Learning to Walk with Hybrid Serial-Parallel Linkages: a Case Study on the Kangaroo Robot

***Abstract*—Humanoid robots increasingly adopt hybrid serial-parallel kinematics to improve structural stiffness, mass distribution, and impact robustness. However, these mechanisms introduce complexity associated with simulation and control, which impacts algorithms for Reinforcement Learning (RL)–based locomotion. This paper presents a case study of an RL end-to-end pipeline that trains walking policies for Kangaroo, a 72 degrees of freedom biped whose legs contain several hybrid serial-parallel chains, without kinematic simplifications. Training is performed using the *Isaac Lab* framework, leveraging the *Isaac Sim*™ built-in constraint capabilities. An ablation study on the observation state is carried out to find evidence in the use of redundant information from the measured state of the robot, i.e., using the passive and/or active joint measurements available in Kangaroo. Validation of a set of trained policies is done in *MuJoCo*, demonstrating a degree of robustness to the *Sim-to-Sim* gap. Closed-loop behaviors successfully transfer for most of the trained policies when equality-constraint stiffness and other simulation parameters are properly tuned. Furthermore, we analyze the impact that policy action rates have on the effectiveness of the *Sim-to-Sim* deployment.**

I. Introduction

In recent years, humanoid bipedal robots have moved beyond laboratory settings into real-world environments. Significant advancements in control systems and mechatronic design have enabled this transition. In particular, Reinforcement Learning (RL) has become essential for achieving robust locomotion in unstructured environments [1], [2], while, on the mechatronics side, there has been a growing adoption of hybrid serial-parallel linkages in robot design.

These mechanisms enable higher structural stiffness and improved dynamic performance by relocating heavy components, such as actuators, closer to the robot's base, thereby reducing the inertia during impacts that happen during locomotion. Such structures can be observed in most modern humanoid robots, including Cassie [3], Kangaroo [4], the Disney bipedal robot [5], and Apptronik Apollo[1], to name a few.

In recent years, the robotics humanoid community has started to show interest in the explicit modeling of hybrid serial-parallel kinematic chains present in bipedal systems for both simulation and control [4], [6]–[10]. For control purposes, all these works aim to model hybrid serial-parallel mechanisms with high fidelity to account for hardware-induced limitations and input nonlinearities, while, at the same time, improving the computational efficiency. However, the modeling and control of hybrid serial-parallel mechanisms still remain a challenge due to their inherent complexity. To cope with this, such mechanisms are often simplified, ignoring specific constraints and dynamic behaviors of the system [11]. This can result in suboptimal control strategies or inaccurate simulations.

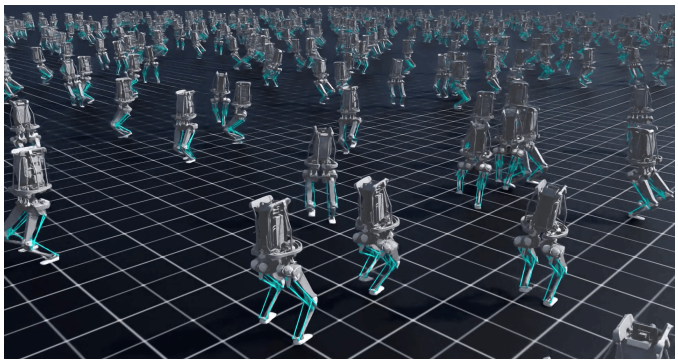

Fig. 1: Multiple Kangaroo robots simulated in *Isaac Sim*™ including serial-parallel closed linkages.

In the context of RL, the possibility of explicitly including hybrid serial-parallel mechanisms is strictly related to the simulation tools used for training. At the time of writing, both *Isaac Sim*™ [2] and *MuJoCo* [12], among the most widely used simulators for RL, support hybrid serial-parallel kinematic chains. Nevertheless, RL-based solutions for robots with hybrid serial-parallel linkages remain uncommon, largely due to the increased computational demands these mechanisms impose. Moreover, it is uncommon for control policies to be trained directly in motor space (the space of active, i.e., actuated joints) [13]; instead, simplified models are often used [14], [15], and their outputs are subsequently transformed into motor space through dedicated routines that are not considered during training [16]. It must also be considered that commonly robot descriptions come as URDF models, which do not support natively the description of hybrid serial-parallel kinematic chains [17], posing an additional practical obstacle.

Motivated by these challenges, this paper presents the pipeline we developed to train locomotion policies for the Kangaroo, a bipedal humanoid robot developed by PAL Robotics, which incorporates numerous hybrid serial-parallel mechanisms in its leg design. Beyond reporting performance outcomes, we place particular emphasis on the technical modeling choices and implementation details that enable successful policy training in such complex systems. The robot is modeled within the NVIDIA *Isaac Sim*™ simulator to take advantage of GPU acceleration [2], while policy training is performed using Proximal Policy Optimization (PPO) [18] with a standard

[1]Apptronik Apollo: https://apptronik.com/apollo

[2]NVIDIA *Isaac Sim*™: https://developer.nvidia.com/isaac/sim

Fig. 2: (Left) Kangaroo robot; (Right) its model in *Isaac Sim*™. The collision meshes are shown in green: convex hulls of the mesh links are used for the torso, the hip, the thigh, and the tibia, while a box mesh is used for the foot.

locomotion reward formulation within the *Isaac Lab* robot learning framework [19] (Figure 1 shows an example of multiple robots simulated in parallel). To assess robustness and transferability, we deployed the trained policies in *MuJoCo*, evaluating a set of *Sim-to-Sim* validation experiments to compare the performance observed in the two simulation environments, as well as confirming each simulator's ability to maintain closed-loop kinematic constraints. Moreover, two ablation studies are presented to provide some more in-depth insight on key factors to take into account during policy training and *Sim-to-Sim* deployment. The first analyzes the training performance obtained by considering different parts of the joint state within the observation space (a study particularly relevant in our case, given the high number of joints present in the Kangaroo robot). The second focuses on the *Sim-to-Sim* deployment, assessing the impact of the action rate when transferring policies in a different simulator.

The remainder of this paper is organized as follows. Section II details the modeling of the Kangaroo robot, with a focus on its hybrid serial-parallel kinematic structure and how it is represented in simulation. Section III describes the RL framework used to train locomotion policies, defining the observation and action spaces, reward design, and domain randomization settings. Policy training results are presented in Section IV, while Section V analyzes the *Sim-to-Sim* deployment. Finally, Section VI draws the conclusions and discusses potential future research directions.

II. Modeling the Kangaroo Robot

Kangaroo is a bipedal humanoid robot developed by PAL Robotics, featuring a unique leg design based on hybrid serial-parallel closed linkages actuated by high-power linear actuators (Figure 2). The full model of Kangaroo includes a total of 72 degrees of freedom (DOFs), comprising 60 rotational passive (i.e., not actuated) and 12 linear active DOFs. Each leg of Kangaroo is composed of four kinematic groups, as described in [4]:

- 1 DOF *Hip External/Internal Rotation*, actuated by **Motor 1** (M1);
- 2 DOFs *Hip Flexion/Extension & Abduction/Adduction*, actuated by **Motors 2 and 3** (M2, M3);
- 1 DOF *Knee Flexion/Extension*, actuated by the **Leg Length Motor** (LLM);
- 2 DOFs *Ankle Dorsiflexion/Plantarflexion & Inversion/Eversion*, actuated by **Motors 4 and 5** (M4, M5).

These groups are illustrated in Figure 3, with the possible motions of each leg sub-mechanism. It is worth noting that below the knee, no electronics are present, making Kangaroo a robust platform for locomotion and agile actions. A distinctive characteristic of the Kangaroo's leg is that, during extension or retraction via the LLM, the relative orientation between the hip and the ankle remains constant.

From a sensing perspective, Kangaroo is equipped with an IMU located at the waist, 10 rotational absolute encoders (5 per leg) placed on selected passive joints, and 2 linear absolute encoders (1 per leg) installed on the LLM actuators (see Figure 3). Joint force sensing is achieved through a combination of motor current measurements and dedicated force sensors.

In the real Kangaroo, because not all linear actuators are equipped with absolute encoders, a calibration procedure is required each time the robot is started. By leveraging the known kinematics of the closed chains and the readings from the absolute rotational encoders, it is possible to initialize the positions of the linear actuators. This enables position control in addition to velocity and force control. Table I presents the actuator limits for each closed sub-mechanism.

TABLE I: Actuator limits and starting configurations for each closed sub-mechanism in Kangaroo's lower body

Closed sub-mechanism	*Min [m]*	*Max [m]*	*Start [m]*
hip yaw	-0.02	0.02	0.0
hip roll/pitch	-0.04	0.04	0.0
knee	0.00	0.15	0.0375
ankle roll/pitch	-0.02	0.02	0.0

A. Simulation

The hybrid serial-parallel closed linkages are modeled as kinematic chains that are "opened" at specific passive joints, with constraints applied to enforce closure. In the case of the Kangaroo, a total of 60 DOFs, corresponding to the passive joints, must be constrained to accurately simulate the closed-chain behavior. The implementation of these constraints varies depending on the simulator.

Isaac Sim™ permits the modeling of hybrid serial-parallel kinematic chains by the inclusion of passive joints to *close* the open chains. In particular, we started from a URDF description of Kangaroo, which includes all the DOFs but the ones that close the chains. We imported it into the *Isaac Sim*™, and we included manually, through the GUI, the missing passive joints in the model (described using the Universal Scene Description (USD) format[3]), thereby enabling the representation of closed

[3]USD: https://openusd.org

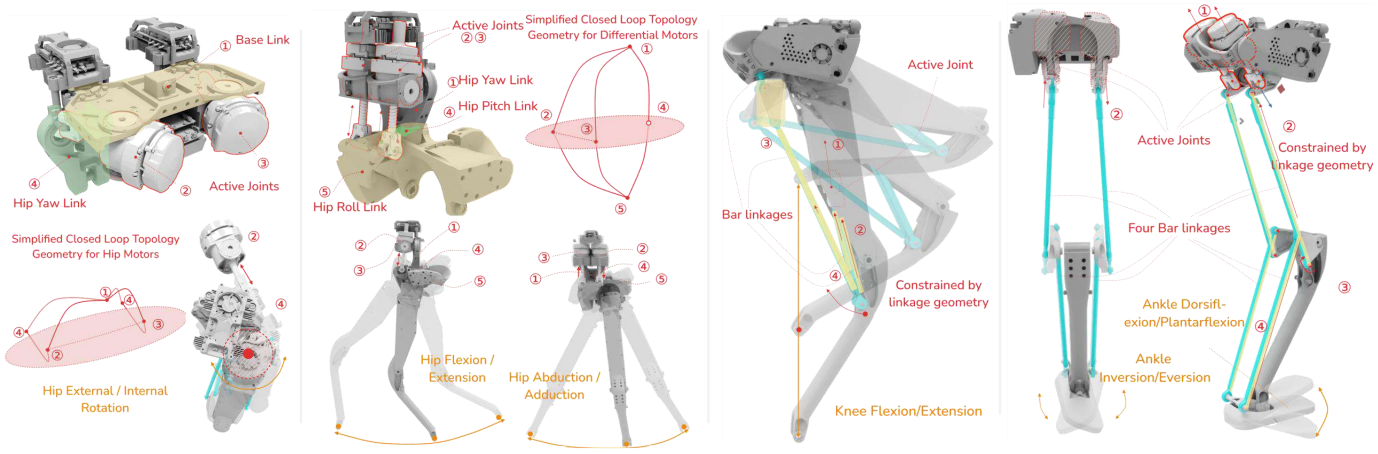

Fig. 3: Kangaroo kinematic groups, from left to right: Hip External/Internal Rotation, Hip Flexion/Extension & Abduction/Adduction, Knee Flexion/Extension, and Ankle Dorsiflexion/Plantarflexion & Inversion/Eversion.

TABLE II: Actuators parameters (simulation).

Motor	K_p $[N/m]$	K_v $[N \cdot s/m]$	*Force limit* $[N]$
Motor 1	$3 \cdot 10^5$	1095	3000
Motor 2	$7 \cdot 10^5$	1673	3000
Motor 3	$7 \cdot 10^5$	1673	3000
Motor 4	$8 \cdot 10^5$	1789	3000
Motor 5	$8 \cdot 10^5$	1789	3000
Leg Length Motor	$10 \cdot 10^5$	2000	5000

kinematic chains. *Isaac Sim*™ permits also using *articulation* to achieve better performance. However, articulations do not support the direct representation of closed-loop kinematic structures. To address this limitation, one joint was excluded from each closed kinematic chain in the articulation hierarchy. Despite removing them from the articulations, the passive joints are still present in the robot description, ensuring closure in the hybrid serial-parallel mechanisms. Actuated joints are controlled through simulated position controllers defined by Table II parameters, and a simulation time step of 0.005 s has been used. These values have been tuned by hand to obtain an effective reference tracking for the actuated linear joints and overall stability of the simulated closed chains.

MuJoCo supports closed loops by enabling explicit constraints between different bodies. Hence, Kangaroo's closed kinematic chains have been implemented using a series of `equality connect` constraints for each kinematic group of both legs. Proper tuning of different simulation parameters has been fundamental to maintain the constraints and the overall stability of the *MuJoCo* simulation. Notably, the solver parameters for equality constraint simulation, i.e., the `solref`, comprising two values: `timeconst` and `damping_ratio`, representing the time constant and damping ratio of the virtual spring-damper system enforcing the constraint, respectively. We found that with `time_constant` value of 0.005 and a `damping_ratio` of 1.0 (critically damped), we were able to simulate the Kangaroo system maintaining the integrity of the closed-chain structure. Regarding the actuation, also in *MuJoCo* we adopted the same position control parameters of Table II, empirically verifying the achievement of a valid reference tracking. To increase numerical stability, we set the simulation time step to 0.001 s, a smaller value w.r.t. the time step used in *Isaac Sim*™.

Finally, in both simulators, collisions were modeled using convex hulls of selected links, without including all the links forming each sub-mechanism to avoid internal collisions that could interfere with joint motion through undesired internal forces. Hence, most non-structural elements, such as leg bars, motor rods, and fragmented link components, were excluded from collision modeling. Only the primary structural components of the legs, the feet, and the torso, see Figure 2 (Right), were included in the collision model.

III. Method

The locomotion control problem is formulated as a classical RL problem where an agent, at each time step t, receives an observation $\boldsymbol{o}_t$ from the environment, selects a control action $\boldsymbol{a}_t$, and collects a reward r_t. Control actions are sampled from a stochastic policy $\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{o}_t)$ at a frequency of 50 Hz. In the considered locomotion setting, the primary objective is to find a policy able to minimize the difference between the robot's actual base velocity and a commanded base velocity. For simplicity, we are considering only the two linear components along the x and y axes in the gravity-aligned robot frame, denoted as $\bar{\boldsymbol{v}}^{xy}$, and the angular velocity around the world frame z axis (yaw), denoted as $\bar{\phi}$. We simulate a Kangaroo robot standing on a flat terrain, initialized with the joints configured in the actuators' starting positions reported in Table I. The duration of each episode is 20 s, at the beginning of which, a random velocity command is sampled from uniform distributions for the robot to follow. Linear velocity references along the x axis are sampled from $\mathcal{U}(0, 1)$ $[m/s]$, while references along the y axis from $\mathcal{U}(-1, 1)$ $[m/s]$ (where $\mathcal{U}(a, b)$ denotes the continuous uniform probability distribution between the minimum a and maximum b). Angular velocity references are instead updated at each policy update steps according to the current orientation error w.r.t. to a target

heading angle sampled from $\mathcal{U}(-\pi, \pi)$ $[rad]$, and clipped between the min-max interval $(-1, 1)$ $[rad/s]$. The episode terminates in case the robot falls, when its base link height goes below a threshold of $0.6\ m$.

We use the PPO algorithm to train the policy (adopting the *RSL RL* implementation[4]), leveraging multiple parallel *Isaac Sim*™ simulations and GPU acceleration provided by the *Isaac Lab* framework[5]. In the following, we provide a more detailed description of the action and observation spaces, the rewards, and the domain randomization settings.

A. Observation Space

We are considering the following components to define the observation vector $\boldsymbol{o}_t$:

- $\boldsymbol{v}_t \in \mathbb{R}^3$, base linear velocity (in body frame);
- $\boldsymbol{\omega}_t \in \mathbb{R}^3$, base angular velocity (in body frame);
- $\tilde{\boldsymbol{g}}_t \in \mathbb{R}^3$, gravity direction projected on the body frame;
- $\bar{\boldsymbol{v}}_t^{xy} \in \mathbb{R}^2$, base linear velocity command (xy axes);
- $\bar{\phi}_t \in \mathbb{R}$, base angular velocity command (yaw);
- $\boldsymbol{q}_t^p \in \mathbb{R}^{10}$, position of the measurable passive joints;
- $\dot{\boldsymbol{q}}_t^p \in \mathbb{R}^{10}$, velocity of the measurable passive joints;
- $\boldsymbol{q}_t^{ll} \in \mathbb{R}^2$, position of the LLM;
- $\dot{\boldsymbol{q}}_t^{ll} \in \mathbb{R}^2$, velocity of the LLM;
- $\boldsymbol{q}_t^m \in \mathbb{R}^{10}$, position of M1, M2, M3, M4, M5;
- $\dot{\boldsymbol{q}}_t^m \in \mathbb{R}^{10}$, velocity of M1, M2, M3, M4, M5;
- $\boldsymbol{a}_{t-1} \in \mathbb{R}^{12}$, actions applied at the previous step.

It is worth noticing that the full joint state of Kangaroo is $\boldsymbol{q} \in \mathbb{R}^{72}$, which encompasses both measurable and non-measurable, passive and active joints. As previously mentioned in Sec. II, in the real system, positions of $\boldsymbol{q}^{ll}$ and $\boldsymbol{q}^p$ can be measured directly via absolute encoders, instead positions of $\boldsymbol{q}^m$ is obtained by integration starting from a calibrated initial position. Given this distinctive characteristic of our system, in IV-A, we analyze the effects of composing the observation vector using a diverse set of joints. In particular, we will consider three different structures for the observation vector $\boldsymbol{o}_t$:

- *Full*: $\left[\boldsymbol{o}_t^c, \boldsymbol{q}_t^{p\,T}, \dot{\boldsymbol{q}}_t^{p\,T}, \boldsymbol{q}_t^{ll\,T}, \dot{\boldsymbol{q}}_t^{ll\,T}, \boldsymbol{q}_t^{m\,T}, \dot{\boldsymbol{q}}_t^{m\,T}\right]^T \in \mathbb{R}^{68}$
- *Measured*: $\left[\boldsymbol{o}_t^c, \boldsymbol{q}_t^{p\,T}, \dot{\boldsymbol{q}}_t^{p\,T}, \boldsymbol{q}_t^{ll\,T}, \dot{\boldsymbol{q}}_t^{ll\,T}\right]^T \in \mathbb{R}^{48}$
- *Actuated*: $\left[\boldsymbol{o}_t^c, \boldsymbol{q}_t^{ll\,T}, \dot{\boldsymbol{q}}_t^{ll\,T}, \boldsymbol{q}_t^{m\,T}, \dot{\boldsymbol{q}}_t^{m\,T}\right]^T \in \mathbb{R}^{48}$

where $\boldsymbol{o}_t^c = \left[\boldsymbol{v}_t^T, \boldsymbol{\omega}_t^T, \tilde{\boldsymbol{g}}_t^T, \bar{\boldsymbol{v}}_t^{xy\,T}, \bar{\phi}_t, \boldsymbol{a}_{t-1}^T\right]^T$ contains the state of the floating base, the velocity commands, and the action applied at the previous step.

B. Action Space

The action space of the bipedal robot is defined as the set of target positions for all actuated joints on each leg. As previously shown, each leg is equipped with six independently controlled linear actuators, resulting in a total of 12 continuous action dimensions. These actuated joints correspond to functionally distinct DOFs, each enabling a specific motion critical for dynamic locomotion. Hence, the action vector $\boldsymbol{a} \in \mathbb{R}^{12}$ describes the position of the linear actuators. The ranges of allowed motor positions are restricted to a tight interval due to the linear nature of Kangaroo's actuators. If not carefully taken into consideration, this characteristic can hinder the learning process of the PPO agent, which could be unable to correctly explore the allowed action space, for instance, because of a too high initial policy's standard deviation parameter. In this work, we decided to post-process the raw actions $\boldsymbol{a}_t$ sampled from the policy distribution to consistently obtain valid references for the actuated joints $\hat{\boldsymbol{q}}_t^m$. The procedure involves two steps: (i) we scale the raw actions into the range $(-1, 1)$ using a $tanh$ function, (ii) re-scale each action's dimension from $(-1, 1)$ to the actual joint limits. Detailed expressions are reported below.

$$\tilde{\boldsymbol{a}}_t = tanh\left(0.1 \cdot \boldsymbol{a}_t + \boldsymbol{\delta}\right)$$
$$\hat{\boldsymbol{q}}_t^m = \tilde{\boldsymbol{a}}_t \cdot (\overline{\boldsymbol{q}}^m - \underline{\boldsymbol{q}}^m)/2 + (\overline{\boldsymbol{q}}^m + \underline{\boldsymbol{q}}^m)/2$$

We refer to the vectors containing the lower and upper bounds of the actuated joints with $\underline{\boldsymbol{q}}^m, \overline{\boldsymbol{q}}^m \in \mathbb{R}^{12}$, while $\boldsymbol{\delta} \in \mathbb{R}^{12}$ is a constant offset vector, which is assigned with the only non-zero values (-0.54) at the two dimensions associated to each leg's LLM to map $a = 0$ to their corresponding starting position $\hat{q}^m = 0.0375\ m$.

C. Rewards and Penalties

The total reward considered during the training comprises different terms, with both rewards (positive) and penalties (negative), that are weighted and summed together. Let us first define the following quantities:

- *Air times* ${}^{L/R}T_t^{air}$, the time that the left (right) foot is spending in the air since the last lift off (0 if in contact);
- *Last air times* ${}^{L/R}\overline{T}_t^{air}$, the time the left (right) foot spent in the air before the last contact;
- *Contact times* ${}^{L/R}T_t^{con}$, the time the left (right) foot is spending in contact since the last contact (0 if not in contact);
- *Last contact times* ${}^{L/R}\overline{T}_t^{con}$, the time the left (right) foot spent in contact before the last detach.

Additionally, we will refer to the robot angular velocity around the world frame z axis (yaw) as ϕ_t, and to the velocity of the feet links expressed in world frame as ${}^{L/R}\boldsymbol{v}_t$. Finally, we are using ^{x}, ^{y}, and ^{z} superscripts to denote the components of vectors $\boldsymbol{v}_t$ and $\boldsymbol{\omega}_t$ relative to the axis x, y, and z, respectively.

All the reward and penalty terms used during the policy training are reported in Table III. Moreover, a fixed termination penalty of -200 is assigned in case the robot falls. The *feet air time* term tends to reward high swing times, keeping only one foot in contact with the ground. The *diff. steps* component penalizes unsymmetrical gaits, i.e., gaits where air and contact times are sensibly different between the left and right foot. The *action rate* component tries to privilege the generation of smoother motor references, while the remainder terms penalize oscillation of the robot's body. The overall reward structure and different components' weights have been manually tuned

[4]*RSL RL*: https://github.com/leggedrobotics/rsl_rl

[5]NVIDIA *Isaac Lab*: https://isaac-sim.github.io/IsaacLab

TABLE III: Rewards and penalties definition.

Name	*Expression*	*Weight*
lin vel xy track	$\exp\left(-\|\|\bar{\boldsymbol{v}}_t^{xy}-\boldsymbol{v}_t^{xy}\|\|^2/0.25\right)$	2.0
ang vel z track	$\exp\left(-(\bar{\phi}_t-\phi_t)^2/0.25\right)$	1.0
feet air time	$\begin{cases}\min({}^{L}T_t^{air},\ {}^{R}T_t^{con},0.4) & \text{if } {}^{R}T_t^{air}=0\\ \min({}^{R}T_t^{air},\ {}^{L}T_t^{con},0.4) & \text{if } {}^{L}T_t^{air}=0\\ 0.0 & \text{otherwise}\end{cases}$	2.0
lin vel z	$\|\|\boldsymbol{v}_t^z\|\|^2$	-0.2
ang vel xy	$\|\|\boldsymbol{\omega}_t^{xy}\|\|^2$	-0.05
flat orient.	$\|\|\tilde{\boldsymbol{g}}_t^{xy}\|\|^2$	-2.0
action rate	$\|\|\boldsymbol{a}_t-\boldsymbol{a}_{t-1}\|\|^2$	-0.01
feet slide	$\sum_{j=L,R}\|\|{}^{j}\boldsymbol{v}_t\|\|\cdot[{}^{f}T_t^{con}>0]$	-0.25
diff. steps	$\|{}^{L}\overline{T}_t^{air}-{}^{R}\overline{T}_t^{air}\|+\|{}^{L}\overline{T}_t^{con}-{}^{R}\overline{T}_t^{con}\|$	-0.5

to produce satisfactory walking patterns; nevertheless, a substantial reward shaping analysis is outside the scope of this paper.

D. Domain Randomization

Observation vectors $\boldsymbol{o}_t$ are corrupted by an additive noise sampled from different uniform distributions for each component taken on the robot, to mimic the presence of measurement noise. Moreover, static and dynamic friction coefficients are randomized together with the robot's initial pose. Table IV provides details of all the measurement noises and domain randomization terms.

TABLE IV: Noise and domain randomization parameters.

Parameter	*Unit*	*Range*	*Operator*
linear velocity $\boldsymbol{v}$	m/s	$[-0.1, 0.1]$	add
angular velocity $\boldsymbol{\omega}$	rad/s	$[-0.2, 0.2]$	add
projected gravity $\tilde{\boldsymbol{g}}$	–	$[-0.05, 0.05]$	add
meas. pass. joint pos. $\boldsymbol{q}^p$	rad	$[-0.01, 0.01]$	add
meas. pass. joint vel. $\dot{\boldsymbol{q}}^p$	rad/s	$[-0.1, 0.1]$	add
LLM pos. $\boldsymbol{q}^{ll}$	m	$[-0.0025, 0.0025]$	add
LLM vel. $\dot{\boldsymbol{q}}^{ll}$	m/s	$[-0.025, 0.025]$	add
M1, …, M5; pos. $\boldsymbol{q}^m$	m	$[-0.0025, 0.0025]$	add
M1, …, M5; vel. $\dot{\boldsymbol{q}}^m$	m/s	$[-0.025, 0.025]$	add
initial robot xy position	m	$[-0.5, 0.5]$	sample
initial robot yaw angle	rad	$[-3.14, 3.14]$	sample
static friction	–	$[0.7, 1.0]$	sample
dynamic friction	–	$[0.4, 0.7]$	sample

IV. Policy Training

Training with PPO was carried out using 2048 parallel simulated environments and the default hyperparameters provided by *Isaac Lab*™. Both the actor and critic networks featured hidden layers of dimensions [256, 256, 128], with Exponential Linear Unit (ELU) activation functions. We maintained the self-collisions enabled in the simulator throughout all the training and evaluation experiments. We found that 2000 learning iterations were enough to converge to a walking policy. Figure 4 shows an example walk obtained at the end of the training process. Training has been carried out on a laptop equipped with a *NVIDIA RTX 2000 Ada GPU* and an *Intel® Core™ Ultra 7 165H × 22*, taking approximately 2 hours to complete a policy training, despite the complexity of the robot and the enabled collision model. The supplementary material includes videos demonstrating examples of the walking policy at various stages of training.

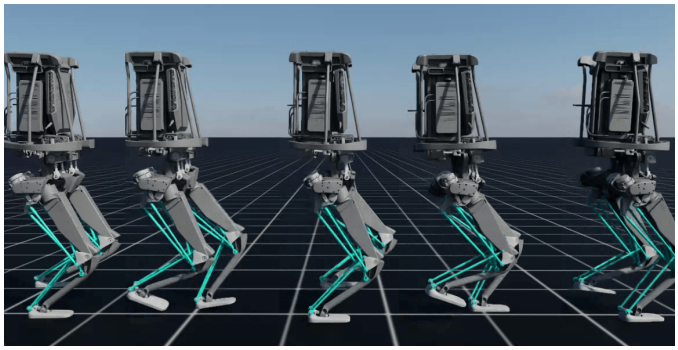

Fig. 4: Time-lapse of one of the obtained walking policies running on *Isaac Sim*™.

Given the complex structure of the robot's kinematics and the number of DOFs, we decide to focus our analysis on understanding which kind of joint measurements are decisive for a successful locomotion policy training. In the following, we present an ablation study in which we compare training performance obtained with the three different observation structures proposed in III-A.

A. Ablation Study: Changing joint state in the observations

For each observation vector structure, *Full*, *Measured*, and *Actuated*, we conducted 10 independent training sessions, each with a different random seed. Figure 5 presents the comparison across these settings in terms of mean reward, along with the most significant differences observed in specific reward and penalty terms. Results are shown using median values and 5th–95th percentile ranges. Policies trained with the *Actuated* observation vector consistently underperform in tracking accuracy compared to those using either the *Full* or *Measured* vectors. Moreover, they lead to more frequent robot falls: while early termination penalties largely disappear after a few iterations in the *Full* and *Measured* cases, they remain prominent with the *Actuated* configuration. Additional differences emerge in the *action rate* and *flat orientation* penalties. Policies based on the *Actuated* vector exhibit lower values and higher variability, particularly during early training, suggesting increased instability and difficulty in maintaining balance. Interestingly, the performance of policies trained with the *Full* and *Measured* observation vectors is largely comparable. In contrast, relying solely on motor states appears insufficient for successful training in this setup. This might possibly be attributed to the limited range of positions attainable by M1, …, M5 (ranging from $-0.02\ m$ to $0.02\ m$), especially when combined with additive uniform noise, which may diminish the relative informativeness of these inputs. Finally, an analysis of the remaining reward and penalty components highlights notable variability in both *feet air time* and *step difference* terms across all observation sets. This variability leads to a diversity of gait patterns, ranging from longer to shorter steps, and in some cases, asymmetric walking between the

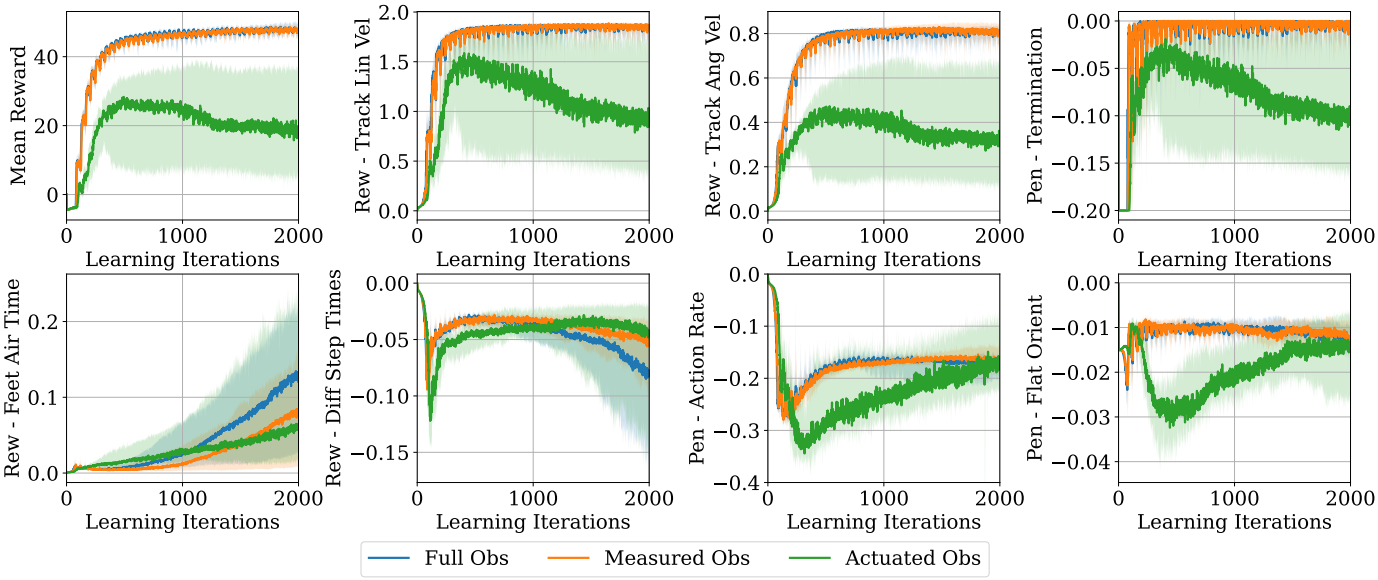

Fig. 5: Comparison of the mean total reward and other different specific terms recorded during training using either the *Full*, *Measured*, or *Actuated* observation vector (see III-A). Results are reported in terms of median values (represented with solid lines) and confidence intervals defined by the 5th and 95th percentiles.

left and right legs. A more refined reward shaping strategy, along with careful tuning of individual reward components, may help future training runs converge toward more consistent and stable locomotion behaviors.

V. Sim-to-Sim Validation

To validate the effectiveness of the trained policies, we deployed them into a different simulation environment, namely *MuJoCo*. The *Sim-to-Sim* process poses significant challenges, due to the differences present in the two simulators, in particular in handling constraints and contacts.

When deploying the trained policies in *MuJoCo*, we replicated the same observation collection and action scaling, as described in III-A and III-B, keeping the same policy update frequency of 50 Hz. For brevity, we deployed only policies that use the *Full* observation vector (Figure 6 shows an example walk recorded in *MuJoCo*). To better cope with the reduced simulation time steps length and avoid major discontinuities in control inputs, we employed a first-order hold (FOH) function that interpolates between the previous and current policy actions across the simulation steps. This ensured smoothness and better joint reference tracking. As an example, Figure 7 shows the reference tracking for the LLM joint obtained in both *Isaac Sim*™ and *MuJoCo* using one of the trained policies passing the same velocity command. The joint trajectory obtained from the deployed policy in *MuJoCo* is similar to the original behavior. Note how the FOH introduces some delay w.r.t. the reference generated by the policy, but this guarantees to avoid "jumps" in the reference signal, improving smoothness and tracking performance.

To evaluate *Sim-to-Sim* performance, we tested 10 different policies, each trained with a different random seed, in both *Isaac Sim*™ and *MuJoCo*. For each policy, we recorded the

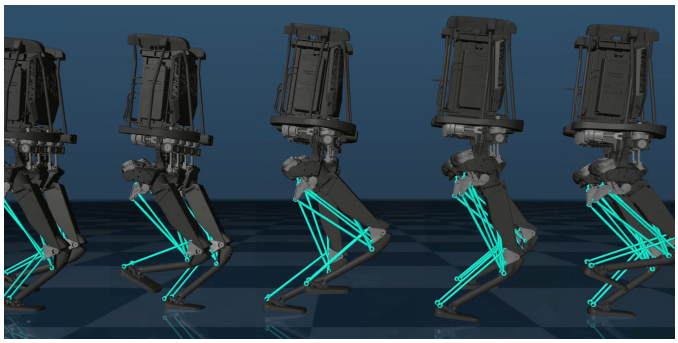

Fig. 6: Time-lapse of one *Sim-to-Sim* policy deployment in *MuJoCo*.

success rate, i.e., whether the robot fell, as well as the linear and angular tracking errors over 10 runs, each following a random command sampled in the same way as during training. Results are reported in Figure 8 using box plots (refer to the *Default Reward* column). Mann-Whitney U-test has been performed to assess the statistical significance of the differences observed in the collected results. While falling is always avoided when running the policies in the original *Isaac Sim*™ simulator, falls are more frequent when running in *MuJoCo*. The majority of the policies show a success rate between 60% and 80% (average 70%), while one case reaches the 90%. Also, tracking error shows a significant increase in the *Sim-to-Sim* deployment. If in *Isaac Sim*™ the mean linear and angular tracking errors are 0.095 m/s and 0.115 rad/s; the policies deployed in *MuJoCo* achieve errors of 0.200 m/s and 1.26 rad/s. One of the factors that could explain such a worsening in performance is the

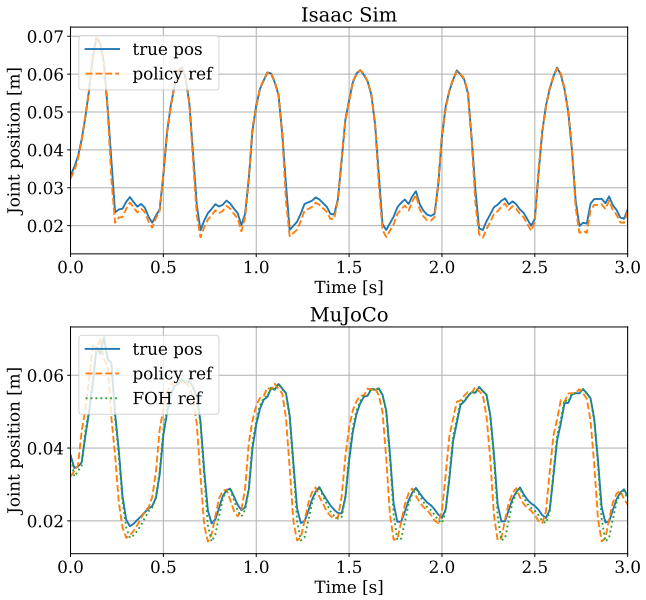

Fig. 7: Tracking of the LLM joint in two example runs of the same policy in *Isaac Sim*™ and *MuJoCo*.

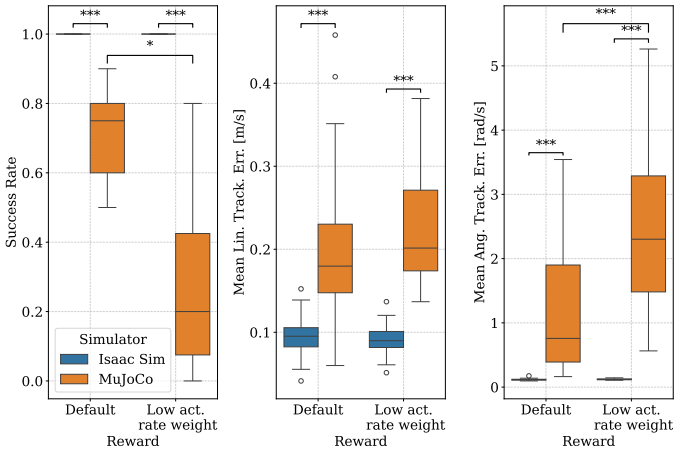

Fig. 8: Box plots of the success rates, linear and angular tracking errors obtained in *Isaac Sim*™ and in *MuJoCo*. The tested policies are 10 trained with the default reward (Table III) and 10 trained with a lower *action rate* penalty weight of -0.005. Statistical significance is marked with * (p-value $<$ 0.05), ** (p-value $<$ 0.01), or *** (p-value $<$ 0.001).

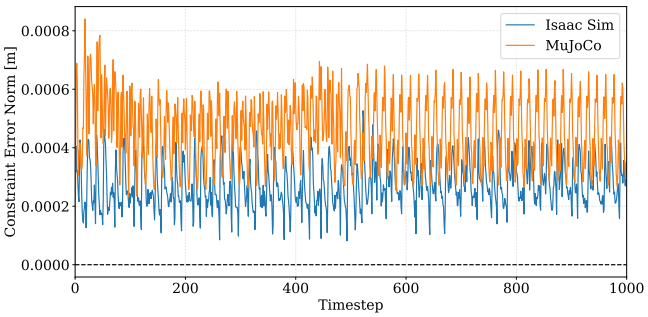

Fig. 9: Average norm error associated with the closed-chain constraints observed in two example runs in *Isaac Sim*™ and *MuJoCo*.

different way the two simulators model contacts, which plays a crucial role in ensuring stable interactions with the ground and deeply affects the resulting locomotion behavior. Another important aspect is the effectiveness in enforcing closed-chain constraints, which are essential for accurately reproducing the system's kinematics. Figure 9 shows, for example, the average norm of the constraint errors in the two simulators while running the same locomotion policy. We can observe a consistently higher error in *MuJoCo* (with an average increase of 67% compared to *Isaac Sim*™).

Despite the inherent challenges of transferring policies across simulators, especially for complex closed-chain systems, it is encouraging to observe that most of the policies were still able to successfully stabilize the robot and achieve locomotion in *MuJoCo*, demonstrating a degree of robustness to the simulation gap. Example videos of *Sim-to-Sim* policy deployments are available in the supplementary material.

A. Ablation Study: Training with a lower action rate weight

During the initial attempts at *Sim-to-Sim* deployment, we observed a drastic improvement when we started to use a *action rate* penalty weight of -0.01. In fact, when using a lower weight of -0.005 (half of the original), we were observing great difficulties in keeping the balance. For this reason, we deployed another set of 8 policies trained using an *action rate* penalty weight of -0.005 (with all other terms identical to those in Table III), and compared the results obtained over 10 runs for each policy. The use of a lower penalty weight resulted in an average increase of the policies' action rate of 29.52%. In Figure 8, we also report the results obtained with this new reward (refer to the column *Low act. rate weight Reward*).

In comparison with the results obtained by training with the default reward, it is possible to observe that lowering the *action rate* penalty weight did not result in major differences in *Isaac Sim*™ performance. Instead, it emerges a statistically significant drop in the success rates is observed in *MuJoCo*, where we reach an average of only 30%. Also, the angular tracking error exhibits a significant increase. These findings confirm the importance of carefully tuning *action rate* penalties to obtain transferable policies across simulators.

VI. Conclusion and Future Work

In this work, we presented a case study on a locomotion training pipeline for Kangaroo, a bipedal humanoid robot featuring hybrid serial-parallel kinematic chains, based on *Isaac Sim*™ and successively transferred to *MuJoCo* to preserve full system dynamics. Unlike simplified approaches that train on reduced models requiring additional mappings, we chose to model the full complexity of Kangaroo and investigate whether

it is possible to train and evaluate a walking policy directly at the motor level using existing tools for RL and simulation.

Our results highlight the importance of observation design according to the type of motors, rotational or linear, as well as the presence of redundancy in the measurements, for example, in passive DOFs.We conducted an extensive *Sim-to-Sim* validation process, deploying 10 trained policies in another simulator, *MuJoCo*, achieving in most cases the stability of the walking policy in closed-loop despite the presence of a significant *Sim-to-Sim* gap, especially for what concerns the simulation of contacts and closed kinematic chains. Furthermore, we highlighted how small differences in the *action rate* penalties used during training could lead to major variations in the behavior of the resulting policies when deployed in a different simulation environment.

As next steps, we aim to refine the locomotion, possibly taking into account imitation learning to achieve more natural gaits [20]–[22] and deploy the trained policies on the physical Kangaroo robot. This will allow us to evaluate their real-world performance and investigate *Sim-to-Real* challenges specific to systems with hybrid closed-chain mechanisms.

REFERENCES

[1] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi9579, 2024.

[2] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.

[3] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Conference on Robot Learning*. PMLR, 2020, pp. 317–329.

[4] E. M. Hoffman, A. Curti, N. Miguel, S. K. Kothakota, A. Molina, A. Roig, and L. Marchionni, "Modeling and numerical analysis of kangaroo lower body based on constrained dynamics of hybrid serial–parallel floating-base systems," *Robotics and Autonomous Systems*, vol. 182, p. 104827, 2024.

[5] K. G. Gim, J. Kim, and K. Yamane, "Design and fabrication of a bipedal robot using serial-parallel hybrid leg mechanism," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 5095–5100.

[6] A. S. Sathya and J. Carpentier, "Constrained articulated body algorithms for closed-loop mechanisms," *HAL Open Science*, 2025.

[7] M. Boukheddimi, R. Kumar, S. Kumar, J. Carpentier, and F. Kirchner, "Investigations into exploiting the full capabilities of a series-parallel hybrid humanoid using whole body trajectory optimization," in *IEEE/RSJ*

[13] Q. Zhang, G. Han, J. Sun, W. Zhao, J. Cao, J. Wang, H. Cheng, L. Zhang, Y. Guo, and R. Xu, "Lips: Large-scale humanoid robot reinforcement learning with parallel-series structures," *arXiv preprint*, 2025.

International Conference on Intelligent Robots and Systems (IROS), 2023, pp. 10 433–10 439.

[8] V. Batto, T. Flayols, N. Mansard, and M. Vulliez, "Comparative metrics of advanced serial/parallel biped design and characterization of the main contemporary architectures," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2023, pp. 1–7.

[9] L. de Matteis, V. Batto, J. Carpentier, and N. Mansard, "Optimal control of walkers with parallel actuation," *arXiv preprint*, 2025.

[10] V. Lutz, L. de Matteïs, V. Batto, and N. Mansard, "Control of humanoid robots with parallel mechanisms using kinematic actuation models," *arXiv*, 2025.

[11] F. Ruscelli, A. Laurenzi, E. Mingo Hoffman, and N. G. Tsagarakis, "A fail-safe semi-centralized impedance controller: Validation on a parallel kinematics ankle," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[12] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[14] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi9579, 2024.

[15] X. Gu, Y.-J. Wang, X. Zhu, C. Shi, Y. Guo, Y. Liu, and J. Chen, "Advancing humanoid locomotion: Mastering challenging terrains with denoising world model learning," in *Robotics: Science and Systems (RSS)*, 07 2024.

[16] L. Tang, D. Liang, G. Gao, X. Wang, and A. Xie, "Modeling and reinforcement learning-based locomotion control for a humanoid robot with kinematic loop closures," *Multibody System Dynamics*, pp. 1–27, 2024.

[17] M. Chignoli, J.-J. Slotine, P. M. Wensing, and S. Kim, "Urdf+: An enhanced urdf for robots with kinematic loops," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2024, pp. 197–204.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[19] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[20] T. He, W. Xiao, T. Lin, Z. Luo, Z. Xu, Z. Jiang, C. Liu, G. Shi, X. Wang, L. Fan, and Y. Zhu, "Hover: Versatile neural whole-body controller for humanoid robots," *arXiv preprint arXiv:2410.21229*, 2024.

[21] A. Tang, T. Hiraoka, N. Hiraoka, F. Shi, K. Kawaharazuka, K. Kojima, K. Okada, and M. Inaba, "Humanmimic: Learning natural locomotion and transitions for humanoid robot via wasserstein adversarial imitation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 13 107–13 114.

[22] Q. Zhang, P. Cui, D. Yan, J. Sun, Y. Duan, G. Han, W. Zhao, W. Zhang, Y. Guo, A. Zhang *et al.*, "Whole-body humanoid robot locomotion with human reference," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 11 225–11 231.