XX Simposio CEA de Control Inteligente
25-27 de junio de 2025, Huelva

Aprendizaje por Refuerzo para el cálculo del Modelo Cinemático Inverso de un robot colaborativo

Menéndez-García, Alejandro.[a,*], Sierra-García, Jesús Enrique.[b]

[a] *Digitalización, Universidad de Burgos, Campus Río Vena, Avda. Cantabria, s/n, 09006 Burgos, España.*

To cite this article: Menéndez García, Alejandro, Sierra García, Jesús Enrique, 2025. "Reinforcement Learning for the computation of the Inverse Kinematic Model of a collaborative robot" XX Simposio CEA de Control Inteligente

Resumen

Este artículo propone un enfoque de aprendizaje por refuerzo para resolver el modelo cinemático inverso (MCI) en manipuladores industriales. Se emplea un agente *Deep Deterministic Policy Gradient* (DDPG) con buffer de repetición, *Hindsight Experience Replay* (HER) y normalización de entradas, que interactúa con un entorno que emula la cinemática directa del robot colaborativo GOFA 5_95. La función de recompensa combina penalizaciones por error de posición y discrepancia angular, y se entrena al agente en 27 subespacios del volumen de trabajo. En la fase de evaluación, se ejecutan un número de inferencias sobre superficies esféricas y esferoidales, obteniéndose tiempos medios de inferencia inferiores a 5 ms, tasas de éxito superiores al 98 %, errores posicionales medios alrededor de 11 mm y errores angulares medios de 0,13 rad. Estos resultados demuestran la viabilidad del aprendizaje por refuerzo como alternativa en tiempo real al cálculo iterativo del Jacobiano para aplicaciones que no requieran de precisión subcentimétrica.

Palabras clave: Aprendizaje por refuerzo, robótica, modelo cinemático inverso, modelado, control inteligente.

Reinforcement Learning for the computation of the Inverse Kinematic Model of a collaborative robot

Abstract

This paper proposes a reinforcement learning approach to solve the inverse kinematic model (IKM) in industrial manipulators. It employs a *Deep Deterministic Policy Gradient* (DDPG) agent with a replay buffer, *Hindsight Experience Replay* (HER), and input normalization, interacting with an environment that simulates the forward kinematics of the GOFA 5_95 collaborative robot. The reward function penalizes both position error and angular discrepancy, and the agent is trained across 27 subspaces of the workspace. In the evaluation phase, a series of inferences are performed on spherical and spheroidal surfaces, yielding average inference times below 5 ms, success rates above 98 %, mean position errors around 11 mm, and mean angular errors of 0.13 rad. These results demonstrate the viability of reinforcement learning as a real-time alternative to iterative Jacobian-based calculations for applications not requiring subcentimeter accuracy.

Keywords: Reinforcement learning, robotics, inverse kinematic model, modelling, intelligent control

1. Introducción

La robótica se ha incorporado de forma masiva en sectores como la manufactura, la logística y la salud. En plantas de producción, los manipuladores automatizados llevan a cabo tareas de montaje, soldadura y pintura con niveles de precisión y repetibilidad imposibles de replicar manualmente, lo que eleva la productividad y disminuye los accidentes (Bogue, R., 2018). En el ámbito sanitario, los sistemas quirúrgicos robóticos facilitan intervenciones mínimamente invasivas de gran exactitud, mientras que, en centros de distribución y almacenes, los vehículos guiados automatizan el traslado de mercancías, optimizando el flujo logístico (Bayona et al.,2024; Sierra-Garcia, et al., 2023). En el ámbito de la automatización de procesos, el proyecto europeo MANiBOT ha sido diseñado con este objetivo, siendo casos de uso como recoger y trasladar maletas desde la cinta de equipaje hasta un carro en aeropuertos (aliviando la carga de trabajo del personal) o como adaptarse a entornos de supermercados con el objetivo de reponer productos y rellenar estanterías con rapidez y precisión, mejorando la eficiencia en la reposición de stock (Manibot, 2025).

*Autor para correspondencia: alejandro.menendez@ubu.es

El modelo cinemático inverso (MCI) convierte una posición y orientación deseadas del end effector en la configuración articular necesaria para alcanzarlas. Gracias a él se pueden planificar trayectorias suaves que respeten los límites mecánicos y evitar configuraciones singulares, lo que resulta clave tanto en tareas finas de ensamblaje o manipulación en entornos tridimensionales como en la generación de animaciones realistas en simulación (Aristidou et al., 2018).

Normalmente, el MCI se resuelve mediante métodos iterativos basados en el Jacobiano, como el método Newton–Raphson, Gauss–Newton o Levenberg–Marquard, los cuales ofrecen buena robustez, pero suelen requerir varios milisegundos para converger en robots de seis o más grados de libertad (Chiacchio & Chiaverini, 1995; Chen et al., 2013). Esto dificulta el control en tiempo real de ciclo inferior a 5 ms, lo que limita su uso en sistemas de alta frecuencia.

Algunos enfoques han tratado de acelerar el cálculo en contextos específicos: Tolani et al. (2000) desarrollaron técnicas en tiempo real para extremidades antropomórficas; Rapetti et al. (2020) propusieron un MCI dinámico aplicado al seguimiento de movimiento; y Pizzolato et al. (2016) combinaron cinemática inversa y dinámica para aplicaciones de miembro inferior. Más recientemente, Zhang et al. (2024) exploraron la optimización en tiempo real de MCI para robots redundantes con restricciones cinemáticas.

Sin embargo, el uso de aprendizaje por refuerzo para abordar el MCI sigue siendo un área poco explotada donde todavía no se ha explorado al completo el aprovechar la capacidad de RL para aprender políticas de control eficientes y adaptativas que podrían operar al margen de iteraciones costosas del Jacobiano, necesario para el cálculo del MCI en el método tradicional. En el aprendizaje por refuerzo (RL), un agente observa el estado del entorno, elige una acción según una política y recibe una recompensa como consecuencia (Sierra & Santos, 2021). La recompensa cuantifica la calidad de la acción con respecto al objetivo. El objetivo es aprender una estrategia que permita maximizar la recompensa acumulada a lo largo del tiempo.

En este artículo se presenta un agente de aprendizaje por refuerzo profundo para resolver el MCI en manipuladores industriales (robot colaborativo GOFA 5_95), alcanzando tiempos de inferencia inferiores a 5 ms. Se evalúa su capacidad para minimizar simultáneamente el error de posición y el error angular, mostrando resultados sobre superficies esféricas y esferoidales que ilustran la distribución espacial de ambos tipos de error.

El resto del artículo se estructura de la siguiente forma. La propuesta para resolver el MCI mediante RL se describe en la sección 2. El entrenamiento del sistema se explica en la sección 3. En la sección 4 se detallan los resultados. Finalmente se exponen las conclusiones y los trabajos futuros.

2. Resolución del modelo cinemático inverso mediante aprendizaje por refuerzo

En un sistema de aprendizaje por refuerzo (RL), la dinámica se establece a través de la interacción entre un agente y su entorno. En cada instante temporal, se recibe del entorno un estado o vector de observaciones que envía la información relevante sobre la situación actual. Empleando ese estado como base, se elige una acción de acuerdo con la política actual. Al ejecutarse la acción, el entorno devuelve un nuevo estado junto con una recompensa que mide el grado de cumplimiento de la meta. Estas transiciones quedan registradas en un buffer de repetición y se emplean para actualizar tanto los parámetros de la política como los de la función de valor, con el fin de maximizar de manera acumulativa las recompensas futuras.

La Figura 1 muestra el esquema general de Agente-Entorno planteado para este problema:

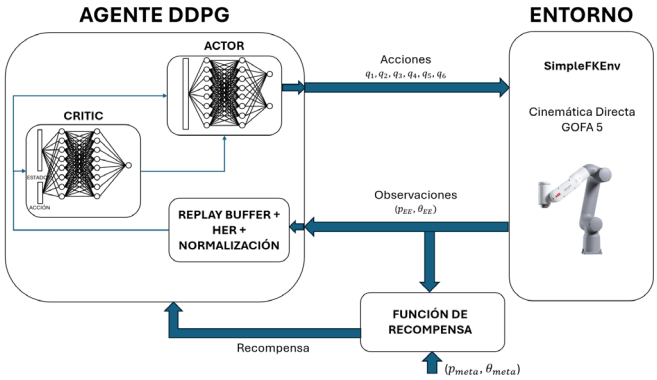

Figura 1:Esquema de Reinforce Learning Agente-Entorno

La figura muestra el ciclo de interacción del agente DDPG con el entorno SimpleFKEnv del GOFA 5: en cada paso el actor genera una configuración articular a partir de la observación de la pose actual del efector (posición y orientación), y el crítico evalúa ese par estado-acción para guiar la mejora de la política; todas las transiciones se almacenan en un replay buffer con HER y normalización. El entorno devuelve la nueva pose del end effector, y la función de recompensa emplea esa observación y la meta objetivo para obtener una recompensa generando así el bucle de aprendizaje.

Cuando se recurre a un modelo de aprendizaje por refuerzo profundo (Deep RL), tanto las políticas como las funciones de valor se representan mediante redes neuronales capaces de procesar entradas de alta dimensionalidad (por ejemplo, posiciones y orientaciones) y de generar salidas continuas (por ejemplo, comandos articulares).

2.1. Agente DDPG

El agente seleccionado para este proyecto emplea el algoritmo DDPG (Deep Deterministic Policy Gradient). DDPG es un método de aprendizaje por refuerzo actor–crítico que hace uso de redes neuronales con el objetivo de aproximar y optimizar directamente una política determinista en espacios de acción continuos. Esta técnica se complementa con el algoritmo HER (Hindsight Experience Replay), el cual es un método que reaprovecha experiencias pasadas asignándoles objetivos alternativos ya cumplidos, con lo que multiplica los ejemplos de aprendizaje. Esto permite aprender una política determinista continua que resuelva la cinemática inversa del robot ABB CRB15000-95 (GOFA). La arquitectura del actor consta de una red neuronal completamente conectada con tres capas ocultas de 256 neuronas cada una, seguidas de una capa de salida cuya activación está escalada a los límites físicos de cada articulación. Antes de alimentar las redes actor y crítico, el vector de estado bruto se normaliza componente a componente según:

$$\widehat{x_i} = \frac{x_i - \mu_i}{\sqrt{\sigma_i^2 - \varepsilon}} \quad (1)$$

Siendo:

- μ_i es la media estimada de la i-ésima componente.
- σ_i^2 es su varianza.
- ε es un pequeño valor para evitar divisiones por cero.

Esta normalización mantiene los valores de entrada en rangos óptimos para las funciones de activación, evita inestabilidades numéricas y acelera la convergencia de la política al reducir el desplazamiento interno de covariables y asegurar una propagación de gradientes más uniforme (Lillicrap et al., 2016).

De manera análoga, el crítico utiliza una red de arquitectura similar (tres capas ocultas de 256 unidades) que recibe como entrada la concatenación del vector de estado normalizado y la acción propuesta, produciendo un único valor escalar Q que estima la calidad de esa pareja estado-acción. Esta configuración actor-crítico permite que el agente aprenda directamente la política determinista de posición articular que aproxima la función inversa.

Para garantizar la estabilidad del entrenamiento, se emplean redes objetivo (actor-objetivo y crítico-objetivo) cuyas ponderaciones se actualizan de forma paulatina mediante interpolación suave (soft update) con un coeficiente tau(τ). Durante la fase de exploración, al vector de acción generado por el actor se le añade ruido gaussiano, comenzando con un desvío estándar elevado que se atenúa gradualmente a medida que avanza el aprendizaje. Estas estrategias combinadas redes objetivo y ruido adaptativo permiten al agente aprender con mayor fiabilidad la función inversa que posiciona y orienta el efector final según la meta deseada. En nuestros experimentos, ambas redes se optimizaron con Adam, usando tasas de aprendizaje conservadoras (por ejemplo, 5×10^{-4} para el actor y 10^{-3} para el crítico) y actualizando las redes objetivo con un factor de suavizado de aproximadamente $tau = 0.005$ en línea con prácticas en otros experimentos de esta índole. Este planteamiento contribuye a una convergencia más sólida y estable de la arquitectura actor-crítico determinista.

2.2. Observaciones, estados y acciones

El vector de estado en el entorno incluye tanto la pose actual del efector final como la meta deseada. Específicamente, el estado es un vector de 12 valores donde se concatenan la posición cartesiana (x,y,z) del efector final y su orientación en ángulos de Euler normalizados (cada componente dividido entre pi), junto con la posición (x,y,z) y la orientación objetivo de la meta, también normalizadas. Esta definición proporciona al agente información completa sobre la ubicación actual del efector y la meta a la que debe llegar. Dado que cada episodio consiste en un solo paso de control, no se incluye velocidad en el estado; toda la dinámica relevante está encapsulada en la pose actual y la meta.

Las acciones corresponden directamente a las coordenadas articulares del robot: el agente emite un vector continuo cuya dimensión coincide con el número de articulaciones activas del manipulador, indicando la posición angular deseada para cada articulación. Para garantizar factibilidad física, las acciones se restringen dentro de los límites definidos en el modelo URDF (Unified Robot Description Format) del robot; en la práctica, la capa de salida del actor está escalada para respetar esos rangos de movimiento. De este modo, el agente opera en un espacio de acción continuo coherente con la cinemática del manipulador. En la implementación, la acción se interpreta como la posición articular absoluta objetivo de cada articulación, ya que no hay pasos adicionales después; tras una acción, el episodio termina.

2.4. Función de recompensa

La función de recompensa incorpora tanto el error de posición como el error de orientación entre la pose alcanzada por el efector final y la meta deseada. Concretamente, la recompensa sin escalar se define como la suma negativa de la distancia euclídea al objetivo y un término ponderado del error angular:

$$r = 80 \times [-(\|p_{EE} - p_{meta}\| + \beta \times \|\theta_{EE} - \theta_{meta}\|)] + bonus \quad (2)$$

$$bonus = \begin{cases} 0, & exito = 0 \\ 50 \times k, & exito = 1 \end{cases} \quad (3)$$

Donde:

- β = 0.2 es un peso que modera la importancia del error de orientación.
- $'bonus'$ es un valor que vale 0 en capítulos en los que no ha habido éxito en llegar al punto objetivo cumpliendo los umbrales de posición y orientación. Mientras que vale cuando el episodio se considera exitoso $50 \times k$ siendo k el número de episodios exitosos consecutivos.

Esta señal se multiplica por una constante de escala (80) para producir recompensas con magnitudes adecuadas al aprendizaje. De esta forma, la recompensa máxima (cero tras escalado) se obtiene únicamente cuando se alcanza la meta exacta en posición y orientación. Además, se asigna un *bonus* de 50 puntos adicional acumulativo al finalizar un episodio exitoso por cada subespacio consecutivo de metas superado. Es decir, cada vez que el agente completa con éxito un subespacio y pasa al siguiente, recibe una recompensa positiva extra. Esto incentiva una progresión ordenada a través de las regiones definidas, premiando al agente por resolver subespacios secuencialmente.

En conjunto, esta función de recompensa motiva al agente a minimizar simultáneamente la distancia al objetivo y la discrepancia angular, favoreciendo soluciones que satisfagan las dos condiciones. El factor β <1 refleja la prioridad de alcanzar la posición objetivo con precisión, relegando el ajuste angular pero aun siendo penalizador, lo que es coherente con las exigencias típicas de aplicaciones en manipuladores industriales.

3. Entrenamiento del sistema

3.1. *Entorno, subdivisión del espacio de metas y criterios de éxito*

Se ha creado un entorno de simulación 'SimpleFKEnv', que fue desarrollado en Python usando la librería *ikpy*, cargando el modelo descriptivo del robot ABB CRB15000-95. Este entorno realiza la cinemática directa: dado un vector de articulaciones, calcula la posición y orientación resultantes del efector final. En cada episodio se ejecuta un único paso: el agente propone una configuración articular, el entorno la aplica y retorna el nuevo estado del efector (posición y orientación), la recompensa y la señal de fin de episodio. El estado devuelto incluye la posición y la orientación actuales del end effector final, junto con la posición y orientación de la meta. En el reinicio de cada episodio, el robot se inicializa en la postura base con todas las articulaciones en 0, calculando así una posición inicial conocida para el efector.

El espacio de metas se definió como un cubo dentro del espacio cartesiano de trabajo. Los límites de posición van de 0.7 a 0.9 m en el eje X, de 0.0 a 0.3 m en Y y de 0.4 a 0.7 m en Z. De esta manera, el agente debe aprender a mover el efector a cualquier punto dentro de ese volumen manteniendo la orientación prescrita. Este diseño de metas permite evaluar de manera exhaustiva la capacidad del agente para resolver la cinemática inversa en el dominio relevante de trabajo del robot, garantizando cobertura completa dentro del cubo definido.

Para organizar el entrenamiento, el cubo de metas se subdividió en 27 subespacios tridimensionales iguales (rejilla de $3 \times 3 \times 3$). El centro de cada subcubo se calculó y se usó como representante de la región correspondiente, como se muestra en la Figura 2.

A continuación, se filtraron los subespacios alcanzables resolviendo la cinemática inversa en el centro de cada uno (con la orientación objetivo fijada). Solo se consideraron *activos* aquellos subespacios cuyo centro podía alcanzarse satisfactoriamente (error posicional < 2 mm y error angular < 0.2 rad). De los 27 subespacios iniciales, se seleccionó el subconjunto alcanzable según estos criterios; únicamente estos subespacios activos se emplearon en el ciclo de entrenamiento.

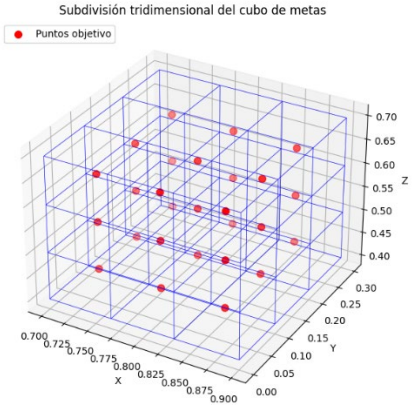

Figura 2: Subdivisión tridimensional del cubo.

Esta Figura 2 ilustra cómo se ha particionado el espacio de metas en 27 celdas cúbicas de igual tamaño. Los centros de cada subcubo (puntos rojos) representan los objetivos clave en cada región del espacio. Solo los subespacios cuyos centros resultaron físicamente alcanzables con la orientación dada se mantienen activos, optimizando así el entrenamiento en metas factibles.

Cada subespacio se considera resuelto si el agente alcanza una meta en esa región cumpliendo los umbrales de error establecidos. Durante el entrenamiento, los subespacios activos se recorrieron de manera secuencial en orden aleatorio cíclico: el agente debía alcanzar exitosamente un subespacio para avanzar al siguiente, acumulando así una racha de éxitos. Este enfoque asegura que el aprendizaje cubra todas las regiones del espacio de metas de forma progresiva y equilibrada.

3.2. *Técnicas de entrenamiento*

Para estabilizar y mejorar el proceso de entrenamiento se implementaron varias técnicas complementarias. Se emplea un normalizador en línea que actualiza incrementalmente la media y la varianza de cada componente de los estados observados, normalizando las entradas de las redes para mantenerlas centradas en cero con varianza unitaria. Esto evita que escalas muy distintas dificulten la convergencia. Asimismo, se utiliza un replay buffer de gran capacidad (aproximadamente 13,000 transiciones) que almacena las experiencias previas del agente (estado, acción, recompensa, nuevo estado). Cada transición recopilada se añade al buffer y, cuando este supera el tamaño mínimo de un minibatch, el agente selecciona un lote aleatorio de muestras para actualizar sus redes actor y crítico. De esta manera se implementa el aprendizaje *off-policy* clásico de DDPG, aprovechando datos no correlacionados.

Además, al finalizar cada episodio se aplica HER: se genera una meta alternativa a partir de los estados finales alcanzados en trayectorias exitosas. Por cada episodio exitoso se crean ejemplos sintéticos usando la posición final del end effector como nueva meta, lo que multiplica las transiciones útiles de entrenamiento. Con estas muestras adicionales se realizan iteraciones de aprendizaje extra en cada actualización. Esto permite que el agente entrene tanto en tiempo real (cada paso dentro del episodio) como con los datos expandidos tras el episodio, acelerando la convergencia en un espacio de metas continuo y potencialmente disperso. En la práctica, el agente entrena con minibatches cada vez que el buffer lo permite, iterando de forma constante durante el entrenamiento. La combinación de normalización de estados, replay buffer y HER permitió que el agente consolide progresivamente una política estable que generaliza bien en todo el espacio de metas definido.

3.3. *Ciclo de entrenamiento y condiciones de parada*

El entrenamiento se realizó en episodios independientes (hasta un máximo de 6000 episodios) con un único paso por episodio (MAX_STEPS=1). En cada episodio se selecciona aleatoriamente un subespacio activo y se genera una meta dentro de sus límites mediante muestreo uniforme. Tras reiniciar el entorno y normalizar el estado inicial, el agente ejecuta la acción propuesta (configuración articular) y recibe la recompensa calculada según el error final. A continuación, la transición (estado, acción, recompensa, nuevo estado) se almacena en el *replay buffer*. Si el buffer contiene suficientes muestras, el agente extrae minibatches aleatorios para

actualizar las redes actor y crítico (proceso off-policy de DDPG).

El episodio se considera exitoso si los errores finales de posición y orientación están dentro de los umbrales establecidos (error posicional < 2 mm y error angular < 0.2 rad). Si es exitoso, se incrementa la racha de éxitos y en el próximo episodio se procede con otro subespacio; si falla, el agente repite el mismo subespacio en el siguiente episodio. El entrenamiento finaliza cuando el agente logra una racha de éxitos consecutivos igual al número de subespacios activos (es decir, ha resuelto todas las regiones alcanzables) o cuando se alcanza el límite de episodios. Este criterio de parada garantiza que el agente haya dominado cada región antes de concluir, promoviendo un aprendizaje completo en todo el espacio de metas.

4. Resultados

En esta sección se presentan primero los resultados del entrenamiento del agente (recompensa por episodio y tasa de éxito), y seguidamente se muestran los resultados de inferencia sobre las superficies de prueba (esfera y esferoide), incluyendo latencia, tasas de éxito, errores medios y distribución espacial de los mismos.

4.1. Rendimiento durante el entrenamiento

En la Figura 3 se muestra la evolución de la recompensa por episodio:

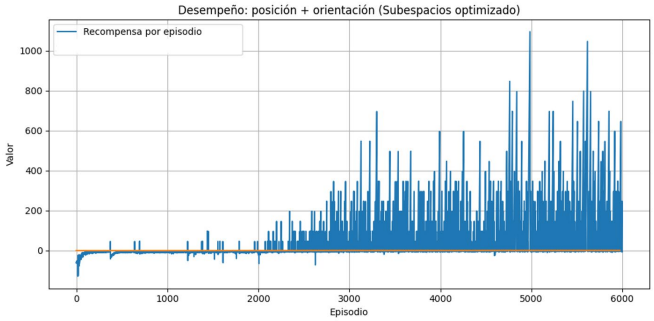

Figura 3: Gráfica Episodio-Recompensa

Al inicio (episodios 0–1000) la recompensa es negativa y la tasa de éxito cercana a cero. A partir del episodio 2000, la recompensa progresa a valores positivos y la tasa móvil supera el 50 %. Hacia el final del entrenamiento (episodios 4000–6000), la recompensa por episodio alcanza valores medios superiores a 200.

4.2. Rendimiento de inferencia

Tras el entrenamiento, se evaluó al agente en 90000 metas repartidas uniformemente en la superficie de una esfera (45000 puntos) y en la de un esferoide de revolución (45000 puntos). El tiempo total de inferencia medido fue:

$$T_{total} = 12{,}532\ s$$

$$t_{inferencia} = \frac{T_{total}}{N_{total}} = \frac{12{,}532}{90000} = 1{,}3924 \times 10^{-4}\ s$$

$$t_{inferencia} = 0{,}139\ ms$$

4.3. Tasas de éxito y error posicional medio

Definimos la tasa de éxito como:

$$\mu = \frac{N_{exitos}}{N_{puntos}} \times 100 \quad (4)$$

El error posicional medio se obtiene empleando:

$$\bar{e} = \frac{1}{N_{puntos}} \times \sum_{i=1}^{N_{puntos}} e_i \quad (5)$$

Y el error de orientación medio se obtiene empleando:

$$\overline{\Delta\theta} = \frac{1}{N_{puntos}} \times \sum_{i=1}^{N_{puntos}} \Delta\theta \quad (6)$$

$$\Delta\theta = |\theta_{EE} - \theta_{meta}| \quad (7)$$

Donde:

- e_i es la distancia euclídea entre la posición alcanzada y la meta en la i-ésima inferencia.
- $\Delta\theta$ es el error de orientación obtenido

Tabla 1: Tasa de éxito y error de posición medio

Superficie	Puntos	Éxitos	%	$\bar{e}$	$\overline{\Delta\theta}$
Esfera	45000	44799	99,5%	11mm	0,116rad
Esferoide	45000	44214	98,2%	10mm	0,141rad

4.4. Visualización de la distribución de errores

La Figura 4 presenta mapas de color en la superficie completa de la esfera (panel izquierdo) y del esferoide (panel derecho), ambos en milímetros, evidenciando la variación espacial del error posicional.

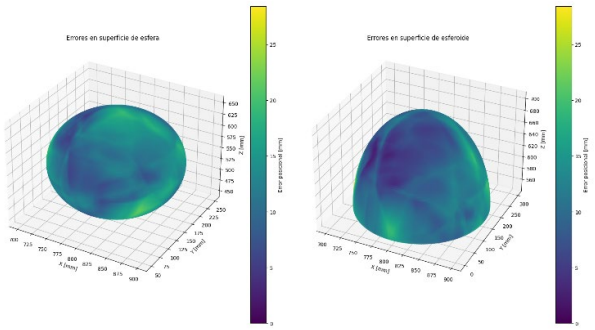

Figura 4: Error de posición en puntos de la superficie en una esfera y esferoide

En ambos mapas esféricos predominan tonos azulados (error de posición aproximado de 10 mm), sin zonas de fallo sistemáticas. Aisladamente se observan tonos más amarillentos (25–30 mm), aunque visualmente se puede decir que se tiene errores de posición cercanos a 10 mm.

Para el error de orientación en los distintos puntos que conforman las superficies de la esfera y el esferoide se ha seguido la misma metodología de representación que en el caso

del error de orientación. La Figura 5 representa dichos errores de orientación en la superficie de la esfera y el esferoide.

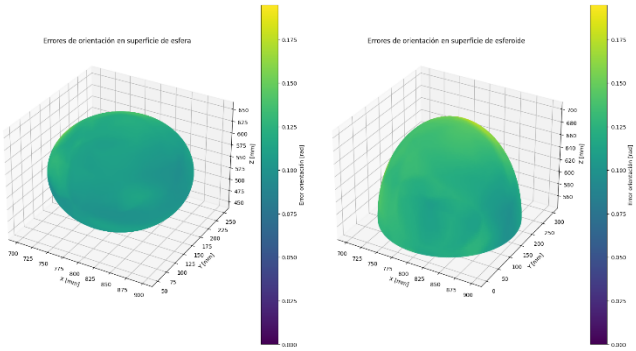

Figura 5: Error de orientación en puntos de la superficie en una esfera y esferoide

En ambos mapas esféricos predominan en su gran mayoría tonos verdosos (error de orientación aproximado de 0,15 rad), sin zonas de fallo sistemáticas.

5. Conclusiones

El agente DDPG implementado demuestra que el aprendizaje por refuerzo profundo puede sustituir al cálculo iterativo del Jacobiano en el MCI de manipuladores colaborativos, alcanzando tiempos de inferencia medios de 0,139 ms (beneficiosos para aplicaciones en tiempo real) y tasas de éxito superiores al 98 % sin sacrificar precisión con mayormente errores posicionales $\lesssim$ 12 mm y errores angulares $\lesssim$ 0,15 rad. Al representar la política de control con una red neuronal, se consigue una solución que elimina la necesidad de bucles de refinamiento en cada paso, lo que facilita su integración en sistemas con frecuencia de control elevada.

Además, el enfoque basado en DDPG con replay buffer, HER y normalización ha mostrado buena estabilidad durante el entrenamiento en 27 subespacios del volumen de trabajo, y ha sido capaz de generalizar con eficacia tanto en superficies esféricas como esferoidales. Estos resultados sugieren que el método no solo es rápido, sino también robusto frente a variaciones geométricas del objetivo, lo cual es crucial para tareas de manipulación en entornos de trabajo como aeropuertos o supermercados, los cuales se alinean con los casos de uso del proyecto MANiBOT.

Para afianzar su aplicabilidad, el siguiente paso consiste en trasladar este esquema al robot físico GOFA 5_95, incorporando modelos dinámicos y variaciones de carga durante el entrenamiento. Paralelamente, se explorarán algoritmos de refuerzo multiobjetivo que permitan optimizar simultáneamente latencia, precisión y eficiencia energética, así como técnicas de transferencia y meta-aprendizaje para adaptar la política a distintos manipuladores y escenarios emergentes con mínima reentrenamiento. De este modo, se avanzará hacia agentes de control por RL capaces de operar con seguridad y fiabilidad en líneas de producción y entornos colaborativos heterogéneos.

Agradecimientos

Este trabajo ha sido realizado parcialmente gracias al apoyo de la Comisión Europea en el marco del proyecto europeo MANiBOT, número de referencia 101120823.

Referencias

Aristidou, A., Lasenby, J., Kyriazis, N., 2018. Inverse kinematics techniques in computer graphics: A survey. *Computer Graphics Forum* 37(6), 305–329. DOI: 10.1111/cgf.13317

Bayona, E., Sierra-García, J. E., Santos, M., & Mariolis, I., 2024. In search of the best fitness function for optimum generation of trajectories for Automated Guided Vehicles. *Engineering Applications of Artificial Intelligence*, 133, 108440.

Bogue, R., 2018. The rise of the service robot. *Industrial Robot: An International Journal* 45(5), 1–7. DOI: 10.1108/IR-02-2018-0023

Chen, L., Pan, H., Mao, H. L., 2013. Real-Time Solution of Inverse Kinematics for 6R Robot Manipulators Using Digital Signal Processor. *Applied Mechanics and Materials* 423-426, 2788–2791. DOI: 10.4028/www.scientific.net/amm.423-426.2788

Chiacchio, P., Chiaverini, S., 1995. Coping with joint velocity limits in first-order inverse kinematics algorithms: analysis and real-time implementation. *Robotica* 13(5), 515–519. DOI: 10.1017/S026357470001835X

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2016. Continuous control with deep reinforcement learning. *arXiv preprint* arXiv:1509.02971.

Manibot, 2025. Advancing the physical intelligence and performance of roBOTs towards human-like bi-manual objects MANipulation. https://manibot-project.eu/

Pizzolato, C., Reggiani, M., Modenese, L., Lloyd, D. G., 2016. Real-time inverse kinematics and inverse dynamics for lower limb applications using OpenSim. *Computer Methods in Biomechanics and Biomedical Engineering* 20(4), 436–445. DOI: 10.1080/10255842.2016.1240789

Rapetti, L., Tirupachuri, Y., Darvish, K., Dafarra, S., Nava, G., Latella, C., Pucci, D., 2020. Model-Based Real-Time Motion Tracking Using Dynamical Inverse Kinematics. *Algorithms* 13(10), 266. DOI: 10.3390/a13100266

Sierra-García, J. E., & Santos, M., 2021. Control of industrial AGV based on reinforcement learning. *In 15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020)* 15 (pp. 647-656). Springer International Publishing.

Sierra-García, J. E., Fernández-Rodríguez, V., Santos, M., & Quevedo, E., 2023. Development and experimental validation of control algorithm for person-following autonomous robots. *Electronics*, 12(9), 2077.

Tolani, D., Goswami, A., Badler, N. I., 2000. Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs. *Graphical Models* 62(5), 353–388. DOI: 10.1006/gmod.2000.0528

Zhang, L., Chen, X., Wang, Y., Zhu, Z., 2024. Real-time optimized inverse kinematics of redundant robots under inequality constraints. *Scientific Reports* 14, 29754. DOI: 10.1038/s41598-024-29754