Learning-based Co-design for Bio-inspired Quadrupeds

F.G.M. Boekel

Master Thesis Robotics

Learning-based Co-design for Bio-inspired Quadrupeds

Master of Science Thesis by

F.G.M. Boekel

In partial fulfillment of the requirements
for the degree of Master of Science in Robotics
at Delft University of Technology.

To be defended at January 9, 2025, on 11:00.

Supervisor: Dr. C. (Cosimo) Della Santina
Dr. J. (Jiatao) Ding

Author: Fabio G.M. Boekel
Student Number: 4855892
Faculty: Mechanical Engineering
Research Group: Learning & Autonomous Control Group
Date: December 25, 2024

Learning-based Co-design for Bio-inspired Quadrupeds

F.G.M. Boekel
Master Thesis Robotics

***Abstract*—Designing robotic systems such as quadrupeds is challenging due to the intricate relationship between motion and design, particularly when aiming to replicate the agility, efficiency, and versatility of animals. Co-design simplifies robotic development by simultaneously optimizing physical design and control algorithms in an integrated way. While most prior work validates co-design approaches in simulation, our research bridges this gap by transitioning optimized designs to real-world implementation. To achieve this, we developed a modular quadruped platform with bio-inspired legs that enables the physical implementation of the optimized designs. Our design space, which includes leg segment lengths, spring stiffness, and engagement angle, was optimized to maximize energy efficiency for real-world tasks. We propose a simplified learning-based co-design framework that combines reinforcement learning to create a universal locomotion controller with Bayesian optimization to select the best design. Real-world tests demonstrate a significant reduction in the cost of transport—18.6% for inspection tasks and 35.7% for payload tasks—compared to the nominal design without springs. In simulations, the universal controller adapts well across robot configurations, and the optimization process remains consistent across runs. Although some discrepancies between simulation and real-world performance remain, our findings underscore the potential of co-design to address complex trade-offs in real-world robotic system design.**

I. Introduction

THE animal kingdom showcases an astounding array of agile, efficient, and versatile creatures, surpassing even the most advanced robots in their ability to adapt to complex and dynamic environments. From the cheetah's high-speed sprint to the gecko's nimble climbing, animals have evolved bodies and brains to tackle diverse challenges. This remarkable integration of physical form, cognitive function, and behavioral adaptability serves as inspiration for roboticists striving to create machines with similar capabilities.

Quadruped robots have made significant strides in areas such as agile jumping [1], [2], efficient walking [3], and show versatile performance in hiking [4] and parkour [5]–[8]. However, a noticeable performance gap remains between robots and their biological counterparts [9]. A contributing factor is the difficulty of designing robotic systems, which arises from the deep interconnection between a robot's motion and its design.

The traditional sequential approach to robot development, treating the mechanical design as fixed and developing the control system to fit it, can lead to suboptimal performance. In this approach, the control system compensates for limitations in the design. This can be fixed with iterative improvements to both controllers and designs. However, this requires significant time, resources, and expert knowledge. This raises a critical question: How can we improve the way we design robots to narrow this performance gap?

Fig. 1: The modular "Dogger" hardware platform with bio-inspired legs.

One way to design improved robotic systems is to exactly replicate anatomical structures and proportions found in nature, such as an ostrich-inspired robot for efficient movement [10] and a rat-inspired robot for confined spaces [11]. However, this approach requires extensive expert knowledge and is complicated by fundamental differences between current robotic technologies and biological systems. For example, robots typically rely on rotary motors, while animals use muscles for movement. Rather than exactly replicating animals, we propose optimizing both structure and movement concurrently through co-design, similar to how mammals evolved diverse abilities while sharing similar limb structures [12].

Co-design is an integrated method of robotic development with the goal of optimizing the physical design and control algorithms simultaneously. Co-design is especially effective in complex design spaces with many parameters and trade-offs. Therefore, we decided to apply co-design to optimize bio-inspired legs. Our bio-inspired legs have elements seen in biological leg structures of mammals, such as having three segments and elasticity. The bio-inspired leg design introduces a more extensive and complex design space compared to traditional two-segment robotic legs, as certain parameters can achieve similar effects; for example, reducing the shank length decreases torque in the knee joint, but a comparable outcome can also be achieved by increasing the stiffness in the knee joint. In our work, we focus on co-designing the bio-inspired legs and their motion to maximize energy efficiency.

II. Related Work

A. Bio-inspired Robots

The bio-inspired leg design which we aim to optimize is based on structures found in mammals, characterized by their

three-segment configuration. Most quadrupeds of the current generation, including ANYmal [13], Go 1 [14], Spot [15], Solo [16], and Mini Cheetah [17], have two-segment legs. The three-segment leg can be seen in some older generation robots such as the Cheetah [18] and the Cheetah Cub [19]. Three-segment legs can reduce joint torque, and thereby help with energy efficiency, compared to two-segment designs [20].

To emulate the natural elasticity found in biological systems, we incorporate parallel elastic actuators (PEAs) into the knee joints of our robot design. In a PEA, the spring is connected in parallel with the motor. This means that a certain deviation in the actuator's joint angle will result in certain spring torque exerted on the actuator. PEAs can contribute to the efficiency of robot actuators by providing additional torques to the joint [21]. This again helps to improve the energy efficiency of quadrupeds. Currently, there are not many quadrupeds with PEAs. In [22], a PEA was integrated directly into a robotic leg, though not in a complete robot. Other existing hardware platforms have been modified to include PEAs, such as in the E-GO robot [23] and the ANYmal robot [24], to improve energy efficiency. Building on these works, our bio-inspired leg design incorporates PEAs from the outset.

B. Co-design

Co-design approaches in robotics can be outlined in what kind of optimization algorithms they use. Following [25], the distinction is made between evolutionary algorithm (EA) based approaches, optimal control (OC) based approaches, reinforcement learning (RL) based approaches, and hybrid approaches which combine these techniques.

EA-based approaches are inspired by natural selection. In [26], Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is used to optimize design parameters such as link lengths and torso dimensions and the parameters of a predefined parameterized gait trajectory. Thanks to co-design optimization, higher walking speeds could be reached.

OC-based approaches added design parameters to the optimization of state and control parameters. In [27], link lengths and actuator placements were optimized along with motion parameters such as actuator inputs and contact forces. This method was validated in simulation and on physical robots, showing a reduction in torque. Other OC-based approaches, such as [20], [28], use a bilevel optimization strategy instead of a monolithic strategy. Here, the lower level computes the state and control trajectories based on task and design parameters, and the upper level optimizes the design parameters using gradient information from the lower level.

RL presents another promising method for co-design. Studies such as [29] and [30] have demonstrated the ability of RL-based approaches to simultaneously optimize robot design and control for tasks such as walking and navigating challenging environments in a monolithic way. These approaches allow design parameters, such as leg lengths, to be treated as part of the policy optimization, ensuring that the design evolves alongside the control strategy. However, these methods can struggle with the exploration-exploitation trade-off, where the RL agent may initially favor suboptimal design configurations to prevent failure, such as standing still.

Hybrid approaches mix the above-based techniques. [31], uses a bilevel approach, where CMA-ES is used to select design parameters at the upper level and OC is used in the lower level.

Recent advances have combined EA or Bayesian Optimization (BO) with RL to address the challenges of design optimization. In [32] and [24], a two-step method was used. First, a universal policy capable of controlling a variety of robot designs was trained, and then an optimization framework selected the best design using the universal policy to control it. These works built on top of [33], which showed that a single policy could be trained to control a bipedal robot with different body sizes. In [24], PEAs are optimized to significantly reduce joint torques and energy consumption, allowing longer operational times on a single battery charge. The ANYmal robot optimized PEAs demonstrated a 30% reduction in joint torques and a 11% increase in walking distance compared to the robot without PEAs.

C. Contributions

Despite these advances, many co-design approaches are only validated in simulation [20], [26], [29], [30], [32], raising the question of how well these optimized designs will perform in the real-world and highlighting the need for more comprehensive real-world testing.

When co-design is applied to a physical robot [27], the approach is limited to simplified planar walking tasks. In another case [24], the approach is restricted to modifications of an existing quadrupedal hardware platform, where the design space is restricted to components such as springs rather than exploring the entire morphological design space, such as the shapes of the legs.

The reason for this gap is the absence of a modular hardware platform capable of implementing and validating optimized physical designs. Our research aims to bridge this gap between theoretical models and practical applications by transitioning optimized designs from simulation to the real-world. We aim to demonstrate that co-design can optimize the whole leg structure of the bio-inspired legs for energy efficiency in various tasks in the real-world. Our contributions are as follows:

- A novel, modular hardware platform featuring bio-inspired legs, allowing for the validation of optimized designs.
- A simplified RL and BO co-design framework inspired by [32] and [24], applied to a quadruped with bio-inspired legs focused on energy efficiency.
- Sim-to-real transfer of optimized designs and controllers with validation in real-world environments, ensuring that our optimizations result in more energy efficient quadrupeds.

Our approach advances how we can better design robot structures and controllers, bringing us closer to creating robots that match the agility, efficiency, and adaptability of their biological counterparts.

III. Methodology

Our method integrates bio-inspired leg design, co-design optimization, and sim-to-real transfer to develop energy-efficient quadruped robots. We start by detailing the design of the modular bio-inspired legs, including their three-segment configuration and PEAs. Next, we present a co-design framework that jointly optimizes the physical leg design and control policies leveraging RL and BO. Finally, we outline the steps taken to ensure successful real-world implementation of the optimized designs and share the implementation details.

A. Bio-inspired Leg Design

The bio-inspired leg consists of two main components: a three-segment leg and a PEA at the knee joint. We selected this bio-inspired setup to have a leg with diverse parameters, which makes it more challenging to choose the correct ones. We designed a modular leg that is easy to manufacture, so that the optimized design can be quickly implemented.

The leg that will be optimized has the following design parameters; the three link lengths (thigh, shank, and foot) and the mechanical properties of the knee's spring system (rotational stiffness and engagement angle of the spring). The link parameters are denoted as the three link scales: thigh (λ_1), shank (λ_2), and foot (λ_3). This means that the total length of each link is the respective scaling factor multiplied by the nominal length of each link in the leg (l_1^n, l_2^n, l_3^n). The rotational joint stiffness and the engagement angle are controlled by the wheel diameter ($\varnothing$) and spring offset (o). These five parameters are shared over all four legs. A design $\boldsymbol{d}$ in the design space $\mathcal{D}$ is parametrized as $\boldsymbol{d} = [\lambda_1, \lambda_2, \lambda_3, \varnothing, o]^T$ and these parameters are visualized in Fig. 2.

1) Three-Segment Leg: To simplify the design of the three-segment leg, we implement a parallel linkage for the third segment instead of introducing an additional degree of freedom. This approach preserves the essential characteristics of a three-segment leg while reducing mechanical complexity.

The leg structure is fully 3D printed and designed to exclude any electronics and belts. Modifications to $\lambda_1, \lambda_2, \lambda_3$ can be made by reprinting individual parts. This modular design allows the subcomponents of the leg to be easily interchanged. In addition, the entire leg structure can be removed in one piece without the need to detach a single motor, as can be seen in Fig. 3. These features enable rapid and cost-effective testing of the co-design approach.

Although an alternative design with adjustable link lengths could potentially make experimentation easier, it would increase both the robot's complexity and mass, possibly introducing more points of failure. Therefore, we opted for the above-mentioned approach of reprinting individual leg segments. By choosing this method, we maintain a simpler and more robust system while still allowing for a wide range of different leg configurations.

We selected the ranges of design parameters for the robot link scales $\lambda_1, \lambda_2, \lambda_3$ to be between $0.70 - 1.30$. This range was selected based on the minimum and maximum leg sizes that could fit onto the robot body. Reducing the lower limit was difficult because the parallel linkage joint in the thigh would interfere with the motor output axis. Increasing the upper limit would result in a collision between the front and back legs of the robot. The nominal length is centered within this range, so that the nominal leg length scaling factor is 1.00.

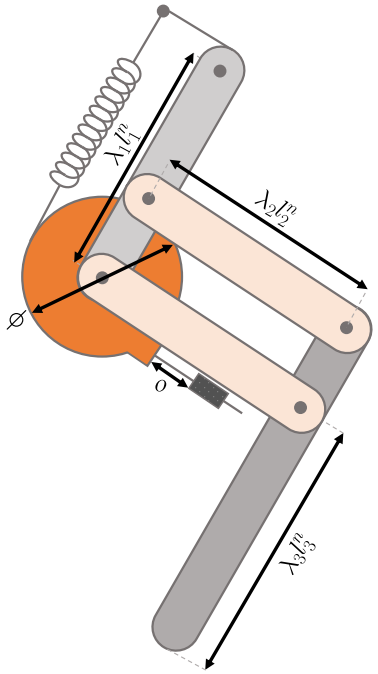


Fig. 2: Design parameters of the bio-inspired leg for co-design optimization: The lengths of the leg's links are scaled from their nominal values (l_1^n, l_2^n, l_3^n) using their scaling factors (λ_1, λ_2, λ_3). The spring wheel's diameter ($\varnothing$) determines the spring stiffness, while the spring offset (o) specifies the engagement angle.

Fig. 3: The "Dogger" hardware platform (Sec. III-D1) is shown in the center, alongside the modular bio-inspired legs, which can be swapped as entire modules without disconnecting electronics. Nearby are the interchangeable components, including the thigh, shank, foot, and spring wheel.

2) Parallel Elastic Actuator (PEA): In addition to the leg structure, we integrated a PEA into the knee joint to emulate

the elasticity found in biological systems.

Our PEA design draws inspiration from the spring wheel-wire mechanism presented by [23], [24], [34]. Our design consists of a spring connected to the thigh, with a wire attached to the spring. This wire runs over a wheel located at the knee joint, and both the wheel and the wire's end are connected to the shank. The diameter $\varnothing$ of the wheel influences the rotational stiffness of the joint. Since the wheel is 3D printed, it can be easily swapped, enabling faster and cheaper co-design experimentation as there is no need to purchase a new spring each time the robot is optimized with a different spring stiffness. The mechanism with different wheels can be seen on the right side of Fig. 4.

Furthermore, the design allows for adjusting o by altering the position of the clamps on the wire, effectively modifying the leg angle at which the spring activates. The position of the clamp can be changed by screwing or unscrewing the screw on the clamp and moving it along the wire, making it easy to select o. The clamps can be seen in the circle to the left of Fig. 4.

The torque that will be exerted by the parallel spring on the knee joint can be calculated by

$$\tau_i(q_i(t), \varnothing, c_s) = \max\left(\frac{1}{4}\varnothing^2 c_s(q_i(t)) + \frac{1}{2}\varnothing c_s o,\, 0\right), \quad (1)$$

where $\tau_i(q_i(t), \varnothing, c_s)$ is the torque that is produced by the spring on joint i which is dependent on the joint angle $q_i(t)$, $\boldsymbol{d}$ and the stiffness of the spring c_s.

For the PEAs, we choose a spring with a stiffness $c_s = 4180$ N/m, and $\varnothing$ range between $5 - 50$ mm and o between $0 - 10$ mm. c_s in combination with $\varnothing$ and o, result in a wide range of joint stiffnesses and engagement angles for the PEA. With these ranges, we can achieve a rotational joint stiffness between $0.03 - 2.62$ Nm/rad and an initial torque on the knee joint in the standing position between $0 - 5.62$ Nm. With these range combinations, we remain within the maximum deviation of the springs before it is deformed plastically. For example, with $\varnothing$ and o set to the middle values, it is still possible to fully collapse the leg. If $\varnothing$ and o are chosen at the maximum values, they can still compensate for gravity with the maximum value of λ_2 during a walking motion when the robot carries a payload. Tab. I lists the respective ranges for each parameter.

TABLE I: Parameter ranges and units.

Parameter	Range	Unit
Scale Length Thigh (λ_1)	$0.70 - 1.30$	[-]
Scale Length Shank (λ_2)	$0.70 - 1.30$	[-]
Scale Length Foot (λ_3)	$0.70 - 1.30$	[-]
Wheel Diameter ($\varnothing$)	$5 - 50$	[mm]
Offset (o)	$0 - 10$	[mm]

B. Co-design Approach

Our co-design approach uses a two-step optimization process to optimize both the physical design and the controller. This process consists of: 1) universal policy training using RL and 2) design selection through BO. Fig. 5 provides an overview of this algorithm.

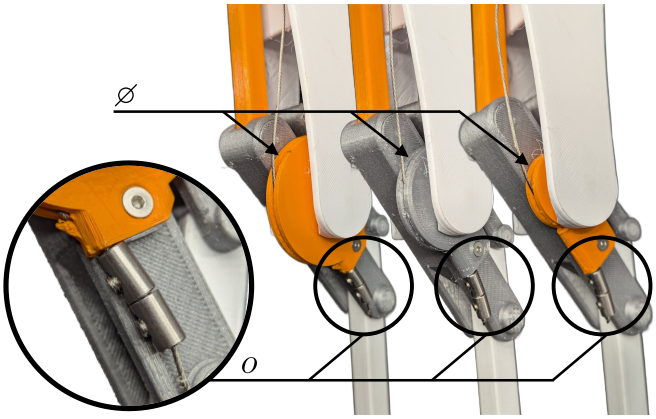


Fig. 4: The spring wheel-wire mechanism on the bio-inspired leg. On the left in the circle, the clamps to set o can be seen. On the right three different setups for the spring wheel, where $\varnothing$ goes from large to small. The wire going to the top is connected to the spring. The spring is not shown in this picture.

The universal policy π_u is a policy that can control and is well adapted to any robot design $\boldsymbol{d}$ in the design space $\mathcal{D}$. In the design selection step, we use π_u to control the different designs during the optimization process for a specific task k. We end up with the best design for a task d_k^* and a controller (π_u) that can control that design.

1) RL-based Universal Design-aware Policy: In the first step, we use RL to train a single policy π_u that can handle any robot design $\boldsymbol{d} \in \mathcal{D}$. Formally, RL is framed as learning a policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ in a Markov Decision Process (MDP). At each timestep t, the agent observes its state $\boldsymbol{s}_t$, in our case a partial observation $\boldsymbol{o}_t$, and takes an action $\boldsymbol{a}_t$. This interaction causes a transition to a new state $\boldsymbol{s}_{t+1}$, and the agent receives a reward r_t that evaluates the quality of its action. Using the new state $\boldsymbol{s}_{t+1}$ and its policy $\pi(\boldsymbol{a}_{t+1}|\boldsymbol{s}_{t+1})$, the agent determines the next action $\boldsymbol{a}_{t+1}$. The goal of the RL algorithm is to optimize the policy to maximize the cumulative rewards obtained over time.

In our design-aware formulation, the agent's observation $\boldsymbol{o}_t$ includes both the standard robot observations variables and the current design parameters $\boldsymbol{d}$. In this way, π_u can adapt its actions to different designs. At the start of training, a set of robot designs $\mathcal{D}' \subset \mathcal{D}$ is sampled, and the agent is trained across multiple environments, each with a randomly selected design $\boldsymbol{d} \in \mathcal{D}'$. By exposing the policy to various designs, we ensure that it can generalize and specialize across $\mathcal{D}$. The following sections describe the observation, action, reward, and termination structures used during training.

a) Observation Space: The agent's observation space includes both the design parameters $\boldsymbol{d}$ and the remaining portion of the observation. This portion is denoted as $\hat{\boldsymbol{o}}_t'$ and consists of: the commanded x and y velocities, the commanded yaw rate, the gravity vector of the robot's body, the actual yaw rate, the joint positions $\boldsymbol{q}_t^{mea}$, and the previous actions $\boldsymbol{a}_t$ taken. Together, these elements form the current observation $\boldsymbol{o}_t' = [\boldsymbol{d}, \hat{\boldsymbol{o}}_t']^T$. To incorporate temporal context, a 15-step

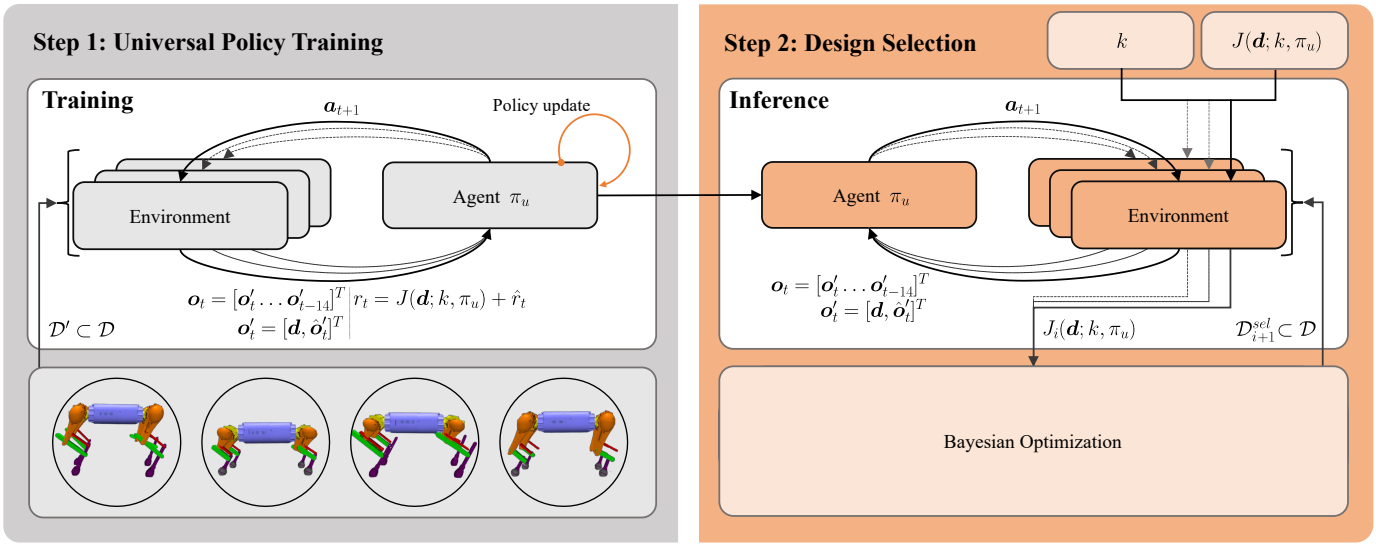


Fig. 5: Overview of the two-step co-design algorithm: **Step 1** trains the universal policy π_u by sampling a set of designs $\mathcal{D}'$ from the design space $\mathcal{D}$. During training, the agent observes $\boldsymbol{o}_t$ (a 15-step temporal context of the current observation $\boldsymbol{o}'_t$, including design parameters $\boldsymbol{d}$) and receives a reward r_t, which includes the cost function $J(\boldsymbol{d}; k, \pi_u)$ from Step 2 and the rest of the reward $\hat{r}_t$, to update the policy, and takes action $\boldsymbol{a}_{t+1}$. **Step 2** uses the trained π_u to generate trajectories for a selected set of designs $\mathcal{D}^{sel}_{i+1}$ and a task k in iteration i. $\mathcal{D}^{sel}_{i+1}$ is chosen by the BO algorithm, which optimizes $J(\boldsymbol{d}; k, \pi_u)$ to find the best design d^*_k.

history of $\boldsymbol{o}'_t$ is used, resulting in the full observation $\boldsymbol{o}_t$. See Fig. 6 for an overview of the observations.

This approach of the temporal context follows the method of [7], [35]. The inclusion of this removes the need to use the linear velocity of the robot as an observation, which is often included in other works [36], [37]. This makes it possible to deploy the policy on hardware that has an affordable inertial measurement unit (IMU).

b) Action Space: The action space represents the agent's direct control over the robot's motion. At each timestep, the policy outputs an action $\boldsymbol{a}_{t+1}$ based on the observation $\boldsymbol{o}_t$. The action $\boldsymbol{a}_{t+1}$ is added to the predefined standing pose of the robot $\boldsymbol{q}^{stand}$, resulting in the desired joint angles $\boldsymbol{q}^{des}_{t+1}$. This approach simplifies the learning process, since a zero action corresponds to the standing pose.

The commands generated by the policy are executed by the robot actuators, which operate using the internal proportional derivative (PD) controller. The PD controller adjusts the torques to track $\boldsymbol{q}^{des}_{t+1}$. See Fig. 6 for an overview of how the actions are processed.

Due to the smaller arm levers of the three-segment legs compared to two-segment robot legs, there is less torque on the knee motors. To facilitate learning and sim-to-real transfer, we use different PD gains and action scales for the knee, thigh, and hip motors.

c) Rewards: The reward function r_t used during training combines task-specific and regularization rewards. The task-based reward is focused on tracking the velocity command. The regularization rewards encourage smooth motion, base stability, and reduced energy consumption. The reward structure is designed to avoid biasing the policy toward specific designs. For example, by excluding reward terms such as body height, which could bias a specific design. Note that the reward structure also incorporates the cost function $J(\boldsymbol{d}; k, \pi_u)$ from Step 2, which ensures alignment between the training and design selection objectives.

To address a requirement of one optimization task which required standing still (Sec. IV-A), the reward structure changes for the zero-velocity command. Therefore, during training, a zero command is sampled 10% of the time to learn to stand still. In addition, a reward is given for keeping $\boldsymbol{q}^{des}_{t+1}$ close to $\boldsymbol{q}^{stand}$. Care is taken to ensure that this reward is not excessively large, as the initial pose may not be optimal for every leg design.

d) Termination: For the termination condition, the policy training includes several conditions to prevent undesirable behaviors and to speed up the learning process. These conditions include: the robot body flipping upside down, reaching joint limits, or surpassing the maximum episode length. The agent receives a negative reward if any of the first two conditions is met.

We have a straightforward implementation in our design choices; as in that we have a single fixed π_u. Unlike [32], which uses a meta-RL approach to adapt policies across different designs, our simpler approach demonstrates sufficiently robust performance across all considered designs, thereby simplifying the training pipeline (Sec. IV-D2). While [24] distinguishes between design-aware and deployment policies, we use a single policy informed by design parameters, ensuring that the deployment policy is also aware of the robot design. In contrast to both papers [24], [32] we also do not use motion primitives for the feet trajectories as part of the reward to give the policy full freedom to explore the best motion for each design.

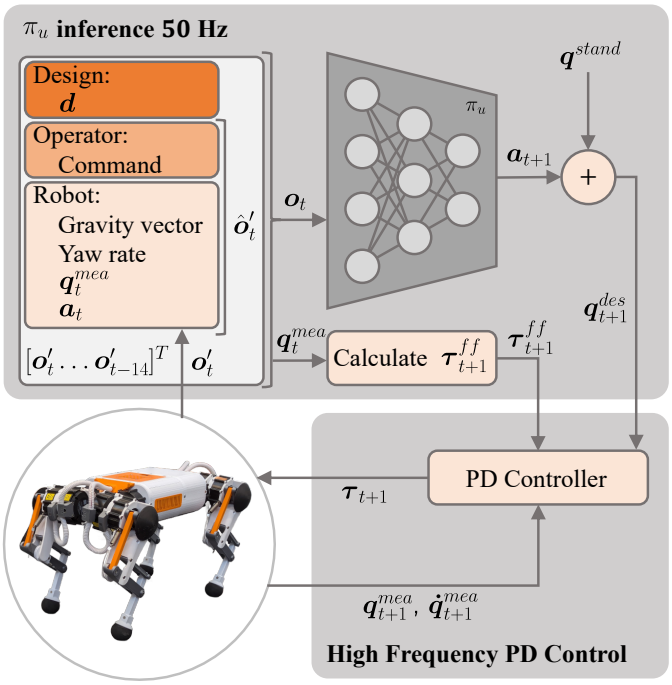


Fig. 6: Overview of the control diagram: Observations $\boldsymbol{o}_t$ includes information about the designs, command given by robot operator, and robot data over a history. The universal policy π_u is a neural network that generates actions $\boldsymbol{a}_{t+1}$ based on $\boldsymbol{o}_t$ at timestep t. The desired joint angles $\boldsymbol{q}_{t+1}^{des}$ are obtained by adding $\boldsymbol{a}_{t+1}$ to the standing joint angles $\boldsymbol{q}^{stand}$ and are tracked by a PD controller. For this tracking, the PD controller uses the position and velocity measurements ($\boldsymbol{q}_{t+1}^{mea}$ and $\dot{\boldsymbol{q}}_{t+1}^{mea}$) from the encoders in the actuators. To aid this tracking $\boldsymbol{o}_t$ provides the actual joint angles $\boldsymbol{q}_t^{mea}$, which are used to calculate the feedforward torque $\boldsymbol{\tau}_{t+1}^{ff}$ (Sec. III-C3). This feedforward torque is combined with the PD control output to generate the total torque $\boldsymbol{\tau}_{t+1}$ on the joints. The frequency of the PD loop is unknown.

2) BO-based Design Selection: After training π_u, we proceed with design selection. In the second step, we use BO to select the best design d_k^* by optimizing a cost function $J(\boldsymbol{d}; k, \pi_u)$ for a specific task k.

This optimization process takes place over multiple iterations i. At each iteration, the BO algorithm receives the cost function scores of a batch of designs $\mathcal{D}_i^{sel}$ from the previous iteration. Based on these scores, it selects a new batch of designs $\mathcal{D}_{i+1}^{sel}$. During each episode, the trained policy π_u performs inference to control the different designs in $\mathcal{D}_{i+1}^{sel}$. At the end of the episode, the scores of $J(\boldsymbol{d}; k, \pi_u)$ for these designs are returned to the BO algorithm. This process is repeated for a set number of iterations to find the best design d_k^*.

The task k defines aspects of the environment or the requirements of the robot. It can include a specific velocity command that the robot must follow, a particular type of terrain, disturbance levels, or variations in the robot, such as a version with a payload.

The cost function $J(\boldsymbol{d}; k, \pi_u)$ is not differentiable with respect to $\boldsymbol{d}$. Therefore, we use a derivative-free optimization method. We chose a Bayesian method over an evolutionary algorithm because of its sample efficiency.

Then we optimize this design space with a specific metric. The co-design optimization framework allows for different cost functions. In this paper, we focus on energy efficiency. As energy efficiency metric, we choose the cost of transport CoT. The aim is to minimize the cost function $J(\boldsymbol{d}; k, \pi_u)$, in our case the CoT, for a task k. The CoT is calculated by

$$CoT = \frac{E_{total}}{\Delta s} = \frac{\int_0^T P_{total}(t)dt}{\Delta s}, \tag{2}$$

where the total energy usage of the robot E_{total} is calculated from the integration of the total power of the system $P_{total}(t)$ over time. Δs is the total distance traveled and T the total time of the test. CoT was chosen because its normalization by Δs enables fair comparisons across different velocity commands and designs with varying velocity tracking performance.

In contrast to [24], [32], we use a more realistic power model. This choice is motivated by the goal to optimize robots for the real-world, where designs must be selected in simulation based on realistic metrics. The total power consumption, $P_{total}(t)$, can be estimated by the power used by the twelve motors and the constant power consumption of other electronics, such as the NVIDIA Jetson computer. This is necessary because, on the hardware platform, we can only measure the total power consumption of the system. This helps to compare performance between the real-world and simulation. The motor power is denoted as $P_{mech}(t)$ and other electronics as P_{system}. This results in

$$P_{total}(t) = P_{mech}(t) + P_{system}. \tag{3}$$

Measurements have shown that P_{system} remains relatively constant during operation and is therefore time independent. For $P_{mech}(t)$, we take inspiration from the power model used in [38]. $P_{mech}(t)$ can be estimated with

$$\begin{aligned} P_{mech}(t) &= \sum_{i=1}^{n} P_{mech,i}(\tau_i(t), \dot{q}_i(t)) \\ &= \sum_{i=1}^{n} \left[\left(\frac{\tau_i(t)}{K_T} \right)^2 R + \frac{K_v \tau_i(t) \dot{q}_i(t)}{K_T} \right], \end{aligned} \tag{4}$$

where $\tau_i(t)$ is the torque applied by the motor i, $\dot{q}_i(t)$ is the rotational speed, R is the motor's winding resistance, K_T is the torque constant, and K_v is the velocity constant. This equation accounts for both winding power dissipation, proportional to $\tau_i(t)^2 R_i$, and rotational power, $\tau_i(t)\dot{q}_i(t)$. More information about the derivation of the energy model can be found in App. A.

An important consideration is that the CoT reflects the total energy consumption of the system divided by distance walked, including the energy consumption of the rest of the electronics, such as the compute. This means that even if the motors consumed no power, the maximum system improvement achievable would not be 100% compared to the nominal design. Therefore, we also introduce the mechanical

CoT, namely CoT_{mech} for the results, which only accounts for the power consumption of the motors.

C. Sim-to-real Transfer of Optimized Control and Design

To ensure successful sim-to-real transfer of both design and controller, we use domain randomization, along with disturbances and noise, during both the training of π_u and the design selection step. In addition, we implement the PEA in a way that the policy does not have to learn the joint offsets created by the springs.

1) Domain Randomization: Domain randomization is applied at the beginning of training, adjusting parameters such as body weight and inertia, PD control constants of the motor, and friction of the ground. This variability reduces the risk of overfitting to the simulator and better prepares the model for real-world conditions, where the exact values of the physical properties differ from those used in simulation. The ranges for these randomized parameters are selected based on testing on the real robot, aiming to encompass the variability observed in real-world environments. Note that this domain randomization is not applied to the design selection step.

2) Disturbances and Noise: During π_u training, random velocity perturbations are introduced to simulate unexpected disturbances that the robot may encounter in real environments. In addition, random noise is added to all of the observation values except for the design parameters. There is also noise added to the offset of the spring that changes every 100 environment steps. This process improves the robustness of the control policy when deployed in real-world scenarios.

For the design selection step, the same disturbances and noise are applied as during training. To ensure a fair comparison, we share the random seed across all tested designs, ensuring that each design experiences identical noise and disturbances during evaluation. We found that introducing these disturbances and noise, in combination with making the evaluation episodes long enough, is crucial for successful optimized design transfer. Without them, the optimized designs tend to overfit the simulation environment, leading to suboptimal performance when applied in the real-world.

3) Implementation of PEA in the Policy: To ensure proper transfer of the PEAs, we compute a torque feedforward $\boldsymbol{\tau}^{ff}_{t+1}$ command to compensate for spring forces. This allows the policy to focus on optimizing the gait and using the spring effectively rather than compensating for spring-related deviations on the joint. The feedforward torque exerted by the springs on the knee joints is calculated using Eq. 1.

D. Implementation

1) Robot Hardware Platform Overview: To fully support our custom-designed bio-inspired legs, we need a versatile robotic platform that can accommodate these legs. Existing platforms, such as the ODRI Solo 12 [39] or the Unitree GO 1 [14], presented integration challenges due to physical and software constraints for new leg designs. As a result, we developed a custom platform from the ground up, named "Dogger", to seamlessly integrate our bio-inspired legs and support our co-design approach.

A key feature of "Dogger" is its use of off-the-shelf electronics, alongside custom 3D printed mechanical parts made from standard materials like PLA, PETG, and TPU that can be printed on conventional FDM 3D printers. This design choice ensures that fabrication and replication are accessible to other researchers, while keeping costs low and supporting adaptability for future modifications. The total material and component cost is approximately 3000 euros, making "Dogger" an affordable platform for quadrupedal robotics research.

Our quadrupedal robot with the nominal legs is sized between the Unitree GO 1 [14] and the ODRI Solo 12 [39]. It weighs 6.5 kg, with a front-to-back distance of 37 cm and a width of 27 cm between the left and right legs. The actuators used are Xiaomi Cybergears, operating at 24 V and producing a peak output torque of 12 Nm per joint. A generic 2 Ah Li-ion drill battery supplies power, distributed via the MJbots power distribution board [40].

An NVIDIA Jetson Orin Nano [41] is used as the onboard computer. Communication with the actuators is facilitated through a PEAK M.2 CAN bus card [42] connected via the PCIe port, with each leg having its own CAN bus. Over these buses, joint position commands are sent, and joint position estimates are retrieved. An additional CAN-FD USB dongle (CANable 2.0) [43] enables communication with the power distribution board to monitor the robot's energy consumption. The robot's orientation is measured using an onboard IMU (MPU-6050) [44] configured in DMP mode, communicating over the I^2C bus with the Jetson. By using the onboard DMP, sensor fusion computations are offloaded to the IMU, reducing the computational load on the Jetson. For more implementation details of the physical robot, see App. B.

2) Training and Optimization Details: We use the Google Brax [45] with the MuJoCo MJX [46] physics engine and use a part of the quadruped environment from [7]. MuJoCo MJX supports closed kinematic chains for the bio-inspired legs and large-scale GPU parallelization to speed up training and design evaluation.

For policy architecture, an actor critic setup is used in which both the actor and the critic use a multilayer perceptron (MLP) with five hidden layers, each containing 128 neurons, and using Swish/Silu activation functions [47]. An input normalization layer processes the observations, and a tanh distribution is used in the last layer to produce the actions. The policy runs at a frequency of 50 Hz.

During training, 8192 agents, each with a different design, are trained simultaneously. We use proximal policy optimization (PPO) as the RL algorithm to update the policy weights, using the implementation provided in the Google Brax environment. Training π_u requires 150 million environment steps on flat terrain and this takes approximately 15 minutes on a desktop GPU (RTX 4070). The policy runs at 50 Hz.

For the design selection process, we use the tree-structured parzen estimator (TPE) solver [48] because it is readily available in the Optuna library [49]. The process begins with a warm-up phase, where 100 random designs are validated in parallel. Each design is tested for 6000 environment steps for task 1 (Sec. IV-A) and 15000 environment steps for task 2 (Sec. IV-B). Following this, we perform 45 iterations with a

batch size of 20 designs per iteration, requiring approximately 45 minutes of computation on the same GPU. More training and optimization details can be found in App. C

3) Cost Function Constants: For CoT (Eq. 2), the following constants are used. Measurements have estimated $P_{system} = 16.2\,\text{W}$. The constants used in $P_{mech}(t)$ for the Cybergear actuators are provided in the datasheet. These are $R = 0.45\,\Omega$, $K_T = 0.87\,\text{Nm/A}$, and $K_v = 0.53\,\text{V/rad/s}$.

IV. Experimental Validation

To evaluate the proposed co-design approach and assess its performance improvements, experiments were conducted both in simulation and in the real-world on the modular hardware platform. We performed two design optimization tasks, task 1 for efficient inspections, and task 2 for efficient payload transport. For both experiments, the performance of each optimized design is tested in simulation and in the real-world. Cross-validation is performed to see how well each optimized design performs in the other task in simulation and in the real-world. In the last experiments performed only in simulation, we look at how well π_u performs compared to specialized policies and look at the consistency of the co-design optimization process.

The optimized designs are denoted as d_1^* for the optimized design for task 1, and d_2^* as the optimized design for task 2. They will be compared to two baselines; d_1^n, which is the design with the link lengths set to the middle value and no springs ($\lambda_1 = 1.00$, $\lambda_2 = 1.00$, $\lambda_3 = 1.00$, $\varnothing = 0$ mm, and $o = 0$ mm), and d_2^n, which is the same leg design only with the springs set to the middle values as well ($\lambda_1 = 1.00$, $\lambda_2 = 1.00$, $\lambda_3 = 1.00$, $\varnothing = 25$ mm, and $o = 5$ mm). An overview of all designs can be found in App. D.

In real-world experiments, velocity commands are transmitted from a remote computer to the robot via a TCP connection. These commands are sent automatically according to a predefined schedule, ensuring consistency across tests. The robot samples all sensor information at 25 Hz, including proprioceptive data, energy consumption, temperatures, and control inputs. These data is sent to a remote computer for post-processing. We measure the real energy consumption on the hardware platform and this represents the energy consumption of the entire system. We emphasize this distinction, as related work often estimates energy consumption using metrics or by monitoring battery charge levels. In simulation, energy consumption is estimated via Eq. 2, which is identical to the metric used during design selection.

A. Task 1: Efficient Inspections

The first task is aimed at inspection routes, a task commonly performed by quadrupedal robots. When the robot performs an inspection route, it follows a fixed path, where it will sometimes stop to take measurements.

For this task, we simplified this route so that the robot moved in a straight line on flat terrain, alternating between walking at 0.5 m/s and standing still. Each test lasted 1 minute, consisting of 30 s of walking and 30 s of being stationary. Eight tests per design were performed in total. The environment was a flat, smooth indoor surface. The walking distance was measured using the floor plans of the hallway and a measuring tape. This environment is shown in Fig. 7.

After optimization, the best resulting design was $\lambda_1 = 0.86$, $\lambda_2 = 1.02$, $\lambda_3 = 1.28$, $\varnothing = 33\,\text{mm}$, and $o = 6\,\text{mm}$ which is visualized in Fig. 9a and Fig. 9b. In simulation, this design, d_1^*, achieved a 12.0% reduction in CoT compared to d_1^n. In real-world tests, the same design achieved a reduction of 18.6% $\pm 1.7\%$ in CoT compared to d_1^n. These results can be found Fig. 9c.

Looking at the adjusted metric CoT_{mech}, which only considers motor power usage, d_1^* achieved a 25% reduction compared to d_1^n. More details of this experiment can be found in App. E.

Fig. 7: Chronophotography of d_2^* performing task 1 (efficient inspections): The frames are spaced to minimize overlap between robots.

B. Task 2: Efficient Payload Transport

The second task focuses on another use case for quadrupeds, namely carrying a payload, for example, for disaster response. In this setting, we simplify the scenario again. The robot is required to execute omni-directional locomotion while carrying a 2 kg payload (approximately 30% of its total mass). For the optimization, 16 random velocity commands were sampled. In the experiment, the robot received 16 different selected commands of varying durations, totaling 2 minutes, to represent a wide range of walking directions and velocities. Four tests for each design were performed in total. The experiments were carried out in a motion capture room to ensure precise distance measurements. This test was also performed on a flat, smooth indoor surface. Fig. 8 shows the environment and d_1^n, d_1^*, and d_2^* motions during the experiment.

The optimized design for this task was $\lambda_1 = 1.30$, $\lambda_2 = 0.70$, $\lambda_3 = 1.30$, $\varnothing = 38\,\text{mm}$, and $o = 2\,\text{mm}$ and can be seen in Fig. 9a and Fig. 9b. In simulation, this design, d_2^*, demonstrated a 33.8% reduction in CoT relative to d_1^n. On the hardware platform, the same configuration reduced CoT by 35.7% $\pm 3.1\%$ relative to d_1^n. These results can be found Fig. 9d.

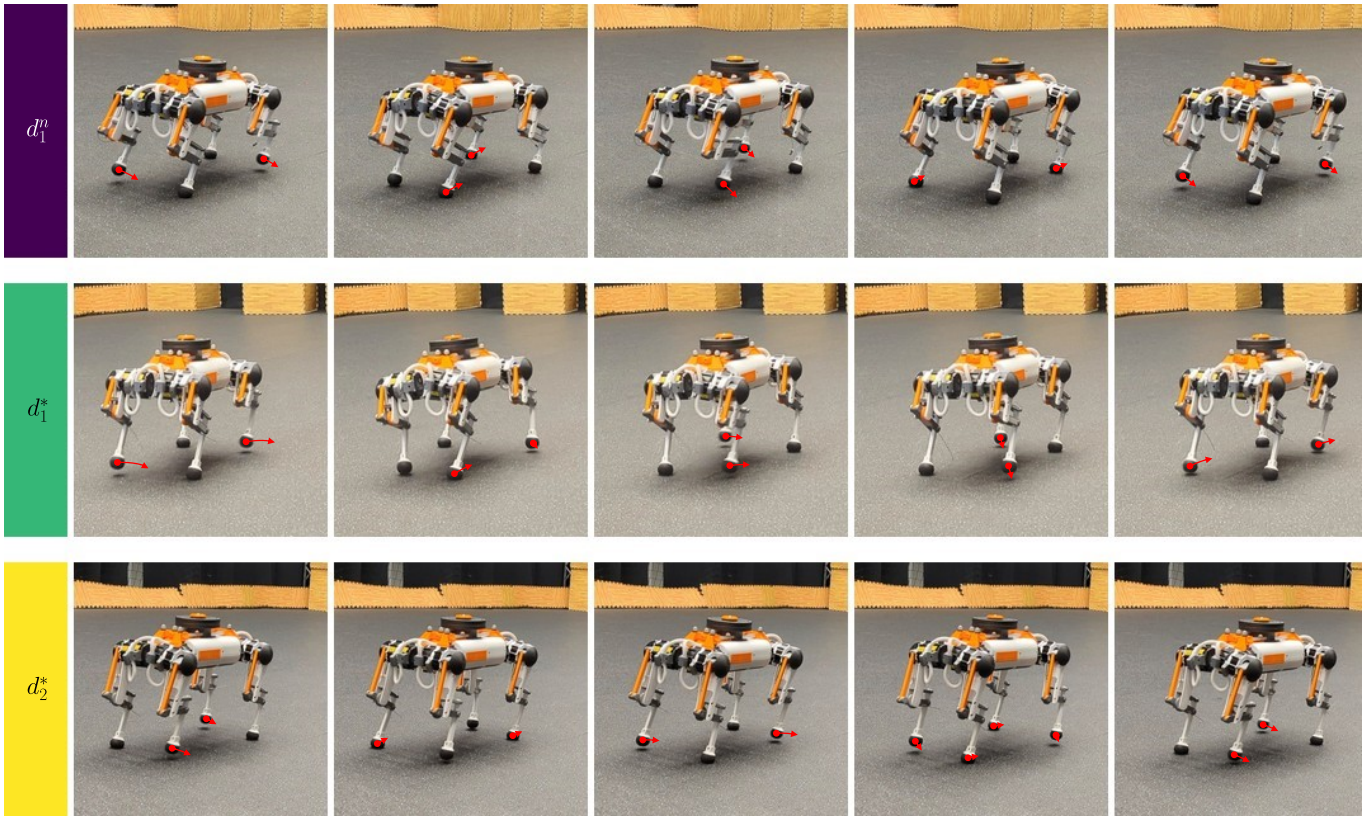


Fig. 8: Dogger hardware platform with bio-inspired legs performing task 2 (efficient payload transport): From top to bottom, the rows show d_1^n (top), d_1^* (middle), and d_2^* (bottom). Each sequence begins at the same time point, with the robot walking sideways to the left at 0.25 m/s. Frames are spaced 0.12 seconds apart. Red arrows indicate the movement of individual feet. The arrows point to the feet location at the next frame and the absence of an arrow means minimal movement.

Taking into account the adjusted metric CoT_{mech}, the improvement reached 49% compared to d_1^n, indicating substantial gains in energy efficiency for payload-carrying tasks.

Even though there is a sim-to-real gap in CoT, these results, together with the results of task 1, demonstrate that the proposed co-design framework can effectively optimize the robot design for both tasks. More details of this experiment can be found in App. F.

C. Cross-validation of Optimized Designs

To further validate the robustness of the co-design approach, we cross-validate optimized designs in both tasks. d_2^* is tested in task 1 and d_1^* is tested in task 2.

d_2^* achieved a 7.0% improvement in CoT over d_1^n in simulation and a 13.5% $\pm 3.3\%$ improvement over d_1^n in the real-world for task 1. These results can be found Fig. 9c. Furthermore, d_1^* improved CoT by 14.6% over d_1^n in simulation and 23.7% $\pm 2.3\%$ over d_1^n in real-world tests for task 2. These results can be found Fig. 9d.

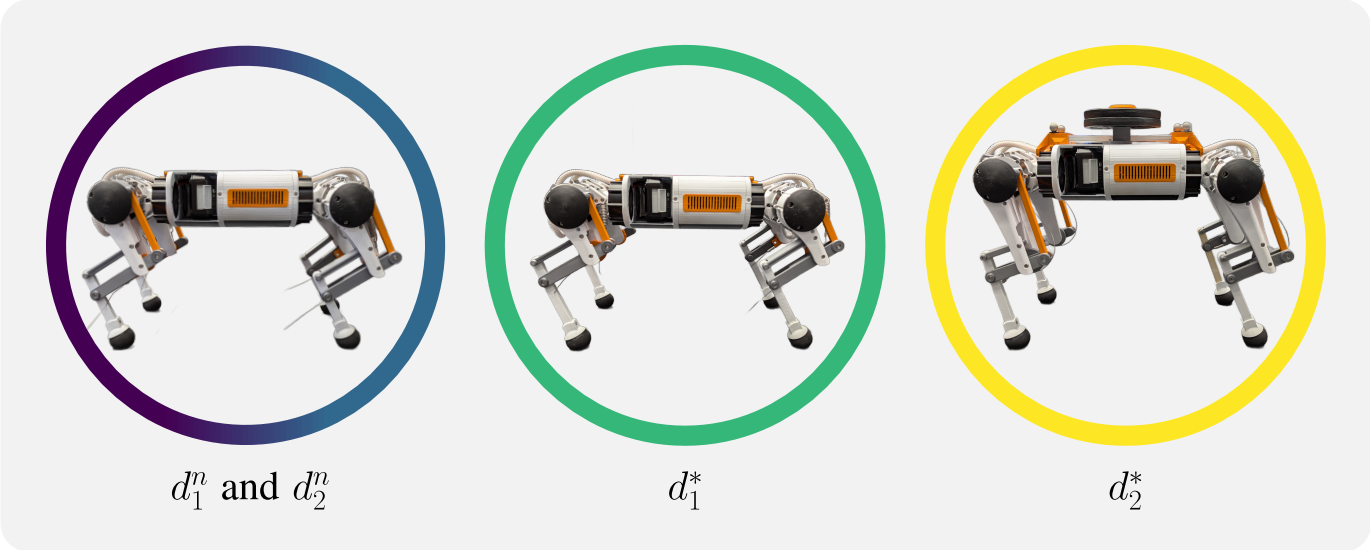
These cross-validation results confirm that while the optimized designs perform well when tested in other tasks, they do not achieve the same level of performance as the designs specifically optimized for their respective task. This outcome underscores the effectiveness of the co-design approach: it does not merely select designs that work slightly better across tasks but identifies configurations that perform best for their intended purpose.

Of course, further testing across a broader range of tasks and environments is required to comprehensively validate this conclusion. Nevertheless, the current results provide evidence that the proposed co-design framework is capable of identifying near-optimal solutions tailored to specific tasks.

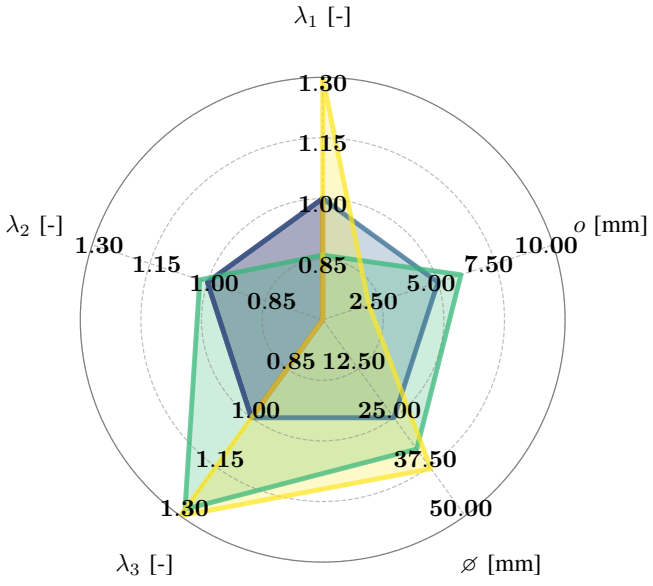
D. Validation of the Co-design Framework

1) Consistency Across Multiple Runs: To assess the consistency of the co-design optimization process, the design optimization step was repeated five times with different random seeds that influenced the initial parameters and sampling for BO, and noise and disturbances for both tasks. For task 2, the seeds also influence the commands that are sampled.

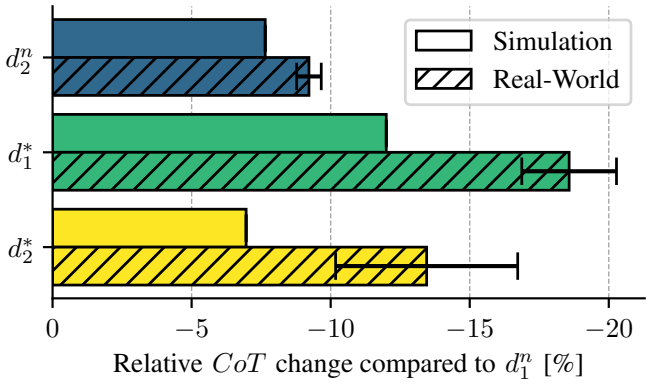
For task 1, all optimized designs achieved CoT between -0.03% and 1.97% compared to d_1^*, which can be seen in Fig. 10c. For task 2, all optimized designs achieved CoT between 1.13% and 5.73% compared to d_2^*, which can be seen in Fig. 10d. In both cases, their specific parameter configurations varied, which can be seen in Fig. 10a and Fig. 10b. This shows that multiple near-optimal solutions exist, reflecting the coupled nature of the design parameters and the underlying trade-offs in the design space. It also shows that our co-design approach consistently picks well-performing



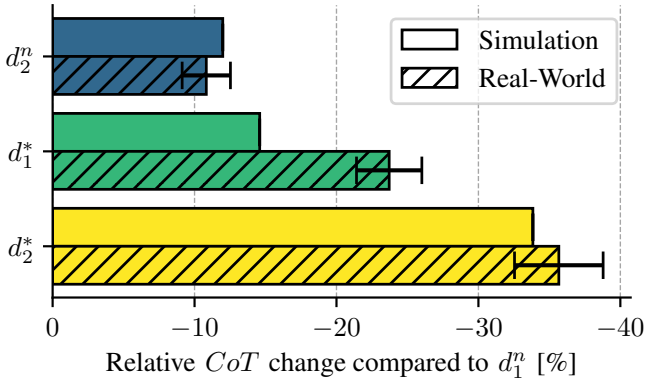
(a) The four evaluated designs: d_1^n, d_2^n, d_1^*, and d_2^*. d_1^n and d_2^n share the same link parameters ($\lambda_1, \lambda_2, \lambda_3$) and therefore share the picture (left). d_1^n has the diameter $\varnothing$ of the wheel and the offset o set to zero, while d_2^n uses mid-range values for these parameters and therefore differ. d_1^* (center) and d_2^* (right) show the optimized designs.



(b) Design parameters visualized for the evaluated designs. d_1^n is less visible due to overlapping design values.

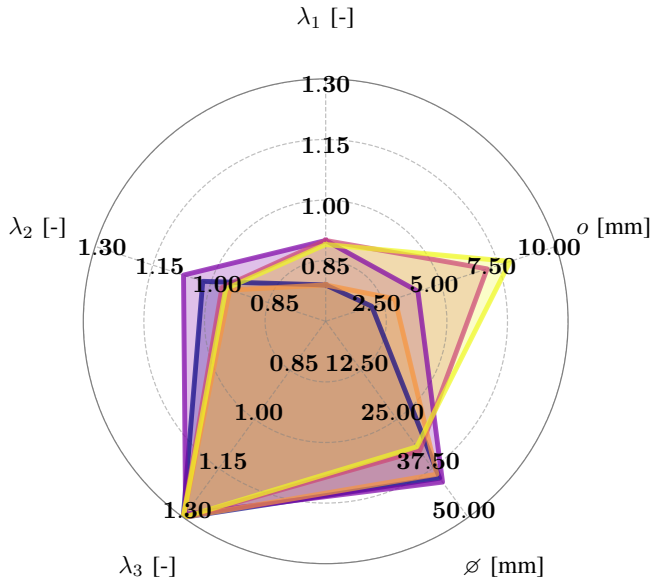


(c) Relative performance for task 1.

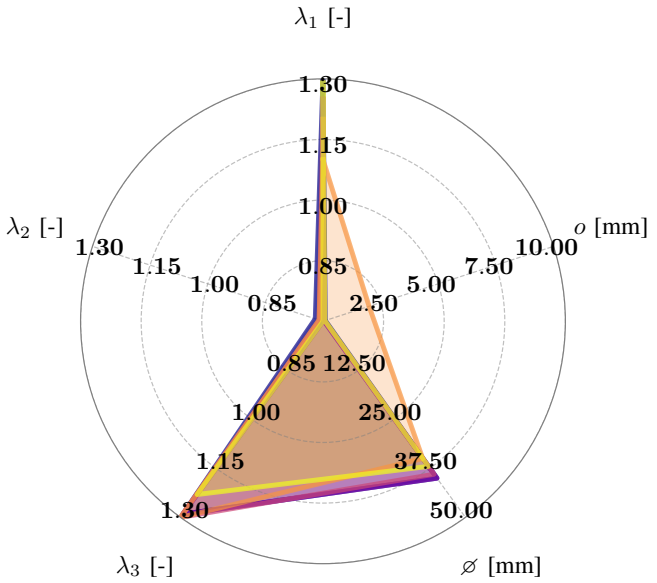


(d) Relative performance for task 2.

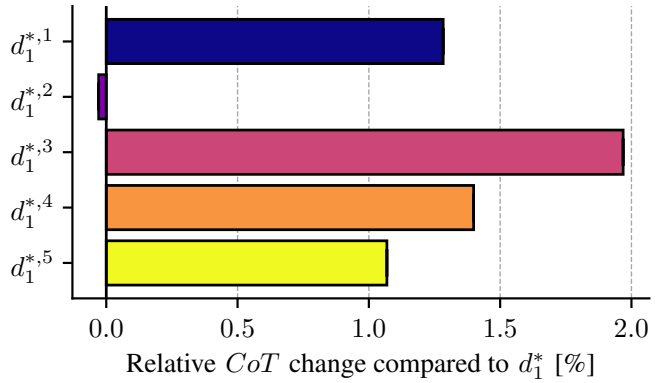
Fig. 9: Comparison of nominal and optimized designs across tasks. At the top, pictures of the designs (d_1^n, d_2^n, d_1^*, d_2^*) illustrate their structural differences. Below-left, design parameters are shown in a visual comparison to better see the differences for the PEAs (parallel elastic actuators). Below-right, relative CoT (cost of transport) gains over d_1^n are presented for tasks 1 and 2, evaluated in both simulation and real-world tests. Note that the CoT includes energy usage from both motors and electronics. Design colors: purple (d_1^n), blue (d_2^n), green (d_1^*), and yellow (d_2^*).



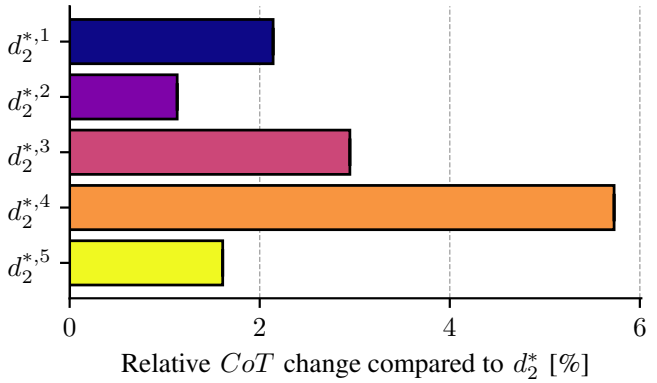
(a) Design parameters for task 1 across different optimization runs.



(b) Design parameters for task 2 across different optimization runs.



(c) Relative performance change for task 1 across optimizations.



(d) Relative performance change for task 2 across optimizations.

Fig. 10: Consistency analysis of the co-design approach for tasks 1 and 2 across different random seeds. The left column corresponds to task 1, and the right column corresponds to task 2. The top row visualizes design parameters across optimization runs for each task, while the bottom row shows the relative CoT change compared to the optimized design for each task.

designs. The raw values of this experiment can be found in App. G.

2) Universal Versus Specialized Policies: To determine whether π_u adapts well to various designs, separate specialized policies π_s were trained for d_1^n, d_2^n, d_1^*, d_2^*. π_s use the same hyperparameters and rewards as π_u.

As illustrated in Fig. 11, the reward score for π_u ranged from -0.2% to 13.8% compared to π_s. This shows that π_u adapts sufficiently to the nominal and optimized designs. Consequently, this shows that we do not need separate policies or fine-tuning for each configuration. This also shows that in the design selection step, each robot design is evaluated in a fair way because the policy is well adapted to each design. The raw values of this experiment can be found in App. H.

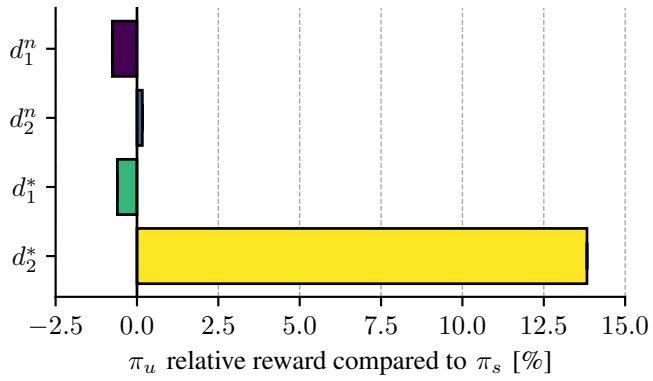


Fig. 11: Comparison of π_u and π_s reward for the four evaluated designs: d_1^n, d_2^n, d_1^*, and d_2^*.

V. Conclusion

In this paper, we aimed to bridge the gap between theoretical models and practical applications by transitioning optimized designs from simulation to real-world implementation. A key contribution to achieve this was the development of a modular, low-cost hardware platform with bio-inspired legs that enabled rapid prototyping and testing of optimized designs. It completed all experiments, tests, and demonstrations without electronic failures and only one broken leg, which could easily be reprinted and replaced.

We demonstrated that our simplified co-design approach, using reinforcement learning (RL) combined with Bayesian optimization, significantly enhanced the energy efficiency of the bio-inspired quadruped robot. By concurrently optimizing both the mechanical design and control strategies, we navigated a complex design space where parameters like shank length and spring stiffness presented conflicting trade-offs. Our co-design approach found well-performing designs, leading to a notable 18.6% and 35.7% reduction in CoT over the nominal design without springs in real-world tests during inspection and payload tasks respectively. Cross-validation showed that our approach does not merely select designs that work slightly better across tasks but identifies designs that perform best for their intended purpose.

In addition to the real-world experiments, we conducted further software validation to support our approach. First, the universal policy, trained across various designs, performed comparably to specialized policies tailored to specific designs, demonstrating the effectiveness of the universal policy within our co-design approach. Second, the design selection remained consistent between different random seeds, indicating the reliability of the proposed approach despite the complexity of the design space.

However, challenges remain. First, the design selection is based on simulation data, and due to the sim-to-real gap, the question arises about whether this selected design is also the best in real-world conditions. Second, the biases introduced during RL training, such as those resulting from initial joint configurations, can influence design preferences. Although some of the above discrepancies between simulation and real-world performance remain, our findings underscore the potential of co-design to address complex trade-offs in real-world robotic system design.

Future work should aim to ensure that the selected design is also the best design in the real-world. Additionally, new metrics and tasks, such as optimizing jumping distance, running speed, or walking endurance, should be explored. Expanding the design space to include more factors, such as body shape, gear ratios, and the spring setup of series elastic actuators, could also lead to further advancements. By continuing to refine the co-design process, we aim to both simplify the design process of robots and make robots that approach the agility, efficiency, and versatility of their biological counterparts.

Acknowledgment

I would like to thank my supervisors, Dr. Jiatao Ding and Dr. Cosimo Della Santina, for their guidance and support throughout my master's program, for the literature review, the research assignment, and the thesis. Their feedback and expertise were essential in shaping this work.

I am also grateful to Dr. Martijn Wisse for encouraging me to explore the field of legged robotics for my thesis and for his assistance in securing funding for the project.

Many thanks to my fellow robotics students and researchers for their helpful discussions and support during experiments, as well as to the lab technicians for their assistance with the motion tracking system.

Finally, I would like to thank my father, brother, girlfriend, and friends for their encouragement and support throughout my studies.

References

[1] V. Atanassov, J. Ding, J. Kober, I. Havoutis, and C. D. Santina, "Curriculum-based reinforcement learning for quadrupedal jumping: A reference-free design," 2024. [Online]. Available: https://arxiv.org/abs/2401.16337

[2] M. Chignoli, S. Morozov, and S. Kim, "Rapid and reliable quadruped motion planning with omnidirectional jumping," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6621–6627.

[3] F. Stella, M. Achkar, C. Della Santina *et al.*, "Paws: A synergy-based robotic quadruped leveraging passivity for robustness and behavioural diversity," 9 2023. [Online]. Available: https://doi.org/10.21203/rs.3.rs-3195331/v1

[4] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[5] D. Hoeller, N. Rudin, D. Sako, and M. Hutter, "Anymal parkour: Learning agile navigation for quadrupedal robots," *Science Robotics*, vol. 9, no. 88, p. eadi7566, 2024.

[6] Z. Zhuang, Z. Fu, J. Wang, C. G. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," in *7th Annual Conference on Robot Learning*, 2023.

[7] K. Caluwaerts, A. Iscen, J. C. Kew, W. Yu, T. Zhang, D. Freeman, K.-H. Lee, L. Lee, S. Saliceti, V. Zhuang, N. Batchelor, S. Bohez, F. Casarini, J. E. Chen, O. Cortes, E. Coumans, A. Dostmohamed, G. Dulac-Arnold, A. Escontrela, E. Frey, R. Hafner, D. Jain, B. Jyenis, Y. Kuang, E. Lee, L. Luu, O. Nachum, K. Oslund, J. Powell, D. Reyes, F. Romano, F. Sadeghi, R. Sloat, B. Tabanpour, D. Zheng, M. Neunert, R. Hadsell, N. Heess, F. Nori, J. Seto, C. Parada, V. Sindhwani, V. Vanhoucke, and J. Tan, "Barkour: Benchmarking animal-level agility with quadruped robots," 2023. [Online]. Available: https://arxiv.org/abs/2305.14654

[8] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 11 443–11 450.

[9] S. A. Burden, T. Libby, K. Jayaram, S. Sponberg, and J. M. Donelan, "Why animals can outrun robots," *Science Robotics*, vol. 9, no. 89, p. eadi9754, 2024.

[10] A. Badri-Spröwitz, A. Aghamaleki Sarvestani, M. Sitti, and M. A. Daley, "Birdbot achieves energy-efficient gait with minimal control using avian-inspired leg clutching," *Science Robotics*, vol. 7, no. 64, p. eabg4055, 2022.

[11] Q. Shi, J. Gao, S. Wang, X. Quan, G. Jia, Q. Huang, and T. Fukuda, "Development of a small-sized quadruped robotic rat capable of multimodal motions," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3027–3043, 2022.

[12] J. K. Lungmus and K. D. Angielczyk, "Antiquity of forelimb eco-morphological diversity in the mammalian stem lineage (synapsida)," *Proceedings of the National Academy of Sciences*, vol. 116, no. 14, pp. 6903–6907, 2019.

[13] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2016, pp. 38–44.

[14] Unitree. Go1. Accessed: 2024-09-13. [Online]. Available: https://www.unitree.com/en/go1/

[15] Boston Dynamics. Spot. Accessed: 2024-09-13. [Online]. Available: https://bostondynamics.com/products/spot/

[16] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[17] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6295–6301.

[18] P. M. Wensing, A. Wang, S. Seok, D. Otten, J. Lang, and S. Kim, "Proprioceptive actuator design in the mit cheetah: Impact mitigation and high-bandwidth physical interaction for dynamic legged robots," *Ieee transactions on robotics*, vol. 33, no. 3, pp. 509–522, 2017.

[19] A. Spröwitz, A. Tuleu, M. Vespignani, M. Ajallooeian, E. Badri, and A. J. Ijspeert, "Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 932–950, 2013.

[20] F. De Vincenti, D. Kang, and S. Coros, "Control-aware design optimization for bio-inspired quadruped robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1354–1361.

[21] A. M. Abate, "Mechanical design for robot locomotion," 2018. [Online]. Available: https://ir.library.oregonstate.edu/concern/graduate_thesis_or_dissertations/nk322k39g

[22] G. Kenneally and D. E. Koditschek, "Leg design for energy management in an electromechanical robot," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 5712–5718.

[23] J. Ding, P. Posthoorn, V. Atanassov, F. Boekel, J. Kober, and C. D. Santina, "Quadrupedal locomotion with parallel compliance: E-go design, modeling, and control," *IEEE/ASME Transactions on Mechatronics*, vol. 29, no. 4, pp. 2839–2848, 2024.

[24] F. Bjelonic, J. Lee, P. Arm, D. Sako, D. Tateo, J. Peters, and M. Hutter, "Learning-based design and control for quadrupedal robots with parallel-elastic actuators," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1611–1618, 2023.

[25] G. Grandesso *et al.*, "Reinforcement learning and trajectory optimization for the concurrent design of high-performance robotic systems," 2023. [Online]. Available: https://iris.unitn.it/handle/11572/381949

[26] K. M. Digumarti, C. Gehring, S. Coros, J. Hwangbo, and R. Siegwart, "Concurrent optimization of mechanical design and locomotion control of a legged robot," in *Mobile Service Robotics*. World Scientific, 2014, pp. 315–323.

[27] S. Ha, S. Coros, A. Alspach, J. Kim, and K. Yamane, "Joint optimization of robot design and motion parameters using the implicit function theorem." in *Robotics: Science and systems*, vol. 8, 2017.

[28] T. Dinev, C. Mastalli, V. Ivan, S. Tonneau, and S. Vijayakumar, "A versatile co-design approach for dynamic legged robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 343–10 349.

[29] C. Schaff, D. Yunis, A. Chakrabarti, and M. R. Walter, "Jointly learning to construct and control agents using deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9798–9805.

[30] D. Ha, "Reinforcement learning for improving agent design," *Artificial Life*, vol. 25, no. 4, pp. 352–365, Nov. 2019.

[31] G. Fadini, S. Kumar, R. Kumar, T. Flayols, A. Del Prete, J. Carpentier, and P. Souères, "Co-designing versatile quadruped robots for dynamic and energy-efficient motions," *Robotica*, pp. 1–22, 2023.

[32] Á. Belmonte-Baeza, J. Lee, G. Valsecchi, and M. Hutter, "Meta reinforcement learning for optimal design of legged robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 134–12 141, 2022.

[33] J. Won and J. Lee, "Learning body shape variation in physics-based characters," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–12, 2019.

[34] X. Liu, A. Rossi, and I. Poulakakis, "A switchable parallel elastic actuator and its application to leg design for running robots," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 6, pp. 2681–2692, 2018.

[35] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *The International Journal of Robotics Research*, vol. 43, no. 4, pp. 572–587, 2024.

[36] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[37] M. Aractingi, P.-A. Léziart, T. Flayols, J. Perez, T. Silander, and P. Souères, "Controlling the solo12 quadruped robot with deep reinforcement learning," *scientific Reports*, vol. 13, no. 1, p. 11945, 2023.

[38] G. Bellegarda, C. Nguyen, and Q. Nguyen, "Robust quadruped jumping via deep reinforcement learning," *Robotics and Autonomous Systems*, vol. 182, p. 104799, 2024.

[39] P.-A. Léziart, T. Flayols, F. Grimminger, N. Mansard, and P. Souères, "Implementation of a reactive walking controller for the new open-hardware quadruped solo-12," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5007–5013.

[40] MJbots. Mjbots power distribution. Accessed: 2024-09-12. [Online]. Available: https://mjbots.com/products/mjbots-power-dist-r4-5b

[41] NVIDIA. Jetson orin. Accessed: 2024-09-12. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/

[42] PEAK. Pcan-m-2. Accessed: 2024-09-12. [Online]. Available: https://www.peak-system.com/PCAN-M-2.473.0.html?&L=1

[43] CANable. An open-source usb to can adapter. Accessed: 2024-09-12. [Online]. Available: https://canable.io/

[44] Invensense. Mpu-6050. Accessed: 2024-09-12. [Online]. Available: https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/

[45] C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem. (2021) Brax - a differentiable physics engine for large scale rigid body simulation. [Online]. Available: http://github.com/google/brax

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

[47] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017. [Online]. Available: https://arxiv.org/abs/1710.05941

[48] Y. Ozaki, Y. Tanigaki, S. Watanabe, M. Nomura, and M. Onishi, "Multiobjective tree-structured parzen estimator," *Journal of Artificial Intelligence Research*, vol. 73, pp. 1209–1250, 2022.

[49] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[50] NVIDIA. Nvidia tensorrt. Accessed: 2024-12-25. [Online]. Available: https://developer.nvidia.com/tensorrt

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[52] J. Bai, F. Lu, K. Zhang *et al.* (2019) Onnx: Open neural network exchange. https://github.com/onnx/onnx.

APPENDIX A
ENERGY MODEL DERIVATION AND DETAILS

This section of the Appendix gives more information about the energy model that is used.

A. Derivation Motor Power

The motor power $P_{mech}(t)$ is calculated using the motor voltage and current:

$$\begin{aligned} P_{mech}(t) &= \sum_{i=1}^{n} P_{mech,i}(\tau_i(t), \dot{q}_i(t)) \\ &= \sum_{i=1}^{n} V_{m,i}(\tau_i(t), \dot{q}_i(t)) I_{m,i}(\tau_i(t)), \end{aligned} \tag{5}$$

where $P_{mech,i}(\tau_i(t), \dot{q}_i(t))$ is the motor power of motor i, $\tau_i(t)$ is the torque of motor i, $\dot{q}_i(t)$ is the joint velocity of motor i, $V_{m,i}(\tau_i, \dot{q}_i)$ is the voltage across motor i, $I_{m,i}(\tau_i)$ is the current through motor i. $I_{m,i}(\tau_i(t))$ is estimated by:

$$I_{m,i}(\tau_i(t)) = \frac{\tau_i(t)}{K_T}, \tag{6}$$

here K_T is the torque constant. $V_{m,i}(\tau_i(t), \dot{q}_i(t))$ is calculated as:

$$V_{m,i}(\tau_i(t), \dot{q}_i(t)) = I_{m,i}(\tau_i(t))R + K_v \dot{q}_i(t), \tag{7}$$

here R is the winding resistance of the motor and K_v is the velocity constant of the motor. The first term represents the voltage drop due to resistive losses in the motor windings, and the second term accounts for the back-emf generated by the motor's rotation.

The total power $P_{total}(t)$ is obtained by adding $P_{mech}(t)$ to P_{system}. This is integrated to obtain the total energy of the system E_{total}. This integration is performed discretely using a Riemann sum at each time step of the environment's operation:

$$E_{total} \approx \sum_{i=0}^{N-1} P_{total}(t) \cdot \Delta t, \tag{8}$$

where Δt is the duration of each time step. This provides an estimate of the system's total energy consumption over the operational period.

B. Estimating Power Usage of System

To estimate the power usage of the other electronic components P_{system}, we measured the energy consumption of the entire system with the software running but no power applied to the motors for a duration of 8 minutes. As shown in the plot, the power consumption remains constant. The total energy usage of the robot system, measured in Wh over time, is illustrated in Fig. 12.

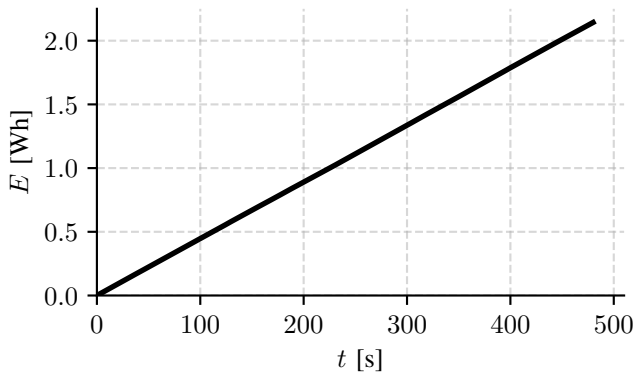


Fig. 12: Measured energy consumption over time for all electronics with the actuators turned off for 8 minutes.

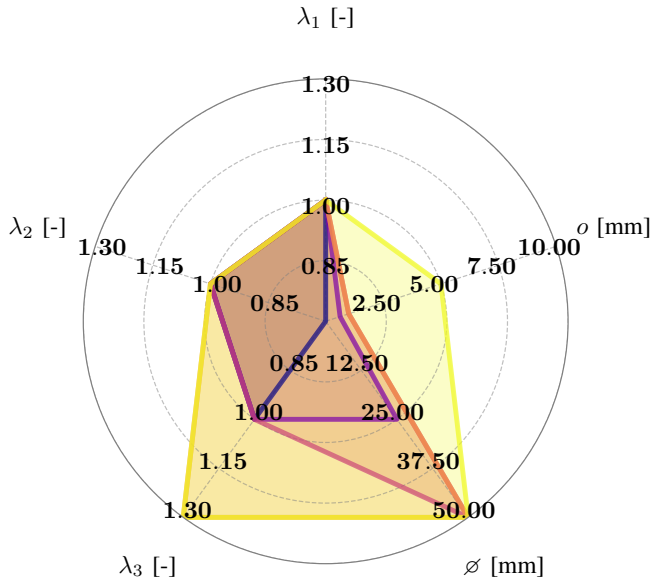
1) Verification of Energy Model: To ensure that our cost function is based on an accurate energy model, we conducted additional energy consumption tests on both the simulator and the real robot, now focused purely on energy E and not on CoT. Note that this test has been done with an older policy than the tests in the main section of this report. In this test, we take five different designs noted as $d_1^n, d_1^a, d_1^b, d_1^c$ and d_1^d with slight changes between them. Then the energy consumption is measured for task 1 (efficient inspections) (Sec. III-D1) both in the simulator and in the real-world. Fig. 13 shows the designs on the radar chart and the comparison between the relative simulation and real-world energy consumption compared to d_1^n. The raw values of the designs can be found in Tab. II. There is still an offset between the energy measurements obtained from the simulation and those from the real-world, with the offset ranging from 4.16% to 6.08% in all designs.

TABLE II: Design parameter values for different designs of the energy experiment.

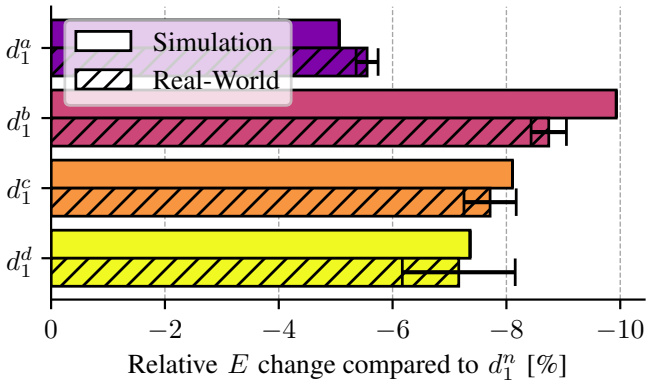
Design	λ_1 [−]	λ_2 [−]	λ_3 [−]	$\varnothing$ [mm]	o [mm]
d_1^n	1.00	1.00	1.00	0	0
d_1^a	1.00	1.00	1.00	25	1
d_1^b	1.00	1.00	1.00	50	1
d_1^c	1.00	1.00	1.30	50	1
d_1^d	1.00	1.00	1.30	50	5

Tab. III shows the raw energy consumption of the five designs for simulation and the real-world. The first column shows the energy consumption measured in the simulation. The second column shows the energy consumption measured in the real-world. The third column shows the relative difference between what is measured in the simulator and what is measured in the real-world.

Tab. IV shows the difference in energy consumption for each design compared to d_1^n. The first column shows these relative differences for the measured simulation energy consumption for each design compared to d_1^n. The second column shows these relative differences for what is measured in the real-world for each design compared to d_1^n. The last column shows the relative difference between these two relative differences for each design compared to d_1^n. This means that if we would



(a) Design parameters visualized for the evaluated designs.



(b) Relative performance for task 1.

Fig. 13: Energy experiment. On the left the different tested designs are visualized. On the right the relative performance differences compared to d_1^n. The test is based on task 1, only a different metric is used.

TABLE III: Raw simulated and real-world energy consumptions for each design and their relative difference.

Design	Sim. Energy [Wh]	Real Energy [Wh]	Diff. [%]
d_1^n	0.989	1.035	+4.70
d_1^a	0.939	0.978	+4.16
d_1^b	0.891	0.945	+6.08
d_1^c	0.909	0.955	+5.14
d_1^d	0.916	0.961	+4.92

TABLE IV: Relative differences between simulation and reality compared to d_1^n and the relative difference between those two.

Design	Sim. Diff. [%]	Real Diff. [%]	Rel. Sim.-Real Diff. [%]
d_1^n	+0.00	+0.00	+0.00
d_1^a	−5.06	−5.55	−0.49
d_1^b	−9.93	−8.74	+1.19
d_1^c	−8.11	−7.71	+0.39
d_1^d	−7.36	−7.16	+0.20

have the same relative improvement in the real world as the simulator, that this sim-to-real difference is 0%. The relative differences between the designs in the simulation compared to the real world range from 0.00% to 1.19%. This last result is important because it shows, despite having an offset between the energy consumption measured between simulation and reality, that the relative differences are small. This is less compared to the sim-to-real differences in CoT seen in Sec. IV-A and Sec. IV-B. This could be due to the smaller design changes or by the exclusion of the walked distance.

APPENDIX B
IMPLEMENTATION DETAILS OF PHYSICAL ROBOT

This section of the Appendix provides additional details regarding the implementation of key components on the physical robot. Although the scope of this project is broad and includes aspects not covered in this work, such as creating MJCF simulation models from CAD or developing tools for the RL training pipelines, we include the following information to highlight implementation details that are often not explained in related work.

A. Data Collection on the Real Robot

Collecting data onboard the robot is a non-trivial task due to the limited computational resources available. To address this, the data collection pipeline is run asynchronously from the main control loop at a frequency of 25 Hz. This setup minimizes computational load on the main processes, which operate at a fixed frequency of 50 Hz.

The data collection pipeline is run during periods when the main loop is idle, which happens when waiting for the time to keep the loop frequency fixed. During these idle periods, all collected data are queued. The queued data are then transmitted to a remote computer in batches every 2.5 seconds, after which the queue is reset. This batching strategy prevents the overhead of frequent data transmissions.

B. Running Neural Networks Efficiently on the NVIDIA Jetson

To achieve efficient inference of the neural network on the robot, we utilize NVIDIA TensorRT, which provides runtime optimizations for neural networks [50]. The neural network, initially developed in the Brax simulation environment using

JAX, underwent several transformations to enable deployment on the NVIDIA Jetson platform.

First, the JAX-based model was converted to a PyTorch model [51]. This was necessary for the export to the ONNX (Open Neural Network Exchange) format [52]. This intermediate format was then used to create a TensorRT (TRT) model, optimized for the Jetson platform. Finally, the TensorRT model was executed using the PyCUDA framework in Python. This pipeline ensured low-latency inference and enabled real-time deployment of the neural network on the physical robot.

APPENDIX C
TRAINING DETAILS

This section of the Appendix, we go in more depth about the training process of π_u and design selection process. In Tab. V the ranges can be found for the domain randomization. In Tab. VI the hyperparameters can be found that are used in the PPO algorithm. In Tab. VII the hyperparamters can be found that are applied to the environment. In Tab. VIII the rewards weights can be found for each reward. In Tab. IX the hyperparameters can be found that are used for the BO algorithm. Finally in Tab. X, the reward functions that are used during training can be found with the symbol explanation presented in Tab. XI.

TABLE V: Domain randomization ranges.

Parameter	Range	Unit	Notes
λ_1	[0.7, 1.3]	[-]	Factor
λ_2	[0.7, 1.3]	[-]	Factor
λ_3	[0.7, 1.3]	[-]	Factor
$\varnothing$	[0, 50]	[mm]	-
o	[0, 10]	[mm]	-
Friction	[1.0, 2.0]	[-]	Factor
Position gain (Kp)	$[-2.5, 2.5]$	$[\mathrm{Nmsrad}^{-1}]$	Addition
Derivative gain (Kd)	$[-0.2, 0.35]$	$[\mathrm{Nmsrad}^{-1}]$	Addition
Joint Damping	[0.0, 0.3]	$[\mathrm{Nmsrad}^{-1}]$	Addition
Torso Mass	[0.75, 1.5]	[-]	Factor
Torso Inertia	[0.5, 2.5]	[-]	Factor
Torso COM Position	[0.035, 0.05]	[m]	Addition

TABLE VI: Training network hyperparameters.

Parameter	Value
Number of Timesteps	$150,000,000$
Episode Length	1000
Normalize Observations	True
Unroll Length	20
Number of Minibatches	32
Number of Updates per Batch	4
Discounting	0.97
Learning Rate	3.0×10^{-4}
Entropy Cost	1×10^{-2}
Number of Environments	8192
Batch Size	256
Policy Hidden Layer Sizes	(128, 128, 128, 128)

TABLE VII: Environment hyperparameters.

Parameter	Value
Velocity Command Bounds	
Linear Velocity X	$[-0.6, 0.6]$
Linear Velocity Y	$[-0.8, 0.6]$
Angular Velocity Yaw	$[-1.0, 1.0]$
Action Space Bounds	
Lower Bounds	$[-1.0, 0.5, -1.75]$
Upper Bounds	[1.0, 2, 0]
Additional Reward Parameters	
Observation Noise	0.065
Action Scale	[0.4, 0.5, 1.2]
Kick Velocity	0.12
Foot Radius	0.02
Push Interval	10
Contact Threshold	0.001
Contact Threshold	0.01
Height Termination Threshold	0.18
Step Height Threshold	0.09
Probability of Zero Command	0.1

TABLE VIII: Reward weights.

Reward Component	Weight
Tracking Reward Weights	
Tracking Linear Velocity	1.5
Tracking Angular Velocity	0.8
Tracking Sigma	0.25
Regularization Reward Weights	
Linear Velocity in Z	-2.0
Angular Velocity in XY	-0.05
Orientation	-35.0
Torques	-0.0015
Action Rate	-0.1
Feet Air Time	10.0
Hip Close to Initial	-0.20
Stand Still Close to Initial	-0.15
Step Height	-1.75
Termination	-1.0
Foot Slip	-0.1

TABLE IX: Bayesian solver hyperparameters.

Parameter	Value
Selected Sampler	tpe
Number of Trials	1000
Batch Size	20
Number of Startup Trials	100
Number of EI Candidates	192
Multivariate	True
Group	True

TABLE X: Reward functions.

Reward Component	Expression
Tracking Reward Functions	
Tracking Linear Velocity	$\exp\left(-\frac{(v_{des,x}-v_x)^2+(v_{des,y}-v_y)^2}{\sigma}\right)$
Tracking Angular Velocity	$\exp\left(-\frac{(\omega_{des}-\omega_z)^2}{\sigma}\right)$
Regularization Reward Functions	
Linear Velocity in Z	v_z^2
Angular Velocity in XY	$\omega_x^2+\omega_y^2$
Orientation	$q_{base,x}^2+q_{base,y}^2$
Torques	$c_{heatloss}\cdot\sum_{i=1}^{12}\tau_i^2+c_{mech\_power}\cdot\sum_{i=1}^{12}(\tau_i\cdot\dot{q}_i)$
Action Rate	$\sum_{i=1}^{12}\left(a_i-a_i^{-1}\right)^2$
Feet Air Time	$\sum_{i=1}^{4}\left((t_i^{air}-0.1)\cdot not\_contact\_filt_i\right)\cdot\mathbb{1}(\boldsymbol{v}^{des}=0)$
Hip Close to Initial	$\sum_{i=1}^{4}\left\lvert q_i^{hip}-q_i^{default\_act\_pose,hip}\right\rvert$
Stand Still Close to Initial	$\sum_{i=1}^{12}\left\lvert q_i-q_i^{default\_act\_pose}\right\rvert\cdot\mathbb{1}(\boldsymbol{v}^{des}=0)$
Step Height	$\sum_{i=1}^{4}(step\_height_i-step\_height\_threshold_i)^2\cdot contact\_filt_i$
Termination	$\mathbb{1}(done\wedge(step<500))$
Foot Slip	$\sum_{i=1}^{4}\left((v_i^{foot,x,y})^2\cdot contact\_filt_i\right)$

TABLE XI: Symbol descriptions for reward functions table.

Symbol	Description
$\boldsymbol{q}$	Joint position vector for all actuators.
$\boldsymbol{q}^{default\_act\_pose}$	Default joint position vector for all actuators.
$\dot{q}_i$	Angular velocity of the i-th actuator.
τ_i	Torque applied to the i-th actuator.
a_i	Action command for the i-th actuator.
a_i^{-1}	Previous action command for the i-th actuator.
$\boldsymbol{v}$	Actual linear velocity vector.
$\boldsymbol{v}_{des}$	Desired linear velocity vector.
$\boldsymbol{\omega}$	Actual angular velocity vector.
ω_{des}	Desired angular velocity around the z axis.
$\boldsymbol{q}_{base}$	Orientation quaternion components representing pitch and roll of the base.
t_i^{air}	Air time of the i-th foot.
$not\_contact\_filt_i$	Boolean indicating if the i-th foot is not in contact with the ground.
$contact\_filt_i$	Boolean indicating whether the i-th foot is in contact with the ground.
$\boldsymbol{v}_i^{foot}$	Linear velocity vector of the i-th foot in the x and y directions.
$step\_height_i$	Step height of the i-th foot.
$step\_height\_threshold_i$	Threshold step height for the i-th foot.
$c_{heatloss}$	Heat loss constant.
c_{mech_power}	Mechanical power constant.
$\mathbb{1}(\cdot)$	Indicator function (1 if the condition inside is true, 0 otherwise).
$done$	Boolean indicating if the episode has terminated.
$step$	Current simulation step.
σ	Variance parameter controlling the sensitivity of the exponential function.

APPENDIX D
DESIGN OVERVIEW

In this section of the Appendix, the values for the design parameters can be seen for the nominal and optimized designs in Tab. XII.

TABLE XII: Design parameter values for nominal and optimized designs.

Design	λ_1 [−]	λ_2 [−]	λ_3 [−]	$\varnothing$ [mm]	o [mm]
d_1^n	1.00	1.00	1.00	0	0
d_2^n	1.00	1.00	1.00	25	5
d_1^*	0.86	1.02	1.28	33	6
d_2^*	1.30	0.70	1.30	38	2

APPENDIX E
HARDWARE EXPERIMENT TASK 1 EFFICIENT INSPECTIONS DETAILS

In this section of the Appendix, more information can be found about optimization experiment of task 1. The velocity sequence given to the robot during the experiment can be seen Fig. 16.

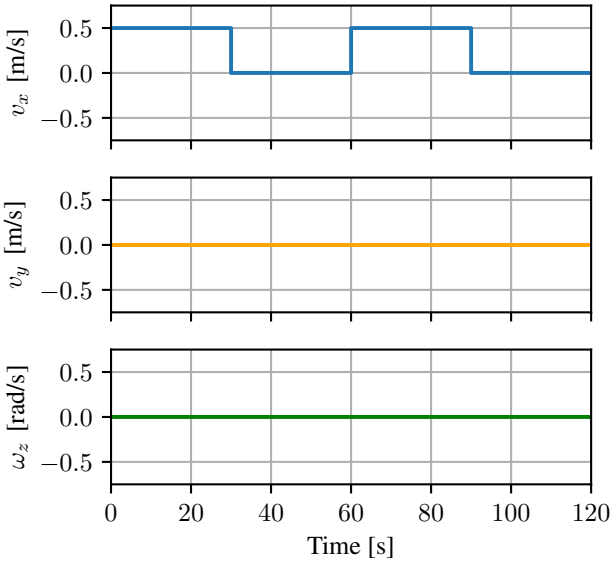


Fig. 14: The velocity command sequence for x (v_x), y (v_y), and yaw rate around z (ω_z) for task 1.

A. Detailed Table of Inspection Route Experiment

Tab. XIII shows the detailed measurements collected on the real robot. The columns show the four tested designs. For the rows, it is split in five main areas: the distance measured Δs, the energy consumed E, adjusted energy consumption E_{mech}, the CoT and CoT_{mech}. For each of these areas, the raw value is displayed for each run, but also the average value is shown. Note that this table shows 4 runs. Each run consists out of two times a 1 minute run.

E_{mech} is calculated by subtracting the energy usage of E_{system} from E_{total}. Here, E_{system} has been estimated to consume $0.54\,\text{Wh}$ during the two-minute combined run. The mechanical energy consumption is then divided by the actual distance walked to obtain CoT_{mech}. The key difference between CoT and CoT_{mech} is that CoT accounts for the total energy consumption of the entire system, while CoT_{mech} only considers the motor's energy consumption.

Tab. XIV shows a similar table as Tab. XIII, only now with the simulation data. Here, only one run is recorded.

TABLE XIII: Hardware results task 1. Comparison of distance walked, energy consumed, energy mechanical, CoT, and CoT mechanical for each design.

Metric	d_1^n	d_2^n	d_1^*	d_2^*
Δs [m]				
Run 1	29.82	31.64	32.47	29.72
Run 2	30.12	31.49	33.59	30.12
Run 3	30.22	31.97	33.47	30.02
Run 4	29.92	31.97	34.27	30.22
Average Δs	30.02	31.77	33.45	30.02
E [Wh]				
Run 1	1.06	1.02	0.97	0.92
Run 2	1.06	1.01	0.95	0.94
Run 3	1.06	1.02	0.96	0.94
Run 4	1.06	1.03	0.97	0.86
Average E	1.06	1.02	0.96	0.92
E_{mech} [Wh]				
Run 1	0.52	0.48	0.43	0.38
Run 2	0.52	0.47	0.41	0.40
Run 3	0.52	0.48	0.42	0.40
Run 4	0.52	0.49	0.43	0.32
Average E_{mech}	0.52	0.48	0.42	0.38
CoT [J/m]				
Run 1	127.47	115.67	107.00	111.81
Run 2	126.30	115.15	101.93	112.76
Run 3	126.33	114.65	102.96	112.28
Run 4	127.82	115.63	101.69	102.75
Average CoT	126.98	115.03	103.40	109.90
CoT_{mech} [J/m]				
Run 1	62.28	54.22	47.13	46.40
Run 2	61.76	53.41	44.05	48.21
Run 3	62.00	53.84	44.88	47.53
Run 4	62.84	54.83	44.96	38.42
Average CoT_{mech}	62.22	54.08	45.26	45.14

TABLE XIV: Software results task 1. Comparison of distance walked, energy consumed, and CoT for each design.

Metric	d_1^n	d_2^n	d_1^*	d_2^*
Δs [m]				
Run 1	32.75	32.28	33.57	29.66
E [Wh]				
Run 1	1.06	0.97	0.96	0.89
CoT [J/m]				
Run 1	116.68	107.76	102.68	108.56

B. Graphs Showing Energy Consumption and Torque over Time

Fig. 15 shows two graphs. The measured energy consumption over time and the effort for the knee motors. The colors

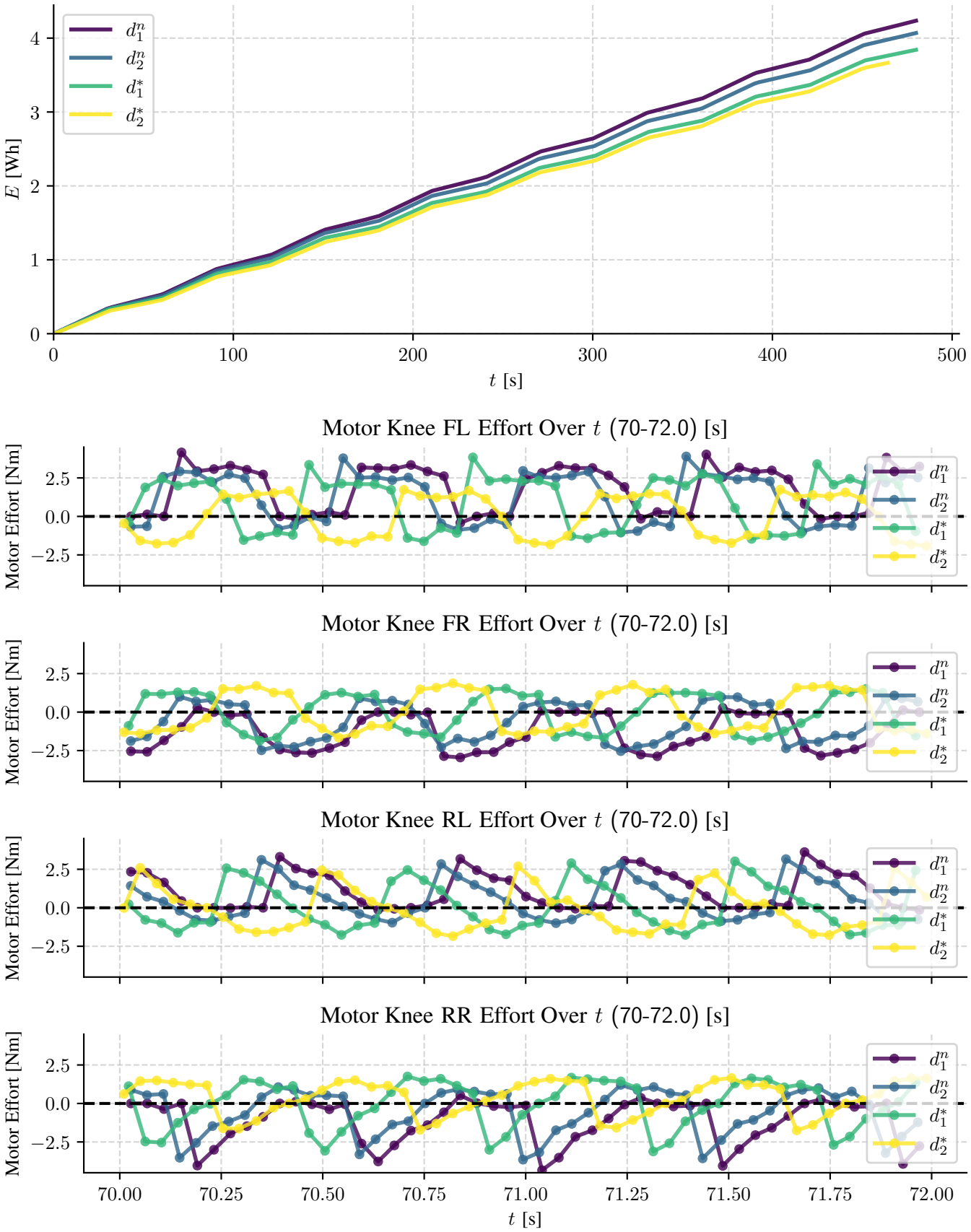


Fig. 15: Energy consumption and motor efforts for task 1. The top chart shows the energy consumption over eight trials for the designs d_1^n, d_2^n, d_1^*, and d_2^*. The bottom chart provides a snapshot of the knee actuator efforts during a 2.5-second window while the robot is walking. Actuator labels: FL = Front Left, FR = Front Right, RL = Rear Left, RR = Rear Right.

correspond to the colors of the designs of Sec. III-D1 (purple: d_1^n, blue: d_2^n, green: d_1^* yellow: d_2^*).

In the energy plot, one can see the difference when the robot is walking (steeper part), and when the robot is standing still (less steep part). Also, the optimized designs consume less energy than d_1^n, d_2^n. Also, a small measurement mistake at the end for d_2^* can be seen where the measurement stopped too early.

In the motor effort plots, the torque of each knee motor is plotted. Here, the effect of the PEA can be seen. For d_1^n, the design without PEA engaged, it can be seen that it has the highest peak torques and that most torque is on one side of the zero line (denoted as a dotted line). With PEA engaged, one can see the torque shift, resulting in lower peak torques and a more balanced torque graph on both sides of the zero line for design d_2^n, d_1^* and d_2^*. The optimized designs d_1^* and d_2^* have the most balanced line with the lowest peak torque. This is important, because the motor torque has a squared relationship with the motor power.

Appendix F
Hardware Experiment Task 2 Efficient Payloads Details

In this section of the Appendix, more information can be found about optimization experiment of task 2. The velocity sequence given to the robot during the experiment can be seen Fig. 16.

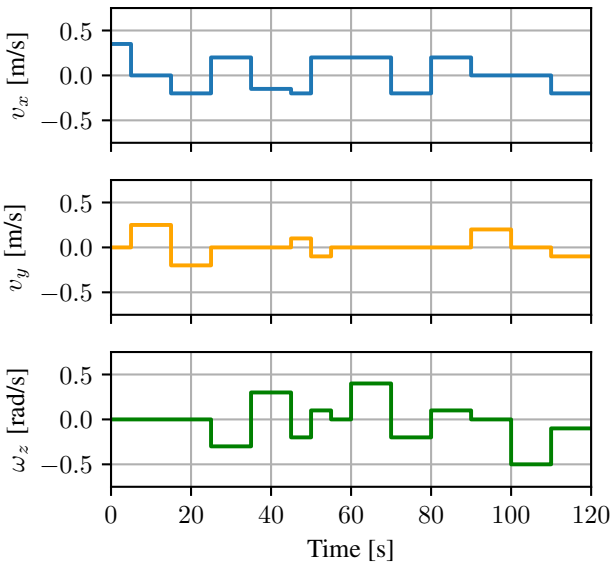


Fig. 16: The velocity command sequence for x (v_x), y (v_y), and yaw rate around z (ω_z) for task 2.

Tab. XV shows the detailed measurements collected on the real robot. The columns show the four tested designs. For the rows, it is split in five main areas: the distance measured Δs, the energy consumed E, adjusted energy consumption E_{mech}, the CoT and CoT_{mech}. For each of these areas, the raw value is displayed for each run, but also the average value is shown.

See App. E to see how E_{mech} and CoT_{mech} are calculated.

Tab. XVI shows a similar table as Tab. XV, only now with the simulation data. Here, only one run is recorded.

TABLE XV: Hardware task 2. Comparison of distance walked, energy consumed, energy mechanical, CoT, and CoT mechanical for each design.

Metric	d_1^n	d_2^n	d_1^*	d_2^*
Δs [m]				
Run 1	24.02	24.10	25.33	32.49
Run 2	24.14	23.93	27.17	31.34
Run 3	23.88	23.74	26.67	29.24
Run 4	24.02	23.56	26.69	30.00
Average Δs	24.02	23.83	26.47	30.77
E [Wh]				
Run 1	1.45	1.28	1.22	1.17
Run 2	1.45	1.28	1.21	1.22
Run 3	1.46	1.29	1.22	1.20
Run 4	1.46	1.30	1.23	1.19
Average E	1.45	1.29	1.22	1.20
E_{mech} [Wh]				
Run 1	0.91	0.74	0.68	0.63
Run 2	0.91	0.74	0.67	0.68
Run 3	0.92	0.75	0.68	0.66
Run 4	0.92	0.76	0.69	0.65
Average E_{mech}	0.91	0.75	0.68	0.66
CoT [J/m]				
Run 1	217.47	191.22	173.24	130.06
Run 2	216.61	192.38	160.60	139.77
Run 3	219.60	195.64	164.81	148.01
Run 4	218.23	198.26	166.50	143.11
Average CoT	217.98	194.38	166.29	140.23
CoT_{mech} [J/m]				
Run 1	136.53	110.54	96.51	70.22
Run 2	136.09	111.15	89.05	77.74
Run 3	138.21	113.75	91.93	81.52
Run 4	137.30	115.75	93.67	78.30
Average CoT_{mech}	137.03	112.80	92.79	76.95

TABLE XVI: Simulation data for task 2. Comparison of distance walked, energy consumed, and CoT for each design.

Metric	d_1^n	d_2^n	d_1^*	d_2^*
Δs [m]				
Run 1	23.47	23.92	23.72	27.08
E [Wh]				
Run 1	1.27	1.14	1.09	0.97
CoT [J/m]				
Run 1	194.57	171.24	166.18	128.73

Appendix G
Optimized Design Consistency Details

This section of the Appendix gives more information about the consistency experiment. Tab. XVII shows the five optimized designs and scores that are optimized with different random seeds for each task.

Note that the random seed influences a few things. For both task 1 and task 2, it changes parameters of the BO algorithm, the initial sampled designs, the noise, and the disturbances. For task 2 it also influences the commands that are sampled.

TABLE XVII: Design parameter values and scores for each policy.

Design	λ_1 [−]	λ_2 [−]	λ_3 [−]	$\varnothing$ [mm]	o [mm]	CoT [J/m]
d_1^*	0.86	1.02	1.28	33	6	107.64
$d_{1,1}^*$	0.79	1.02	1.30	40	2	109.02
$d_{1,2}^*$	0.90	1.07	1.29	41	4	107.61
$d_{1,3}^*$	0.90	0.97	1.30	33	7	109.76
$d_{1,4}^*$	0.79	0.95	1.30	39	3	109.14
$d_{1,5}^*$	0.89	0.96	1.30	32	8	108.79
d_2^*	1.30	0.70	1.30	38	2	127.60
$d_{2,1}^*$	1.30	0.72	1.29	40	0	130.34
$d_{2,2}^*$	1.25	0.70	1.29	40	0	129.05
$d_{2,3}^*$	1.20	0.71	1.30	39	0	131.37
$d_{2,4}^*$	1.10	0.71	1.30	35	2	134.92
$d_{2,5}^*$	1.29	0.70	1.23	37	0	129.66

APPENDIX H
UNIVERSAL VERSUS SPECIALIZED DETAILS

This section of the Appendix gives more information about the universal versus specialized policy experiment. Tab. XVIII shows the absolute reward scores for the specialized policy π_s and universal policy π_u and their relative differences for the four evaluated designs: d_1^n, d_2^n, d_1^*, and d_2^*. The specialized policies are trained using the exact same hyperparameters as the universal policy.

TABLE XVIII: Universal and specialized reward for each design.

Design	π_u Reward[-]	π_s Reward[-]
d_1^n	251.08	253.00
d_2^n	252.73	252.31
d_1^*	251.52	253.04
d_2^*	248.36	218.19