno sin sombras con ruido gaussiano.

En la Figura 5 se recogen los resultados de los entrenamientos para ambos modelos. En el caso del *Baseline*, el mejor modelo, es decir, el que mayor recompensa tiene, está en el *step* 53M. Obtiene una recompensa media de aproximadamente 56 y el porcentaje de éxito medio alcanzado en su evaluación post-entrenamiento es del 98%.

En el caso del agente entrenado con las observaciones ruidosas, el mejor modelo se da en el *step* 55M, obtiene una recompensa media de aproximadamente 34 y el porcentaje de éxito medio en la evaluación post-entrenamiento es del 90%.

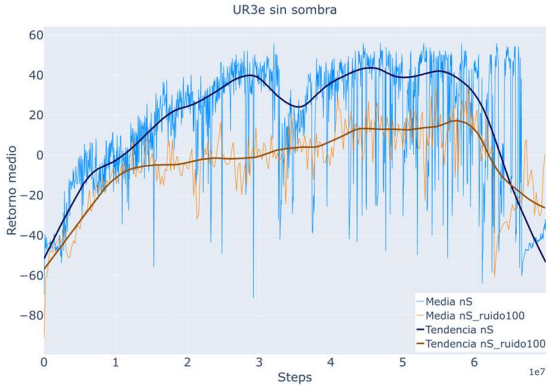


Figura 5. Entrenamiento en simulación de los modelos sin sombra. En azul el agente sin ruido y en naranja el agente entrenado con ruido.

Cabe destacar las diferencias entre los porcentajes de éxito y la rapidez de cada proceso de aprendizaje que se observa en la Figura 5 debido al incremento en la complejidad en el entorno al que se le añade el ruido. Como se puede observar en las figuras que representan las observaciones de ambos escenarios, es más difícil ser capaz de inferir la posición del robot y del cubo rojo si aparece ruido en los píxeles de la imagen que si no hay nada y está limpia.

El tercer modelo entrenado es ya con las sombras habilitadas, pero sin aplicar el ruido gaussiano. Como se puede ver en la Figura 6, la distribución de los tres colores principales es muy parecida al primer modelo ya que ambos no tienen ruido añadido y la sombra apenas afecta a los píxeles de la imagen.

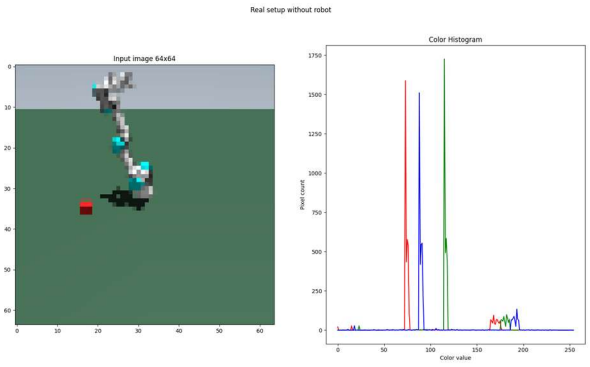


Figura 6. Imagen e histograma del entorno con sombras sin ruido.

Finalmente, el cuarto modelo es con las sombras y aplicando el ruido gaussiano. Como se puede ver en la Figura 7, la distribución de los tres colores principales es muy parecida al segundo modelo ya que ambos tienen el mismo ruido añadido.

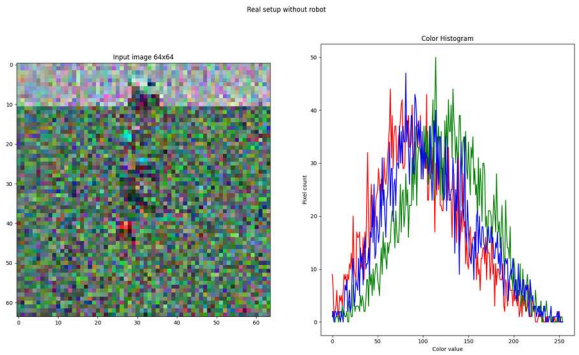


Figura 7. Imagen e histograma del entorno con sombras y ruido gaussiano.

En la Figura 8 se presenta el resultado de los entrenamientos para estos dos modelos con sombra. En el caso del entorno sin ruido, el mejor modelo que es en el *step* 40M, obtiene una recompensa media de aproximadamente 56, siendo su porcentaje de éxito del 95%. Por otro lado, cuando se aplica ruido, el mejor agente, que es en el *step* 46M, obtiene una recompensa media de aproximadamente 33 y un porcentaje de acierto del 83%. Al igual que en el caso anterior, se observa que el modelo con ruido es más costoso en términos de aprendizaje y recompensa que se alcanza.

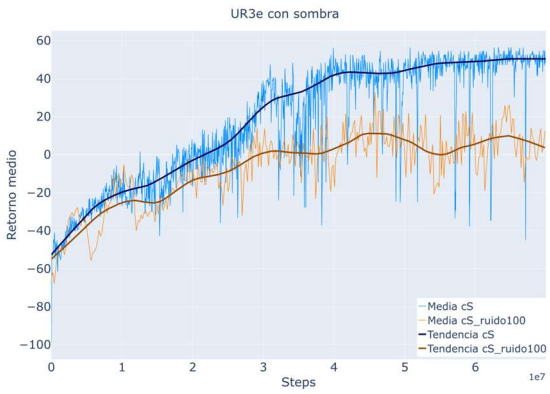


Figura 8. Entrenamiento en simulación de los modelos con sombra. En azul el agente sin ruido y en naranja el agente entrenado con ruido.

Capítulo 8. DISEÑO DEL ENTORNO REAL

El sistema con el que se trabaja está formado por el UR3e, el agente, el objetivo, la cámara que captura la imagen del entorno real, el PC, y la estructura que se sustenta el entorno. La comunicación entre los distintos elementos y el diseño del entorno se detalla a continuación.

8.1. COMUNICACIÓN PC-UR3E

En esta sección, se detallará el proceso de comunicación entre el PC y el robot UR3e. Se describirá la utilización del lenguaje propio de Universal Robots en el controlador del UR3e y Python en el PC, así como la conexión a través de una red compartida para la transmisión de información mediante sockets. Se abordarán los objetivos y métodos de esta comunicación, que incluyen el monitoreo de los valores cinemáticos y el control de los movimientos del robot.

CONTROLADOR UR3E

El UR3e cuenta con una interfaz de usuario llamada el *polyscope*. Tiene diversas funciones como mover el brazo de forma manual ver los estados de todas las articulaciones o el extremo. Además de esto se puede programar el brazo para que desempeñe distintas funciones de forma autónoma. La utilidad que se le da en este caso es la de recibir desde el PC las instrucciones de movimiento y enviar el estado de las articulaciones.

La estructura básica de un programa en el *polyscope* está formada por tres partes. *BeforeStart*, la sección donde se inicializan las configuraciones iniciales necesarias para la comunicación y una variable vector de seis elementos donde se irán guardando los ángulos objetivo de cada articulación.

```
BeforeStart:
tmp := [0,0,0,0,0,0] # inicialización de un vector de dimensión 6 del propio programa
write_output_integer_register(0,0)
# asignar un valor de 0 al primero de los registros output de tipo entero
```

```
rtde_set_watchdog("input_int_register_0", 1, "IGNORE")
```

Código 2. Polyscope BeforeStart.

A continuación, están el *RobotProgram* y el Thread1. Estas dos secciones van a estar ejecutándose en paralelo cada una desempeñando su función.

En la sección *RobotProgram*, primero se coloca el robot en una posición de inicio y sincroniza con el Thread1. Después va a estar ordenando al brazo que ponga sus articulaciones en la posición que indica la variable tmp.

```
RobotProgram:
movej([0, -1.6, 0, -1.6, 0, 0, 0])
# mueve cada articulación a las posiciones angulares iniciales
write_output_integer_register(0,0) # asignar un valor de 0 al primero de los registros output de tipo entero
sync() # función que soncroniza los hilos que se ejecutan en paralelo

Loop: # inicializa el bucle
movej(tmp, 1.5, 0,008) # establece en cada articulación (i) la posicion de tmp[i]
sync()
```

Código 3. Polyscope RobotProgram.

En el Thread1 es un bucle infinito que continuamente está actualizando el valor del vector tmp para que contenga los ángulos objetivo de las articulaciones.

```
Thread_1:
Loop: # inicializa un bucle infinito
tmp[i] = read_input_float_register(i)
# establece en la posición i - ésima del vector tmp el valor del i
sync()
```

Código 4. Polyscope Thread_1.

Estos ángulos objetivo los lee del registro. Los registros son un buffer de memoria que tiene declarado el robot para la comunicación. Para declarar como se estructura se hace mediante un archivo .xml que se incluye en la aplicación de *Python*. En este archivo se definen cuantas variables hay y de que tipo son. El .xml está dividido en dos partes, las variables tipo *state* y las variables tipo *setp*.

Las variables tipo *state*, son las que se van a poder ver desde el PC y guardan información de velocidades y posiciones de las diferentes articulaciones, así como del extremo del robot. Estas variables nos serán útiles para hacer el control de movimiento del robot y para saber en todo momento su posición, su estado y otros parámetros.

```
<recipe key="state">
    <field name="runtime_state" type="UINT32"/>
    <field name="actual_q" type="VECTOR6D"/>
    <field name="actual_qd" type="VECTOR6D"/>
    <field name="actual_TCP_speed" type="VECTOR6D"/>
    <field name="actual_TCP_pose" type="VECTOR6D"/>
    <field name="output_bit_registers0_to_31" type="UINT32"/>
    <field name="output_int_register_0" type="INT32"/>
</recipe>
```

Código 5. RTDE configuración state.

Las variables tipo *setp*, por el contrario, son las variables que se envían desde el PC al brazo. En este caso como la variable que hemos definido anteriormente es un vector de seis elementos, necesitamos seis espacios.

```
<recipe key="setp">
    <field name="input_double_register_0" type="DOUBLE"/>
    <field name="input_double_register_1" type="DOUBLE"/>
    <field name="input_double_register_2" type="DOUBLE"/>
    <field name="input_double_register_3" type="DOUBLE"/>
    <field name="input_double_register_4" type="DOUBLE"/>
    <field name="input_double_register_5" type="DOUBLE"/>
</recipe>
```

Código 6. TDE configuración setp.

APLICACIÓN PYTHON

Universal Robots tiene una librería llamad RTDE (Real Time Data Exchange). La aplicación desarrollada se construye sobre esta librería. Facilita todo lo relacionado con enviar y recibir los datos de esos registros comentados anteriormente.

Esta aplicación lo que te permite es tanto recibir como enviar información al robot. En cuanto a recibir, se obtiene la posición del *gripper* en coordenadas cartesianas con origen en la base

del robot y además el ángulo en el que están cada una de las 6 articulaciones que tiene el robot. En cuanto a enviar, se le envía al robot las posiciones angulares objetivo de cada una de las articulaciones.

El funcionamiento del código es muy similar al anterior, se tienen dos hilos de ejecución, el primero se encarga de mantener la comunicación con el robot activa en todo momento, utilizando la librería RTDE y el otro hilo se encarga de la interfaz, gestionar las peticiones, ya sea del usuario o de otras aplicaciones. Ambos hilos se comunican para intercambiar la información.

El .xml del que se habla en la sección del controlador del UR3e, es un archivo de configuración que interpreta la librería RTDE para que el robot y la propia aplicación Python tengan el mismo esquema de memoria y no haya conflicto. Además, el propio UR3e se inicia teniendo en cuenta esto para la preparación de sus variables.

8.2. COMUNICACIÓN PC-CÁMARA

La comunicación entre el PC y la cámara de Intel D435 se lleva a cabo mediante un controlador programado en Python, utilizando la librería pyrealasene2, que permite desde otra aplicación interactuar con la cámara y recibir las imágenes que va capturando. Este controlador fue desarrollado en otra beca de colaboración de un estudiante en ICAI.

8.3. ESTRUCTURA Y DISEÑO DEL ENTORNO REAL

El entorno real es semejante al entorno virtual en cuanto a que el área de trabajo del robot está igual conformada. Tiene una plancha horizontal sobre el que está el brazo y en la parte trasera una plancha perpendicular a la base. La base y de la mitad de debajo del plano perpendicular son de color verde y la parte de arriba de la parte perpendicular es de color gris. Tanto el robot como estas planchas se montan sobre una estructura de perfiles de aluminio extruido que se ensamblan formando un soporte cuadrado. Esta estructura es versátil, ya que se puede adaptar para otras tareas o para otros brazos robóticos. Además, en

frente de esta estructura se encuentra la cámara apoyada en un trípode y conectada al ordenador donde se está ejecutando todo el proyecto.

Para el bloque objetivo, no sería óptimo tener que estar recolocando manualmente el cubo después de cada episodio, por lo que para solventar este problema se utilizan marcadores Aruco. Estos se colocan en un plano que permite a la cámara identificarlos y localizar su posición con un código Python. A continuación, se guardan estas posiciones y se asignan como posibles puntos en los que puede estar situado el cubo rojo. Posteriormente, una vez que se ejecuta el algoritmo de aprendizaje, ya sea entrenando o evaluando, la imagen del entorno, antes de ir al agente, pasa por un preprocesado en el que con técnicas de realidad aumentada se "pinta" un cubo rojo sobre una de las posiciones guardadas previamente. De este modo, la observación que le llega al agente del entorno contiene todos los elementos que necesita para realizar la toma de decisión.

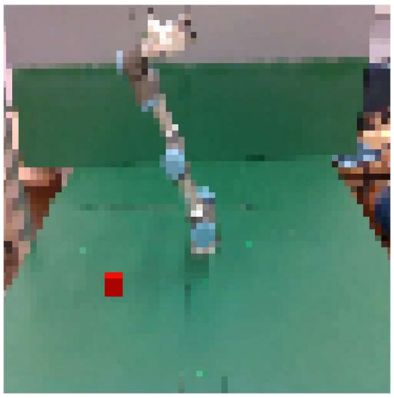


Figura 9. Ejemplo de observación del entorno real con el brazo UR3e y el cubo rojo a alcanzar diseñado con realidad aumentada a partir de las posiciones de los Arucos.

Capítulo 9. SIM2REAL

Para la transferencia de conocimiento al entorno real se va a probar directamente con un *Zero-Shot.* Los modelos entrenados anteriormente en el entorno virtual se vuelcan directamente y sin ningún tipo de re-entrenamiento o adaptación al entorno real. Es el método más eficiente ya que no requiere de nada más que las simulaciones, pero su desventaja es que el parecido entre el entorno virtual y real debe ser muy grande. Si por el contrario la diferencia es muy alta no se produce transferencia del aprendizaje entre entornos.

Como se ve en la Figura 10, el histograma de colores de la imagen del entorno real es muy diferente a la de el entorno virtual sin ruido. Por esto, al realizar los experimentos con estos modelos directamente en el entorno real la ratio de éxito del agente fue practicante nulo. Los movimientos del brazo robótico ni siquiera tienden a acercarse al target desde un inicio. Por este motivo se desestiman los modelos sin ruido.

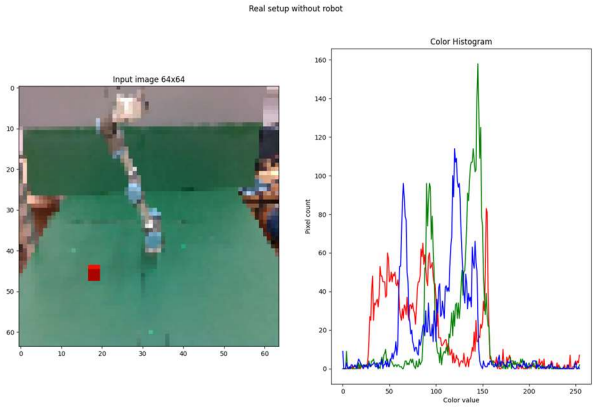


Figura 10. Imagen e histograma del entorno real.

Al utilizar en el entorno real los modelos con y sin sombra, pero con ruido, se obtienen mejores resultados, aunque no se llega a una transferencia completa. Como se puede ver en la Tabla 3, la precisión no alcanza ni el 50% en ninguno de los dos casos, pero si se observa en el movimiento del robot que éste tiende a ir hacia el objetivo en la mayoría de los episodios, faltándole el ajuste fino de los últimos movimientos de aproximación al cubo.

Modelo	Precisión entorno virtual	Precisión entorno real
Sin Sombra con ruido	90%	30%
Con Sombra con ruido	83%	20%

Tabla 3. Resultados de la transferencia zero-shot de los modelos con ruido al entorno real.

Dado que el porcentaje de éxito aún se queda lejos del obtenido en el entorno virtual, se deberían buscar técnicas complementarias para poder cerrar la brecha entre entorno virtual y entorno real.

Capítulo 10. CONCLUSIONES Y FUTURO DESARROLLO

En la virtualización del brazo robótico UR3e, se utilizó el archivo .xml de *Mujoco*, para ensamblar los .stl y dar la estructura al robot. También se incluyeron en este archivo los elementos para acotar el entorno de simulación como pared y suelo, además del bloque objetivo que es un cubo rojo.

El diseño y los resultados obtenidos en el entorno virtual mostraron que los algoritmos de aprendizaje por refuerzo profundo, como el A3C, pueden entrenar eficientemente a un agente para cumplir tareas específicas en un entorno simulado. La elección de MuJoCo como plataforma de simulación fue acertada debido a su capacidad para simular dinámicas complejas de manera precisa y eficiente.

Para el diseño del entorno real, aparte de crear un entorno similar al virtual, se crearon los puentes de comunicación entre los diferentes elementos que conforman este sistema. Comunicación Agente-Brazo y Comunicación Agente-Cámara.

Al transferir los modelos entrenados al entorno real utilizando un *Zero-Shot*, se observó que los modelos entrenados sin ruido no tuvieron un buen desempeño debido a las diferencias significativas entre los entornos virtual y real. Sin embargo, los modelos entrenados con ruido gaussiano mostraron mejores resultados. Sin embargo se considera que éstos están aún lejos del desempeño que se espera de un agente al realizar la tarea de aproximación a una pieza, por lo que se propone como futuro desarrollo la aplicación de otras técnicas que permitan una transferencia más eficiente de conocimiento entre entornos.

Capítulo 11. ALINEACIÓN CON LOS ODS

- **ODS 4: Educación de Calidad**

A través de la investigación y el desarrollo de estrategias para la transferencia de experiencia entre entornos virtuales y reales, el proyecto contribuye a mejorar la calidad y el acceso a la educación en robótica y en tecnología. Esto podría tener impacto positivo en la formación de profesionales para estos campos.

- **ODS 8: Trabajo Decente y Crecimiento Económico**

Al mejorar la capacidad de los agentes para adaptarse a entornos industriales, se promueve el crecimiento económico al aumentar la eficiencia y la productividad en sectores que utilizan tecnologías robóticas. Esto puede llevar a la creación de empleos y ayudar al desarrollo económico sostenible.

- **ODS 9: Industria, Innovación e Infraestructura**

El proyecto contribuye a la mejora de la infraestructura industrial al explorar estrategias para la transferencia efectiva de conocimiento entre entornos virtuales y reales en el ámbito de la robótica. Al promover la innovación en la adaptación de agentes de aprendizaje por refuerzo a entornos industriales, se apoya el desarrollo de tecnologías que puedan beneficiar a la industria.

- **ODS 11: Ciudades y Comunidades Sostenibles**

Si estos avances tecnológicos se implementan de manera amplia, podrían contribuir a la creación de comunidades más sostenibles al mejorar la eficiencia en entornos industriales y urbanos, reduciendo los residuos y optimizando los procesos de producción.

BIBLIOGRAFÍA

[1] Universal Robots, "Robot UR3," disponible en: https://www.universal-robots.com/es/productos/robot-ur3/, último acceso: 14/06/2024. (Página web).

[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, y A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process Mag, vol. 34, no. 6*, pp. 26–38, 2017, doi: 10.1109/MSP.2017.2743240.

[3] H. Sasaki, T. Horiuchi, y S. Kato, "A study on vision-based mobile robot learning by deep Q-network," 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, Japan, del 19 al 22 de September de 2017, pp. 799-804, doi: 10.23919/SICE.2017.8105597.

[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, y O. Klimov, "Proximal Policy Optimization Algorithms" arXiv preprint arXiv:1707.06347, 2017. (Preprint).

[5] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," en International Conference on Machine Learning, conference, pp. 1928-1937, PMLR, 19 al 24 Junio de 2016, Nueva York.

[6] C. Garrido Urbano, "Sistema de navegación para robot móvil basado en aprendizaje por refuerzo," Tesis (Grado), Universidad de los Andes, ingeniería electrónica, 2020.

[7] A. López Sánchez, "Aplicación de aprendizaje profundo por refuerzo a problemas de robótica aérea," Tesis (Máster), E.T.S. de Ingenieros Informáticos (UPM), 2021.

[8] W. Zhao, J. P. Queralta, y T. Westerlund, "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey," en 2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020, del 1 al 4 de December de 2020, Canberra, Australia. pp. 737–744, doi: 10.1109/SSCI47803.2020.9308468.

[9] E. Salvato, G. Fenu, E. Medvet, y F. A. Pellegrino, "Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning," *IEEE Access, vol. 9*, pp. 153171-153187, 2021.

[10] F. Darema, "Dynamic data driven applications systems: A new paradigm for application simulations and measurements," en International Conference on Computational Science, del 6 al 9 de junio de 2004, pp. 662-669, Berlín, Heidelberg: Springer Berlin Heidelberg, junio de 2004.

[11] B. Ince, H. S. Shin, y A. Tsourdos, "Optimization of a Robust Reinforcement Learning Policy," en AIAA SCITECH Forum, del 23 al 27 de Enero de 2023, Maryland, Estados Unidos. p. 0967.

[12] M. S. P. de Melo, J. G. da Silva Neto, P. J. L. da Silva, J. M. X. N. Teixeira, y V. Teichrieb, "Analysis and comparison of robotics 3D simulators," en Proceedings – 2019 21st Symposium on Virtual and Augmented Reality, SVR 2019, 28 al 31 de octubre de 2019, Río de Janeiro, Brasil. pp. 242–251, doi: 10.1109/SVR.2019.00049.

[13] J. K. Haas, "A History of the Unity Game Engine," 2014. Universidad de Wisconsin-Madison (Informe).

[14] E. Todorov, T. Erez, y Y. Tassa, "MuJoCo: A physics engine for model-based control," en IEEE International Conference on Intelligent Robots and Systems, del 7 al 12 de octubre de 2012, Algarve, Portugal., pp. 5026–5033, doi: 10.1109/IROS.2012.6386109.

[15] N. Koenig y A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," en 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), del 28 de septiembre al 2 de octubre de 2004, Sendai, Japón. vol. 3, pp. 2149-2154, IEEE.

[16] S. Jordan, Y. Chandak, D. Cohen, M. Zhang, y P. Thomas, "Evaluating the performance of reinforcement learning algorithms," en International Conference on Machine Learning, del 21 al 23 de noviembre de 2020, Telematico, pp. 4962-4973, PMLR.

[17] W. Zhu, X. Guo, D. Owaki, K. Kutsuzawa, y M. Hayashibe, "A survey of sim-to-real transfer techniques applied to reinforcement learning for bioinspired robots," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[18] J. Tobin et al., "Domain randomization for transferring deep neural networks from simulation to the real world," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), del 24 al 28 de September de 2017, Vancouver, British Columbia, Canada. pp. 23-30, IEEE.

[19] Z. Chen, Y. Deng, Y. Li, y Q. Gu, "Understanding Transferable Representation Learning and Zero-shot Transfer in CLIP," arXiv preprint arXiv:2310.00927, 2023. (Preprint).

[20] M. Wang y W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing, vol. 312*, pp. 135-153, 2018.

ANEXO A. CÓDIGO XML PARA MUJOCO

En este anexo se presenta el código completo que define la estructura del modelo virtual del UR3e.

```
<mujoco model="ur3e">
  <compiler inertiafromgeom='true' angle="radian" settotalmass="11.1" meshdir="stl/" />
  <option timestep="0.002" gravity="0 0 -9.81"/>
  <asset>
    <mesh name="base" file="base.stl" />
    <mesh name="shoulder_1" file="shoulder_1.stl" />
    <mesh name="shoulder_2" file="shoulder_2.stl" />
    <mesh name="upperarm_1" file="uperarm_1.stl" />
    <mesh name="upperarm_2" file="uperarm_2.stl" />
    <mesh name="upperarm_3" file="uperarm_3.stl" />
    <mesh name="upperarm_4" file="uperarm_4.stl" />
    <mesh name="upperarm_5" file="uperarm_5.stl" />
    <mesh name="forearm_1" file="forearm_1.stl" />
    <mesh name="forearm_2" file="forearm_2.stl" />
    <mesh name="forearm_3" file="forearm_3.stl" />
    <mesh name="forearm_4" file="forearm_4.stl" />
    <mesh name="wrist1_1" file="wrist1_1.stl" />
    <mesh name="wrist1_2" file="wrist1_2.stl" />
    <mesh name="wrist2_1" file="wrist2_1.stl" />
    <mesh name="wrist2_2" file="wrist2_2.stl" />
    <mesh name="wrist3" file="wrist3.stl" />
    <mesh name="mano" file="mano.stl"/>

    <texture type="skybox" builtin="gradient" rgb1=".2 .3 .4" rgb2="1 1 1" width="100"
height="100"/>

    <texture name="groundplane" type="2d" builtin="checker" rgb1=".25 .26 .25" rgb2=".22 .22 .22"
width="100" height="100" mark="none" markrgb=".8 .8 .8"/>

    <texture name="greenground" type="2d" builtin="gradient" rgb1="0.19 0.28 0.23" rgb2="0 0 0"
width="100" height="100" mark="none" markrgb=".8 .8 .8"/>

    <material name="MatViz" specular="1" shininess=".1" reflectance="0.5" rgba="0.07 0.07 0.1 1"/>
```

```
    <material name="MatGnd" texture="groundplane"  texrepeat="5 5" specular="1" shininess="0"
reflectance="0"/>

    <material name="silver" rgba="1 1 1 1" reflectance="0"/>
    <material name="blue" rgba="0.34 0.67 0.824 1"/>
    <material name="grey" rgba="0.7 0.7 0.7 1" />
  </asset>

  <contact>
    <pair geom1="geo_link1_1" geom2="geo_base" condim="1" />
  </contact>

  <default class ="ground">
    <geom type="plane" margin="0.001" contype="1" />
  </default>

  <worldbody>
    <light cutoff="200" diffuse="1.2 1.2 1.2" dir="-0 0 -1.3" directional="true" exponent="1"
pos="0 0 1.3" specular=".1 .1 .1" castshadow="true"/>

    <camera name="view1" pos="0.2 0.1 2.0" zaxis="0 0 1" fovy="45" ipd="0.068"/>

    <geom name="ground" class="ground" type='plane' pos="0 0 0" rgba="0.19 0.30 0.23 1" size="2 2
1"/>

    <geom name="ground2" class="ground" type='plane' pos="1 0 1" quat="0.707107 0 0.707107 0"
rgba=".25 .26 .25 1" size="1 1 0.5" />

    <geom name="target" pos="-0.3 0 0.03" rgba="0.9 0.1 0.1 1" size="0.03 0.03 0.03" type="box"
contype='0' conaffinity="0"/>

    <body name="robot0:base_link" pos="0 0 0">
      <geom name = "geo_base" type="mesh" material="silver" mesh="base" />
      <body name="link1" pos="0 0 0.1519">

        <joint name="joint0" pos="0 0 0" axis="0 0 1" limited="true" range="-4.1015 0.9599" />
        <geom name = "geo_link1_0" type="mesh" material="blue" mesh="shoulder_2" />
        <geom name = "geo_link1_1" type="mesh" material="grey" mesh="shoulder_1" />
        <body name="link2" pos="0 0.1198 0" quat="0.707107 0 0.707107 0">

          <joint name="joint1" pos="0 0 0" axis="0 1 0" limited="true" range="-2.7925 0.0" />
          <geom name="geo_link2_0" type="mesh" material="grey" mesh="upperarm_1" />
          <geom name="geo_link2_1" type="mesh" material="blue" mesh="upperarm_2" />
```

```
        <geom name="geo_link2_2" type="mesh" material="silver" mesh="upperarm_3" />
        <geom name="geo_link2_3" type="mesh" material="blue" mesh="upperarm_4" />
        <geom name="geo_link2_4" type="mesh" material="grey" mesh="upperarm_5" />
        <body name="link3" pos="0 -0.0925 0.24365">

          <joint name="joint2" pos="0 0 0" axis="0 1 0" limited="true" range="-2.0 0.0" />
          <geom name="geo_link3_0" type="mesh" material="grey" mesh="forearm_1" />
          <geom name="geo_link3_1" type="mesh" material="silver" mesh="forearm_2" />
          <geom name="geo_link3_2" type="mesh" material="blue" mesh="forearm_3" />
          <geom name="geo_link3_3" type="mesh" material="grey" mesh="forearm_4" />
          <body name="link4" pos="0 0 0.21325" quat="0.707107 0 0.707107 0">

            <joint name="joint3" pos="0 0 0" axis="0 1 0" limited="true" range="-4.3633 -0.565"/>
            <geom name="geo_link4_0" type="mesh" material="blue" mesh="wrist1_1" />
            <geom name="geo_link4_1" type="mesh" material="grey" mesh="wrist1_2" />
            <body name="link5" pos="0 0.08505 0">

              <joint name="joint4" pos="0 0 0" axis="0 0 1" limited="true" range="-2.094 2.094"/>
              <geom name="geo_link5_0" type="mesh" material="blue" mesh="wrist2_1" />
              <geom name="geo_link5_1" type="mesh" material="grey" mesh="wrist2_2" />
              <body name="link6" pos="0 0 0.08535">

                <joint name="joint5" pos="0 0 0" axis="0 1 0" limited="true" range="-4.22 4.22"/>
                <geom name="geo_link6_0" type="mesh" material="silver" mesh="wrist3" />
                <geom name="geo_link6_1" type="mesh" material="silver" mesh="mano" />
                <site name="robot0:grip" pos="0 0.19 0" rgba="1 0 0 0" size="0.01 0.01 0.01" />
              </body>
            </body>
          </body>
        </body>
      </body>
    </body>
  </worldbody>
  <actuator>
    <position ctrlrange="-4.1015 0.9599" ctrllimited="true" name="joint0_motor" joint="joint0"/>
    <position ctrlrange="-2.7925 0.0" ctrllimited="true" name="joint1_motor" joint="joint1"/>
    <position ctrlrange="-2.0 0" ctrllimited="true" name="joint2_motor" joint="joint2"/>
    <position ctrlrange="-4.3633 -0.565" ctrllimited="true" name="joint3_motor" joint="joint3"/>
    <position ctrlrange="-2.0944 2.0944" ctrllimited="true" name="joint4_motor" joint="joint4"/>
    <position ctrlrange="-4.22 4.22" ctrllimited="true" name="joint5_motor" joint="joint5"/>
  </actuator>
</mujoco>
```