Reinforcement Learning-Based Model Matching to Reduce the Sim-Real Gap in COBRA

Adarsh Salagame[1†], Harin Kumar Nallaguntla[1†], Bardia Ardakanian[3],
Eric Sihite[2], Gunar Schirner[1], Alireza Ramezani[1*]

Abstract— This paper employs a reinforcement learning-based model identification method aimed at enhancing the accuracy of the dynamics for our snake robot, called COBRA. Leveraging gradient information and iterative optimization, the proposed approach refines the parameters of COBRA's dynamical model such as coefficient of friction and actuator parameters using experimental and simulated data. Experimental validation on the hardware platform demonstrates the efficacy of the proposed approach, highlighting its potential to address sim-to-real gap in robot implementation.

I. INTRODUCTION

Snake robots exhibit a multitude of actuated joints, engaging in locomotion patterns involving intricate interactions with the environment [1]. These systems present formidable challenges in modeling and control due to the complex interplay of unilateral contact forces, leading to intricate complementarity conditions [2]. Traditional approaches have previously tackled these force inclusion issues with promising outcomes [3], [4].

Recent advancements in learning methodologies offer expedited solutions, circumventing the need for manually tuned parameter sets and expanding beyond constrained or supervised settings [5]–[9]. The models in these studies are further simplified through various assumptions to minimize the number of adjustable parameters. While these methods eloquently showcase how neural modulations can give rise to diverse locomotion strategies [10]–[14], there remains limited exploration on extending these models to actively utilize high-dimensional complex observations from onboard sensors to modulate actuators.

This paper addresses the "sim-to-real problem" encountered in the COBRA snake robot [15]–[17], as depicted in Fig. 1, where substantial disparities in joint angle trajectories and final head positions emerge between the physical robot and its mathematical model. COBRA is capable of executing programmed 3D interactions with the environment surface.

Driven by the necessity to refine the accuracy of COBRA's locomotion model for effective policy transfer, we employ the Proximal Policy Optimization (PPO) algorithm [18] to

[1]This author is with the Department of Electrical and Computer Engineering, Northeastern University, Boston MA `salagame.a, nallaguntla.h, G.Schirner, a.ramezani@northeastern.edu*`

[2] This author is with California Institute of Technology, Pasadena CA `esihite@caltech.edu`

[3] This author is with the University of Saskatchewan `bardia.ardakanian@usask.ca`

†These authors have equal contribution to this work.

∗Indicates the corresponding author.

Fig. 1. Illustrates COBRA platform while performing sidewinding over flat ground.
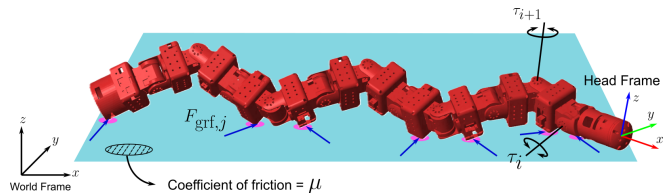

Fig. 2. Shows free-body-diagram of COBRA and model parameters unknown to us and found as part of this work.

determine model parameters, encompassing inertial terms, friction coefficients, and unknown actuator parameters, to synchronize the model's behavior with the hardware platform during predefined locomotion patterns.

Our proposed reinforcement learning-based approach aims to bridge the simulation-reality gap, enhancing predictive accuracy and facilitating the transfer of acquired control policies. This research significantly advances snake robot locomotion by addressing the challenges of sim-to-real disparities, thereby contributing to the formulation of robust and adaptable control strategies in complex environments.

A. Quick Overview of COBRA

COBRA, an acronym for Crater Observing Bio-inspired Rolling Articulator, is a snake robot designed after the intricate locomotion of serpentine creatures. This robot is specifically engineered to navigate and endure rugged terrains and environments, such as craters, where traditional wheeled or legged robots may struggle [19]–[21].

Comprising 11 articulated joints, COBRA exhibits flexibility and fluidity in its movements, enabling it to perform a variety of maneuvers, including sidewinding or forming a loop-shape configuration for tumbling. Each module in

COBRA emboides a LIPO battery, an actuator and a microprocessor. The modules are connected to each other in a daisy chain. The COBRA robot is equipped with a C++ API that controls a series of Dynamixel servos.

The paper's structure is as follows. We go over the dynamics of Cobra in Section II, then present the reinforcement learning framework used to identify unknown parameters in the model in Section III. Finally, we present results showing the change in performance of the model after training and quantify the improvements made.

II. COBRA FULL-DYNAMICS

The dynamical equations of motion of COBRA is given by

$$
\begin{bmatrix} D_H & D_{Ha} \\ D_{aH} & D_a \end{bmatrix} \begin{bmatrix} \ddot{q}_H \\ \ddot{q}_a \end{bmatrix} + \begin{bmatrix} H_H \\ H_a \end{bmatrix} = \begin{bmatrix} 0 \\ B_a \end{bmatrix} u + \\ + \begin{bmatrix} J_H^\top \\ J_a^\top \end{bmatrix} F_{GRF} \tag{1}
$$

where D_i, H_i, B_i, and J_i are partitioned model [22]–[24] parameters corresponding to the head 'H' and actuated 'a' joints. F_{GRF} denotes the ground reaction forces. u embodies the joint actuation torques. The model considered for actuators is adopted from M. Spong's book [25]. The actuator parameters include: transmission inertia, internal damping, and dc motor constant. The reference trajectories used in the actuator models are generated by

$$
y(t) = A\sin(\omega t + \phi) \tag{2}
$$

where A is the amplitude, ω is the frequency, and ϕ is the phase difference of the signal respectively.

The ground model [26]–[29] used in our simulations is given by

$$
F_{GRF} = \begin{cases} 0 & \text{if } p_{C,z} > 0 \\ [F_{GRF,x}, \ F_{GRF,y}, \ F_{GRF,z}]^\top & \text{else} \end{cases} \tag{3}
$$

where $p_{C,i}$, $i = x, y, z$ are the $x - y - z$ positions of the contact point; $F_{GRF,i}$, $i = x, y, z$ are the $x - y - z$ components of the ground reaction force assuming a contact takes place between the robot and the ground substrate. In Eq. 3, the force terms are given by

$$
F_{GRF,z} = -k_1 p_{C,z} - k_2 \dot{p}_{C,z}, \tag{4}
$$

$$
F_{GRF,i} = -s_i F_{GRF,z}\,\text{sgn}(\dot{p}_{C,i}) - \mu_v \dot{p}_{C,i} \quad \text{if } i = x, y \tag{5}
$$

In Eqs. 4 and 5, k_1 and k_2 are the spring and damping coefficients of the compliant surface model. In Eq. 5, the term s_i is given by

$$
s_i = \left(\mu_c - (\mu_c - \mu_s)\exp\left(-|\dot{p}_{C,i}|^2/v_s^2\right) \right) \tag{6}
$$

where μ_c, μ_s, and μ_v are the Coulomb, static, and viscous friction coefficients; and, $v_s > 0$ is the Stribeck velocity. Specifically speaking, from Eqs. 1 and 3, the following parameters are unknown to us: actuator models parameters and Stribeck terms.

In Section. III, we will use RL to obtain these unknown parameters. The unknown model parameters are encoded within the simulator as part of sim-to-real gap research.
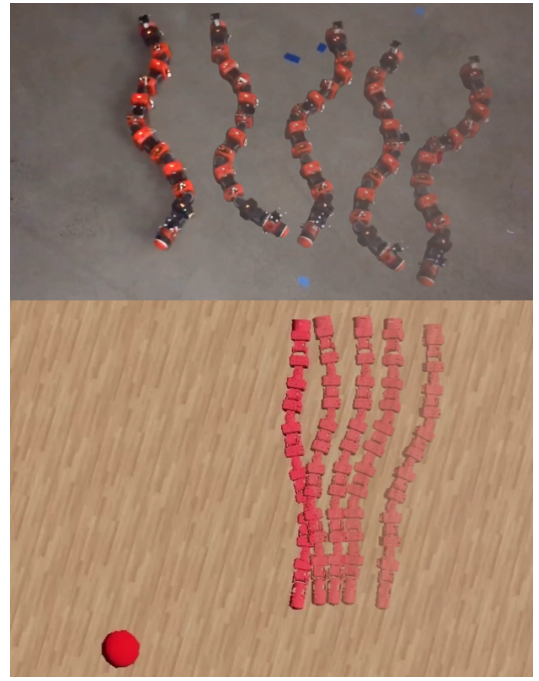

Fig. 3. Highlights a significant disparity between the behaviors of COBRA in experiments and simulations. The paper's primary contribution lies in aligning the dynamic behavior of COBRA across both simulation and real-world experiments.
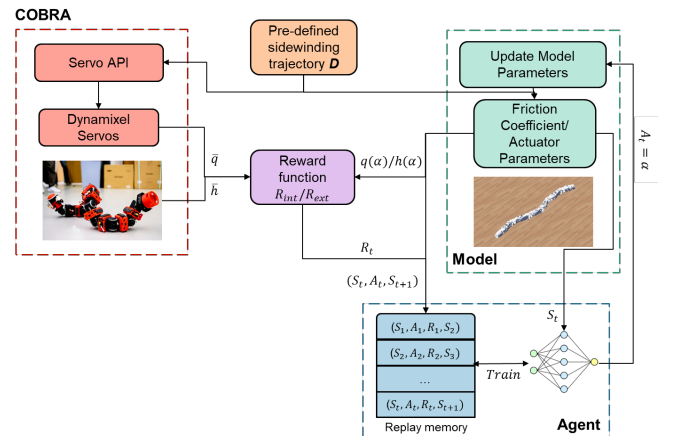

Fig. 4. Reinforcement Learning-guided model identification setup used in this work.

A. Selection of Simulator

To provide a good starting point for the model matching, the right simulator must be selected. Simulators such as Matlab Simscape, Drake and MuJoCo are commonly used for simulating dynamics, and others such as Gazebo are more suited for high level perception and navigation tasks. In this work, Webots is chosen for its robust physics simulation capabilities and adaptability to multi-degree-of-freedom robot locomotion complexities, as well as the ability to easily add several sensing modalities for training a closed loop locomotion policy in the future. Developed by Cyberbotics, Webots offers a realistic and dynamic simulation environ-
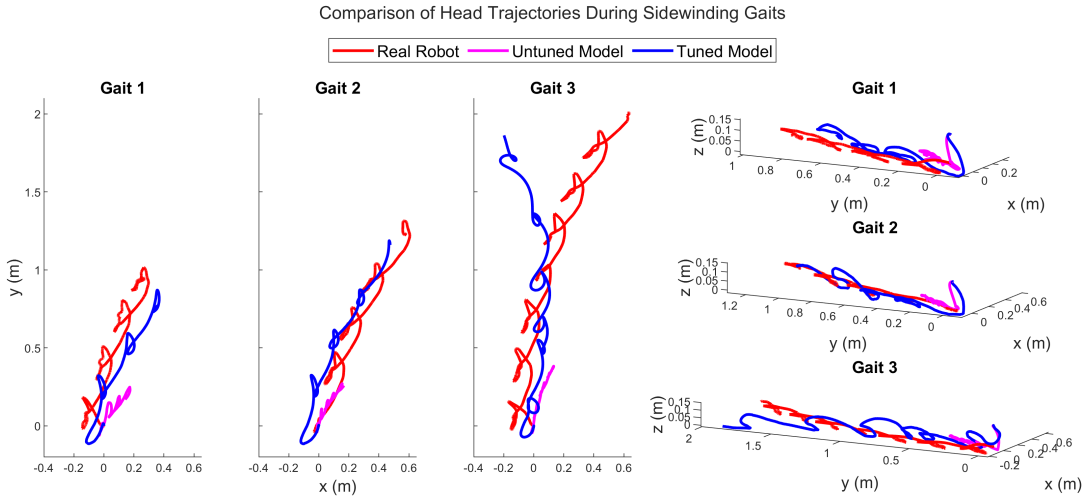
Fig. 5. Illustrates a comparison between the head positions in the actual hardware platform (blue), tuned model (orange) and untuned model (red) for a sidewinding trajectory 0.35, 0.5, and 0.65 Hz.

ment, accurately modeling interactions between the COBRA snake robot and its surroundings. Its physics engine enables precise evaluation of robot behavior while executing predefined trajectories. Webots also has a user-friendly interface and extensive documentation, and is a popular choice in the RL community for its ability to change model parameters programmatically with ease, enabling encoding unknown model parameters for Learning applications.

III. REINFORCEMENT LEARNING-GUIDED IDENTIFICATION OF ACTUATOR AND STRIBECK PARAMETERS

A. Urgency

Figure. 3 illustrates the behavior gap between the hardware platform and mathematical model. The sim-to-real problem in COBRA locomotion manifests as variations in final head positions between the simulated robot and its physical counterpart when exactly the same joint motions are commanded in both models. These disparities hinder the seamless transfer of control policies developed in simulation to the real robot, limiting the effectiveness of learned behaviors in practical applications. Hence, addressing the sim-to-real issue is crucial for ensuring the reliability and robustness of COBRA when deployed in diverse and dynamic environments.

B. Approach

Our methodology provides a structured approach to bridging the sim-to-real gap for the COBRA robotic system. We combine physical hardware, a virtual simulation environment, and a reinforcement learning framework to iteratively refine the fidelity of the simulation, thereby ensuring that it mirrors the real-world performance of the robot as closely as possible.

The simulation of the COBRA robot is conducted within Webots, which incorporates an ODE physics engine to reproduce the dynamics of robotic systems. The physics engine takes into consideration the mechanical properties, such as

mass and friction, as well as the actuator models to simulate the robot's interaction with its environment.

C. Performance Comparison Based on Underactuated and Actuated Dynamics from Eq. 1

A sidewinding trajectory command q_r is employed to guide both the real COBRA robot and its virtual model through identical movements. The performance analysis occurs in two separate domains as suggested by the partitioned model in Eq. 1:

- *Underactuated dynamics:* The imposition of q_r leads to the head translations $\ddot{q}_H$, which are underactuated. Comparing the head's position and orientation provides a clear performance metric for discerning contact force discrepancies (friction coefficients) between the physical robot and the simulation while completely excluding actuator dynamics.
- *Actuated dynamics:* In addition to matching passive dynamics models, we also address actuated dynamics, specifically the joint angle trajectories $\ddot{q}_a$ in Eq. 1. Aligning actuator models is crucial for evaluating the precision of the robot's movements. This comparison can be accomplished by calculating the norm distance of output joint motions between the physical robot and the simulated model.

D. Reinforcement Learning Framework

The PPO algorithm is used to iteratively refine the model parameters. This is done with the aim of minimizing the observed discrepancies in the underactuated and actuated dynamics between the model and the actual robot. Below, we explain the state, action, and reward function defined for the PPO algorithm.

1) State, Action, and Reward Definition: The state S_t encapsulates the actuator parameters for internal tuning and Steibek terms for external tuning. The actions A_t are the modifications applied to these simulation parameters.

The reward function R is given by

$$R_{\text{external}} = -\sqrt{((x_{\text{des}} - x_{\text{actual}})^2 + (y_{\text{des}} - y_{\text{actual}})^2)} \quad (7)$$

where x_{des} and y_{des} denote desired x- and y-positions of the head module, respectively. x_{actual} and y_{actual} are the OptiTrack data. And, R_{internal} is given by

$$\begin{aligned} R_{\text{internal}} = &- (\phi_{\text{des}} - \phi_{\text{actual}})^2 - (\omega_{\text{des}} - \omega_{\text{actual}})^2 \\ &- (A_{\text{des}} - A_{\text{actual}})^2 \end{aligned} \quad (8)$$

where ϕ, ω, and A are CPG variables. These reward functions are crafted to penalize the deviation in both passive dynamics and actuated dynamics. They are defined as the sum of the L^2-norm of the joint angle trajectory differences and the L^2-norm of the final head position differences.

2) Policy Updates: The policy search recruited here operates in a cycle of simulation runs and updates. During simulation runs, the algorithms collects data on state transitions and rewards, which are aggregated into sequences (S_t, A_t, R_t, S_{t+1}) and stored in a replay memory. The training phase involves updating the policy network with this data, where PPO adjusts the policy in a manner that maximizes the cumulative reward while maintaining a degree of similarity to the previous policy, using a mechanism known as clipping to avoid drastic policy changes [18].

Through these iterative training cycles, PPO tunes the model parameters to align the model performance with the actual COBRA robot, following a predefined sidewinding trajectory. The process continues until a convergence is reached. The following is a line-by-line breakdown of the search approach as reported in [18]:

a) Input: The algorithm starts with initial policy parameters θ_0 and initial value function parameters ϕ_0. These parameters are what the algorithm will learn to adjust as it interacts with the environment to improve its policy.

b) Collect Trajectories: A set of trajectories D_k is collected by running the current policy $\pi(\theta_k)$ in the environment. A trajectory is a sequence of states, actions, and rewards experienced by the agent.

c) Compute rewards-to-go: For each time step t, compute the total expected rewards from that time step until the end of the trajectory, denoted as R_t. This is used to estimate how good it is to be in a particular state.

d) Compute advantage estimates: The advantage A_t indicates how much better or worse an action is compared to the average action in a given state. This is calculated using value function V_ϕ and the rewards-to-go.

e) Update the Policy: First, we calculate the probability ratio, which is a fraction where the numerator is the probability of taking action a_t in state s_t under the new policy π_θ, and the denominator is the probability of taking action a_t in state s_t under the old policy $\pi_{\theta_{\text{old}}}$:

$$\text{ratio}(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

This ratio measures how the new policy differs from the old policy in terms of the likelihood of taking the same actions.

We then clip this ratio to be within a range of $[1-\epsilon, 1+\epsilon]$, where ϵ is a hyperparameter typically set to a small value like 0.1 or 0.2. This clipping limits the amount by which the new policy can differ from the old one, regardless of the advantage estimate:

$$\text{clipped}(\theta) = \text{clip}(\text{ratio}(\theta), 1 - \epsilon, 1 + \epsilon)$$

The objective function incorporates the clipped ratio and the advantage function A_t. It takes the minimum of the unclipped and clipped objectives, ensuring that the final objective doesn't take too large of a step (hence, "clipping"):

$$L(\theta) = \min\left(\text{ratio}(\theta)A_t, \ \text{clipped}(\theta)A_t\right)$$

The expectation of this objective function over all timesteps and all trajectories is what the algorithm seeks to maximize. This expectation is approximated by averaging over a finite batch of timesteps and trajectories:

$$\theta_{k+1} = \arg\max_\theta \left(\frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} L(\theta) \right)$$

Finally, the parameters of the policy θ are updated using stochastic gradient ascent. This means we compute gradients of the objective function with respect to the policy parameters and adjust the parameters in the direction that increases the objective:

$$\theta_{\text{new}} \leftarrow \theta_{\text{old}} + \alpha \nabla_\theta \left(\frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} L(\theta) \right)$$

where α is the learning rate.

The Adam optimizer is often used for this step because it adapts the learning rate for each parameter, helps in converging faster, and is more robust to the choice of hyperparameters.

By maximizing this objective, PPO-Clip seeks to improve the policy by making sure that the actions that would increase the expected return are taken more frequently, while ensuring that the policy does not change too drastically, which could lead to poor performance due to overfitting to the current batch of data.

IV. RESULTS

Experiments were performed on the real robot and motion capture data for the precise trajectory of the head link, the tail link, and the middle link was recorded. This is illustrated in Fig. 6. In addition, joint trajectory data for all eleven joints were recorded. Odd-numbered joints (1,3,5,...,11 from the head) execute a pitching motion and even-numbered joints (2,4,...,10 from the head) execute a yawing motion. A sidewinding gait created by supplying a sinusoidal input signal to each of the joints is used as a standard input with varying parameters to change the behavior of the robot. The input signal is given by a sine-wave according to Eq. 2 with some phase difference for each joint, as follows:

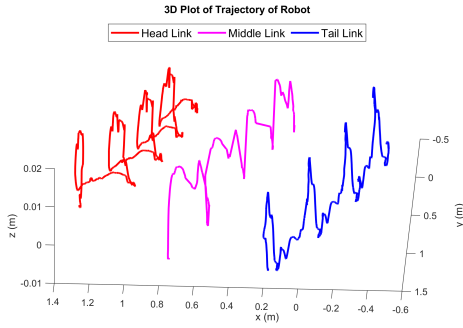$$A_{\text{pitching}} = 14^\circ, \ A_{\text{yawing}} = 60^\circ$$

Fig. 6. Illustrates body positions (head module, link 6, and tail module) during slithering towards the desired position on a flat ground collected from the actual hardware platform using an OptiTrack system.
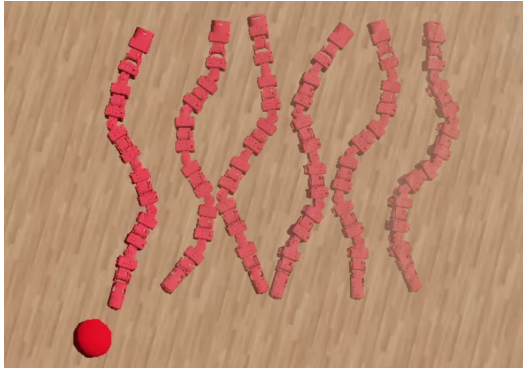

Fig. 7. Show COBRA in Webots simulator with tuned parameters performing the sidewinding motion. The red ball shows the location the actual robot achieved when similar joint trajectories were applied.

$$\phi = \frac{\pi}{2}[0, 0, 1, 1, 2, 2, 3, 3, 0, 0, 1]$$

The frequency for this sine-wave input was varied for diversity of data, creating three gaits with frequencies 0.35 Hz, 0.5 Hz, 0.65 Hz respectively. The recorded joint trajectories for two joints performing the 0.5 Hz frequency input is shown in Fig. 8. The actuators on the robot almost perfectly track the desired input signal. A total of 25 experiments were conducted, recording the robot moving from various initial conditions for the given set of three gaits. The robot performance was highly repeatable and one representative sample was kept for validation while the remaining was used for training of the simulation model.

Each of the three desired gaits was then executed in simulation, matching the initial conditions of the robot to reality for all training data. Fig. 3 presents snapshots that illustrate the difference in the behavior of the robot in simulation as compared to reality before tuning. The model was then tuned by training using the framework presented in Fig. 4. Fig. 9 shows the reward curve from tuning the internal dynamics of the robot model. Fig. 7 presents snapshots of the same experiment in simulation after tuning. The trajectory of the head module is compared in Fig. 5, showing significant improvements with the simulation matching the trajectory of the real robot. The Euclidean distance between the reference trajectory from the real robot and the trajectory from the simulation is plotted in Fig. 10, showing that for all three
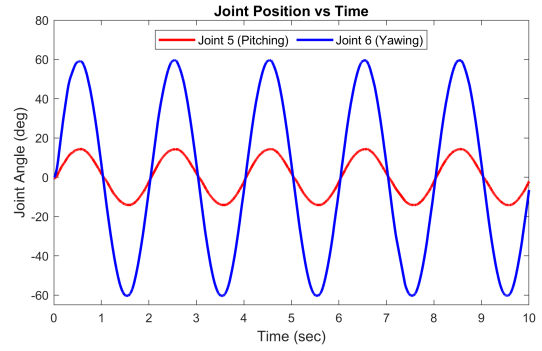

Fig. 8. Shows joint angles (e.g., joints 5 and 6) collected from the hardware platform.
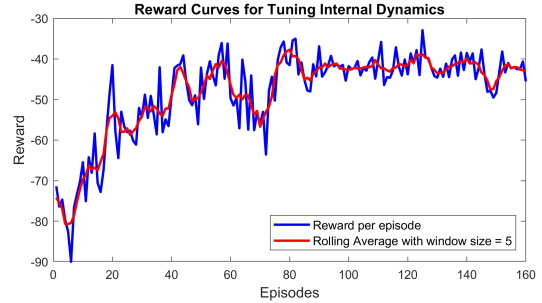

Fig. 9. Shows the training reward curve for tuning model parameters.

gaits, the error for the tuned model is lower. From this figure, it may be observed that as the movement of the robot becomes more aggressive (by increasing the frequency of the input sine wave), the tuned model reduces in performance. This is expected to some degree but can be mitigated by increasing the diversity of training data to improve the quality of the model. The performance is further inspected at the joint level by comparing the joint trajectories of the simulation model with the desired trajectories. Fig. 11 shows the signal for two representative joints, with the tuned model showing significantly closer tracking of the desired signal. Fig. 12 tracks the overall performance of all joints using sliding window correlation.

V. CONCLUSION

Despite limited training data, the framework has significantly improved the ability of the simulation to predict the behaviour of the robot. However, due to the limited training data, the model shows a dropoff in accuracy when attempting to generalize to more environments and types of inputs as observed earlier. This may be solved by increasing the diversity of training data. However, expanding the training set using data from real hardware is expensive both in terms of wear and tear to the robot and the time required for operation. This is circularly related to the primary motivation behind the presented work on model matching and underscores the importance of these results. With the tuned model, a locomotion policy trained in simulation has a higher chance of success when transferred to the real robot, reducing the sim-to-real gap. Demonstrating a successful sim-to-real

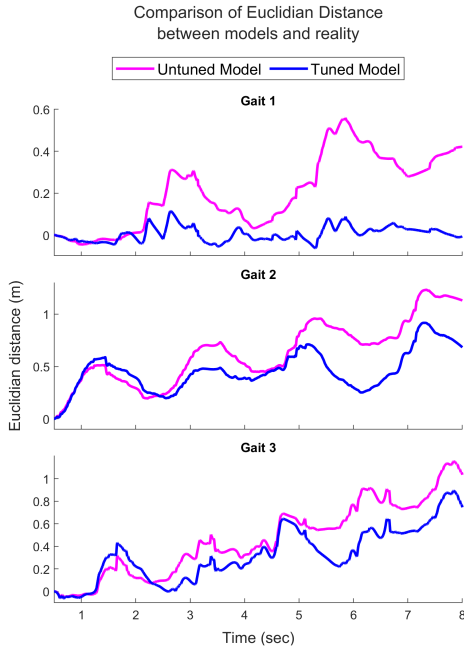**Comparison of Euclidian Distance between models and reality**

Fig. 10. Comparison of the Euclidean distance (error metric) between the actual platform's head position captured by OptiTrack and tuned/untuned models for sidewinding @ 0.35, 0.5, and 0.65 Hz.
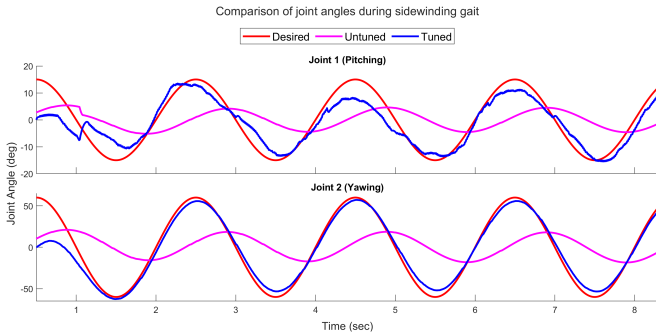

Fig. 11. Shows a comparison between the actuator joint responses from the actual hardware platform (red), the tuned (blue) and untuned (magenta) models for a sidewinding gait at @ 0.5 Hz.

transfer of locomotion policy will be presented in future work.

REFERENCES

[1] D. Rollinson and H. Choset, "Gait-based compliant control for snake robots," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 5138–5143.

[2] C. Studer, *Numerics of Unilateral Contacts and Friction: Modeling and Numerical Time Integration in Non-Smooth Dynamics* (Lecture Notes in Applied and Computational Mechanics), en. Berlin Heidelberg: Springer-Verlag, 2009, ISBN: 978-3-642-01099-6. DOI: 10.1007/978-3-642-01100-9. [Online]. Available: https://www.springer.com/gp/book/9783642010996 (visited on 05/30/2020).

[3] S. L. Cleac'h, T. Howell, S. Yang, *et al.*, *Fast Contact-Implicit Model-Predictive Control*, Jan. 2023.

[4] J. Carius, R. Ranftl, V. Koltun, and M. Hutter, "Trajectory Optimization With Implicit Hard Contacts," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3316–3323, Oct. 2018.

[5] A. M. Deshpande, E. Hurd, A. A. Minai, and M. Kumar, "DeepCPG Policies for Robot Locomotion," en, *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1–1, 2023.
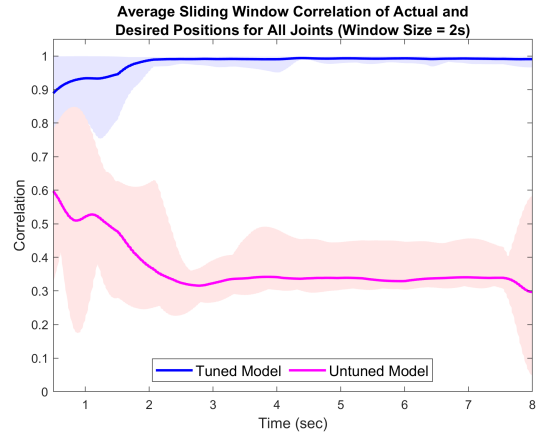
Fig. 12. Shows a comparison of average sliding window correlation (obtained by taking average of all the 11 correlation curves displayed in the above plot) of yawing joint positions for tuned and default simulators with desired signal (for the sidewinding gait @ 0.5 Hz). The model matching improved the actuator model in the the simulator as seen from the improved average correlation factor between desired and actual joint positions. Shaded regions represent the range between min and max correlation across all joints.

[6] X. Liu, R. Gasoto, C. Onal, and J. Fu, *Learning to Locomote with Deep Neural-Network and CPG-based Control in a Soft Snake Robot*, en, Mar. 2020.

[7] Z. Wang, Q. Gao, and H. Zhao, "CPG-Inspired Locomotion Control for a Snake Robot Basing on Nonlinear Oscillators," en, *Journal of Intelligent & Robotic Systems*, vol. 85, no. 2, pp. 209–227, Feb. 2017.

[8] Y. Gong, R. Hartley, X. Da, *et al.*, "Feedback Control of a Cassie Bipedal Robot: Walking, Standing, and Riding a Segway," in *2019 American Control Conference (ACC)*, Jul. 2019, pp. 4559–4566.

[9] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, *ANYmal Parkour: Learning Agile Navigation for Quadrupedal Robots*, en, Jun. 2023.

[10] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu, *Sim-to-Real Transfer for Biped Locomotion*, Aug. 2019.

[11] B. Qin, Y. Gao, and Y. Bai, "Sim-to-real: Six-legged Robot Control with Deep Reinforcement Learning and Curriculum Learning," in *2019 4th International Conference on Robotics and Automation Engineering (ICRAE)*, Nov. 2019, pp. 1–5.

[12] Z. Li, X. Cheng, X. B. Peng, *et al.*, *Reinforcement Learning for Robust Parameterized Locomotion Control of Bipedal Robots*, Mar. 2021.

[13] T. Johannink, S. Bahl, A. Nair, *et al.*, "Residual Reinforcement Learning for Robot Control," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 6023–6029.

[14] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning Quadrupedal Locomotion over Challenging Terrain," en, *Science Robotics*, vol. 5, no. 47, eabc5986, Oct. 2020.

[15] A. Salagame, N. Bhattachan, A. Caetano, *et al.*, *How Strong a Kick Should be to Topple Northeastern's Tumbling Robot?* Nov. 2023.

[16] S. Jiang, A. Salagame, A. Ramezani, and L. Wong, *Snake Robot with Tactile Perception Navigates on Large-scale Challenging Terrain*, Dec. 2023.

[17] S. Jiang, A. Salagame, A. Ramezani, and L. Wong, *Hierarchical RL-Guided Large-scale Navigation of a Snake Robot*, Dec. 2023.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, *Proximal Policy Optimization Algorithms*, Aug. 2017.

[19] G. Meirion-Griffith, I. Nesnas, L. Kerber, *et al.*, "Accessing mars recurring slope lineae: Mobility systems analysis," in *2018 IEEE Aerospace Conference*, Mar. 2018, pp. 1–13.

[20] H. Marvi, C. Gong, N. Gravish, *et al.*, "Sidewinding with minimal slip: Snake and robot ascent of sandy slopes," *Science*, vol. 346, no. 6206, pp. 224–229, Oct. 2014.

[21] H. Kolvenbach, P. Arm, E. Hampp, *et al.*, "Traversing Steep and Granular Martian Analog Slopes With a Dynamic Quadrupedal Robot," *arXiv:2106.01974 [cs]*, Jun. 2021.

[22] A. Ramezani and J. Grizzle, "Atrias 2.0, a new 3d bipedal robotic walker and runner," in *Adaptive Mobile Robotics*, WORLD SCIENTIFIC, May 2012, pp. 467–474.

[23] P. Dangol and A. Ramezani, "Feedback design for Harpy: A test bed to inspect thruster-assisted legged locomotion," in *Unmanned Systems Technology XXII*, vol. 11425, SPIE, May 2020, pp. 49–55.

[24] A. Ramezani, J. W. Hurst, K. Akbari Hamed, and J. W. Grizzle, "Performance Analysis and Feedback Control of ATRIAS, A Three-Dimensional Bipedal Robot," *Journal of Dynamic Systems, Measurement, and Control*, vol. 136, no. 021012, Dec. 2013.

[25] *Robot Modeling and Control, 2nd Edition — Wiley*, en-us.

[26] K. Liang, E. Sihite, P. Dangol, A. Lessieur, and A. Ramezani, "Rough-Terrain Locomotion and Unilateral Contact Force Regulations With a Multi-Modal Legged Robot," in *2021 American Control Conference (ACC)*, May 2021, pp. 1762–1769.

[27] P. Dangol, E. Sihite, and A. Ramezani, "Control of Thruster-Assisted, Bipedal Legged Locomotion of the Harpy Robot," eng, *Frontiers in robotics and AI*, vol. 8, p. 770 514, 2021.

[28] A. Salagame, M. Gianello, C. Wang, *et al.*, *Quadrupedal Locomotion Control On Inclined Surfaces Using Collocation Method*, Dec. 2023.

[29] E. Sihite, B. Mottis, P. Ghanem, A. Ramezani, and M. Gharib, *Efficient Path Planning and Tracking for Multi-Modal Legged-Aerial Locomotion Using Integrated Probabilistic Road Maps (PRM) and Reference Governors (RG)*, May 2022.