CEA Comité Español de Automática
XLV JORNADAS AUTOMÁTICA MÁLAGA 2024
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Optimización de Caminata con Aprendizaje por Refuerzo en Humanoide TEO

Mas, J.[1], Victores, J. G., Balaguer, C.

RoboticsLab, Dpto. de Ing. Sistemas y Automática, Universidad Carlos III de Madrid, Av. de la Universidad, nº 30, 28911 Leganés, Madrid, España.

To cite this article: Mas, J., Victores, J. G., Balaguer, C. 2024. Reinforcement Learning with the humanoid TEO. Jornadas de Automática, 45. https://doi.org/10.17979/ja-cea.2024.45.10950

Resumen

En los últimos años, el aprendizaje por refuerzo en entornos de simulación robótica ha emergido como una herramienta valiosa para entrenar plataformas robóticas en la ejecución de tareas complejas, como la marcha. El aprendizaje por refuerzo permite al robot descubrir un camino viable para realizar una tarea previamente definida, eliminando la necesidad de una programación exhaustiva y un control detallado de los movimientos. El propósito de este trabajo es mostrar la implementación de algoritmos de aprendizaje por refuerzo con el objetivo de conseguir que nuestro modelo del robot humanoide TEO aprenda a caminar sin necesidad de programar un controlador de manera explicita. Este artículo incluye como se ha desarrollado el modelo del humanoide, que medida de aprendizaje se ha desarrollado y que algoritmos se han implementado durante el entrenamiento, así como los resultados que se han obtenido de este entrenamiento.

Palabras clave: Aprendizaje por refuerzo y aprendizaje profundo en control, Aprendizaje automático en modelado, predicción, control y automatización, Guía, navegación y control, Arquitectura del software de control, Sistemas robóticos autónomos

Optimization of Walking with Reinforcement Learning in Humanoid TEO

Abstract

In recent years, reinforcement learning in robotic simulation environments has emerged as a valuable tool for training robotic platforms to perform complex tasks, such as walking. Reinforcement learning allows the robot to discover a viable path to perform a predefined task, eliminating the need for exhaustive programming and detailed control of movements. The purpose of this work is to demonstrate the implementation of reinforcement learning algorithms with the aim of enabling our humanoid robot model TEO to learn to walk without the need for explicit controller programming. This paper includes the development of the humanoid model, the learning metric developed, and the algorithms implemented during training, as well as the results obtained from this training.

Keywords: Reinforcement learning and deep learning in control, Machine learning in modelling, prediction, control and automation, Guidance navigation and control, Control software architecture, Autonomous robotic systems

1. Introducción

La coordinación y eficiencia en la ejecución de tareas por parte de robots continúan siendo desafíos complejos en la robótica. Los robots humanoides, en particular, enfrentan mayores dificultades debido a su estructura intrincada y la necesidad de imitar movimientos humanos.

El robot humanoide TEO, con una altura de 1,65 metros y 28 grados de libertad, ilustra estos desafíos. La programación y el control de los movimientos necesarios para realizar cualquier tarea con este robot son extremadamente complejos. Esta complejidad se debe, en parte, al alto número de grados de libertad y a la dificultad inherente en la coordinación y estabilidad de todos los movimientos del humanoide.

En el artículo de (Radosavovic et al., 2024) se presenta un enfoque completamente basado en el aprendizaje para la lo-

*Autor para correspondencia: sanmasja@gmail.com

comoción de robots humanoides en el mundo real. El controlador propuesto es un transformador causal que toma como entrada el historial de observaciones proprioceptivas y acciones y predice la siguiente acción. Este modelo se entrena mediante aprendizaje por refuerzo sin modelo a gran escala en un conjunto de entornos aleatorios en simulación y se despliega en el mundo real sin ajustes adicionales. Los resultados obtenidos muestran que el controlador es capaz de caminar sobre diversos terrenos al aire libre, es robusto ante perturbaciones externas y puede adaptarse en contexto.

La locomoción de robots humanoides ha sido históricamente abordada mediante métodos de control clásico, que aunque han mostrado resultados impresionantes en varios escenarios, presentan dificultades para generalizar y adaptarse a nuevos entornos. En contraste, los métodos basados en aprendizaje han demostrado ser efectivos en tareas de manipulación y locomoción de robots cuadrúpedos y bipedales. Los enfoques previos incluyen el uso de redes neuronales recurrentes como LSTM y redes convolucionales temporales (TCN) para estimar propiedades del entorno y mejorar la adaptación en tiempo real.

El enfoque de Radosavovic et al. destaca por su capacidad de adaptarse dinámicamente durante la prueba, sin necesidad de ajustar los pesos del modelo, basándose únicamente en el historial de observaciones y acciones1. Este tipo de aprendizaje en contexto es similar al encontrado en modelos transformadores grandes como GPT-3. El modelo propuesto se valida en un robot humanoide de tamaño completo y demuestra una locomoción confiable en exteriores, robustez ante perturbaciones externas y la capacidad de llevar cargas de distintas masas.

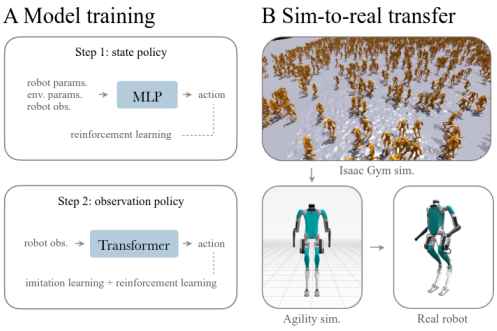

Figura 1: Proceso de entrenamiento y transferencia de conocimiento (Radosavovic et al., 2024).

Estos avances sugieren que los controladores basados en aprendizaje simples y generales son capaces de manejar sistemas humanoides complejos y de alta dimensión en el mundo físico, lo que fomenta nuevas investigaciones en enfoques de aprendizaje escalables para la robótica humanoide.

1.1. MuJoCo

El entorno de simulación empleado durante el proceso de aprendizaje ha sido *MuJoCo* (Todorov et al., 2012). Este simulador de código abierto permite la simulación de modelos predefinidos en archivos XML dentro de un entorno estructurado. Los modelos son altamente configurables, lo que permite definir cada componente del robot de manera precisa y realista. Aunque existen varias diferencias entre estos dos formatos, la principal distinción con respecto a los modelos URDF, utilizados en otros simuladores como Gazebo, radica en la ausencia de las etiquetas parent y child. En consecuencia, la herencia entre articulaciones y elementos del modelo se organiza en una estructura arbórea, donde del nodo raíz (en nuestro caso, la cadera) emergen el resto de las articulaciones como hojas de un árbol.

Durante el proceso de aprendizaje, hemos empleado un modelo simplificado del humanoide TEO. Este modelo conserva la estructura del robot real, incluidos los grados de libertad, la disposición y los pesos. La estrategia consiste en utilizar este modelo simplificado durante las fases iniciales del aprendizaje, y luego incrementar progresivamente su complejidad en función de los resultados obtenidos. De este modo, se pretende llegar a trabajar con un modelo que sea completamente fiel al robot real.

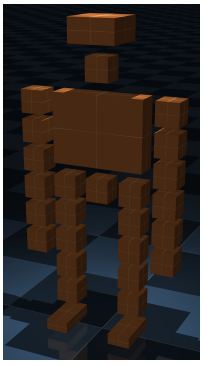

Figura 2: Modelo de TEO en el simulador MuJoCo.

1.2. Aprendizaje por refuerzo

Durante el proceso de aprendizaje, se ha empleado el framework *RL Baselines3 Zoo* (Raffin et al., 2021), el cual proporciona un conjunto de implementaciones de algoritmos de aprendizaje por refuerzo.

Utilizando el modelo del humanoide TEO en MuJoCo, se han aplicado diversos algoritmos disponibles en el framework, ajustando los hiperparámetros hasta optimizar la tarea definida por la función de aprendizaje. La elección de los algoritmos se basó en su capacidad para manejar la complejidad de la dinámica del humanoide y en la eficiencia computacional, factores cruciales para garantizar resultados precisos y reproducibles. Se llevaron a cabo múltiples iteraciones de entrenamiento y evaluación para refinar el rendimiento del modelo, considerando métricas clave como la estabilidad, la eficiencia energética y la naturalidad de los movimientos generados.

La tarea principal abordada ha sido la marcha, con el objetivo de que el modelo del humanoide TEO aprenda a caminar de manera similar a un humano. Para ello nos hemos basado en el trabajo hecho con el modelo *Humanoid de Gym*, (Hugging Face, Year) (de Lazcano et al., 2023), adaptando técnicas y estrategias probadas para el control motor complejo. El proceso incluyó la definición de recompensas específicas orientadas a fomentar patrones de marcha natural, la evitación de caídas y la minimización del consumo de energía durante el desplazamiento. Además, se implementaron mecanismos de regularización para prevenir el sobreajuste y garantizar que el modelo generalice adecuadamente a diferentes escenarios de marcha.

La combinación de RL Baselines3 Zoo y MuJoCo nos ha permitido desarrollar un modelo de entrenamiento robusto para el humanoide TEO, demostrando la eficacia de los algoritmos de aprendizaje por refuerzo en la resolución de tareas complejas en robótica. Este enfoque no solo facilita avances en la locomoción de robots humanoides, sino que también sienta las bases para futuras investigaciones en control adaptativo y aprenMuJoCodizaje autónomo en sistemas robóticos avanzados.

2. Metodología

Para lograr que el humanoide TEO aprenda a caminar utilizando algoritmos de aprendizaje por refuerzo, es esencial una correcta descripción del modelo. En esta sección se detallan los aspectos clave del diseño del modelo, la configuración del entorno de simulación y el proceso de entrenamiento, destacando cómo se han aplicado y adaptado técnicas avanzadas para optimizar el rendimiento del robot en la tarea de caminar.

2.1. Descripción del modelo

Uno de los elementos clave que más influye en el proceso de aprendizaje de tareas como la caminata es la correcta descripción del modelo sobre el cual se inferirá conocimiento. En este estudio, se ha utilizado un modelo simplificado del humanoide TEO, representando al robot mediante formas básicas rectangulares. Cada articulación se ha modelado como un cubo de 1 m³, excepto la cabeza, el pecho y los pies, que tienen formas rectangulares para aproximarse mejor al modelo real del robot.

Cada una de estas articulaciones ha sido definida utilizando el formato XML propio de MuJoCo, por lo que se ha construido este modelo desde cero, siguiendo el esquema de articulaciones de TEO (UC3M, 2024). Las articulaciones están descritas de acuerdo con el manual de desarrollo de TEO, permitiendo que cada una se mueva en el plano frontal o sagital según corresponda.

Se ha asegurado que las propiedades físicas del modelo, como la masa y la inercia, se correspondan con las del robot real. Esto es crucial para que las dinámicas de movimiento observadas en la simulación sean representativas del comportamiento del robot en el mundo real. Cada junta y actuador ha sido cuidadosamente parametrizado para reflejar las capacidades y limitaciones de los componentes físicos de TEO, incluyendo la potencia de los motores y los límites de los ángulos de las articulaciones.

La correcta descripción del modelo no solo implica una representación geométrica precisa, sino también la inclusión de detalles mecánicos y dinámicos que afectan significativamente el rendimiento del aprendizaje. El uso de un modelo detallado y realista permite que los algoritmos de aprendizaje por refuerzo generen políticas de control que sean aplicables y eficientes cuando, en un futuro, se transfieren del entorno de simulación a la implementación en el robot físico.

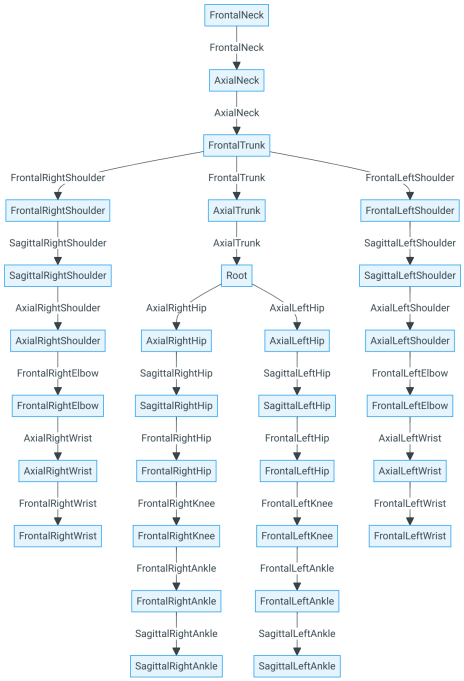

Figura 3: Diagrama de articulaciones de TEO.

La detallada y precisa descripción del modelo de TEO en MuJoCo ha sido un factor fundamental durante del entrenamiento con algoritmos de aprendizaje por refuerzo, permitiendo simular y optimizar de manera efectiva las capacidades de locomoción del robot humanoide.

2.2. Diseño de la función de recompensa

Para llevar a cabo el proceso de aprendizaje, es fundamental definir un entorno en MuJoCo que detalle la función de aprendizaje a minimizar, los pasos que debe seguir el modelo para avanzar en la función, y los tipos de recompensas que debe recibir el modelo al realizar una acción determinada según el resultado.

El entorno de MuJoCo utilizado es *Humanoid-v4*, empleado para entrenar al modelo Humanoid Gym en la tarea de caminar. Este entorno evalúa la caminata del robot basándose en su desplazamiento a lo largo del eje x. Para ello, se calcula el centro de masa del modelo antes y después de cada iteración, obteniendo el coeficiente de desplazamiento en el eje x del modelo, de la siguiente manera:

Sea m_i la masa de la i-ésima parte del cuerpo y $\mathbf{x}_i = (x_i, y_i, z_i)$ la posición de la i-ésima parte del cuerpo. El centro de masa $\mathbf{R}_{CM} = (X_{CM}, Y_{CM}, Z_{CM})$ en tres dimensiones se calcula como:

$$\mathbf{R}_{CM} = \frac{\sum_i m_i \mathbf{x}_i}{\sum_i m_i}$$

Dado que la función devuelve solo las componentes X e Y del centro de masa, la fórmula específica en el plano XY es:

$$(X_{CM}, Y_{CM}) = \left(\frac{\sum_i m_i x_i}{\sum_i m_i}, \frac{\sum_i m_i y_i}{\sum_i m_i} \right)$$

Ya que es vital que el modelo de TEO no se caiga, se añade una nueva recompensa de salud que varía según la posición sobre el eje z del modelo completo del robot.

El reward puede expresarse como:

$$\text{is\_healthy}(z) = \begin{cases} 2.25 & \text{si } z_{\text{mín}} < z < z_{\text{máx}} \\ 0 & \text{si no} \end{cases}$$

Donde: - z es la coordenada z de la posición, es decir, z = self.data.qpos[2]. - $z_{\text{mín}}$ es el valor mínimo del rango saludable. - $z_{\text{máx}}$ es el valor máximo del rango saludable.

Durante las pruebas de entrenamiento, observamos que el modelo encontraba un mínimo local del cual no era capaz de salir, estableciéndose en una posición de estabilidad sin avanzar. Esto se debía a que la recompensa por supervivencia era suficiente para ignorar la recompensa por avance en el eje xx. Por esta razón, fue necesario modificar y personalizar los costos de las recompensas para nuestro modelo.

Para asegurar que el modelo aprendiera a caminar de una manera similar a la humana, se fijaron determinadas articulaciones, especialmente las de los brazos, con el objetivo de evitar posiciones no naturales. El **head_height_reward**, a diferencia del **is_healthy**, contribuye a que el modelo aprenda a caminar de manera erguida. Las pruebas realizadas sin este reward mostraron que el modelo avanzaba en el eje xx adoptando una postura cuadrúpeda, cumpliendo así con el objetivo de is_healthy. Sin embargo, con el head_height_reward, se obliga al modelo a mantener la cabeza a una altura determinada mientras se desplaza, logrando así una marcha bípeda.

$$\text{head\_height\_reward}(z) = \begin{cases} 2{,}0 & \text{si } z_{\text{mín}} < z < z_{\text{máx}} \\ 0{,}0 & \text{si no} \end{cases}$$

De esta manera, estas tres recompensas forman la función de aprendizaje que el modelo debe minimizar a lo largo del período de aprendizaje. Las variables involucradas son las siguientes:

- $\mathbf{r}_{\text{before}} = (x_{\text{before}}, y_{\text{before}})$: posición del centro de masa antes de la acción.
- $\mathbf{r}_{\text{after}} = (x_{\text{after}}, y_{\text{after}})$: posición del centro de masa después de la acción.
- Δt: intervalo de tiempo.
- $\mathbf{v} = (v_x, v_y)$: velocidad en el plano XY.
- ctrl_cost: costo de control.
- forward_reward: recompensa por avance.
- healthy_reward: recompensa por estar saludable.
- head_height_reward: recompensa por mantenerse erguido.
- $\mathbf{a}$: acción aplicada.
- w_f: peso de la recompensa de avance (4.0).

La velocidad en el plano XY del centro de masa se calcula como:

$$\mathbf{v} = \frac{\mathbf{r}_{\text{after}} - \mathbf{r}_{\text{before}}}{\Delta t}$$

Descomponiendo en componentes x e y:

$$v_x = \frac{x_{\text{after}} - x_{\text{before}}}{\Delta t}$$

$$v_y = \frac{y_{\text{after}} - y_{\text{before}}}{\Delta t}$$

La recompensa de avance (forward reward) se calcula como:

$$\text{forward\_reward} = w_f \cdot v_x$$

La recompensa total se calcula como la suma de la recompensa de avance y la recompensa por estar saludable:

rewards = forward_reward+healthy_reward+head_height_reward

2.3. *Entrenamiento*

Como ya se ha mencionado, para el entrenamiento se ha utilizado el framework *RL Baselines3 Zoo* (Raffin, 2020) y, por ende, los algoritmos que este tiene implementados. Para seleccionar el algoritmo que mejor optimice el problema de la caminata en nuestro modelo, es fundamental analizar el tipo de problema que se presenta. Hacer que un robot humanoide aprenda a caminar utilizando algoritmos de aprendizaje por refuerzo es un problema de control continuo. Para resolverlo, el modelo dispone de un rango de acciones conocido como "action space". Este conjunto de acciones representa las fuerzas aplicadas a cada articulación; así, a mayor número de grados de libertad en nuestro modelo, mayor será el "action space y, por tanto, la complejidad computacional del problema.

Asimismo, el algoritmo que utilicemos debe lograr un equilibrio entre la exploración de nuevas estrategias de movimiento y la explotación de estrategias ya aprendidas, con el fin de optimizar su rendimiento.

Tras analizar el tipo de problema al que nos enfrentamos, decidimos explorar el entrenamiento del modelo con tres algoritmos diferentes: *SAC (Soft Actor-Critic)* (Haarnoja et al., 2018), *TD3 (Twin Delayed DDPG)* (Dankwa and Zheng, 2019) y *PPO (Proximal Policy Optimization)* (Schulman et al., 2017). Durante el entrenamiento, observamos el progreso comparando la *media del aprendizaje de exploración*. Este valor nos indica la media de la recompensa calculada a lo largo de una serie de episodios de simulación; por tanto, cuanto mayor sea este valor, mejor responderá nuestro modelo al problema.

Los resultados obtenidos durante el entrenamiento fueron los siguientes:

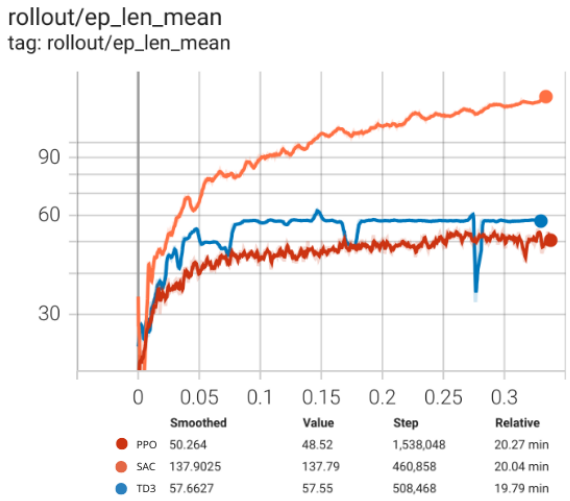

Figura 4: Promedio de la longitud de los episodios *ep_len_mean* durante el entrenamiento de los algoritmos SAC, PPO y TD3.

Esta gráfica muestra la evolución del promedio de la longitud de los episodios *ep_len_mean* durante el entrenamiento de los tres algoritmos. En el eje X se representan los pasos de entrenamiento, mientras que en el eje Y se muestra la longitud media del episodio. El algoritmo *SAC* (naranja) se estabiliza en una longitud media de episodio más alta que *PPO* (rojo) y *TD3* (azul), lo que indica una mayor eficiencia en términos de mantener episodios más largos. Esto sugiere que el agente entrenado con *SAC* es más estable y capaz de sobrevivir más tiempo en el entorno. En contraste, *PPO* tiene un rendimiento significativamente menor, mientras que *TD3* supera a *PPO* pero no alcanza la eficiencia de *SAC*.

Dado que los resultados obtenidos muestran que el algoritmo *Soft Actor-Critic* (SAC) ofrece un rendimiento superior en comparación con otros algoritmos evaluados, hemos decidido centrarnos exclusivamente en su implementación. A continuación, se detallan los componentes y pasos clave para la implementación del algoritmo SAC.

2.3.1. *Soft Actor-Critic (SAC)*

Desarrollado en 2018 por miembros de la Universidad de Berkeley en colaboración con Google, *SAC* es considerado actualmente uno de los algoritmos de aprendizaje por refuerzo más eficientes en el campo de la robótica. Este algoritmo se basa en el aprendizaje por refuerzo de entropía máxima, donde el objetivo es encontrar la política óptima que maximice la recompensa esperada a largo plazo y la entropía a largo plazo, definido por la siguiente ecuación:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot \mid s_t)) \right].$$

- $J(\pi)$: Función objetivo de la política π.
- t: Paso temporal.
- T: Horizonte temporal.
- $\mathbb{E}_{(s_t,a_t)\sim\rho_\pi}$: Esperanza sobre la distribución de estados s_t y acciones a_t siguiendo la política π.
- $r(s_t, a_t)$: Recompensa obtenida al realizar la acción a_t en el estado s_t.
- α: Coeficiente de ponderación de la entropía.
- $\mathcal{H}(\pi(\cdot \mid s_t))$: Entropía de la política π en el estado s_t.
- ρ_π: Distribución de probabilidad inducida por la política π.

3. Resultados

El proceso de entrenamiento del modelo tiene la siguiente forma:

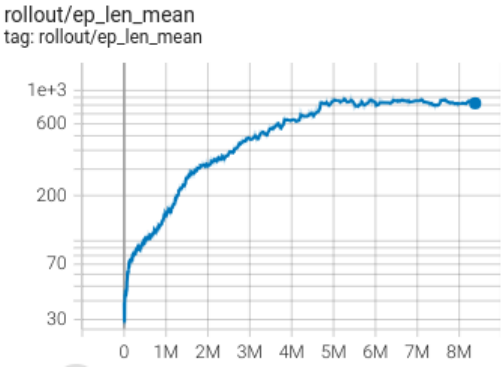

Figura 5: Entrenamiento con SAC.

Al comienzo del entrenamiento (0 a 1 millón de pasos), se observa un rápido incremento en la longitud media de los episodios, indicando que el modelo está aprendiendo rápidamente a mantenerse en pie y moverse de manera más estable. Entre 1 y 3 millones de pasos, la gráfica sigue mostrando una tendencia ascendente, aunque a un ritmo más lento, sugiriendo que el modelo sigue mejorando, pero a medida que se enfrenta a desafíos más complejos en la tarea de caminar, el progreso se vuelve más gradual. A partir de los 3 millones de pasos y hasta los 8 millones, la gráfica muestra una tendencia a estabilizarse alrededor de una longitud media de episodios superior a 1000, lo que indica que el modelo ha alcanzado un punto de rendimiento consistente, donde las mejoras adicionales son mínimas y el modelo puede caminar de manera estable durante largos períodos.

El algoritmo *SAC* ha demostrado ser efectivo en el entrenamiento del modelo para la tarea de caminar del robot humanoide. La longitud media de los episodios aumenta significativamente durante las primeras fases del entrenamiento y se estabiliza en un alto nivel, lo que sugiere que el robot ha aprendido a caminar de manera efectiva y sostenida.

Una vez entrenado el modelo con el algoritmo *SAC*, es momento de comprobar cómo ha aprendido nuestro modelo de TEO a resolver el problema planteado. Para ello, se lanza la simulación con los valores de control que han maximizado la función de aprendizaje dada. Estos valores se guardan en un archivo zip que, al pasárselos al modelo en simulación, hacen que el modelo se comporte como muestra la figura 6, enlace a vídeo.

Figura 6: Secuencia de la caminata de TEO.

Como se puede observar en la imagen superpuesta, el modelo es capaz de caminar. Para ello, necesita valerse de los brazos para mantener el equilibrio mientras avanza con las rodillas flexionadas y los pies. En esta simulación, el modelo es capaz de avanzar, en la mayoría de los casos, indefinidamente a lo largo del eje x sin caerse ni perder el equilibrio en ningún momento.

Otra prueba realizada fue entrenar al modelo para que fuera capaz de caminar hacia atrás, obteniendo los resultados de la figura 7, enlace a vídeo.

Figura 7: Secuencia de la caminata hacia atrás de TEO.

Al igual que cuando avanza hacia adelante, aquí el modelo sigue valiendo de los brazos para mantenerse en equilibrio y utiliza las rodillas y pies para desplazarse. A diferencia del modelo anterior, en este caso se observó que, a lo largo del tiempo, el modelo no es capaz de mantenerse indefinidamente de pie. Esto se debe al error acumulado durante la caminata. Para solucionar esto, necesitaríamos más tiempo de simulación para refinar el modelo.

Cabe destacar que la función de aprendizaje es vital para que el modelo aprenda a realizar la tarea de manera "normal" ya que de lo contrario nos podemos encontrar con casos como el de la figura 8, en el que el modelo aprende a caminar, pero no de la manera en que nosotros deseamos, enlace a vídeo.

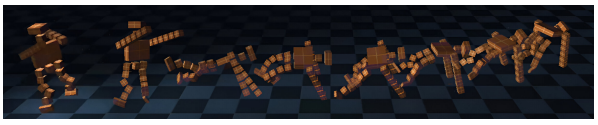

Figura 8: Secuencia de un modo de "caminar" de TEO.

Esto se debía en parte a que el centro de masas de la cabeza no tenía relevancia en los primeros entrenamientos.

4. Conclusiones

El problema de la marcha en robots humanoides sigue siendo un desafío complejo sin solución generalizada. La implementación efectiva es crucial, dado su potencial para tareas en entornos humanos, desde asistencia en el hogar hasta rescate.

El aprendizaje por refuerzo ofrece una alternativa prometedora a los métodos de control clásico, pero introduce desafíos como la calibración precisa de hiperparámetros y costos específicos para cada modelo, limitando su generalización.

Además, definir matemáticamente la tarea presenta otro reto. Aunque mover al robot en el eje x es sencillo, especificar cómo debe caminar es complicado. Durante los experimentos, se observó que sin una función de costo adecuada, el robot avanzaba de formas no deseadas, como arrastrándose o saltando. Esto subraya la importancia de definir el problema con precisión.

En este trabajo, logramos que el humanoide TEO camine de manera "humana"mediante múltiples entrenamientos y ajustes de parámetros y costos, encontrando un conjunto que generaliza bien el problema.

El siguiente desafío es la implementación en el robot real, una transición crítica aún no completamente resuelta. Las simulaciones no capturan todas las complejidades del mundo real, como imperfecciones del hardware y variaciones ambientales. Transferir un modelo de simulación a un robot físico implica adaptar el modelo a sensores y actuadores reales y gestionar la incertidumbre en el entorno físico.

Agradecimientos

La investigación que ha conducido a estos resultados ha recibido financiación del proyecto COMPANION-CM: Inteligencia artificial y modelos cognitivos para la interacción simétrica humano-robot en el ámbito de la robótica asistencial, con referencia Y2020/NMT-6660, financiado por Proyectos Sinérgicos de I+D la Comunidad de Madrid.

Referencias

Dankwa, S., Zheng, W., 08 2019. Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. pp. 1–5.
DOI: 10.1145/3387168.3387199

de Lazcano, R., Andreas, K., Tai, J. J., Lee, S. R., Terry, J., 2023. Gymnasium robotics.
URL: http://github.com/Farama-Foundation/Gymnasium-Robotics

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.
URL: https://openreview.net/forum?id=HJjvxl-Cb

Hugging Face, Year. TQC-Humanoid-v3 Trained Model. Hugging Face Model Hub, retrieved from: https://huggingface.co/sb3/tqc-Humanoid-v3.

Radosavovic, I., Xiao, T., Zhang, B., Darrell, T., Malik, J., Sreenath, K., 2024. Real-world humanoid locomotion with reinforcement learning. Science Robotics 9 (89), eadi9579.
URL: https://www.science.org/doi/abs/10.1126/scirobotics.adi9579
DOI: 10.1126/scirobotics.adi9579

Raffin, A., 2020. Rl baselines3 zoo. https://github.com/DLR-RM/rl-baselines3-zoo.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N., 2021. Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research 22 (268), 1–8.
URL: http://jmlr.org/papers/v22/20-1364.html

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms.

Todorov, E., Erez, T., Tassa, Y., 2012. Mujoco: A physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 5026–5033.
DOI: 10.1109/IROS.2012.6386109

UC3M, R., 2024. Teo developer manual. https://github.com/roboticslab-uc3m/teo-developer-manual, accessed: 2024-05-30.