Simposio CEA de Robótica, Bioingeniería, Visión Artificial y Automática Marina 2025

RBVM Simposio CEA de Robótica, Bioingeniería, Visión por Computador y Automática Marina

Diseño de recompensas mediante LLMs para tareas de manipulación robótica

Fernández-Herrero, M.[1,*], Puente, S.T.[1], Páez-Ubieta, I.d.L.[1]

AUtomatics, RObotics, and Artificial Vision Lab. University Institute for Computer Research. University of Alicante. San Vicente, Spain.

Resumen

El diseño manual de funciones de recompensa para el Aprendizaje por Refuerzo (RL) en robótica es complejo y propenso a errores. Este trabajo investiga la automatización de dicho proceso mediante Grandes Modelos de Lenguaje (LLMs), ampliando la metodología *Eureka*. Se evaluó la capacidad de diversos LLMs del mercado, más allá de GPT-4 y GPT-3.5 estudiados en el trabajo original, para generar recompensas en tres tareas de manipulación robótica con las manos *Shadow* y *Allegro* en el simulador *Isaac Gym*. Los resultados muestran que los LLMs, especialmente modelos recientes y aquellos con razonamiento *Chain-of-Thought*, pueden superar las recompensas diseñadas por expertos humanos en el 100 % de las tareas evaluadas, logrando éxito en entornos de alta complejidad como *pen spinning* con la *Allegro Hand*. Modelos como O1 y algunas variantes de Claude destacan significativamente. El estudio confirma el gran potencial de los LLMs para optimizar el diseño de recompensas en RL aplicado a la realización de tareas complejas en robótica.

Palabras clave: Aprendizaje por Refuerzo, Robots inteligentes, Grandes Modelos de Lenguaje, Diseño de Recompensas, Cadena de Razonamiento

Reward design via LLMs for robot manipulation tasks

Abstract

Manual reward design for Reinforcement Learning (RL) in robotics is both complex and error-prone. This work examines automating that process through Large Language Models (LLMs), extending the *Eureka* methodology. We assess various commercial LLMs, beyond GPT-4 and GPT-3.5 from the original study, to generate rewards in three robotic manipulation tasks involving the *Shadow* and *Allegro* hands in the *Isaac Gym* simulator. Findings indicate that LLMs, especially recent models and those with *Chain-of-Thought* reasoning, outperform human-engineered rewards in 100 % of the tasks tested, achieving success in high-complexity settings such as *pen spinning* with the *Allegro Hand*. Models like O1 and certain Claude variants stand out. The study confirms the strong potential of LLMs to optimize reward design for RL in complex robotics.

Keywords: Reinforcement Learning control, Intelligent robotics, Large Language Models, Reward Design, Chain-of-Thought

1. Introducción

En los últimos años, la influencia de la Inteligencia Artificial (IA), entendida como el campo científico dedicado al estudio y diseño de agentes inteligentes, capaces de percibir su entorno y actuar racionalmente para maximizar sus posibilidades de éxito (Russell and Norvig, 2010, cap. 2), se ha expandido de manera acelerada en ámbitos tan diversos como la medicina, la industria y el transporte, demostrando un gran potencial de cambio que afecta también a la vida cotidiana (Stone, 2021). En este contexto, la robótica ocupa una posición destacada, ya que dotar a las máquinas de percepción, razonamiento y acción autónoma para adaptarse a entornos cambiantes ha sido una aspiración histórica de este campo (Russell and Norvig, 2010, cap. 25). Sin embargo, pese a los grandes avances en control, diseño mecánico y planificación de tareas, la creación de este tipo de sistemas robóticos continúa presentando desafíos a considerar, motivados en parte por la alta dimensionalidad de los entornos, la incertidumbre sensorial y la necesidad de garantizar robustez y seguridad (Siciliano and Khatib, 2016). Esto se debe a que la complejidad intrínseca de la robótica supera ampliamente la posibilidad de programar todas las situaciones de antemano; se requiere, por el contrario, dotar a los robots de in-

*Autor para correspondencia: mfh13@alu.ua.es

teligencia cognitiva que les permita anticiparse y adaptarse de forma autónoma a condiciones cambiantes.

Para abordar esta necesidad de aprendizaje autónomo y adaptativo, el Aprendizaje por Refuerzo (RL) se erige como un paradigma fundamental, al plantear que un agente, en este caso, el robot, aprenda a maximizar recompensas acumuladas por sí solo mediante su interacción con el entorno (Sutton and Barto, 2018) o mediante la utilización de demostraciones de la tarea a realizar (Frau-Alfaro et al., 2024a). Aun así, los éxitos del RL en robótica siguen siendo relativamente aislados fuera del ámbito académico, en parte por dos problemas principales: primero, el elevado coste y los riesgos de que el agente falle en un entorno real, a diferencia de un videojuego (Dulac-Arnold et al., 2021); y segundo, la cuidadosa definición de la función de recompensa, la cual indica al agente qué acciones conducen a lograr los objetivos. En entornos de alta complejidad, diseñar recompensas efectivas mediante ensayo y error puede ser muy costoso y, además, conlleva el riesgo de comportamientos imprevistos si se omiten restricciones o condiciones esenciales, lo que se conoce como *reward hacking* (Ng et al., 1999). Para mitigar estos riesgos y agilizar la fase de ingeniería de recompensas, han surgido enfoques que aprovechan la potencia de los Grandes Modelos de Lenguaje (LLMs) para automatizar la generación de dichas funciones. Esto es debido a que, a día de hoy, la mayoría de los investigadores y profesionales de RL no siguen un comportamiento unificado en el diseño de recompensas, sino que las obtienen a base de prueba y error (Booth et al., 2023; Frau-Alfaro et al., 2024b).

La idea de emplear LLMs para asistir a tareas robóticas mediante RL no es nueva. Inicialmente, modelos menos avanzados como *GPT-2* o *GPT-3*, fueron utilizados como planificadores jerárquicos capaces de descomponer tareas complejas en secuencias de pasos más sencillos. Por ejemplo, abrir un cajón (Brohan et al., 2022; Singh et al., 2023). Con la llegada de LLMs más sofisticados, esta metodología se ha extendido considerablemente, permitiendo abordar tareas más complejas dentro del ámbito del RL y ampliando notablemente su funcionalidad, como la creación de modelos del mundo especializados en tareas robóticas (Brohan et al., 2023; Driess et al., 2023; Guruprasad et al., 2024; Wang et al., 2024).

Uno de los primeros trabajos destacados en utilizar LLMs para generar funciones de recompensa es *Language to Rewards for Robotic Skill Synthesis* (Yu et al., 2023). Este estudio emplea un modelo basado en GPT-4 que convierte instrucciones generales, como "haz que el perro robot se ponga sobre dos patas", en funciones de recompensa parametrizables. El método consta de dos etapas: primero, interpreta semánticamente la instrucción (por ejemplo, "el torso debe mantenerse vertical"); para luego, transformar esta interpretación en una función de recompensas mediante plantillas predefinidas con parámetros específicos. Aunque los resultados son favorables, la dependencia de plantillas limita la generalización, ya que cada nueva tarea requiere crear nuevas plantillas de forma manual.

A partir de esta limitación, surgieron propuestas para iterar y refinar las recompensas aprovechando la capacidad intrínseca del LLM para generar y evaluar funciones de recompensa, eliminando la dependencia de plantillas predefinidas. Bajo este enfoque, el LLM inicialmente propone una recompensa, se entrena al agente con ella, se evalúa el desempeño obtenido y, finalmente, se ajusta la recompensa según los resultados, consiguiendo funciones que superan habitualmente a las diseñadas manualmente (Song et al., 2023).

El trabajo *Eureka* (Ma et al., 2023) profundiza aún más en esta idea mediante un algoritmo evolutivo guiado por LLMs. *Eureka* genera múltiples funciones candidatas en cada ciclo, variando penalizaciones y otros parámetros relevantes. Cada recompensa candidata entrena agentes específicos cuyos resultados retroalimentan al LLM para evaluarlas. Aprovechando recursos computacionales masivos, *Eureka* evoluciona estas recompensas mediante mutaciones y combinaciones sucesivas, superando a las diseñadas por humanos en el 83 % de los casos evaluados. El logro más notable de *Eureka* fue entrenar exitosamente la mano robótica *Shadow Hand* en tareas complejas como "pen spinning". Un trabajo posterior de los mismos investigadores, en el que se exportaron las funciones de recompensa generadas por *Eureka* al mundo real, demostró la eficacia del método al extenderlo a entornos físicos reales (Ma et al., 2024).

Posteriormente, han surgido sistemas similares que profundizan en la idea de que las recompensas generadas por LLMs pueden superar diseños humanos expertos (Guo et al., 2024; Xie et al., 2024). También se han explorado enfoques complementarios que utilizan LLMs no solo para generar recompensas, sino también para la planificación y evaluación crítica, integrando todas estas fases en un mismo bucle (Li et al., 2024a; Zeng et al., 2024a), lo cual simplifica aún más estos procesos.

Tras haber analizado la aplicabilidad satisfactoria de los LLMs al RL, se ha desarrollado el presente artículo, en el que las contribuciones principales son:

- Evaluar la eficacia de las recompensas generadas por un amplio espectro de LLMs, más allá de los explorados en *Eureka* y otros trabajos recientes.
- Demostrar la aplicabilidad de este enfoque automático para el diseño de recompensas en tareas altamente complejas de manipulación robótica.
- Identificar que los LLMs que incorporan procesos de razonamiento (*Chain-of-Thought*) tienden a generar funciones de recompensa que favorecen políticas más estables.

Este artículo se organiza de la siguiente manera: en el Capítulo 2 se describe la metodología empleada para modificar el esquema de *Eureka*; en el Capítulo 3 se detalla el hardware y software utilizados, así como las tareas desarrolladas durante la experimentación y los resultados obtenidos, y finalmente, en el Capítulo 4, se exponen las conclusiones obtenidas.

2. Metodolog´a

A partir del trabajo de *Eureka*, se han realizado modificaciones para permitir el uso de diferentes LLMs y evaluar su rendimiento en diferentes tareas de manipulación robótica (ver Figura 1). En este apartado se detallan los simuladores, las Application Programming Interfaces (APIs) y las métricas empleadas en este artículo.

En primer lugar, como simulador se ha empleado Isaac Gym – Preview 4. Este es un simulador físico desarrollado por NVIDIA, orientado al entrenamiento eficiente y realista de agentes

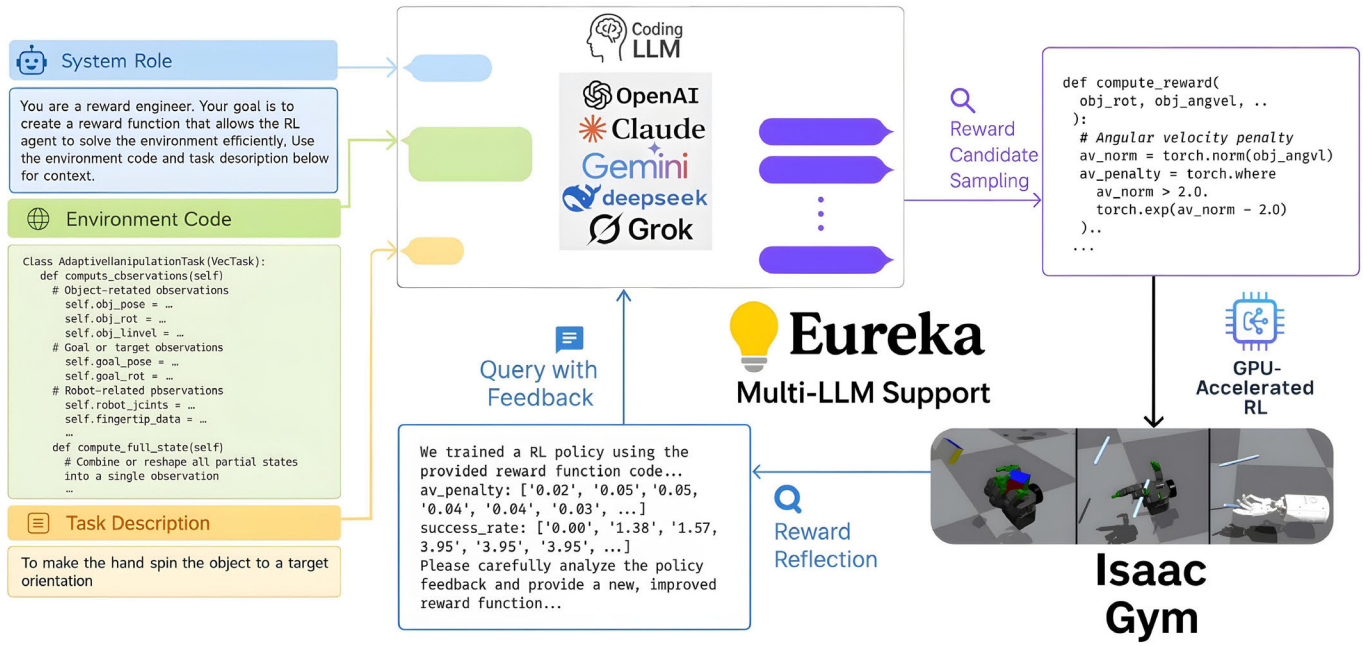

Figura 1: Esquema de la metodología adaptada de *Eureka* para el soporte de múltiples LLM. El proceso recibe como entrada el contexto de la tarea (rol del sistema, código del entorno y descripción textual), el cual es procesado por diferentes LLMs (OpenAI, Claude, Gemini, etc.) para generar funciones de recompensa candidatas. Estas son evaluadas mediante RL acelerado por GPU en el simulador Isaac Gym. Los resultados del entrenamiento se utilizan en un bucle de retroalimentación (*reward re ection* o reflexión de recompensa) para refinar iterativamente la función de recompensa.

mediante RL. Utiliza la aceleración mediante Unidad de Procesamiento Gráfico (GPU) usando Compute Unified Device Architecture (CUDA) para ejecutar múltiples agentes y entornos en paralelo, reduciendo significativamente los tiempos de entrenamiento (Gomez et al., 2024). Se seleccionó Isaac Gym sobre Isaac Lab (una versión más reciente enfocada a la precisión física e integración con hardware real) debido a su mayor capacidad para ejecutar numerosos entornos de RL simultáneamente (Makoviychuk et al., 2021). Aunque se prevén futuras mejoras de NVIDIA en Isaac Lab dentro de la plataforma Omniverse (la plataforma robótica de NVIDIA), actualmente no dispone de actualizaciones ni documentación suficiente para los objetivos del trabajo propuesto.

Respecto a las APIs de LLMs, se han utilizado varias ofrecidas por compañías como OpenAI, Anthropic y Gemini, entre otras. Estas interfaces facilitan la comunicación con modelos alojados en la nube y permiten automatizar tanto la generación de *prompts* como la obtención de respuestas. Para asegurar una evaluación coherente y justa entre modelos, se ha mantenido constante un conjunto de parámetros, como la temperatura de la respuesta, y un esquema uniforme de *prompts* en todas las llamadas a las distintas APIs. La mayoría de las peticiones se gestionan mediante la biblioteca `requests`, combinada con otras librerías oficiales proporcionadas por las empresas. A continuación, se describen las peculiaridades de las principales APIs utilizadas:

- **OpenAI:** esta API permite definir el cuerpo del mensaje con roles específicos como `system`, `user` y `assistant`. *Eureka* emplea el mensaje tipo `system` para establecer el contexto (por ejemplo, "Eres un asistente que genera recompensas en Python. . . ") y utiliza un mensaje tipo `user` para la petición concreta. Algunos modelos como O1 no aceptan roles diferenciados, lo que requiere unificar todo en un mensaje tipo `user`. Por ello, según el modelo utilizado, los mensajes se adaptan o consolidan de forma correspondiente.

- **Gemini:** se optó por emplear la librería estándar `requests`. El cuerpo de la solicitud a Gemini difiere del de OpenAI: en lugar de usar roles, se envía un objeto JSON con un campo `contents` que encapsula una lista de "bloques" a enviar al modelo. Cada bloque contiene una lista `parts`, donde cada parte lleva el contenido en un campo `text`. Esta estructura obliga a fusionar todo el contenido en un único texto antes del envío (similar a lo requerido por modelos como O1 en OpenAI). Este formato exigió ajustes en la lógica del proyecto, pero se conservó intacta la esencia del *pipeline*.

- **Anthropic:** Anthropic cuenta con una librería oficial estable para gestionar su API. En cuanto a la estructura de mensajes, Anthropic difiere de otras APIs al exigir que la directiva `system` se envíe como un parámetro de nivel superior (por ejemplo, `system=...`), dejando en el cuerpo del mensaje únicamente las entradas con roles `user` y `assistant`. No obstante, se respetan los mismos *prompts* y parámetros básicos para mantener la coherencia con el resto de modelos.

- **Grok:** la API de Grok es compatible con la biblioteca cliente de OpenAI, ya que su *endpoint* (la dirección específica (Uniform Resource Locator (URL)) a la que se envían las peticiones) acepta la misma estructura de llamadas. En la práctica, basta con reconfigurar la URL base y la clave API en el cliente de OpenAI para redirigir

las peticiones a Grok, requiriendo mínimos cambios en el código existente.

- **DeepSeek:** DeepSeek ofrece una compatibilidad análoga a la de Grok, permitiendo utilizar la biblioteca de OpenAI para interactuar con sus modelos. En este caso se accedió a DeepSeek mediante OpenRouter, configurando la URL base y empleando la clave API correspondiente.

Por último, para evaluar el rendimiento de los LLMs se utilizan dos métricas principales:

1. **Consecutive Successes:** esta métrica cuantifica la mayor racha de éxitos consecutivos alcanzada durante el entrenamiento, es decir, el valor máximo obtenido en la variable "éxitos" definida para cada entorno. Por ejemplo, en el caso de girar el bolígrafo, se interpreta como el mayor número de giros consecutivos logrados sin perder el control del mismo.
2. **Human Normalized Score:** se calcula normalizando el resultado de la política con respecto a tres referencias: por un lado, la recompensa *sparse*, que corresponde a una señal mínima o básica (+1 únicamente en caso de éxito final); por otro, la recompensa diseñada manualmente por expertos humanos (*human*), y finalmente la recompensa generada por *Eureka* (*method*). De este modo, *method* representa la tasa media de éxitos de *Consecutive Successes* lograda por cada modelo, mientras que *human* corresponde a la tasa media de éxitos de *Consecutive Successes* obtenida con las funciones de recompensa diseñadas manualmente para esta tarea. Esta puede verse en (1).

$$\text{HNS} = \frac{\text{method} - \text{sparse}}{\text{human} - \text{sparse}} \tag{1}$$

3. Experimentación y resultados

En este apartado se especifica el hardware y software empleados, además de describir los diferentes experimentos realizados y los resultados obtenidos.

3.1. Hardware y software empleados

Respecto al hardware empleado, se seleccionaron dos manos robóticas antropomórficas para realizar diferentes tareas de manipulación y agarre de objetos: la *Shadow y Allegro Hand*.

- **Shadow Hand:** mano robótica avanzada con cinco dedos y 24 Grados de Libertad (DoFs), diseñada para emular la destreza humana. Cuenta con sensores de posición y par en sus articulaciones, además de sensores táctiles en las yemas de los dedos, lo que proporciona una retroalimentación precisa en tareas que requieren alta sensibilidad.

- **Allegro Hand:** de diseño más sencillo que la *Shadow Hand*, posee cuatro dedos y 16 DoFs. Aunque también incorpora sensores de posición, par y sensores táctiles en las yemas de los dedos; su menor complejidad estructural supone un desafío adicional en tareas de manipulación fina, ofreciendo así un buen contraste experimental.

En cuanto al software, gran parte del ecosistema de IA, como PyTorch, Docker, CUDA e Isaac Gym, ofrece soporte prioritario para Ubuntu, por lo que se escogió esta distribución Linux como entorno base de experimentación. Inicialmente se utilizó Ubuntu 20.04, aunque se ha migrado parcialmente a Ubuntu 22.04 para asegurar la continuidad a largo plazo. Además, se empleó Docker para crear, distribuir y ejecutar aplicaciones mediante contenedores. Estos contenedores contienen diversos paquetes para cumplir con todas las dependencias necesarias de una aplicación (Merkel, 2014). Su uso en este proyecto facilita ejecutar experimentos con diferentes configuraciones de *Eureka* en distintos dispositivos hardware sin necesidad de realizar cambios manuales concretos, simplificando considerablemente el proceso. Por último, CUDA es una plataforma de computación paralela desarrollada por NVIDIA que aprovecha el paralelismo masivo de las GPUs para acelerar tareas intensivas, como el entrenamiento de modelos profundos y agentes de RL en Isaac Gym (Li et al., 2024b). Se optó por la versión 11.8 por su compatibilidad con diversas versiones de Ubuntu y controladores NVIDIA, facilitando migraciones fluidas entre diferentes máquinas y GPUs (por ejemplo, de una RTX 4070 a una RTX 1080Ti o a una A100) sin necesidad de cambiar la versión en los contenedores Docker, lo que contribuye a la robustez del proyecto.

Los entrenamientos fueron realizados en distintas GPUs, dependiendo de la fase del proyecto y los requisitos computacionales. Inicialmente, se utilizó una NVIDIA RTX 4070 con 12 GB de Video Random Access Memory (VRAM) en un equipo personal para realizar pruebas preliminares, ajustes iniciales, configuración del entorno y experimentos a pequeña escala. Posteriormente, la mayor parte de la experimentación se llevó a cabo utilizando un servidor de la Universidad de Alicante (UA), equipado con dos GPUs NVIDIA RTX 1080Ti, cada una con 12 GB de VRAM. Finalmente, en la etapa más avanzada, se ejecutó el sistema sin restricciones en el número de simulaciones, utilizando una GPU NVIDIA A100 con 40 GB de VRAM perteneciente a la plataforma DGX de la UA (Universidad de Alicante, 2020).

3.2. Descripción de los experimentos

Para evaluar objetivamente el rendimiento de las recompensas generadas por distintos LLMs, se definieron tres tareas diferentes de manipulación en Isaac Gym (Figura 2). Cada tarea se repite dos veces por cada modelo, con un total de ocho iteraciones y cinco propuestas de recompensa en cada iteración, garantizando así resultados más robustos.

1. **Pen Spinning con *Shadow Hand*:** la mano debe rotar un bolígrafo de manera sostenida. Este entorno también se utiliza en los datos originales de *Eureka*, lo que permite comparar resultados mediante la métrica de *Human Normalized Score*.
2. **Pen Spinning con *Allegro Hand*:** se emplea la mano *Allegro* para llevar a cabo la misma tarea de rotación continua de un bolígrafo, en lugar de la *Shadow Hand*.
3. **Manipulación de cubo con *Allegro Hand*:** el agente recibe indicaciones basadas en color e intenta mover o rotar el cubo hasta la posición o el ángulo asociado al color indicado.

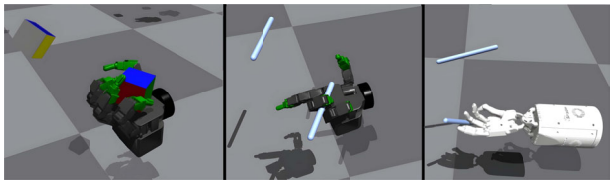

Figura 2: Tareas de entrenamiento: manipulación de cubo (izq.), *pen spinning* con Allegro (centro) y con Shadow Hand (der.).

3.3. Resultados obtenidos

Los experimentos se agruparán en dos bloques diferenciados. El primero de ellos consistirá en realizar la tarea de *pen spinning* con la *Shadow Hand* y la evaluación mediante la métrica *Human Normalized Score*, mientras que en el segundo la métrica *Consecutive Successes* será empleada para cada una de las tres tareas explicadas anteriormente en el Capítulo 3.2.

La Figura 3 muestra la métrica *Human Normalized Score* para la tarea de *pen spinning* con la *Shadow Hand*. Utilizando los datos del trabajo original de *Eureka* con la *Shadow Hand*, es posible comparar directamente la recompensa humana con las métricas obtenidas por los distintos modelos, permitiendo observar claramente su evolución. Se aprecia que algunos modelos, como *O1*, *Gemini 2.0 Flash Thinking Exp* o *Claude 3.7 Sonnet*, superan ampliamente el rendimiento humano. Destaca especialmente el modelo *O1*, que alcanza un valor cercano a 3, situándose como el más alto entre todos y superando con margen al propio *GPT-4*, empleado originalmente como referencia. En general, la mayoría de los modelos evaluados sobrepasan la referencia humana.

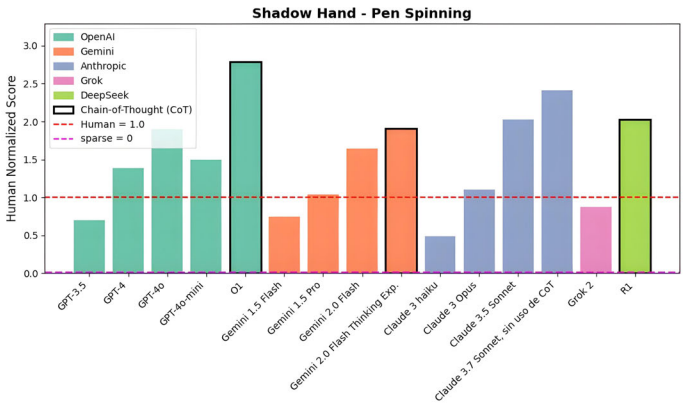

Figura 3: Resultados de la métrica *Human Normalized Score* en la tarea de *pen spinning* con *Shadow Hand*. La mayoría de modelos supera la referencia humana (línea roja), destacando O1, Claude 3.7 Sonnet y Gemini 2.0 Flash Thinking Exp

Por su parte, la Figura 4 presenta tres diagramas de barras que muestran la métrica *Consecutive Successes* para cada una de las tres tareas evaluadas. En general, los modelos más recientes aventajan a *GPT-4*, destacando la estabilidad en términos de desviación típica en aquellos que implementan razonamiento tipo *Chain-of-Thought*. Estas soluciones suelen entrenarse explorando múltiples cadenas de razonamiento y, en la fase de inferencia, aplican técnicas de búsqueda para seleccionar la mejor respuesta según una política aprendida mediante RL (Zeng et al., 2024b). No obstante, algunos modelos tradicionales preentrenados, como *Claude 3.5 Sonnet*, también logran resultados muy elevados. Resulta especialmente relevante la tarea de *pen spinning* con la *Allegro Hand*, donde los modelos recientes aventajan considerablemente a versiones anteriores. Esto sugiere que, al tratarse de una mano más sencilla y con menos sensores (y, por tanto, una tarea más exigente), los modelos de última generación son más efectivos para resolver los desafíos de control fino en manipulaciones complejas.

4. Conclusiones

Los experimentos realizados confirman la eficacia de las soluciones basadas en LLMs para la generación automática de recompensas, demostrando que varios modelos (especialmente aquellos con razonamiento tipo *Chain-of-Thought*) pueden superar significativamente la recompensa humana de referencia y guiar de manera muy efectiva las políticas de RL. En particular, el modelo *O1* destaca por su versatilidad, obteniendo puntuaciones máximas en las métricas *Human Normalized Score* y *Consecutive Successes*. Asimismo, ciertas variantes de Anthropic, como *Claude 3.7 Sonnet*, muestran un rendimiento estable en la mayoría de las pruebas.

Estos resultados corroboran que la combinación del enfoque evolutivo *Eureka* con LLM avanzados constituye una vía altamente prometedora para el diseño automatizado de recompensas en RL. Entre las futuras líneas de investigación se plantea extender este análisis a más modelos de lenguaje y considerar métricas adicionales que evalúen la estabilidad de las recompensas a lo largo de iteraciones o modificaciones del *prompt*. En conjunto, este estudio evidencia el potencial de estas técnicas para afrontar retos en manipulación robótica con alta dimensionalidad y necesidad de precisión, facilitando así su aplicación en diversos escenarios robóticos complejos.

Agradecimientos

The research work was supported by grant PID2021-122685OB-I00 funded by MICIU/AEI/10.13039/501100011033 and by ERDF/EU. Besides, computer facilities were provided by Valencian Government and FEDER through the IDIFEDER/2020/003.

Referencias

Booth, S., Knox, W. B., Shah, J., Niekum, S., Stone, P., Allievi, A., 2023. The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 37. pp. 5920–5929. DOI: 10.1609/aaai.v37i5.25733

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Driess, D., Dubey, A., Finn, C., Hausman, K., et al., 2023. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In: Conference on Robot Learning. PMLR, pp. 2165–2183. DOI: 10.48550/arXiv.2307.15818

Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., et al., 2022. Do as i can, not as i say: Grounding language in robotic affordances. In: Conference on robot learning. PMLR, pp. 287–318. DOI: 10.48550/arXiv.2204.01691

Driess, D., Xia, F., Sajjadi, M. S., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., et al., 2023. Palm-e: an embodied multimodal language model. In: Proceedings of the 40th International Conference on Machine Learning. pp. 8469–8488. DOI: 10.48550/arXiv.2303.03378

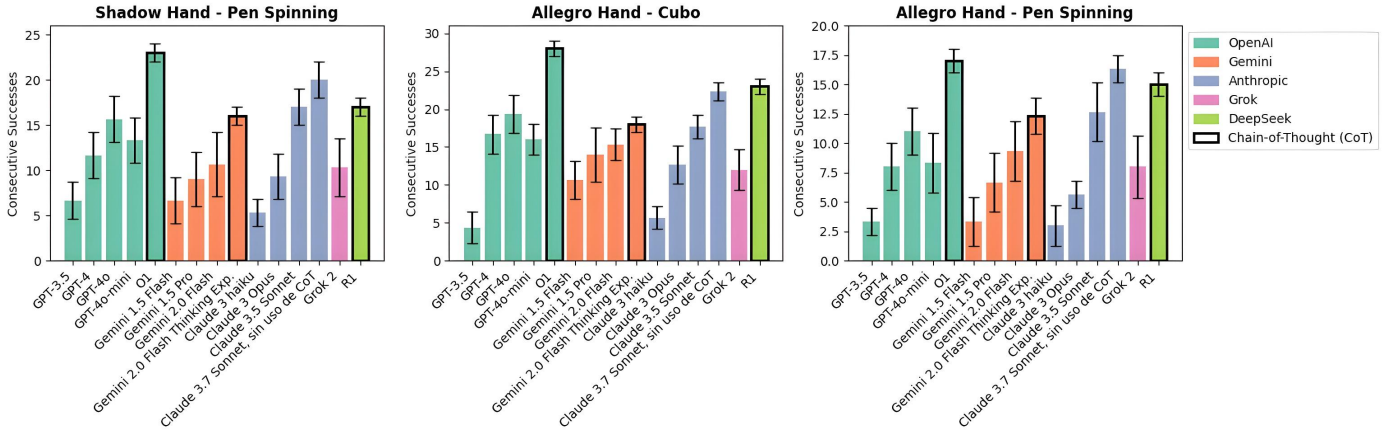

Figura 4: Resultados de la métrica *Consecutive Successes* en las tres tareas evaluadas: *Shadow Hand – Pen Spinning*, *Allegro Hand – Cubo* y *Allegro Hand – Pen Spinning*. Los modelos recientes muestran mejoras frente a GPT-4, especialmente aquellos con razonamiento tipo Chain-of-Thought.

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., Hester, T., 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Machine Learning 110 (9), 2419–2468. DOI: 10.1007/s10994-021-05961-4

Frau-Alfaro, D., Puente, S. T., Páez-Ubieta, I. d. L., 2024a. Trajectory generation using dual-robot haptic interface for reinforcement lerning from demonstration. Robot 2023: Sixth Iberian Robotics Conference. ROBOT 2023. Lecture Notes in Networks and Systems. 976, 444–455. DOI: 10.1007/978-3-031-58676-7_36

Frau-Alfaro, D., Puente, S. T., Páez-Ubieta, I. D. L., Velasco-Sánchez, E., 2024b. Robotic approach trajectory using reinforcement learning with dual quaternions. In: 2024 7th Iberian Robotics Conference (ROBOT). IEEE, pp. 1–6. DOI: 10.1109/ROBOT61475.2024.10796878

Gomez, I., de Cerio, U. D., Parra, J., Rivas, J. M., Gutiérrez, J. J., 2024. Uso de gpus en aplicaciones de tiempo real: Una revisión de técnicas para el análisis y optimización de parámetros temporales. Revista Iberoamericana de Automática e Informática Industrial 21 (1), 1–16. DOI: 10.4995/riai.2023.20321

Guo, Q., Liu, X., Hui, J., Liu, Z., Huang, P., 2024. Utilizing large language models for robot skill reward shaping in reinforcement learning. In: International Conference on Intelligent Robotics and Applications. Springer, pp. 3–17. DOI: 10.1007/978-981-96-0783-9_1

Guruprasad, P., Sikka, H., Song, J., Wang, Y., Liang, P. P., 2024. Benchmarking vision, language, & action models on robotic learning tasks. arXiv preprint arXiv:2411.05821. DOI: 10.48550/arXiv.2411.05821

Li, H., Yang, X., Wang, Z., Zhu, X., Zhou, J., Qiao, Y., Wang, X., Li, H., Lu, L., Dai, J., 2024a. Auto mc-reward: Automated dense reward design with large language models for minecraft. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 16426–16435. DOI: 10.1109/CVPR52733.2024.01554

Li, M., Bi, Z., Wang, T., Wen, Y., Niu, Q., Liu, J., Peng, B., Zhang, S., Pan, X., Xu, J., et al., 2024b. Deep learning and machine learning with gpgpu and cuda: Unlocking the power of parallel computing. arXiv preprint. DOI: 10.48550/arXiv.2410.05686

Ma, Y. J., Liang, W., Wang, G., Huang, D.-A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L. J., Anandkumar, A., 2023. Eureka: Human-level reward design via coding large language models. In: Proceedings of ICLR 2023. DOI: 10.48550/arXiv.2310.12931

Ma, Y. J., Liang, W., Wang, H., Wang, S., Zhu, Y., Fan, L., Bastani, O., Jayaraman, D., 2024. Dreureka: Language model guided sim-to-real transfer. arXiv preprint. DOI: 10.48550/arXiv.2406.01967

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., State, G., 2021. Isaac gym: High performance gpu-based physics simulation for robot learning. arXiv preprint. DOI: 10.48550/arXiv.2108.10470

Merkel, D., 2014. Docker: Lightweight linux containers for consistent development and deployment. Linux Journal 2014 (239). DOI: 10.5555/2600239.2600241

Ng, A. Y., Harada, D., Russell, S. J., 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In: Proceedings of the Sixteenth International Conference on Machine Learning. pp. 278–287.

Russell, S. J., Norvig, P., 2010. Artificial intelligence: A modern approach, 3rd Edition. Pearson Education Limited, consultar caps. 2 y 25.

Siciliano, B., Khatib, O., 2016. Springer handbook of robotics, 2nd Edition. Springer, consultar Cap. 12.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., Garg, A., 2023. Progprompt: Generating situated robot task plans using large language models. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 11523–11530. DOI: 10.1109/ICRA48891.2023.10161317

Song, J., Zhou, Z., Liu, J., Fang, C., Shu, Z., Ma, L., 2023. Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics. arXiv preprint. DOI: 10.48550/arXiv.2309.06687

Stone, P. e. a., 2021. The one hundred year study on artificial intelligence (ai100) 2021 study panel report. Report.

Sutton, R. S., Barto, A. G., 2018. Reinforcement learning: An introduction, 2nd Edition. MIT Press, consultar Cap. 1.1.

Universidad de Alicante, 2020. Plataforma de computación dgx - a100. Consultado el 21 de marzo de 2025.

Wang, C., Hasler, S., Tanneberg, D., Ocker, F., Joublin, F., Ceravola, A., Deigmoeller, J., Gienger, M., 2024. Lami: Large language models for multi-modal human-robot interaction. In: Extended Abstracts of the CHI Conference on Human Factors in Computing Systems. pp. 1–10. DOI: 10.1145/3613905.3651029

Xie, T., Zhao, S., Wu, C. H., Liu, Y., Luo, Q., Zhong, V., Yang, Y., Yu, T., 2024. Text2Reward: Reward shaping with language models for reinforcement learning. In: Proceedings of the International Conference on Learning Representations (ICLR). PMLR. DOI: 10.48550/arXiv.2309.11489

Yu, W., Gileadi, N., Fu, C., Kirmani, S., Lee, K.-H., Arenas, M. G., Chiang, H.-T. L., Erez, T., Hasenclever, L., Humplik, J., et al., 2023. Language to rewards for robotic skill synthesis. In: Proceedings of the Conference on Robot Learning (CoRL). PMLR, pp. 374–404. DOI: 10.48550/arXiv.2306.08647

Zeng, Y., Mu, Y., Shao, L., 2024a. Learning reward for robot skills using large language models via self-alignment. In: Proceedings of the 41st International Conference on Machine Learning (ICML). PMLR. DOI: 10.48550/arXiv.2405.07162

Zeng, Z., Cheng, Q., Yin, Z., Wang, B., Li, S., Zhou, Y., Guo, Q., Huang, X., Qiu, X., 2024b. Scaling of search and learning: A roadmap to reproduce o1 from reinforcement learning perspective. arXiv preprint. DOI: 10.48550/arXiv.2412.14135