Enhancing Robot Design and Locomotion with Model-Agnostic Meta-Learning and Markov Decision Processes

Shen Yining, Xu Zherui, Yong Zhitao, Fan Wangzhuo, Zhang Caomeng, Qiao Kai (in ID order)

Abstract—**Reinforcement learning is now widely used in robotics for tasks like setting design parameters and training movement policies. In this project, we focused on a locomotion control problem using the ANYmal quadrupedal robot. We built and used a locomotion model by applying the Markov Decision Process (MDP) and Multi-Layer Perceptron (MLP). The MDP helped model the decision-making process, defining the robot's states, actions, and rewards to improve its movements and interactions with the terrain. The MLP processed input data such as the robot's state, terrain properties, and contact information to create a representation of the current state. The model then decided on actions for effective movement. By using additional machine learning techniques, we made the model perform better, allowing the robot to adapt to different terrains and conditions. This approach shows the potential of reinforcement learning to improve robotic abilities, making robots more efficient, adaptable, and intelligent for various tasks in different environments.**

I. Introduction

Designing a robot involves many decisions about various parameters, each of which can greatly affect the robot's performance. However, this process is often slowed down by not fully understanding how these parameters interact and influence the robot's behavior. This lack of clarity makes it hard to predict the outcomes of design choices, leading to a trial-and-error method that is time-consuming and inefficient.

One major challenge in this process is the limited knowledge about the relationships between design parameters and performance metrics. For example, changes in a robot's joint configurations, actuator strengths, or sensor placements can greatly impact its agility, precision, and stability. Without a good understanding of these effects, designers struggle to optimize the robot's functionality. This uncertainty can result in designs that do not fully utilize the potential of the available technology.

Additionally, the design process is often hampered by insufficient design principles that do not offer enough guidance or inspiration. Many current design frameworks are not robust enough to address the complexities of modern robotic systems. They may overlook important aspects such as adaptability to different environments, energy efficiency, or the integration of advanced algorithms for autonomous operation. As a result, designers may rely on outdated or simplistic models that do not capture the complexities of real-world applications.

This lack of effective design principles and insights can lead to several inefficiencies. Designers may spend too much time iterating on prototypes, making small adjustments without a clear direction. This not only delays the development process but also increases costs. Additionally, the final product may still not meet the desired performance standards, requiring further modifications and refinements.

To overcome these challenges, a more systematic and informed approach to robot design is essential. This involves developing comprehensive models that accurately represent the interplay between design parameters and performance outcomes. Advanced simulation tools and data-driven methods can play a key role in this, enabling designers to visualize and predict the effects of their decisions. Furthermore, establishing a set of robust design principles that incorporate the latest advancements in robotics can provide a solid foundation for innovation and efficiency.

By enhancing our understanding of parameter interactions and refining design methodologies, we can significantly improve the efficiency and effectiveness of the robot design process, leading to more capable and reliable robotic systems.

II. Related Work

In conventional design paradigm, designers typically rely on approximations, simulations, or bio-inspired solutions to make design decisions . Examples of quadrupedal robots designed using this conventional approach include Mini Cheetah from MIT[4], HyQ[7], and ANYmal from ETH[2]. Although some sources mention considerations like range of motion and inertia in leg design or certain performance objectives, the process for determining final values is often unclear.[1]

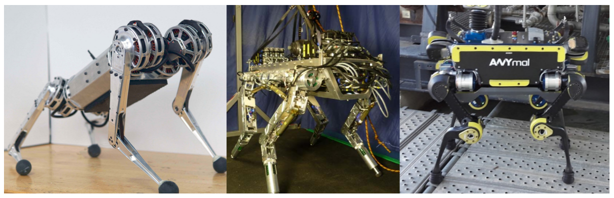


Fig. 1. MiniCheetah, HyQ, and ANYmal

As the proposal of machine learning resolves the problem by using high level computation capability through almost every possible solution to the design, robotic design can be greatly powered, powering up the efficiency in obtaining the possible best parameters combination.

To adopt a more quantitative approach to robot design, computational optimization methods have been introduced to find the best designs. As a bilevel optimization problem, the outer optimization focuses on refining design parameters, while the inner problem determines the best control parameters for each design. The inner loop typically involves multiple sub-objectives and is generally non-differentiable concerning the design parameters.

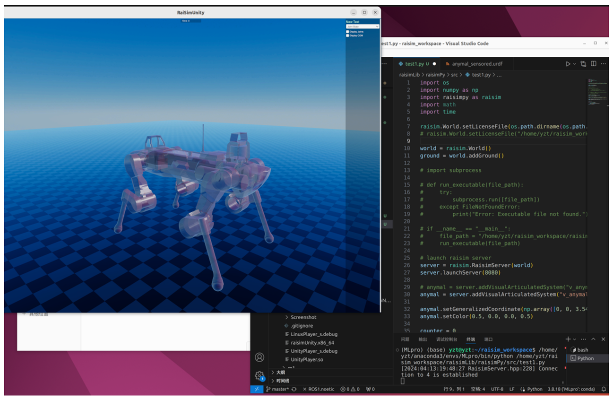
Existing methods can be categorized into two main approaches: gradient-based and gradient-free. Gradient-based methods aim to establish a differentiable relationship between control performance (the outcome of the inner loop) and design parameters. While Gradient-free methods are more suitable for non-convex problems. Initially developed in the computer graphics community, this approach has expanded to mechanical design optimization.[8] For instance, some techniques perform joint optimization of design and control parameters to maximize the speed of a quadrupedal robot using the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). This strategy has also been integrated with genetic algorithm to achieve optimal leg designs for quadrupedal and walking robots.

A common element among these works is their reliance on model-based control approaches. While these methods are generalizable and intuitive, they have several limitations for complex systems like legged robots. First, they often use simplified models to reduce complexity. For example, some trajectory optimization methods rely on centroidal dynamics and ignore limb masses, leading to significant dynamics mismatch, which means the optimized controller may not work on the actual system. Second, the resulting motions depend on handcrafted primitives and are limited to predefined tasks and trajectories, such as specific gait patterns or base trajectories. Lastly, since motion parameters and simplified dynamics models are often manually developed or tuned for specific instances, it is difficult to claim that the optimized motion is truly the best for each design. [1]

III. Data

The Robotic Systems Lab at ETH Zürich demonstrated the application of model-agnostic meta-learning to optimize the design parameters for the legged robot ANYmal. This process involved two main components: an outer circle to optimize the link lengths of the robot's thigh and shank, and an inner circle to train a decision policy using model-agnostic meta-learning techniques on a small dataset. The primary goal was to refine the physical parameters of ANYmal's leg links for optimal performance and to develop a robust, versatile decision policy.

To replicate this work, we acquired ANYmal robot data[3] and a terrain generator[5] for the simulation environment through GitHub open source. This setup was crucial for creating a realistic simulation environment capable of real-time data transfer, which is essential for accurate policy training. We concentrated on training the policy for quadrupedal locomotion, defining the parameters for the training program, and developing a comprehensive evaluation method to assess the policy's performance.



```python
import os
import numpy as np
import raisimpy as raisim
import math
import time

raisim.World.setLicenseFile(os.path.dirname(os.path.
# raisim.World.setLicenseFile("/home/yzt/raisim_work

world = raisim.World()
ground = world.addGround()

# import subprocess

# def run_executable(file_path):
#     try:
#         subprocess.run([file_path])
#     except FileNotFoundError:
#         print("Error: Executable file not found.")

# if __name__ == "__main__":
#     file_path = "/home/yzt/raisim_workspace/raisin
#     run_executable(file_path)

# launch raisim server
server = raisim.RaisimServer(world)
server.launchServer(8080)

# anymal = server.addVisualArticulatedSystem("v_anym
anymal = server.addVisualArticulatedSystem("v_anymal

anymal.setGeneralizedCoordinate(np.array([0, 0, 3.54
anymal.setColor(0.5, 0.0, 0.0, 0.5)

counter = 0
```

```
(MLpro) (base) yzt@yzt:~/raisim_workspace$ /home/
yzt/anaconda3/envs/MLpro/bin/python /home/yzt/rai
sim_workspace/raisimLib/raisimPy/src/test1.py
[2024:04:13:19:48:27 RaisimServer.hpp:228] Connec
tion to 4 is established
```

Fig. 2. Simulation Environment[6]

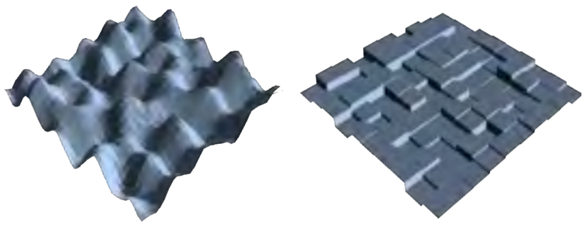


Fig. 3. Parameterized Terrain of Hills and Steps[6]

The simulation environment mimicked real-world conditions, providing relevant data for refining the decision policy. Ensuring simulation fidelity was critical, as it needed to accurately represent the dynamics and constraints of ANYmal's movements. This environment allowed for the collection of data that was used to train the machine learning model effectively.

The training program parameters included various aspects of the robot's kinematics and dynamics, such as joint angles, link lengths, and motor torques. Careful selection and tuning of these parameters were essential to create a training regimen that would teach the robot efficient and stable movement. The iterative training process involved repeated cycles of training and evaluation, utilizing model-agnostic meta-learning algorithms designed to generalize well even with small datasets. These algorithms helped develop a robust decision policy adaptable to different conditions.

Evaluation methods were critical in this process. We developed a thorough evaluation framework to test the robot's performance under various scenarios, measuring metrics like stability, energy efficiency, and speed. Systematically testing the policy in different conditions allowed us to identify areas for improvement and refine the policy accordingly.

The combined approach of optimizing physical parameters and developing a robust decision policy significantly enhanced ANYmal's performance. By employing a precise simulation environment, well-defined training parameters, and comprehensive evaluation methods, we ensured that the resulting policy was both effective and adaptable. This work not only advances robotic locomotion but also demonstrates the powerful potential of machine learning in complex engineering tasks. The integration of open-source data and tools underscores the collaborative and innovative nature of modern robotics research.

IV. Methods

We harness Meta-RL for training policies across an array of design parameters and terrains. Through random sampling, we ensure versatility and robustness in policy learning, adapting to varied conditions. This approach facilitates broader exploration and enhances the adaptability of policies to unforeseen challenges in diverse environments.

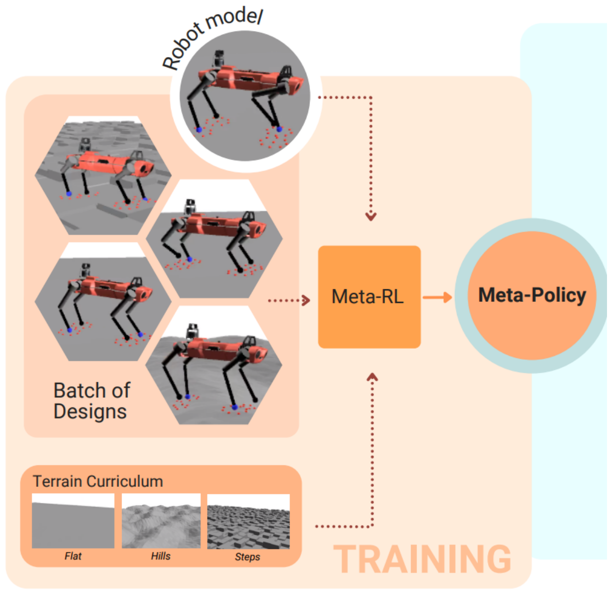


Fig. 4. Overview of the Workflow[1]

A. Markov Decision Process Modeling

The locomotion control problem is modeled as a Markov Decision Process (MDP), a mathematical framework designed to model decision-making in environments where outcomes result from a combination of random events and the decisions made by an agent. An MDP is characterized by a set of states representing different scenarios, a set of actions available to the decision-maker, transition probabilities that define the likelihood of moving from one state to another based on a chosen action, and a reward function that assigns a numerical value to each action-state pair to reflect the desirability of outcomes.

The primary objective within an MDP is to discover an optimal policy, which is a strategy that specifies the best action to take in each state to maximize the expected cumulative reward over time. This involves making a series of decisions that balance immediate and future rewards. Solving an MDP to find this optimal policy typically employs dynamic programming techniques such as value iteration or policy iteration. Value iteration systematically updates the value of each state by considering the expected rewards of possible actions, while policy iteration involves evaluating and improving a given policy iteratively.

In our context, we constructed the Markov Decision Process model to address the complexities of quadrupedal locomotion control. This model incorporated the unique states of the robot's movement dynamics, including joint angles, velocities, and positional coordinates. The actions corresponded to various motor commands that could be executed by the robot's actuators. Transition probabilities were derived from the robot's physical properties and environmental interactions, ensuring realistic modeling of the robot's behavior.

The reward function was designed to encourage stable and efficient movement patterns. It considered factors such as energy consumption, balance, speed, and adherence to desired trajectories. By maximizing this reward function, the robot learns to move in a way that is both effective and efficient.

Through iterative application of value iteration and policy iteration, we refined the decision-making policy to enhance the robot's locomotion. This comprehensive approach allowed us to systematically address the challenges of robot control, ensuring robust performance in various environments. In this context, we model the MDP with its state space $\mathcal{S}$, action space $\mathcal{A}$, transition probability function $\mathcal{P}(s_{t+1}|s_t, a_t)$, and reward function $\mathcal{R}(s_t, a_t, s_{t+1}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ determined as the Table I.

TABLE I
Action Space of the Markov Decision Process[6]

f_{l_1}	Leg 1 Frequency
r_{f_1}	Foot 1 Position Residual
f_{l_2}	Leg 2 Frequency
r_{f_2}	Foot 2 Position Residual
f_{l_3}	Leg 3 Frequency
r_{f_3}	Foot 3 Position Residual
f_{l_4}	Leg 4 Frequency
r_{f_4}	Foot 4 Position Residual
θ_{11}	Leg 1 Joint 1 Angle
θ_{12}	Leg 1 Joint 2 Angle
θ_{21}	Leg 2 Joint 1 Angle
θ_{22}	Leg 2 Joint 2 Angle
θ_{31}	Leg 3 Joint 1 Angle
θ_{32}	Leg 3 Joint 2 Angle
θ_{41}	Leg 4 Joint 1 Angle
θ_{42}	Leg 4 Joint 2 Angle

The state space is a comprehensive set of variables crucial for effective locomotion control. It includes the velocity command, which specifies the desired speed and direction of movement. Additionally, it encompasses the linear and angular body velocities, providing information on the robot's current motion dynamics. The joint states, representing the positions and velocities of each joint, are also integral to the state space. Furthermore, the state space includes the frequency and phase of the gait pattern generators for each foot, which are essential for coordinating leg movements. Notably, the policy learns to modulate these periodic leg phases. Finally, the state space incorporates the two most recent actions taken by the policy. These elements are detailed in Table II.

TABLE II
STATE SPACE OF THE MARKOV DECISION PROCESS[6]

Desired direction	2D
Desired turning direction	1D
Gravity vector	3D
Base angular velocity	3D
Base linear velocity	3D
Joint position/velocity	24D
FTG phases	8D
FTG frequencies	4D
Base frequency	1D
Joint position error history	24D
Joint velocity history	24D
Foot target history	24D

To minimize redundant data acquisition and maximize the speed of policy training, we implement a strategic trade-off by simplifying the terms in the reward function. This approach reduces computational complexity and enhances efficiency during the training process. By focusing on the most critical elements of performance, we streamline the learning process without significantly compromising the quality of the learned policy. The simplified reward function, which prioritizes essential performance metrics such as stability, energy efficiency, and adherence to desired trajectories, is detailed in Table III. This balance ensures faster convergence and more effective policy optimization.

TABLE III
$R(s) = 0.5r_v + 0.2r_w + 0.1r_{v_stability} + 0.1r_{w_stability}$[1]

Linear Velocity r_v	$exp(-1.5\|\|v_{xy_{target}} - v_xy\|\|^2)$
Angular Velocity r_w	$\exp(-2.0 \cdot (\omega_{z_{target}} - \omega_z))$
Linear Base Stability $r_{v_{stability}}$	$\exp(-1.5 \cdot v_z^2)$
Angular Base Stability $r_{w_{stability}}$	$\exp(-1.5 \cdot \|\|w_{xy}\|\|^2)$

B. Multi Layer Perceptron Policy

The decision policy relies on multi-layer perceptron (MLP) to process inputs, including the robot's current state, terrain properties, and its contact with the terrain. These MLP compute a latent embedding, which captures the present state of the robot and the environment. Simultaneously, the model determines an action. The primary goal of the training process is to encourage effective locomotion in specified directions. By rewarding movement that adheres to these directions, the policy becomes adept at navigating various terrains while maintaining stability and efficiency. This method ensures that the robot can adapt to dynamic and diverse environments.

C. Adaptation with Model-Agnostic Meta-Learning

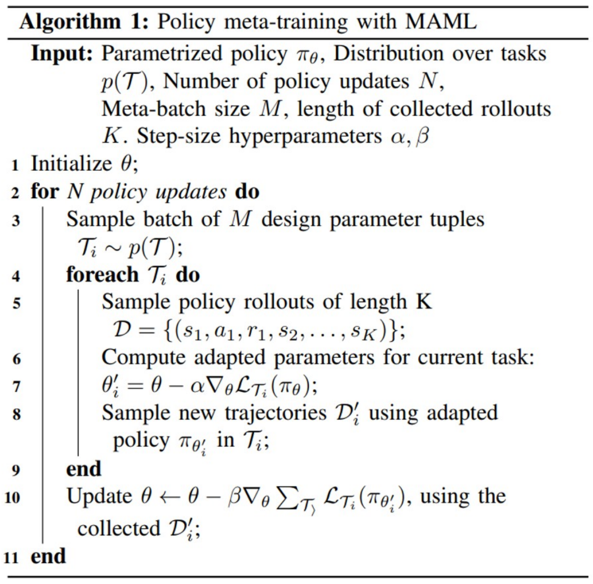
The training objective is to optimize the design lengths of the upper leg (thigh) and the lower leg (shank) of a robotic system. Following the approach by Belmonte-Baeza [1], we set the initial lengths of both the thigh and the shank to 350mm. For testing, we define a length range from 210mm to 490mm, representing a ratio of $[0.6, 1.4]$ relative to the initial length. The task $p(\mathcal{L})$ for the MAML training is uniformly sampled within these bounds.

It is crucial to note that variations in leg length not only impact the kinematic locomotion but also alter the dynamics due to changes in inertia. These factors significantly increase the complexity of the training process. The challenge lies in ensuring that the policy can adapt to these variations and maintain effective performance across different leg lengths.

By sampling tasks within the specified bounds, the training aims to develop a robust and adaptable policy that can handle the diverse kinematic and dynamic scenarios resulting from changes in leg length. This approach ensures that the robotic system can navigate effectively and efficiently, regardless of the specific leg dimensions within the defined range. The variability in design parameters necessitates a comprehensive training regime that addresses both the kinematic and dynamic aspects of locomotion, ultimately leading to a more versatile and resilient robotic system.

V. EXPERIMENTS

The research validates the Meta-RL approach for training design-conditioned policies by comparing a meta-policy against a naive policy, which is trained over uniformly sampled design parameters. The comparison shows the average rewards obtained by the two policies across different parameters. The results demonstrate that the meta-policy consistently outperforms the naive multi-task policy in all scenarios.



Algorithm 1: Policy meta-training with MAML

```
Input: Parametrized policy $\pi_\theta$, Distribution over tasks
       $p(\mathcal{T})$, Number of policy updates $N$,
       Meta-batch size $M$, length of collected rollouts
       $K$. Step-size hyperparameters $\alpha, \beta$
1  Initialize $\theta$;
2  for $N$ policy updates do
3      Sample batch of $M$ design parameter tuples
        $\mathcal{T}_i \sim p(\mathcal{T})$;
4      foreach $\mathcal{T}_i$ do
5          Sample policy rollouts of length K
            $\mathcal{D} = \{(s_1, a_1, r_1, s_2, \ldots, s_K)\}$;
6          Compute adapted parameters for current task:
7          $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\pi_\theta)$;
8          Sample new trajectories $\mathcal{D}'_i$ using adapted
            policy $\pi_{\theta'_i}$ in $\mathcal{T}_i$;
9      end
10     Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_\rangle} \mathcal{L}_{\mathcal{T}_i}(\pi_{\theta'_i})$, using the
        collected $\mathcal{D}'_i$;
11 end
```

Fig. 5. Policy Training with MAML[1]

To further verify the performance of the meta-policy, the research compares it against a set of specialized policies trained for specific designs. After undergoing adaptation steps, the meta-policy achieves rewards comparable to those of the specialized policies, showcasing its ability to reach near-optimal performance levels. This analysis confirms the effectiveness of the meta-policy and justifies its use in subsequent design optimization experiments.

In the design optimization experiments, the research applies the meta-policy to evaluate different design instances. For each instance, the meta-policy is fine-tuned to adapt to the specific design parameters. This fine-tuning process allows the meta-policy to adjust and optimize its performance for the given design, ensuring that it can handle a wide range of design variations effectively.

The superior performance of the meta-policy over the naive policy can be attributed to its ability to generalize across different design parameters. Unlike the naive policy, which is trained uniformly over a broad parameter space, the meta-policy is specifically designed to adapt to variations in design. This targeted adaptability enables the meta-policy to perform better across diverse scenarios, providing a more robust and efficient solution.

Moreover, the meta-policy's performance is validated through comparisons with specialized policies. These specialized policies are trained for specific design configurations, making them highly optimized for those particular settings. Despite this, the meta-policy, after adaptation, manages to achieve similar reward levels, indicating its strong capability to approximate optimal performance across various designs.

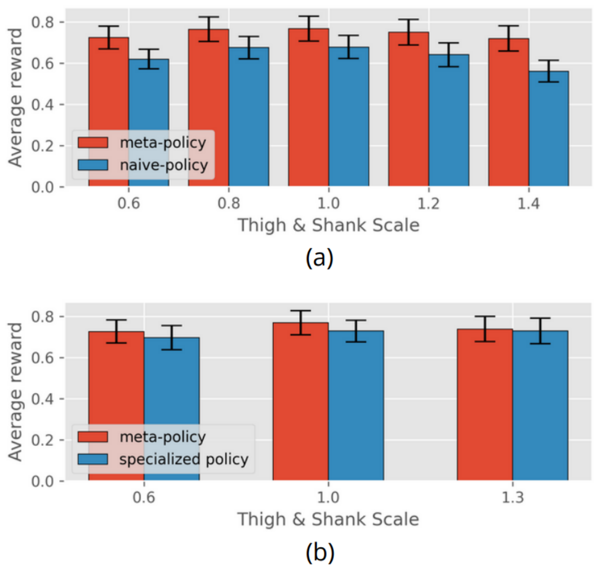


Fig. 6. Comparison between Meta-Policy and Naive-Policy[1]

This adaptability is a significant advantage in practical applications where design parameters can vary widely. By using a meta-policy, it is ensured that the robot or system can perform optimally without the need for extensive retraining for each new design. This not only saves time but also enhances the overall efficiency of the development process.

Furthermore, the fine-tuning of the meta-policy for specific design instances demonstrates its flexibility and robustness. Each design instance may present unique challenges and requirements, and the ability of the meta-policy to adapt through fine-tuning ensures that it can meet these demands effectively. This adaptability is crucial for developing versatile and reliable robotic systems capable of performing well under different conditions.

In conclusion, the validation of the Meta-RL approach through comparisons with both naive and specialized policies highlights the effectiveness of the meta-policy in handling diverse design parameters. Its superior performance, adaptability, and robustness make it a valuable tool for design optimization in robotics. By leveraging the meta-policy, efficient and effective solutions that are capable of adapting to a wide range of design variations can be achieved, ultimately leading to the development of more capable and reliable robotic systems.

VI. Conclusion

This project explores the Markov Decision Process (MDP) and Model-Agnostic Meta-Learning (MAML). By closely following the research paper's steps, the advantages of MAML in task training become evident when compared to the normal training process. The knowledge gained through this project has practical applications in robot design and tasks such as locomotion.

The MAML framework significantly enhances the efficiency and effectiveness of training policies for various tasks. Unlike traditional methods, MAML enables quick adaptation to new tasks with minimal adjustments. This quality makes it highly valuable for designing robots that can operate in diverse environments and perform different functions without extensive retraining.

In this project, the benefits of MAML were demonstrated through a series of experiments. The meta-policy, trained using MAML, consistently outperformed the naive policy, which was trained uniformly across various design parameters. This performance was assessed by comparing the average rewards obtained by both policies across different parameters, with the meta-policy showing superior results in all scenarios.

Additionally, the meta-policy's performance was validated against specialized policies trained for specific designs. After adaptation steps, the meta-policy achieved rewards comparable to those of the specialized policies, indicating its ability to reach near-optimal performance levels across various designs. This adaptability is crucial for practical applications where design parameters can vary widely.

The potential of MAML extends beyond this project. With ongoing improvements, a similar framework could be applied to other robotic design and simple locomotion problems. The ability to quickly adapt to new tasks and environments makes MAML an invaluable tool for developing versatile and reliable robotic systems. Future work can further refine this approach, enhancing its applicability to a broader range of tasks and challenges in the field of robotics.

In summary, the project highlights the significant advantages of MAML in task training and its potential applications in robot design and locomotion. With continuous improvement, this framework promises to advance the development of adaptable and efficient robotic systems capable of handling various tasks and environments.

References

[1] Álvaro Belmonte-Baeza, Lee Joonho, Giorgio Valsecchi, and Marco Hutter. Meta reinforcement learning for optimal design of legged robots. *IEEE ROBOTICS AND AUTOMATION LETTERS*, 2022. URL arXiv:2210.02750v1.

[2] Marco Hutter. Anymal - toward legged robots for harsh environments. *Advanced Robotics*, (17), 2017. doi: https://doi.org/10.3929/ethz-b-000194231.

[3] Jemin Hwangbo. Raisimlib, 2024. URL https://github.com/raisimTech/raisimLib.

[4] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. *2019 International Conference on Robotics and Automation*, 2019.

[5] Joonho Lee. Leggedrobotics/learning quadrupedal locomotion over challenging terrain supplementary, 2020. URL https://github.com/leggedrobotics/learning_quadrupedal_locomotion_over_challenging_terrain_supplementary.

[6] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, (47), 2020. doi: https://doi.org/10.1126/scirobotics.abc5986.

[7] Claudio Semini. Design of hyq – a hydraulically and electrically actuated quadruped robot. *Systems and Control Engineering*, 2011.

[8] Karl Sims. Evolving virtual creatures. 1994.