SoftwareX 35 (2026) 102800

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original Software Publication

RehabBot: Reinforcement learning environment for upper and lower limb robot-assisted rehabilitation

Victoria Oguntosin [a,b], Juan-Ignacio Vazquez [a,*]

[a] Faculty of Engineering, University of Deusto, Avda. Universidades 24 48007, Bilbao, Spain
[b] Department of Electrical & Information Engineering, Covenant University, Ota, Ogun State, Nigeria

ARTICLE INFO

Keywords:
Rehabilitation robotics
Reinforcement learning
Limb rehabilitation
Simulation environment
Benchmarking framework
MuJoCo

ABSTRACT

We present RehabBot, an open-source simulation environment for reinforcement learning (RL) research in robot-assisted upper- and lower-limb rehabilitation. Built on the MuJoCo physics engine, it integrates a torque-controlled UR5e collaborative manipulator with an articulated humanoid patient model to implement physiotherapy-inspired exercises for the shoulder, elbow, hip, and knee. The platform provides a modular RL-ready API with defined continuous observation and action spaces, task-aligned reward functions with safety constraints, and configurable evaluation scenarios. Exercises are formalized as Markov Decision Processes, ensuring compatibility with standard RL libraries. Validation through representative deep RL algorithms demonstrates stable training behaviour and compatibility with standard pipelines. RehabBot offers an extensible framework for reproducible research at the intersection of robotics, rehabilitation, and learning-based control.

Metadata

Nr	Code metadata description	*Metadata*
C1	Current code version	V 1.0
C2	Permanent link to code/ repository used for this code version	https://github.com/vickkiee/rehabbot
C3	Permanent link to reproducible capsule	https://github.com/vickkiee/rehabbot/releases/tag/v1.0.0
C4	Legal code license	MIT Licence
C5	Code versioning system used	Git
C6	Software code languages, tools and services used	Python $\geq$ 3.10 Gymnasium $\geq$ 0.29.1
C7	Compilation requirements, operating environments and dependencies	Ubuntu 24.04
C8	If available, link to developer documentation/manual	https://github.com/vickkiee/rehabbot
C9	Support email for questions	victoria.oguntosin@covenantuniversity.edu.ng ivazquez@deusto.es

1. Motivation and significance

Musculoskeletal and stroke-related impairments are leading causes of disability, affecting daily living and work. Accessible, high-quality rehabilitation is essential because many patients face long and demanding therapy journeys. Personalized robotic therapy can improve outcomes and independence while reducing healthcare costs [1]. Within this context, reinforcement learning (RL) offers a principled way to adapt assistance to individual needs—provided there are simulation tools that approximate upper- and lower-limb movement and the constraints of safe human–robot interaction.

RehabBot is a simulation environment built for this purpose. It instantiates physiotherapy-inspired exercises involving the shoulder, elbow, hip, and knee, coupling a collaborative robotic arm with an articulated humanoid patient model so the robot can assist and guide the limb through prescribed motions. Tasks are expressed with physiologically bounded ranges of motion and safety-aware terminations, and the interface defines clear actions, observations, rewards, and episode criteria to enable reproducible RL studies. The environment is Gymnasium-compatible and can be used with standard RL libraries [2–4] on tasks such as trajectory following and target reaching. The same abstractions can facilitate sim-to-real transfer studies of rehabilitation protocols, subject to biomechanical calibration and clinical validation.

RehabBot is implemented with MuJoCo, a physics engine widely used in robotics for efficient simulation of articulated-body dynamics

* Corresponding author.
E-mail addresses: victoria.oguntosin@covenantuniversity.edu.ng (V. Oguntosin), ivazquez@deusto.es (J.-I. Vazquez).

https://doi.org/10.1016/j.softx.2026.102800
Received 4 March 2026; Received in revised form 4 June 2026; Accepted 9 June 2026
Available online 16 June 2026

and contact interactions [5] due to its computational efficiency and physical accuracy [14]. We also leverage the MuJoCo Menagerie repository of XML models for commercial platforms (bipeds, humanoids, manipulators, grippers, hands, drones, biomechanical systems) when constructing the scene [6]. While such resources offer strong hardware coverage, they do not, by themselves, provide task-specific rehabilitation environments or an RL evaluation protocol.

1.1. Background and related work

Several simulation platforms have addressed related aspects of rehabilitation robotics and reinforcement learning, albeit typically focusing on isolated components such as robotic control, biomechanical modelling, or algorithmic benchmarking.

MyoSuite [7] focuses on bio-musculoskeletal modelling and fine motor control but does not include collaborative robot–human interaction for physiotherapy tasks. Assistive Gym [8] focuses on activities of daily living (e.g., feeding, dressing, bathing), yet it lacks physiotherapy-related exercises with standardized RL task definitions. Work on multi-goal dexterous manipulation demonstrates impressive hand control but does not target rehabilitation-oriented objectives or human–robot assistance [9]. Robosuite [10] and Robohive [11] offer modular MuJoCo-based environments for manipulation and locomotion, however they do not natively integrate articulated human models with rehabilitation-oriented tasks. While MyoAssist [12] offers high-fidelity musculoskeletal modelling for bionics and dexterity, it primarily focuses on the internal mechanics of the human body. Human-Robot Gym [13] is a significant contribution to Human-Robot Collaboration (HRC) in general industrial or domestic settings, although not specifically oriented to physiotherapy exercises.

RehabBot is a novel open-source simulation environment that simultaneously integrates a collaborative robot and an articulated human model to realize RL-based upper- and lower-limb physiotherapy tasks under physiologically bounded constraints.

Table 1 compares RehabBot with the previously discussed alternatives. Existing platforms typically address either robotic control or rehabilitation modelling, but not their integration within an RL-ready software framework. RehabBot brings these elements together: a collaborative robot coupled with a humanoid patient model; physiotherapy-inspired tasks with physiology-aware bounds; and a reproducible RL interface suitable for benchmarking and extension.

2. Software description

2.1. Software architecture

RehabBot is designed as a Gymnasium-compliant [18] reinforcement learning training environment developed using the MuJoCo physics simulator [5] and featuring a UR5e collaborative robot model along with an articulated human model. Fig. 1 illustrates the modular software architecture and control flow of RehabBot.

The training script interacts with the environment exclusively through the Gymnasium API, ensuring compatibility with standard reinforcement learning libraries. Actions generated by the learning agent are passed to the RehabBot environment, which performs action preprocessing, integrates task-specific definitions (e.g., exercise targets, reward functions, and safety constraints), and coordinates the robot and human biomechanical models. The robot model (UR5e) and the humanoid model are represented as separate but physically coupled entities, each encapsulating their respective kinematic and actuation

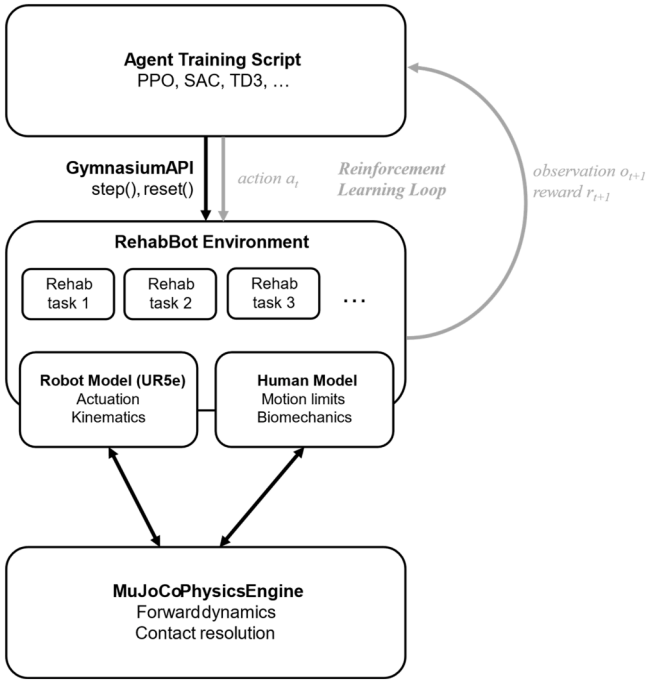

Fig. 1. Software architecture of RehabBot.

Table 1
Summary comparison with related simulation platforms.

Platform	C1 - Human–robot coupling	C2 - Physiotherapy limb-related exercises	Engine	Notes
RehabBot (ours)	✓ collaborative robot + articulated human	✓ shoulder, elbow, hip, knee with ROM bounds & safety terminations (upper + lower)	MuJoCo	Designed for robot-assisted physiotherapy tasks
MyoSuite [7]	× (human models only; no cobot assisting human)	◐ detailed musculoskeletal exercises; fine-motor emphasis; limited leg coverage	MuJoCo	Lacks collaborative-robot–human coupling and unified physiotherapy task set
Assistive Gym [8]	✓ scripted human–robot ADLs	× physiotherapy joint-exercise library; ◐ ADL-specific limb coverage	PyBullet	ADL focus; contact fidelity differs; no standardized physiotherapy API
Multi-goal hand [9]	× (robot hand only)	× physiotherapy exercises; × limb coverage	MuJoCo	Dexterous manipulation; not rehabilitation-oriented
Robosuite [10]	× (robot only)	× physiotherapy exercises; × limb coverage	MuJoCo	General robot-learning suite; no native human model or rehab tasks
Robohive [11]	◐ (broad tasks; some musculoskeletal)	◐ varied tasks, not rehab-specific; ◐ partial limb coverage	MuJoCo	No cobot–human physiotherapy coupling
MyoAssist [12]	✓ (Internal)	◐ framework is specialized for dexterity and agility in bionic humans	MuJoCo	Focuses on musculoskeletal modelling and bionic/prosthetic control; less emphasis on external cobot-assisted therapy.
Human-Robot Gym [13]	✓ (External)	× physiotherapy exercises	MuJoCo	Designed for general HRC in industrial tasks rather than specific clinical protocols.

(✓—criterion met; ◐—partially; ×—not provided natively).

properties. The underlying physics computations, including forward dynamics and contact resolution, are delegated to the MuJoCo simulation engine. The resulting state updates are used to construct observations and rewards that are returned to the training script, thereby closing the reinforcement learning loop.

The robot interacts with the humanoid through kinematic and contact constraints that enable the controller to assist and guide limb motion along task-defined trajectories and targets. Tasks are constructed with physiology-aware bounds, including joint limits and safe workspaces, as well as safety-aware termination conditions that reflect clinical constraints while preserving reinforcement learning trainability. This layered design separates control logic, task specification, biomechanical modelling, and physics simulation, thereby facilitating extensibility, reproducibility, and maintainability.

RehabBot provides a set of predefined training scenarios, formalized as specific Gymnasium-compliant environments (see Table 2), corresponding to upper- and lower-limb rehabilitation tasks, including target-reaching and trajectory-following activities for the shoulder, elbow, hip, and knee joints. Each scenario defines its own reset conditions, target generation logic, reward configuration, and safety-aware termination criteria.

The global reference frame is defined at the robot base, and all target poses and measurements explicitly specify their associated reference frame (world frame or robot base frame). For each task, the wrist/tool frame is aligned with the corresponding assisted limb segment to ensure consistent kinematic interpretation. Additionally, a 90° reorientation of the robot base can be configured when a task requires greater emphasis on specific motion ranges, such as abduction/adduction or pronation/supination.

The default control input is joint torque at a selected subset of robot joints with bounded magnitudes; optional position/velocity wrappers are provided for impedance or kinematic control. Contact models (capsules/spheres) and damping are configured to prevent interpenetration while allowing compliant assistance.

2.2. Software functionalities

2.2.1. Robot and human models

The UR5e is a 6-DoF robot arm with each joint having a working range of 360° In the default scenario, the UR5e, available in MuJoCo Menagerie [14], is mounted on a cylindrical stand positioned at the approximate arm and foot height of a seated human for rehabilitation purposes (see Fig. 2).

The UR5e kinematics are based on the Denavit–Hartenberg (D-H) formulation. Each revolute joint transformation $T_{i-1}^{i}(q_i)$ is parameterized using standard D–H parameters, yielding the end-effector pose $T_0^6(q)$ as the ordered product of homogeneous transformations.

$$T_0^6(q) = \prod_{i=1}^{6} T_i^{i-1}(q_i) \tag{1}$$

Each revolute joint angle q_i corresponds directly to the actuator coordinate. Robot dynamics follow the standard rigid-body formulation

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \tag{2}$$

solved in MuJoCo using articulated-body forward dynamics, including contact interactions with the humanoid.

These combined kinematic and dynamic models allow the robot to apply compliant, physiotherapy-oriented joint torques while respecting anatomical safety bounds. The final robot wrist link is attached to the humanoid hand (or foot in lower-limb tasks), enabling controlled assistive motion.

The humanoid model used in our environment is based on the standard humanoid provided by Gymnasium, originally developed by Google DeepMind. This model, first introduced in [16], is designed to approximate human biomechanics and has been widely employed in tasks such as bipedal locomotion, including environments where the objective is to walk as fast as possible without falling [17].

For our purposes, we have adapted the Gymnasium humanoid to simulate limb rehabilitation. For upper-limb tasks, only the shoulder and elbow joints are actively engaged; for lower-limb tasks, the hip and knee joints are selectively activated. Additionally, the humanoid's hands and feet are not connected via articulated joints and thus not involved in the rehabilitation tasks. This simplification is consistent with our focus on proximal limb rehabilitation, particularly targeting motor recovery at the hip, knee, shoulder and elbow.

To ensure physiotherapy exercises remain anatomically appropriate, the motion of the UR5e robotic arm is constrained to approximate physiologically plausible ranges of the shoulder, elbow, hip, and knee joints. Accordingly, both the robot configuration space and the target end-effector poses are restricted within these bounded ranges. Table 3 provides a compact summary of joint-level mappings between human limbs and the corresponding UR5e joints, together with their effective angular limits.

Although the UR5e has six actuated joints, rehabilitation tasks primarily utilize the shoulder_lift, shoulder_pan, and elbow joints to generate the therapeutic motion. The remaining joints are used to adjust end-effector orientation and maintain stable contact. For certain movements (e.g., abduction or rotation), equivalent mappings are achieved through appropriate reorientation of the robot base, enabling coverage of a broader set of limb motions within a unified control framework.

While the high-level objective involves reaching a target joint angle, the agent is operating within a constrained manifold defined by the human patient's Range of Motion (ROM) and must find a path that achieves the physiotherapy goal without triggering a "safety termination" caused by human joint hyperextension or collision.

2.2.2. The rehabbot training and simulation environment

Using the UR5e model, different scenes of the integrated human-robot environment were developed to support different physiotherapy exercises, as illustrated in Fig. 2. In this case, the final wrist link of the UR5e robot arm is attached to the right hand of the humanoid model. The combined system consists of 27 controllable degrees of freedom (DoF), excluding the initial 7 DoF corresponding to the free-floating base of the humanoid.

Because the original humanoid model in Gymnasium was designed for locomotion, we adjusted the shoulder and elbow joints in the original humanoid description file to better reflect human biomechanics for upper- and lower-limb manipulation. Instead of using MuJoCo's default motor actuators—which apply torque in a manner inconsistent with natural human motion—we replaced them with general class actuators that allow for customized force modelling through direct parameterization of joint dynamics.

2.2.3. Reinforcement learning formal specification of the environment

Our RL-trainable environment consists of Action Space (*A*),

Table 2
Predefined Gymnasium scenarios of RehabBot.

Scenario ID	Limb	Task Type	Description	Control Mode
rehab-reach-v0	Upper	Reaching	Joint space target reaching for shoulder, elbow, hip, and knee joints	Torque
rehab-reach-l-v0	Upper	Trajectory	Trajectory following for shoulder, elbow, hip, and knee joints	Torque
rehab-hip-v0	Lower	Reaching	Hip flexion/extension	Torque
rehab-knee-v0	Lower	Trajectory	Knee rehabilitation	Torque

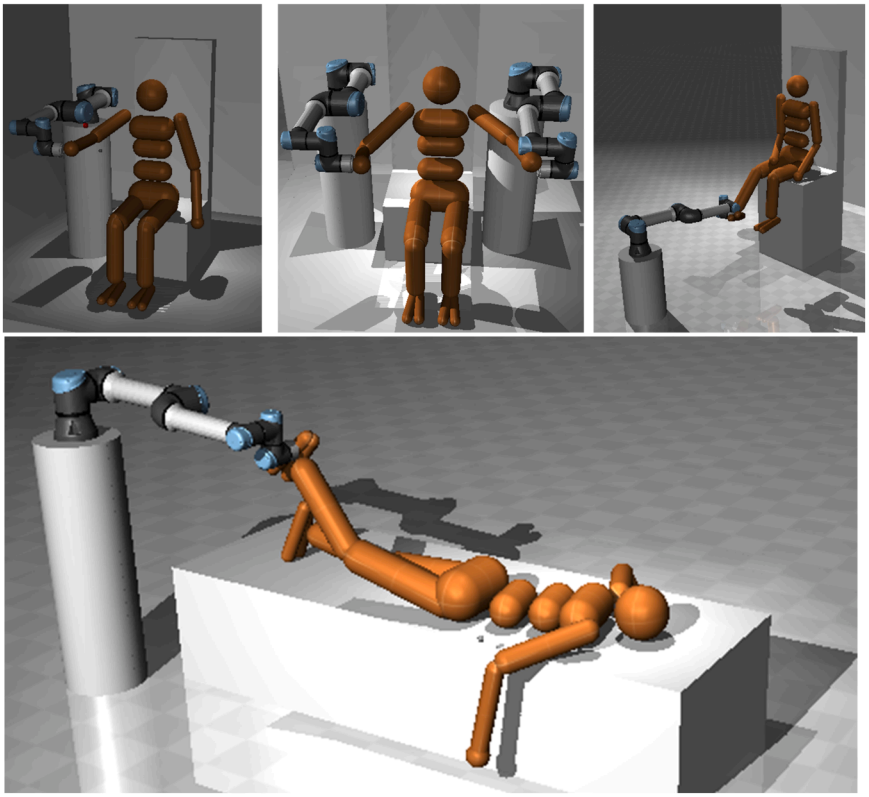

Fig. 2. Different environments of RehabBot for upper and lower limb physiotherapy exercises. In clockwise order: (1) one-arm shoulder and elbow joint exercises, (2) two-arm exercises, (3) hip joint exercises, and (4) knee and hip joint exercises.

Table 3
Summary of joint-level mappings between human limb movements and UR5e joints, with approximate physiological ranges and configuration-dependent considerations.

Joint	Human Range of Motion (rad)	Mapped UR5e Joint	Notes
Shoulder	$[-\frac{\pi}{2}, +\frac{3\pi}{4}]$	shoulder_lift	Covers flexion/extension/ abduction (orientation-dependent)
Shoulder	$[-\frac{\pi}{2}, +\frac{\pi}{2}]$	shoulder_pan	Internal/external rotation
Elbow	$[-\frac{3\pi}{4}, 0]$	elbow	Flexion/extension
Hip	$[-\frac{\pi}{6}, +\frac{\pi}{2}]$	shoulder_lift	Mapped via reorientation
Hip	$[-\frac{\pi}{3}, +\frac{\pi}{6}]$	shoulder_pan	Rotation (configuration-dependent)
Knee	$[-\frac{3\pi}{4}, 0]$	shoulder_lift	Flexion/extension

Observation Space (*S*), Rewards (*R*) and Termination conditions. In this section, we fully describe the design of these sets for the upper-limb rehabilitation domain, as the lower-limb counterpart is analogous.

Action Space: The action space is composed of 6 actions a_i, that can take maximum angular torques $\tau_{max} = \frac{\pi}{400}$ applied at the 6 hinge joints of the UR5e in each decision.

$$\begin{aligned} &A \subset \mathbb{R}^6 \\ &a_i \in [-\tau_{max}, \tau_{max}] \end{aligned} \tag{3}$$

Observation Space: The observation space consists of 23 elements:

- θ^* *(6 elements)*: Target angles for the UR5e joints according to the rehab exercise.
- p^* *(3 elements)*: Target pose for the UR5e according to the rehab exercise.
- θ_t *(6 elements)*: Current joint angles of the UR5e robot at time t.
- p_t *(3 elements)*: Current pose of the UR5e robot at time t.
- ϕ_t *(3 elements)*: Current joint angles of the humanoid's arm joints at time t.
- e *(1 element)*: Type of rehabilitation exercise being carried out (binary value).
- ℓ *(1 element)*: An integer from 0 to 4 indicative of the amount of weight of the humanoid right arm that the robot has to lift at each episode so that the robot can learn to adapt to lifting arms of varying weights.

Therefore, each observation o_t at timestep t can be specified as:

$$\begin{aligned} &o_t = (\theta^*, p^*, \theta_t, p_t, \phi_t, e, \ell) \\ &o_t \in \mathbb{R}^6 \times \mathbb{R}^3 \times \mathbb{R}^6 \times \mathbb{R}^3 \times \mathbb{R}^3 \times \{0,1\} \times \{0,1,2,3,4\} \end{aligned} \tag{4}$$

Rewards: To facilitate the learning of the rehabilitation sequence, the reward function R_t at each timestep t is defined as a combination of a dense tracking penalty and a sparse sequencing bonus. Let θ_t represent the joint configuration of the robot at time t, and θ_i^* represent the i-th target configuration in a sequence of N targets ($i \in \{0,1,\ldots,N-1\}$). The reward is formulated as:

$$R_t = -\| \theta_t - \theta_i^* \|_2 + I\{\| \theta_t - \theta_i^* \|_\infty < \epsilon\} B_i \tag{5}$$

where $\| . \|_2$ is the Euclidean norm representing the distance to the active target, and $I\{.\}$ is an indicator function that equals 1 if the target is reached and 0 otherwise. A target is considered reached when the condition of the indicator function

$$\| \theta_t - \theta_i^* \|_\infty < \epsilon$$

is satisfied, where $\| . \|_\infty$ is the infinity norm, ensuring that all joint coordinates are within the specified tolerance $\epsilon = 0.05$ rad. The target-sequencing bonus B_i is defined as:

$$B_i = \begin{cases} 10, & \text{if } i < N-1 \\ 100, & \text{if } i = N-1 \end{cases}$$

Upon reaching an intermediate target ($i < N-1$), the index i is incremented, and the agent begins tracking the next configuration in the sequence. If the final target ($i = N-1$) is reached, the episode is terminated with a higher bonus.

Episode Ends: There are two situations that end an episode:

- *Termination*: The environment terminates when the final target in the sequence is reached.
- *Truncation*: The default maximum duration of an episode is 1000 timesteps to achieve the goal.

According to this design, the action, observation, and reward variables follow a standard goal-directed RL formulation with rehabilitation-specific constraints. Torque-bounded actions ensure physically consistent and safe control, while the observation space combines target and current states with humanoid and task information to capture human–robot interaction. The reward combines a dense distance-based tracking term with a sparse target-sequencing bonus, providing a stable learning signal. These components are configurable: users may extend observations, redefine rewards (e.g., adding safety or smoothness terms), or adapt control modes and task parameters.

2.2.4. Evaluation of the functionality

The evaluation aims to verify that RehabBot functions as a stable, reproducible, and RL-trainable simulation environment. We assess its correctness by training agents using established deep RL algorithms (PPO, TD3, and SAC). Successful convergence across on-policy and off-policy methods indicates coherent state transitions, consistent reward signals, stable training behaviour, and reliable termination conditions. These experiments therefore constitute a software validation procedure demonstrating compatibility and robustness within standard RL pipelines, rather than a comparative study of algorithmic performance.

Python-based simulation scripts were developed in which the UR5e executes predefined rehabilitation trajectories within a Gymnasium-wrapped environment using Stable-Baselines3. The UR5e acts as the learning agent, interacting with the humanoid model to perform goal-directed physiotherapy exercises.

A visual representation of a reaching task is shown in Fig. 3, where the robot's end-effector trajectories are rendered in purple. These trajectories illustrate the robot's performance over multiple episodes of the reaching task.

To show that our RehabBot environment is RL-trainable, three different DRL algorithms were explored (Fig. 4): Twin Delayed Deep Deterministic policy gradient (TD3), Proximal Policy Optimization (PPO) algorithm and Soft Actor-Critic (SAC). We used the implementation provided by the well-known and widely-used StableBaseline3 library. PPO shows near-deterministic convergence, which is strong evidence for numerical stability. The shaded region is narrow, indicating extremely low variance across 5 different seeds. SAC shows higher variance, which is typical for off-policy maximum entropy algorithms. However, the fact that the mean consistently trends upward, despite the variance, indicates that the environment is "learnable" and robust. After 1 million timesteps, all three algorithms reach a stable plateau. A demonstration video is available in [15].

In relation to the upper-limb, the velocities are continuous and bell-shaped as illustrated in Fig. 5 (left), without step-function discontinuities. In physiotherapy, bell-shaped velocity profiles are a hallmark of natural human movement. This suggests the robot is providing gradual assistance rather than jerky, mechanical impulses. Note that the peak velocity stays below 0.5 rad/s. The trained policy naturally respects safe speed limits without requiring manual overrides. For the joint accelerations, there is some oscillation, but it remains within a bounded range (1.0 rad/s^2). This indicates controlled braking and acceleration. Since it does not spike to extreme values, the motor torque is being applied progressively, reducing the risk of sudden "shocks" to the patient's limb. For the joint jerk, transient peaks are observed during initialization and at isolated points throughout the exercise. However, these peaks are brief and not sustained; for most of the trajectory, the jerk remains bounded and relatively low in amplitude, indicating minimal sustained jerk and supporting smooth, safe interaction during patient mobilization.

In addition to the upper-limb results, the lower-limb rehabilitation environments exhibit performance characteristics consistent with the upper-limb results: the agent produces smooth and safe trajectories in both cases as illustrated in Fig. 5 (right). The lower limb environment exhibits velocity, acceleration, jerk, and torque profiles that mirror the safety and smoothness characteristics observed in the upper limb validation. The reinforcement learning algorithm adapts to the increased inertia of the lower limb model, producing stable, continuous trajectories that remain within the established safe limits for clinical rehabilitation, suggesting that the proposed framework is robustly capable of providing stable mobilization for both the shoulder/elbow and the hip/knee complexes.

Therefore, although the RL agents were trained using a distance-based reward, the post-training kinematic analysis shows that the resulting policies exhibit emergent smoothness. These results support the evidence that RehabBot successfully translates simple objectives into smooth and physiologically viable trajectories.

To ensure the safety of the human-robot interaction, we monitored the generalized actuator forces. As shown in Fig. 6, the torques applied by the UR5e remain bounded throughout the rehabilitation task, reducing the likelihood of sudden impulses that could lead to patient discomfort or injury. Apart from short initial transients and the final saturation event, the torques stay within a reasonable range (e.g., ± 0.25 Nm). These values are well within the safe physical limits of both the robot and the human patient's tolerance. These torque profiles indicate stable and compliant control behaviour, which is desirable for safe

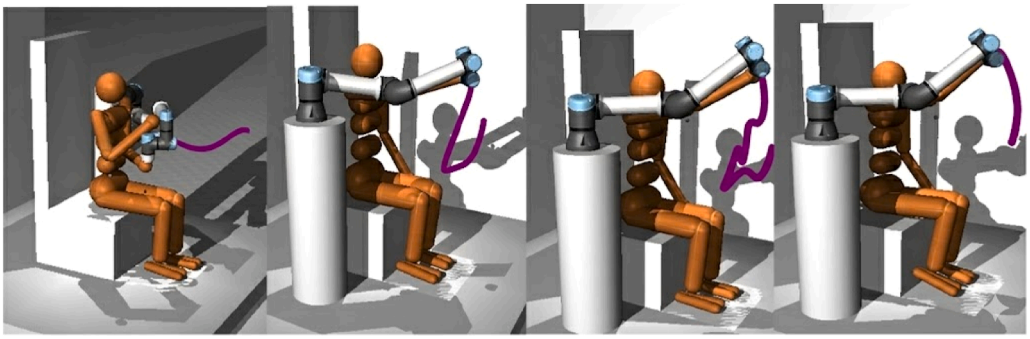

Fig. 3. Purple line shows the path taken by RehabBot during rehabilitation exercises for the shoulder and elbow joints rehabilitation when connected to both right and left hands. The path shows movement in both sagittal and frontal planes.

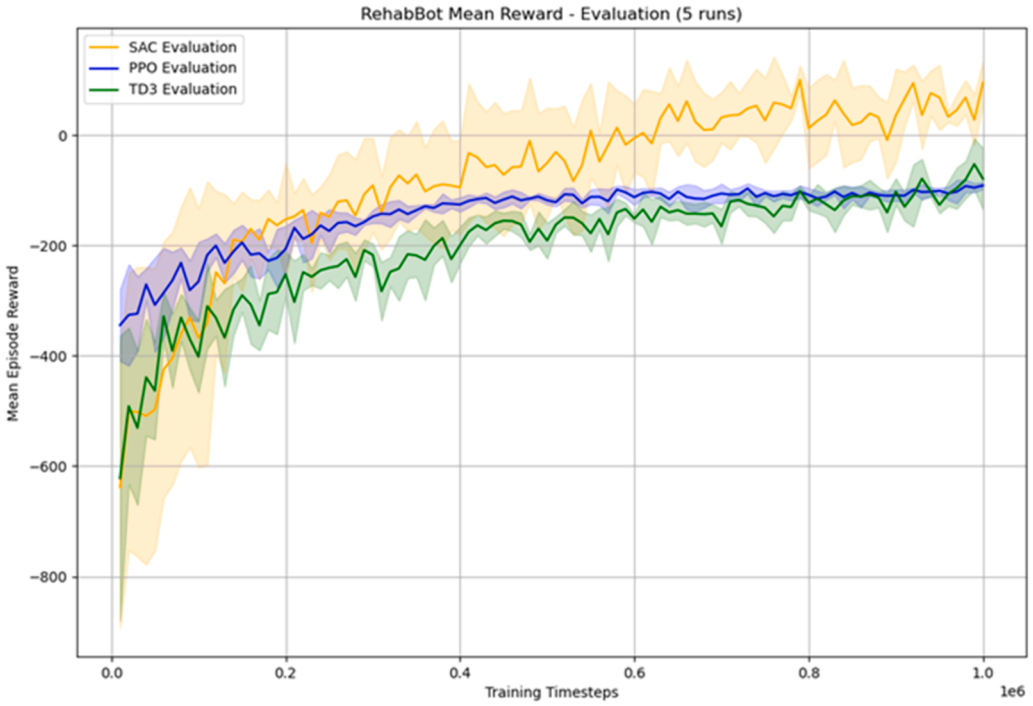

Fig. 4. Graph showing 5 different training logs for SAC, PPO and TD3 policies.

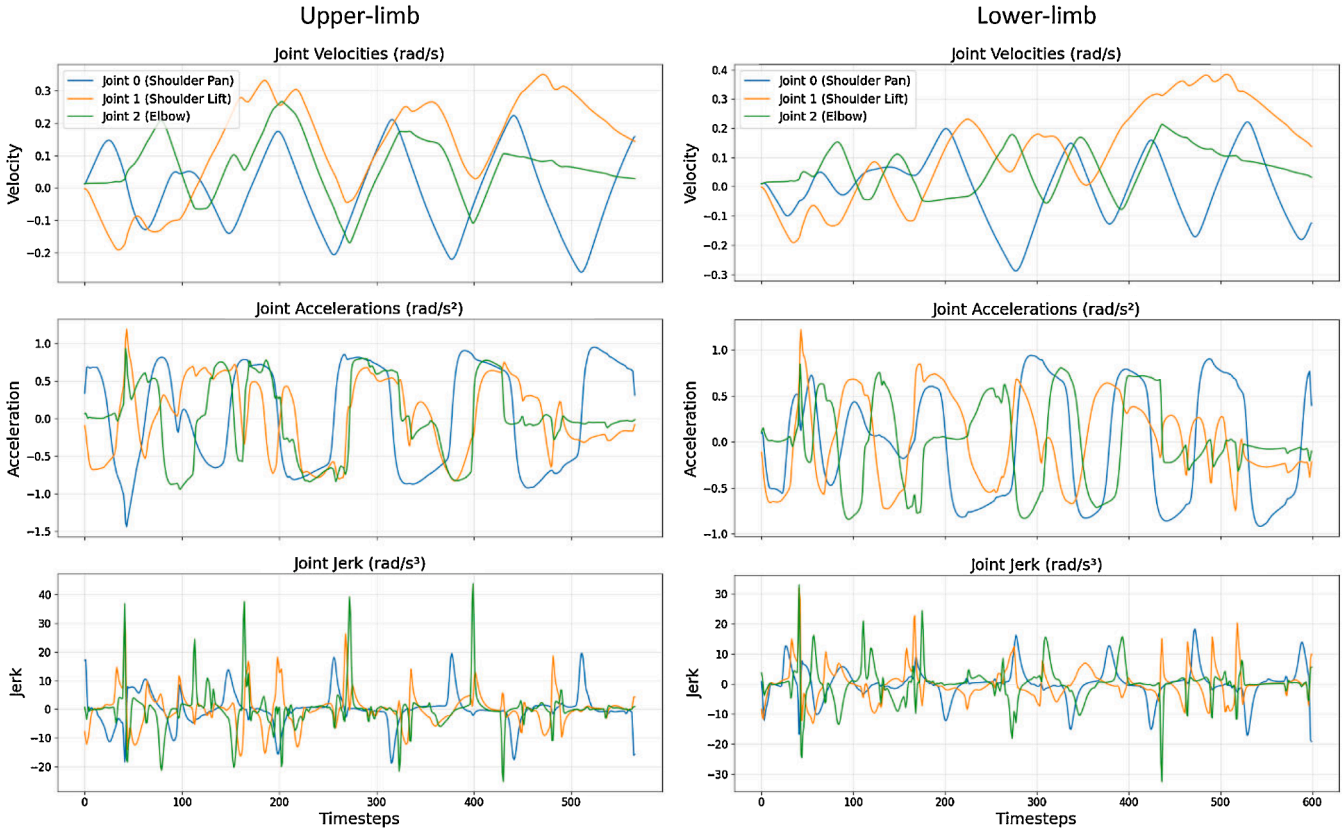

Fig. 5. Kinematic analysis of the velocity, acceleration and jerk of the robot during rehabilitation for the upper-limb (left) and lower-limb (right).

patient-oriented interaction. Furthermore, physical safety is inherently limited by the actuator torque saturation of ± 1.0 Nm, which prevents the robot from exerting forces that could cause structural or musculoskeletal damage to the articulated patient model.

3. Illustrative examples

This section provides examples of several training tests using the RehabBot environment.

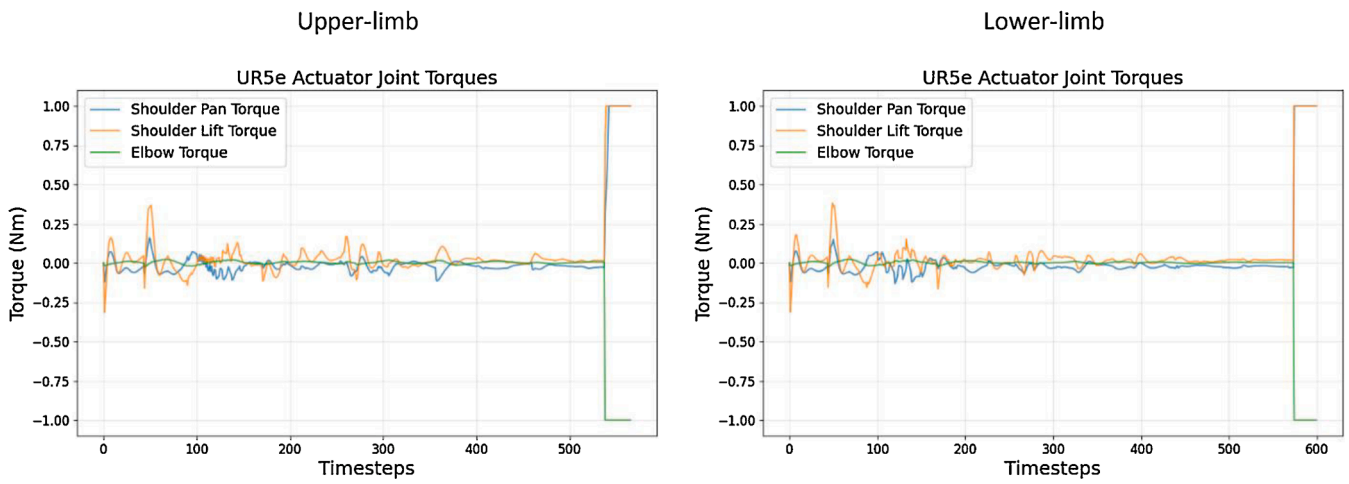

Fig. 6. UR5e actuator joint torques during rehabilitation for the upper-limb (left) and lower-limb (right).

Listing 1. Executing a basic visualization test.

```
import gymnasium as gym
  import rehabbot # registers env IDs

  env = gym.make("rehabbot/rehab-reach-v0",
  render_mode="human")
  obs, info = env.reset(seed=0)

  done = False
  while not done:
    action = env.action_space.sample()
    obs, reward, terminated, truncated, info = env.step(action)
    done = terminated or truncated
  env.close()
```

Listing 2. Training a policy using the SAC algorithm.

```
import gymnasium as gym
  from stable_baselines3 import SAC
  from stable_baselines3.common.evaluation import
  evaluate_policy
  import rehabbot

  env = gym.make("rehabbot/rehab-reach-v0", render_mode=None)

  model = SAC("MlpPolicy",env,verbose=1,
    tensorboard_log="./UR5e_tensorboard/")
  model.learn(total_timesteps=200_000)
```

Listing 3. Evaluating an already trained policy.

```
import gymnasium as gym
  from stable_baselines3 import SAC
  from stable_baselines3.common.evaluation import
  evaluate_policy
  import rehabbot

  env = gym.make("rehabbot/rehab-reach-v0", render_mode=None)

  model = SAC.load("SAC-1.5M-R-2.5", env=env,
  print_system_info=True)

  mean_reward, std_reward = evaluate_policy(model, model.get_env
  (),
    n_eval_episodes=5)

  print(f"Mean reward: {mean_reward:.2f} ± {std_reward:.2f}")
```

4. Impact

As discussed in the review of the state of the art, while several simulation environments have been developed for assistive tasks such as itch scratching, drinking, feeding, dressing, and bathing, no dedicated simulation environment currently exists for delivering limb physiotherapy exercises within a reinforcement learning framework. RehabBot addresses this gap by providing an innovative RL-trainable, physics-based simulation specifically designed for knee, hip, shoulder and elbow rehabilitation.

A key potential impact of this work is its seamless integration with established robot learning frameworks. This interoperability can facilitate broader adoption and extension of the environment within both academic research and clinical development pipelines.

Beyond its integration potential, the environment supports the development of personalized rehabilitation strategies. By leveraging the observation space for patient-specific monitoring and feedback, therapy plans can be dynamically adapted to individual progress and biomechanical constraints. This capability may facilitate future research on adaptive rehabilitation strategies and personalized therapy planning. Moreover, by offering more comfortable, interactive, and data-driven therapy sessions, the environment may contribute to higher patient adherence to prescribed rehabilitation protocols, an essential factor in long-term recovery success.

5. Conclusions

This paper has introduced RehabBot, a reinforcement learning-trainable simulation environment for limb physiotherapy, designed to align closely with real-world rehabilitation setups. The environment enables evaluation of RL algorithms for delivering therapeutic exercises targeting shoulder, elbow, hip, and knee rehabilitation. By integrating a commercially available UR5e robot with a modified humanoid model in a physics-based MuJoCo simulation, we provide a reproducible and extensible platform for advancing research in human-robot interaction for rehabilitation.

A distinguishing feature of this work is the use of the UR5e, a widely available, cost-effective collaborative robot, rather than a custom-built rehabilitation platform. This design choice not only lowers barriers to adoption in clinical and research contexts but also facilitates real-world deployment by bridging the gap between simulation and physical implementation.

While RehabBot provides an RL-compatible framework for robot-assisted rehabilitation, the current version abstracts several aspects of real therapy. In particular, it does not include detailed musculoskeletal or muscle dynamics, and rehabilitation tasks are modelled as simplified target-reaching objectives rather than clinically validated protocols. These simplifications also introduce challenges for sim-to-real transfer, including discrepancies between simulated and real-world dynamics,

unmodeled contact effects, and the need for careful system calibration and validation. Addressing these limitations defines the current scope of the environment and motivates the future developments.

Future work will focus on enhancing the physiological realism and versatility of the environment, as well as supporting sim-to-real transfer. One promising direction involves the integration of musculoskeletal models, such as those provided in MyoSuite [7], to enable detailed study of muscle activation and force dynamics during robot-assisted exercises. This would allow for deeper investigation into biomechanics-informed control, improved model fidelity, and more reliable transfer to physical systems through calibration and validation.

Additionally, the modularity of the environment supports adaptation to other robot models available, allowing exploration of alternative manipulators for limb rehabilitation. Further extensions could include the simulation of bilateral (two-arm) rehabilitation protocols and more complex functional tasks such as lifting, pick-and-place, and object assembly—thereby broadening the scope from pure rehabilitation to recovery-focused functional motor tasks.

A comprehensive documentation and open-source implementation of RehabBot is available on GitHub [19], including installation instructions, dependency specifications, example scripts, and test cases. These resources support reproducibility of the reported experiments and facilitate community-driven development and extension of the environment.

CRediT authorship contribution statement

Victoria Oguntosin: Writing – original draft, Visualization, Software, Resources, Methodology, Investigation, Formal analysis. **Juan-Ignacio Vazquez:** Writing – review & editing, Supervision, Project administration, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Victoria Oguntosin reports financial support was provided by Women for Africa Foundation. The other author declares no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the Science by Women programme of the Fundacion Mujeres por África.

References

[1] Amina Nalongo J. Robotics in physical therapy: enhancing patient outcomes. Res Output J Eng Sci Res 2025;4(2):87–94. https://doi.org/10.59298/ROJESR/2025/4.2.8794.
[2] Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N. Stable-Baselines3: reliable reinforcement learning implementations. J Mach Learn Res 2021;22(268):1–8.
[3] Liang E, Liaw R, Nishihara R, Moritz P, Fox R, Goldberg K, Gonzalez J, Jordan M, Stoica I. RLlib: abstractions for distributed reinforcement learning. In: Proceedings of the International Conference on Machine Learning. PMLR; 2018. p. 3053–62.
[4] Huang S, Fernand Julien Dossa R, Ye C, Braga J, Chakraborty D, Mehta K, Araújo JGM. CleanRL: high-quality single-file implementations of deep reinforcement learning algorithms. J Mach Learn Res 2022;23(274):1–18.
[5] Todorov E, Erez T, Tassa Y. MuJoCo: a physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2012.
[6] K. Zakka, Y. Tassa, MuJoCo Menagerie: a collection of high-quality simulation models for MuJoCo, 2022.
[7] V. Caggiano, H. Wang, G. Durandau, M. Sartori, V. Kumar, MyoSuite—A contact-rich simulation suite for musculoskeletal motor control, arXiv preprint arXiv:2205.13600, 2022.
[8] Erickson Z, Gangaram V, Kapusta A, Liu CK, Kemp CC. Assistive Gym: a physics simulation framework for assistive robotics. In: Proceedings of the International Conference on Robotics and Automation (ICRA); 2020.
[9] M. Plappert, et al., Multi-goal reinforcement learning: challenging robotics environments and request for research, arXiv preprint arXiv:1802.09464 2018.
[10] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, Y. Zhu, robosuite: a modular simulation framework and benchmark for robot learning, arXiv preprint arXiv:2009.12293; 2020.
[11] Kumar V, Shah R, Zhou G, Moens V, Caggiano V, Gupta A, Rajeswaran A. Robohive: a unified framework for robot learning. Adv Neural Inf Process Syst 2023;36:44323–40.
[12] Tan CK, Wang C, Lyu S, Hodossy BK, Schumacher P, Wilson EB, et al. Myoassist 0.1: myosuite for dexterity and agility in bionic humans. In: 2025 International Conference on Rehabilitation Robotics (ICORR); 2025. p. 437–42.
[13] Thumm J, Trost F, Althoff M. Human-robot gym: benchmarking reinforcement learning in human-robot collaboration. In: 2024 IEEE International Conference on Robotics and Automation (ICRA); 2024. p. 7405–11.
[14] M. Nazarczuk, K. Stepanova, J.K. Behrens, M. Hoffmann, K. Mikolajczyk, MuBlE: muJoCo and Blender simulation environment and benchmark for task planning in robot manipulation, arXiv preprint arXiv:2503.02834; 2025.
[15] RehabBot: mujoco-based rehabilitation environment for upper-limb physiotherapy-based reaching tasks, YouTube, 2025, https://youtube.com/shorts/kQRI4Jr-AR8 (accessed June 16, 2025).
[16] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, et al., DeepMind control Suite, arXiv preprint arXiv:1801.00690; 2018.
[17] Tassa Y, Erez T, Todorov E. Synthesis and stabilization of complex behaviors through online trajectory optimization. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems; 2012. p. 4906–13. https://doi.org/10.1109/IROS.2012.6386025.
[18] M. Towers, A. Kwiatkowski, J. Terry, J.U. Balis, G. De Cola, T. Deleu, M. Goulao, A. Kallinteris, M. Krimmel, A.K. G, R. Perez-Vicente, Gymnasium: a standard interface for reinforcement learning environments, arXiv preprint arXiv:2407.17032; 2024.
[19] V. Oguntosin, RehabBot, GitHub, 2025, https://github.com/vickkiee/rehabbot (accessed June 16, 2025).