Entorno de Gym basado en Impedancia para el robot colaborativo IIWA de cara a interacción humano robot

Ignacio Montesino Valle, Bartek Łukawski, Juan G. Victores, Alberto Jardon Huete, Carlos Balaguer

Robotics Lab, Departamento de Sistemas y Automática, Universidad Carlos III de Madrid

{imontesi, blukawsk, jcgvicto, ajardon, balaguer}@ing.uc3m.es

Resumen

A medida que aumenta la fisioterapia de rehabilitación, también aumenta el deseo de soluciones robóticas robustas y adaptables para estandarizar y automatizar procedimientos comunes. Sin embargo, las técnicas de control modernas todavía tienen que migrar a entornos centrados en la interacción humana como los que requiere la fisioterapia. Esto se debe en gran parte a la falta de entornos de aprendizaje que puedan aprovechar los modernos robots con control de fuerzas. El objetivo de este artículo es introducir un nuevo entorno de aprendizaje por refuerzo (RL) para el entrenamiento de un manipulador robótico controlado por fuerza tanto en simulación como en el mundo real. Este problema puede dividirse en tres componentes, cada uno de los cuales depende del anterior. En primer lugar, se requiere un controlador de control robusto que sirva de puente entre el lenguaje de programación nativo de los robots (C++) y el lenguaje de programación Python, donde se implementan la mayoría de los algoritmos de RL. de los algoritmos de RL. En segundo lugar, un controlador de impedancia cartesiana que pueda ajustarse para funcionar en la simulación y en el mundo real, abstrayendo la mayor parte de la complejidad de la de la dinámica de cuerpos rígidos. Y, por último, la construcción del entorno RL basado en la omnipresente interfaz OpenAI Gym.

Palabras clave: Control Impedancia, Aprendizaje por refuerzo, robot colaborativo, rehabilitación robótica

1. Introducción

La creciente demanda de fisioterapia para la rehabilitación ha generado una necesidad de soluciones robóticas para automatizar y estandarizar los tratamientos. Cada vez son más las personas afectadas por distintos tipos de dolencias musculares, y las previsiones más recientes esperan que ese número siga creciendo [1].

Son muchas las soluciones que se han desarrollado tanto para el diagnóstico [2] como para el tratamiento de la fisioterapia. En ellas se ha visto los beneficios que ofrecen: menores costes, mayor repetibilidad y medidas objetivas.

Figura 1: Robot IIWA con maneta para rehabilitación

Paralelamente, los algoritmos basados en redes neuronales y aprendizaje por refuerzo siguen ganando popularidad. Cada vez más aplicaciones de manipulación robótica son resueltas mediante estas técnicas. Como por ejemplo apilar piezas de lego [3] o resolver un cubo de Rubik [4]. Incluso en el ámbito de la salud se han observado los beneficios que aportan [5].

No obstante, sigue siendo de gran dificultad implementar algoritmos de aprendizaje profundo en el campo de la interacción humano robot, en especial al aplicarlos a la rehabilitación. Esto es debido en parte a la gran complejidad que tiene la dinámica de robots manipuladores, definida por un gran número de ecuaciones no lineares. También a la dificultad técnica que requiere enlazar las librerías usadas en aprendizaje por refuerzo y las usadas en el control a bajo nivel.

Por ello, en este artículo mezclamos métodos tradicionales de control seguros con control de fuerza,

 https://doi.org/10.17979/spudc.9788497498418.0762

junto con interfaces predominantes en el mundo del aprendizaje por refuerzo.

Para ello separamos la problemática en tres partes, cada una construyendo sobre la anterior.

1.1. Abstracción del control en tiempo real del robot IIWA a Python.

El primer paso es obtener un *driver* que permita el control en fuerzas de forma robusta en Python, dado que este es el lenguaje por defecto en RL.

1.2. Obtención de un modelo de impedancia cartesiana.

Aun así la problemática del control de fuerzas es demasiado compleja para que un algoritmo de Rl se pueda usar para resolverla. Para evitarlo se deriva un control de impedancia Cartesiana que simplificara la compleja interacción de las fuerzas en las articulaciones a posiciones en el espacio cartesiano.

1.3. Creación de un entorno de aprendizaje por refuerzo y entrenamiento

Por último usaremos la interfaz Gym para crear un entorno de aprendizaje por refuerzo, montado por encima del controlador cartesiano. La tarea a realizar como prueba será la de seguir una trayectoria en la cual se encuentran obstáculos intentando minimizar la desviación y las fuerzas de contacto

2. Plataforma robótica y driver

2.1. Robot KUKA LBR IIWA

La plataforma robótica de rehabilitación consiste en un robot de KUKA LBR IIWA equipado con una maneta en el efector final mostrado en Fig. 1. El robot consta de siete articulaciones, lo que lo convierte en un robot de cinemática redundante.

Consta además de sensores de par en cada una de dichas articulaciones, lo que permite crear controles basados en fuerzas de forma precisa.

2.2. Protocolo FRI

Para controlar el robot IIWA se utiliza el protocolo FRI (Fast Robot Interface) [6]. Este es un protocolo de comunicación en tiempo real para control *on-line* de posiciones y esfuerzos del robot IIWA.

Al mandar un comando al robot, el controlador interno sustituye la salida del interpolador de trayectoria. En el caso de control en posición, la posición es sustituida por la posición deseada. En el caso de control en esfuerzo, el esfuerzo comandado es sumado a la salida del interpolador. Si el interpolador se configura para no ejercer ningún esfuerzo, el controlador solo aplicará compensación de gravedad y de fricción.

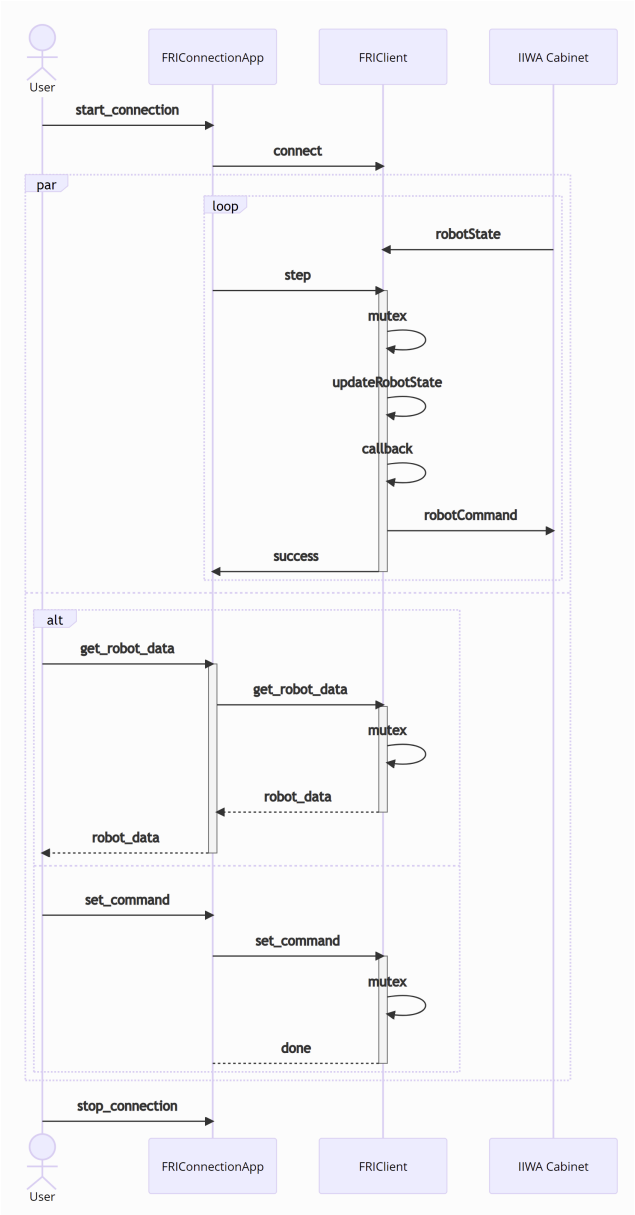


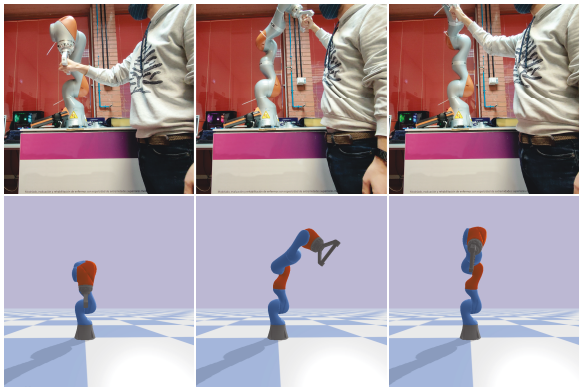
Figura 2: Protocolo de conexión con el driver de KUKA LBR IIWA

El funcionamiento de FRI es de tiempo real estricto, esto quiere decir que si el controlador (FriServer) no recibe un comando del cliente (FriClient) con la frecuencia programada (1-5ms) corta la comunicación y detiene el robot. Para sobrepasar este requisito se ha creado un controlador en C++ encargado de gestionar la conexión en tiempo real. En caso de no recibir un comando del usuario a tiempo se ejecuta un comando de posi-

https://doi.org/10.17979/spudc.9788497498418.0762

ción por defecto o se ejecuta un comando vacío en caso del esfuerzo tal como de muestra en Fig. 2.

2.3. Modelo Simulado



Cuadro 1: Capturas de trayectorias en modelo real y simulado

Para el modelo simulado se ha utilizado el paquete PyBullet [7]. PyBullet es un motor de simulación de robótica de código abierto. Es de uso frecuente en tareas de aprendizaje profundo por su versatilidad, facilidad de paralelización y su eficiencia [8].

Se ha usado un modelo en formato URDF, que es un formato de archivo de texto que describe una estructura de robótica. Al importarse en Pybullet obtenemos un mecanismo con las mismas características físicas y visuales que el modelo real, tal como se observa en las figuras del Cuad. 1.

Al usar el control en fuerza o *wrench* mediante conexión FRI el controlador interno compensa la gravedad propia del robot. Para igualar este efecto, se ha implementado el *solver* Orocos-KDL [9] para calcular las componentes dinámicas del modelo.

Por lo tanto, se crean dos clases en Python: una para el robot real y otra el simulado. Ambas clases heredan de una clase base que define una interfaz común. La interfaz consiste únicamente en los métodos para enviar comandos de control y para recibir información de los sensores tal como se muestra en Fig. 3.

3. Impedancia Cartesiana

Una vez se tiene una interfaz común para el control tanto en simulación como real, se puede definir un control de impedancia cartesiana que pueda ser usado en ambos entornos.

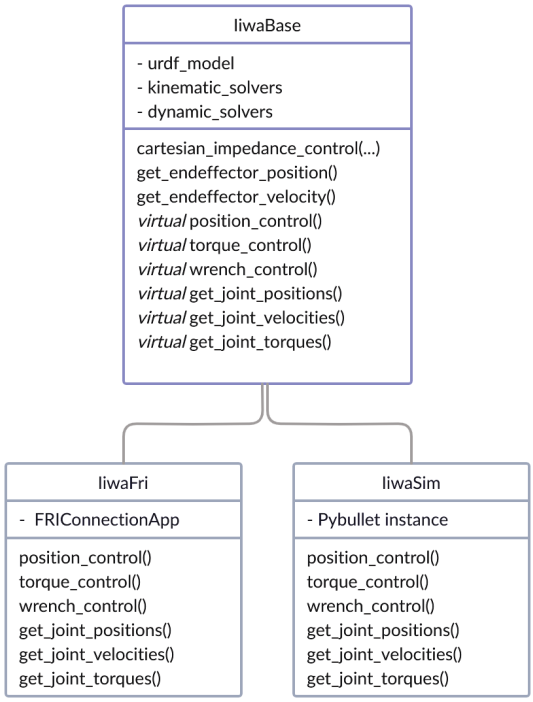


Figura 3: Clases implementadas

3.1. Control de Impedancia

El control de impedancia es un modo de control que se basa en intentar obtener una dinámica similar a la de un muelle amortiguado. Dicho control es de mayor utilidad en los entornos en los que se espera que el robot entre en contacto con objetos o en este caso humanos.

El control de impedancia se puede definir en el espacio articular donde se asimila a una elasticidad en las articulaciones del robot. O se puede definir en el espacio del efector final, llamándose impedancia cartesiana.

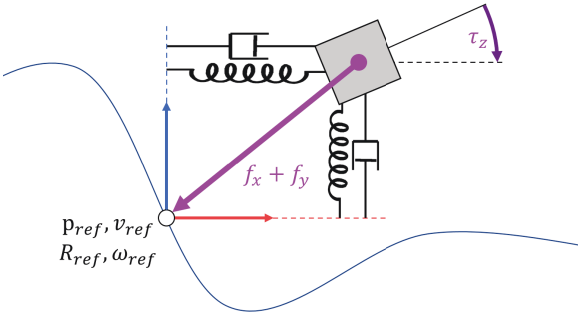


Figura 4: Impedancia Cartesiana en SE2

Este último será el que se utilizará para esta tarea, ya que nos interesa definir la rigidez percibida

https://doi.org/10.17979/spudc.9788497498418.0762

en el espacio de la tarea de la misma forma que un fisioterapeuta modificaría la rigidez con la que moviliza la extremidad de un paciente.

La expresión de las fuerzas y pares en el efector final en función del el error en posición cartesiana e se muestra en la Ec. 1.

$$(f, \tau) = K \cdot e + D \cdot \dot{e} + \Lambda \cdot \ddot{e} \tag{1}$$

(f, τ) es un vector R^6 cuyos tres primeros componentes corresponden a las fuerzas en x, y y z respectivamente, y los tres últimos componentes corresponden a los pares en esos mismos ejes.

El resultado es una dinámica idéntica aun mecanismo que tuviera un muelle amortiguado para cada dirección cartesiana. En la Fig. 4 observa la representación gráfica de este efecto para el caso del movimiento en el plano.

K es una matriz diagonal $R^{6\times6}$ que define la rigidez en las direcciones principales. Los valores de la rigidez los elegimos en función de si queremos un comportamiento más o menos elástico.

La matriz Λ se denomina como la masa generalizada del robot en torno a una posición cartesiana. Para obtenerla expresamos la matriz de inercia articular M en coordenadas cartesianas usando la inversa de la matriz jacobiana.

$$\Lambda = {J^+}^T M_q J^+ \tag{2}$$

La matriz J^+ es la pseudo-inversa de Penrose de la matriz jacobiana, ya que al ser un robot redundante la matriz de jacobiano no es cuadrada sino $R^{6\times7}$.

3.2. Diseño de la matriz de Amortiguación

Queda únicamente definir la matriz de amortiguación D para obtener una dinámica estable. Para obtener la matriz de amortiguación D, usamos la descomposición de Cholesky de la matriz de masa Λ para obtener los autovalores generalizados de la matriz K respecto a Λ.

$$\Lambda = QQ^T \tag{3}$$

$$K_d = {Q^{-1}KQ^{-1}}^T \tag{4}$$

$$D = 2QD_\xi\sqrt{K_d}Q^T \tag{5}$$

D_ξ es una matriz diagonal en la que los elementos de la diagonal son los factores de amortiguación entre 0.1 y 1 para las seis direcciones cartesianas. La obtención de las ecuaciones 4 y 5 se encuentra detallada en [10]. Como nota en el libro citado se denomina $\Lambda = Q^TQ$ en vez de $\Lambda = QQ^T$. Ambas expresiones son equivalentes, pero usamos la segunda al ser la convención en la descomposición de Cholesky.

3.3. Validación del Control

Observamos la respuesta tanto del modelo en simulación como del modelo real, con distintos factores de amortiguación. En color rojo la respuesta del modelo real, y en color azul la respuesta del modelo en simulación.

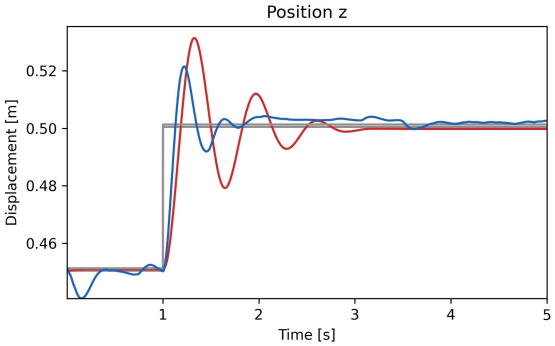


Figura 5: Respuesta en posición con $\xi = 0{,}2$

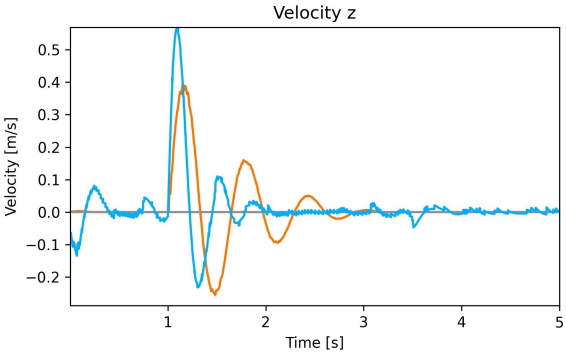


Figura 6: Respuesta en velocidad con $\xi = 0{,}2$

En la Fig. 5 se observa que la respuesta en posición del modelo real y el modelo en simulación son muy similares. Con similitud también en la respuesta en velocidad, tal como muestra la Fig. 6.

Vemos que obtenemos una respuesta estable en el rango de los factores de amortiguación válidos. Y que el sobreimpulso y la oscilación disminuyen adecuadamente al aumentar el factor de amortiguación. Tanto en el robot real como en el simulado.

https://doi.org/10.17979/spudc.9788497498418.0762

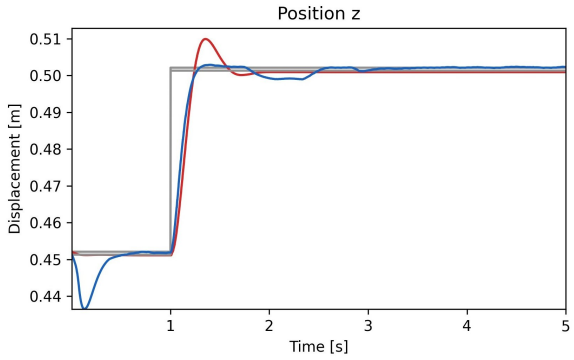


Figura 7: Respuesta con $\xi = 0{,}7$

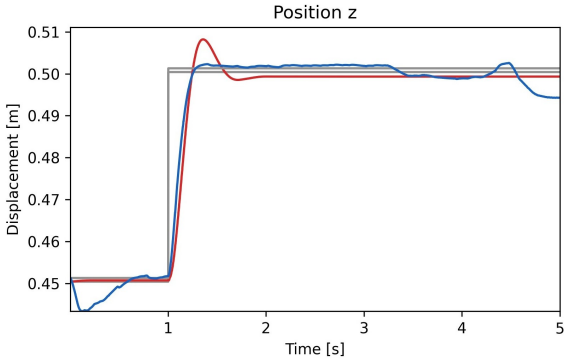


Figura 8: Respuesta con $\xi = 1{,}0$

4. Entorno Gym

Por último, queda implementar un entorno de aprendizaje por refuerzo que permita aprender leyes de control basadas en fuerzas. Se implementará usando la interfaz Gym de OpenAI [11], dada su popularidad en la comunidad del aprendizaje por refuerzo.

El aprendizaje por refuerzo es una de las técnicas de aprendizaje automático con mayor crecimiento en la comunidad robótica. Esto se debe a que, a diferencia del aprendizaje profundo tradicional, no necesitamos de una base de datos etiquetada para entrenar un modelo. Si no que se crea un entorno sobre el que un agente toma acciones.

En Fig. 9 se muestra un esquema simplificado de un entorno de aprendizaje por refuerzo. Las componentes principales [12] son:

1. Un entorno observable.
2. Un agente.
3. Una función de recompensa.

4.1. Entorno

El entorno en este caso es la versión simulada del robot IIWA descrita en la sección 2, con obstáculos

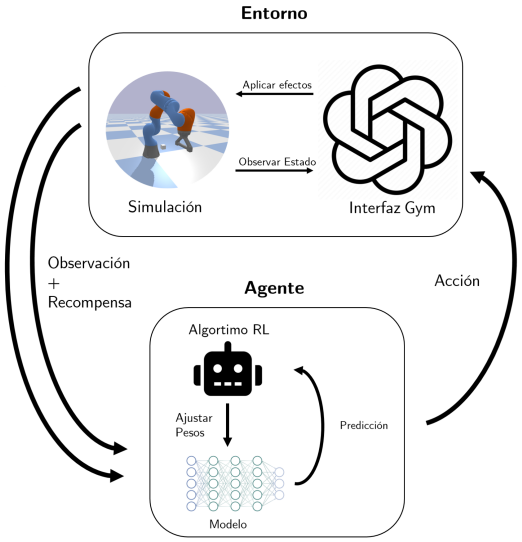


Figura 9: Entorno Aprendizaje por refuerzo

puestos de forma aleatoria, Fig. 9, a lo largo de una trayectoria que debe seguir. Las acciones que puede tomar el agente es establecer la posición de referencia para el controlador de impedancia Cartesiana definido en la sección 3. La observación que devuelve está compuesta de la posición p_{ref} de referencia de la trayectoria τ_{ref} para el instante t. La posición p del efector final, la velocidad v del efector final.

Al no dar información sobre las fuerzas, el modelo debe aprender a interpretarla basándose en la diferencia entre la posición actual y la esperada.

Con todo ello, el objetivo del algoritmo es optimizar el valor óptimo $V^*(s)$ para un estado inicial s.

$$V^*(s) = \max_a \mathbf{E}_{\pi^*}\Big[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k+1}) \Big| s_t = s, a_t = a\Big] \quad (6)$$

Este valor es la expectativa de la suma de todos las recompensas futuras $r(s_{t+k+1})$ en función de los estados alcanzados, cada uno ponderado por un factor de descuento γ para priorizar recompensas a corto plazo. Las acciones a_t son las elegidas por el modelo $\pi^*(s_t) \rightarrow a_t$ $(s_t, a_t) \rightarrow s_{t+1}$

El modelo de agente elegido es el algoritmo PPO [13] por ser un algoritmo actual de estado del arte de probada eficacia en tareas de robótica [14, 15].

Por último, la función de recompensa será la resta de a distancia euclidiana entre la posición actual y la esperada, la velocidad y la fuerza de contacto con el obstáculo. De esta forma, el agente aprende a controlar el robot para que se mueva a la posición deseada, pero detectando colisiones y evitándolas.

https://doi.org/10.17979/spudc.9788497498418.0762

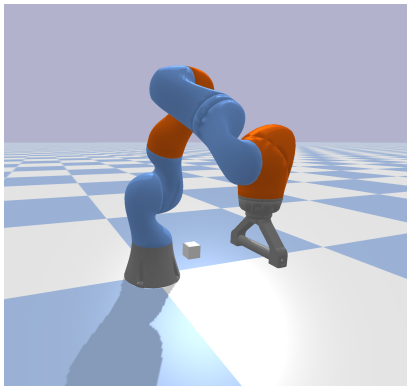


Figura 10: Robot IIWA con obstáculo

4.2. Resultados de entrenamiento

Para el entrenamiento se ha usado el método de *curriculum learning* [16], mediante el cual se va exponiendo al agente a tareas de cada vez mayor complejidad, evitando la problemática del aprendizaje nulo en entornos de bajas recompensas y alta complejidad.

Primero se enseña al agente a seguir la trayectoria sin obstáculos.

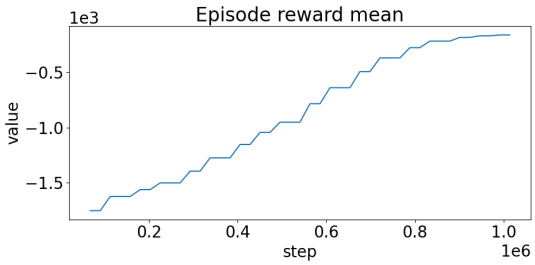


Figura 11: Evolución de la recompensa en entrenamiento sin colisiones

Y después se introducen los obstáculos.

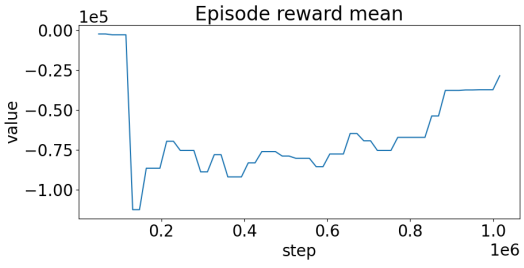


Figura 12: Evolución de la recompensa en entrenamiento con colisiones

Se observa que ambos entornos consiguen optimizar sus funciones de refuerzo. Como era de esperar, al reentrenar el agente en el entorno con colisiones inicialmente se obtiene una menor recompensa, ya que no ha aprendido a detectar colisiones. Pero eventualmente consigue optimizar su aprendizaje.

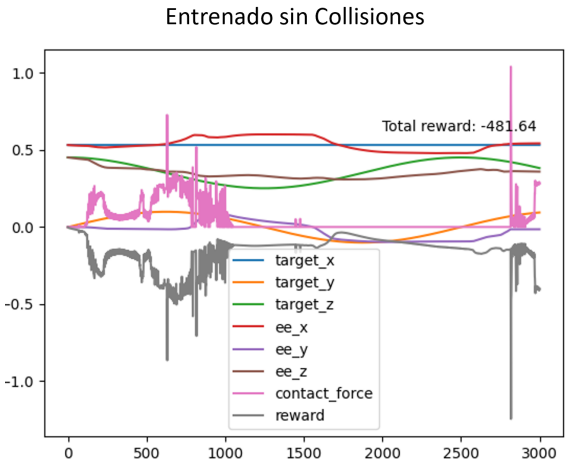


Figura 13: Resultado de control del agente entrenado sin colisiones

Haciendo un análisis cualitativo de ambos los agentes obtenidos tras las dos etapas de entrenamiento, se observa que el agente entrenado sin colisiones, Fig. 13, no es capaz de reaccionar ante un obstáculo, produciendo grades fuerzas de contacto (en rosa), incluso tratandose de un control de impedancia.

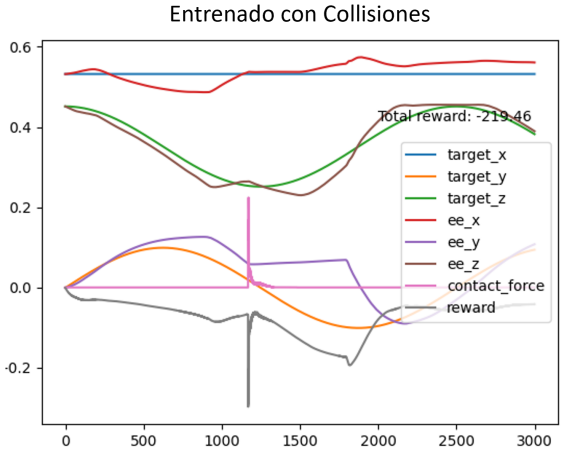


Figura 14: Resultado de control del agente entrenado con colisiones

Sin embargo, el agente reentrenado con colisiones, Fig. 14, Al chocar inicialmente con el obstáculo, detecta el pico de fuerza de contacto y modifica su trayectoria para evitarlo.

5. Conclusión y trabajo futuro

En conclusión, hemos presentado un *framework* completo de aprendizaje por refuerzo para tareas

de control en fuerza para el robot colaborativo IIWA. Se han validado tanto la transferencia del control del modelo simulado al robot real como la implementación del entorno de aprendizaje por refuerzo.

En un futuro esperamos aumentar las capacidades del entorno introduciendo datos reales de fuerzas ejercidas por pacientes de rehabilitación, así como el uso de aprendizaje de trayectorias generadas para el tratamiento de patologías musculares.

Agradecimientos

Esta investigación ha recibido fondos de los proyectos ROBOASSET PID2020-113508RB-I00; ALMA H2020-EIC-FETPROACT-2019; proyecto de I+D+i PLEC2021-007819 financiado por MCIN/AEI/10.13039/501100011033 y por la Unión Europea NextGenerationEU/PRTR; RoboCity2030-DIH-CM, Madrid Robotics Digital Innovation Hub, S2018/NMT-4331, financiado por "Programas de Actividades I+D en la Comunidad de Madrid".

English summary

Impedance based Gym environment for the IIWA Collaborative robot towards human robot interaction

Abstract

As rehabilitation physiotherapy is on the rise, so too is the desire for robust and adaptable robotic solutions to standarise and automate common procedures. Yet modern control techniques have still to migrate to human-interaction centered environments such as the ones required in physiotherapy. This is in great part due to the lack of learning environments that can leverage modern force controlled robots. The aim of this paper is therefore to introduce a new reinforcement learning (RL) environment for learning and training of a force controlled robotic manipulator both in simulation and in the real world. This problem can be divided into three components, each one depending on the last one. First, a robust control driver is required that bridges between the robots native programming language (C++) and the Python programming language, where most RL algorithms are implemented. Second, a cartesian impedance controller that can be fitted to work both in simulation and in the real world, abstracting away most of the complexity of rigid body dynamics. And finally the construction of the RL environment based on the ubiquitous OpenAI Gym interface.

Keywords: Impedance Control, DeepRL, Collaborative robot, robotic rehabilitacion.

Referencias

[1] Francesca Gimigliano and Stefano Negrini. The World Health Organization "Rehabilitation 2030: a call for action". *European Journal of Physical and Rehabilitation Medicine*, 53(2):155–168, April 2017.

[2] F. Biering-Sørensen, J. B. Nielsen, and K. Klinge. Spasticity-assessment: a review. *Spinal Cord*, 44(12):708–722, December 2006.

[3] Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable Deep Reinforcement Learning for Robotic Manipulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6244–6251, May 2018. ISSN: 2577-087X.

[4] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand, 2019.

[5] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics*, 19(6):1236–1246, 05 2017.

[6] G Schreiber. The Fast Research Interface for the KUKA Lightweight Robot. page 31.

[7] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[8] Marian Körber, Johann Lange, Stephan Rediske, Simon Steinmann, and Roland Glück. Comparing Popular Simulation Environments in the Scope of Robotics and

 https://doi.org/10.17979/spudc.9788497498418.0762

Reinforcement Learning, March 2021. arXiv:2103.04616 [cs].

[9] R. Smits. KDL: Kinematics and Dynamics Library. `http://www.orocos.org/kdl`.

[10] Christian Ott. *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*, volume 49 of *Springer Tracts in Advanced Robotics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISSN: 1610-7438, 1610-742X.

[11] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[12] Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction. page 352.

[13] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. arXiv:1707.06347 [cs].

[14] Zackory Erickson, Vamsee Gangaram, Ariel Kapusta, C. Karen Liu, and Charles C. Kemp. Assistive gym: A physics simulation framework for assistive robotics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10169–10176, 2020.

[15] Asad Ali Shahid, Loris Roveda, Dario Piga, and Francesco Braghin. Learning continuous control actions for robotic grasping with reinforcement learning. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 4066–4072, 2020.

[16] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum Learning: A Survey, April 2022. arXiv:2101.10382 [cs].