Learning torque control for quadrupeds

Daniel Schmidt[1], Lina Gaumann[1], Nico Bohlinger[1]

Abstract— **Quadrupedal locomotion can explore terrains that are hard to navigate. In recent research, reinforcement learning of torque-based policies has emerged as control paradigm promising better robustness and overall performance than state-of-the art PD-controllers. However, learning torque control struggles to find an upright standing position, which leads to long training processes. We present several approaches aiming to speed up the learning process and exploit the advantages of torque-based policies.**

I. INTRODUCTION

Quadrupeds can explore a wide range of hitherto unreachable and thus underexplored environments. They can handle steps, gaps, rubble and slippery surfaces [1]. However, quadrupeds are high-dimensional, nonlinear and underactuated systems whose contact states change over time. Therefore the system dynamics cannot be predefined [2]. To deal with this, corresponding controllers are oftentimes complex state machines that show low adaptability and rely on parameter tuning [3].

Approaches using reinforcement learning to optimize a reward function and thus derive a control policy can overcome the deficits of traditional controllers and have achieved state-of-the-art performance in navigating difficult terrain [4]. Learned policies can achieve higher rewards and fulfil the requirements of multiple tasks. They handle modelling and sensing uncertainties better than standard control methods [5].

The action space of reinforcement learning based legged locomotion is usually formed by the joint positions. They are scaled and then added to or subtracted from a nominal position vector, which represents a standing pose [2]. A PD-controller then converts them to joint torques, which form the motor input [6]. PD controllers typically operate on frequencies around 50 Hz and their gains have to be tuned manually. As those gains are set to a constant value, adaptability of those controllers to unknown environments is limited [2]. In novel approaches, changing the action space to joint torques shows promising results [7]. The intermediate PD controller can be rationalised as the policy computes directly applicable torques. The system seems to be more resistant against external disturbances and able to navigate difficult environments, especially when running on a higher frequency than with PD control.

However, in torque-based control, it is difficult to define a nominal position such as an upright posture because the associated torques vary depending on the environment and initial configuration [8]. Experiments based on [2] showed that quadrupeds operating on a torque-based policy struggled to find an initial standing position, leading to a long learning process and mediocre results.

A. Related work

Traditional approaches on controller design for legged locomotion yield complex states machines [3, 9]–[11]. Reinforcement learning has emerged as a more flexible method to derive control policies which yields state-of-the-art performance results [1, 2, 4, 5, 8, 12, 13]. Extensive research has been conducted studying position-based controllers [1, 6, 14, 15], while studies by [2, 5, 7] tested torque-based controllers.

B. Contributions

In this paper, we present and evaluate several approaches to speed up the learning of torque control for quadrupedal locomotion and try to achieve better results compared to the baseline position-based-approach. We pay special attention to achieving an initial standing position. The approaches differ in their action-space, policy and reward function and can be divided into two categories.

First, we pursue a physics-based approach, where we introduce an inductive bias to our policy. The bias is derived from a model-based gravity pre-training in which the robot tries to compensate its own gravity force.

In the second group of approaches, we aim to improve the architecture of the reinforcement learning setup through various modifications. In a pre-training curriculum, a reward is granted based on proximity to a predefined target position. Lastly, a multi-head neural network is designed to blend PD-control and torque control in order to combine their advantages of quick learning (PD) and overall better performance and robustness (torque).

II. SETUP

A. Reinforcement learning

We formulate our quadrupedal control problem as a discrete-time Markov Decision Process (MDP). At every time step t our agent observes a state s_t from the environment, samples an action $a_t \sim \pi(a_t|s_t)$ from a stochastic policy π, applies the action with a control function, transitions to a new state s_{t+1} and receives a reward $r_t = r(s_t, a_t, s_{t+1})$. Iterating this process generates a trajectory $\tau = \{(s_0, a_0, r_0), (s_1, a_2, r_3), \dots, (s_T, a_T, r_T)\}$ of horizon T. The goal is to find a policy π^* that maximizes the expected discounted sum of rewards over an finite horizon by collecting data by trial and error in an environment:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right],$$

[1]Technische Universität Darmstadt

where $\gamma \in [0,1]$ is a discount factor and $p(\tau|\pi)$ denotes the likelihood of trajectory τ under policy π.

While a variety of RL algorithms can be applied to tackle this problem, we follow [2] and [5] and use Proximal Policy Optimization (PPO) [16] to optimize our policy. The hyperparameters we use are listed in Table II. We model both actor and critic as a multilayer perceptron. The critic network consists of three layers of 512, 256, and 128 units, while the structure of the policy network depends on the respective approach we are using (see section III). Similar to [2], all layers use exponential linear activation (ELU).

B. Observation Space

As in [2], we decided to use a concise representation of the robot to avoid overfitting to simulated dynamics. All observations available for the policy network could be inferred from measurements in real-world.

C. Action Space

The action space depends on the respective approach. In all cases the policy outputs samples $a_t \sim \mathcal{N}(\mu_\pi, \Sigma_\pi)$ of a Gaussian distribution whose mean μ_π and standard deviation Σ_π are learned by the network, with the mean controlling the performance and the standard deviation controlling the exploration. The sampled actions consist of either a 12-dimensional (PD and torque approaches) or 24-dimensional vector (multi-head approaches) which is converted into the applied joint torques using scaling and a corresponding control function (see section III for more details).

D. Rewards

The general reward function is defined such that an RL agent receives a higher reward if it follows a given set of high-level user commands whilst satisfying additional requirements to produce a natural-looking gait. It consists of positively weighted terms that encourage the agent to behave in certain ways, and negatively weighted terms that serve as regularization. A complete composition of the total reward with corresponding parameters can be found in Table III. The most important terms are the command tracking rewards $r_t^{\text{track}_{xy}}$ and $r_t^{\text{track}_{\text{angular}}}$ that encourage the robot to track a given linear velocity $v^\star = [v_x^\star, v_y^\star, v_{\text{angular}}^\star]$ on locomotion tasks:

$$r_t^{\text{track}_{xy}} = w_t^{\text{track}_{xy}} \exp\left(-\gamma^{\text{track}_{xy}} \left\| v_{xy}^\star - v_{xy,t} \right\|_2^2\right),$$
$$r_t^{\text{track}_{\text{angular}}} = w_t^{\text{track}_{\text{angular}}} \exp\left(-\gamma^{\text{track}_{\text{angular}}} \left\| v_{\text{angular}}^\star - v_{\text{angular},t} \right\|_2^2\right).$$

Inspired by [12], we use a curriculum learning approach for the weights meaning that $w_t^{\text{track}_{xy}}$ and $w_t^{\text{track}_{\text{angular}}}$ are time dependent and following a specific curriculum defined by:

$$w_t^{\text{track}} = \begin{cases} w_{\text{start}}^{\text{track}}, & t < t_{\text{start}} \\ w_{\text{start}}^{\text{track}} + \left(w_{\text{end}}^{\text{track}} - w_{\text{start}}^{\text{track}}\right)\left(\frac{t - t_{\text{start}}}{t_{\text{end}} - t_{\text{start}}}\right), & t \in [t_{\text{start}}, t_{\text{end}}] \\ w_{\text{end}}^{\text{track}}, & t > t_{\text{end}}. \end{cases} \tag{1}$$

Here, t_{start} and t_{end} denote the number of total training steps that have to be passed to start or end the curriculum and t denotes the total number of steps that have elapsed when the reward is computed. This aims to structure the reward landscape such that the robot first learns how to achieve the target x and y velocities and then learns to achieve the objective yaw velocities while adhering to various constraints and regularizations.

After summing up all individual terms we clip rewards below zero. This prevents the reward from being negative in early training stages (due to low tracking reward and higher regularization costs), which could cause termination (i.e. falling to the ground) to be the most rewarding strategy, which we want to avoid learning.

E. Platform

We used the PPO implementation of RL-X [17]. We did not test our results on a real robot, but used the Unitree A1 for simulations. The simulations were performed in a gym environment [18] using the MuJoCo physics simulator [19].

F. Domain adaption

A complete list of all randomized aspects can be found in the appendix subsection I-B. The two most important aspects are a sampling of the target linear velocities approximately every 10 seconds and the addition of random domain perturbations. Those differ in strength and probability of occurrence and enable us to measure and improve robustness against external disturbances.

III. APPROACHES FOR IMPLEMENTING TORQUE CONTROL

In this project we developed different strategies for learning torque control that differ in their action space, policy and the reward function that was used for learning. They will be compared against each other and against the traditional position-based approach.

In traditional position-based RL, the policy network outputs samples of joint position values. Those are scaled and then added or subtracted from a reference or nominal position vector q_{nominal}, like a standing pose for the robot [2]. This yields better survival rates in the early stages of learning and establishes a good starting point for policy exploration [2].

After the target joint positions are computed they are tracked by a low-level PD controller converting them into the applied joint torques. See section II-A in the appendix for details on the control function. We use the same controller parameters whenever a PD-control is part of our approaches (e.g. in multi-head, see subsection III-C). The policy network usually updates the target joint positions at low frequency, while the PD controller runs at high frequency [2, 5]. An illustration of the common position-based approach can be found in the appendix (Figure 4).

In contrast, when using torque control as presented in [2], the RL policy network directly outputs joint torques samples which get scaled and then applied directly on the robot (s. Figure 1). Since the numerical range of torque control (e.g., -33 to 33 Nm on the A1 robot) is much larger than that of

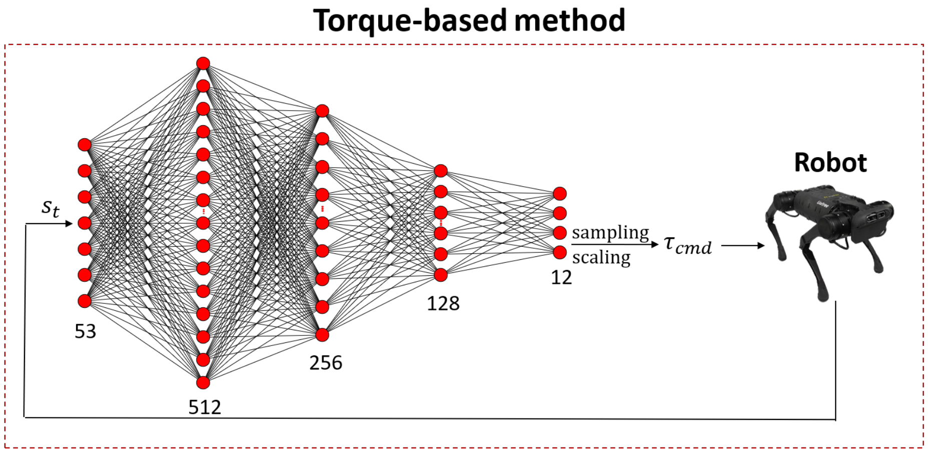


Fig. 1: Graphical representation of the default torque-based method. It merges the two modules of the position-based approach into a neural network module, which directly outputs torque commands. The setup is used for the default-torque approach (section III), gravity pre-training (section III-A.4) and curriculum pre-training (section III-B.2).

position control (e.g., -3.14 to 3.14 rad) [2], we use a large value of 4 as the action scale for all torque or torque related models.

However, the default torque control approach has the issue that determining nominal torques like in the position-based approach is difficult, since such torques might vary with different environments and initial configurations [8]. This causes torque-based policies without any additional requirements to be slow in training and sample inefficient [5], which we were able to confirm in our experiments (see Figure 3). The resulting policies varied significantly in performance. While it was able to learn an upright position in some runs after a considerable number of steps, it failed to learn completely in other experiments. In the early stages the default torque policy always struggled trying to find a standing position. The robot showed abrupt and seemingly random behaviour, with quick sudden joint movements that resulted in a high number of collisions and short episodes.

Therefore, all our approaches aim to find methods that stabilize the robot in the early stages of learning, accelerate initial training and improve the sample efficiency to learn following desired locomotion commands better. We developed various strategies for learning torque control that differ in their initialization, action space, policy and the reward function.

In our first approach, gravity pre-training, we exploit physics to achieve better results. In a separate training process, the robot is taught to compensate gravity force and thus stand upright. In our other strategies, we modify the machine learning architecture: in our second approach, curriculum pre-training, a new reward term is introduced: the nominal position reward. In the early stages of training, it gives high rewards to positions that are close to the nominal position as used in PD-control. Our last approach combines the position-based approach and the torque approach in one multi-head policy network which outputs actions of doubled size interpreted as target positions and torques that will be mixed in a special controller.

A. *Gravity pre-training*

1) General idea: During the initial stages of training for torque control, the robot consistently failed to achieve an upright position. To address this matter, a physics-based approach was investigated to enhance the early learning of standing. This approach, inspired by [5], pre-trains the model with torques computed to compensate for gravitational forces acting on the robot. We call it gravity pre-training.

2) Model: The contact-consistent whole-body model [20] was used to describe the system dynamics and obtain a corresponding control framework. It is applicable to quadrupeds, can handle changes of arbitrary contact states and under-actuated systems. As [5] has shown, the provided control framework is suitable for gravity pre-training. Furthermore, the control torques are designed to minimise the acceleration energy of the system, resulting in lower power consumption. Short descriptions of other eligible models can be found in appendix II-B.

The movements of the robots relative to its surrounding are described by six virtual joints. The dynamic equations of motion, formulated in operational space for a constrained setup, are

$$\Lambda(q)\ddot{x} + \mu(q,\dot{q}) + p(q) = F, \tag{2}$$

where q denotes the vector of joint angles, x is the operational space coordinate, $\Lambda(q)$ is the joint space inertia tensor, $\mu(q,\dot{q})$ are the Coriolis and centrifugal forces and $p(q)$ the gravity force. F is the resulting force vector.

The proposed control framework consists of three parts: a task control, a posture control and a reaction force control:

$$\Gamma = \Gamma_{\text{task}} + \Gamma_{\text{posture}} + \Gamma_{\text{force}} \tag{3}$$

$$\text{Task control} \quad \Gamma_{\text{task}} = J^{\mathrm{T}}F = J^{\mathrm{T}}\Lambda f^{*} + \mu + p \tag{4}$$

$$\text{Posture control} \quad \Gamma_{\text{posture}} = N^{\mathrm{T}}\Gamma_0^{\mathrm{k}} \tag{5}$$

$$\text{Force control} \quad \Gamma_{\text{force}} = J_{\mathrm{c}}^{\mathrm{T}} f_{\text{csd}} - S_{\mathrm{c}} f_{\mathrm{c,task+posture}}. \tag{6}$$

The task control term (4) ensures that the system fulfils the set task, like performing movements specified by the user. For gravity pre-training, we require accelerations $f^* = \mathbf{0}$, as the robot should assume a standing position and not move further. J denotes the Jacobian matrix, N is the null-space projection matrix in the actuated torque space.

The posture control term (5) keeps the robot posture close to a nominal posture Γ_0^{k} derived from a force equilibrium of the stationary robot.

Finally, the force control term (6) ensures that the robot remains in its current contact state (ideally placing four feet on the ground) without violating the limit of the maximum acceptable contact forces. As this term is not used in our model further explanations can be found in appendix II-C.

3) Challenges: The contact Jacobian J_{c} defines the relationship between the joint position and the positions of the contact points (9). During the implementation of force control, this Jacobian presented challenges due to its large magnitudes after inversion, leading to unrealistically high estimates of the contact forces (11), which in turn caused frequent violations of the control range limit of $\Gamma_{\max,\min} = \pm 33.6\,\mathrm{Nm}$. Consequently, the force control outputs were unreliable, which is why we opted not to use reaction force control in our setup. After omitting the reaction force control term, the model is not guaranteed to maintain a consistent contact state. The contact state describes the contact points between the robot and its environment. Whenever the contact state changes, for example when an additional knee makes contact with the ground or when a foot is lifted, the dimensions of the contact Jacobian J_{c} are altered. This poses challenges for task control, which relies on the derivative of the contact Jacobian, as its computation requires consistent matrix dimensions. A solution was implemented that dynamically adjusts the size of the previous Jacobian matrix to match the current one.

4) Application: The model is employed during the data collection phase of gravity pre-training, where the robot is initialized in 20 000 randomly generated configurations. For each initial configuration, the control torques to reach a standing posture are computed, and the corresponding experience pairs of initial states and control torques are recorded. Pairs with mean absolute torques exceeding $30\,\mathrm{Nm}$ are excluded, as values approaching the control range limit suggest inaccurate calculation outcomes. These experience-pairs are processed in a supervised learning process and initialize the torque-based policy network (Figure 1) in form of a weak inductive bias. The pre-trained model can be reused as long as both state and action spaces remain the unchanged.

B. Curriculum pre-training

1) General idea: Like gravity pre-training, the curriculum pre-training approach does also perform a kind of pre-training, but in contrast to the procedure described in III-A we do not use an extra gravity model and supervised learning process. Instead, the pre-train approach is placed inside our torque-based RL framework (Figure 1) by adapting the reward function described in II-D. The goal is to shape the initial reward landscape such that in the beginning, the policy is strongly attracted to standing in a nominal position and then later on learns a locomotion policy and to satisfy other criteria starting from this position.

2) Realisation via curriculum reward: To realise this, we added an extra reward term that has a variable multiplicative curriculum weight similar to the curriculum of the tracking reward. This term describes the deviation of our joint positions to the nominal standing positions that are used in our PD baseline model (see section IV-B). The reward weight for this term is adapted trough a planned curriculum where the weight starts at $w_{start}^{\mathrm{nominal}}$ and is then linearly decreased over time until the end weight $w_{end}^{\mathrm{nominal}}$ is reached.

As we want to minimize the deviation of our joint positions to the nominal positions, this weight is then multiplied with some kind of reward that is larger when we are close to the nominal positions and lower if we are far away. Additionaly, we need to ensure that the reward is positive at all times. See section II-D in the appendix for further information as to why that is necessary and what other options for this reward term were considered.

The approach we found most useful is to use an exponential reward term. This directly ensures a lower and upper bound of the reward:

$$r_t^{\mathrm{nominal}} = w_t^{\mathrm{nominal}} \exp\left(-\gamma^{\mathrm{nominal}} \left\| q_t - q_{\mathrm{nominal}} \right\|_2^2\right),$$

where $\gamma^{\mathrm{nominal}}$ denotes a suitable temperature.

In addition to the new nominal position curriculum, the standard tracking curriculum for the reward described in section II-D is shifted such that is starts after $3 \cdot 10^6$ steps and ends after $13 \cdot 10^6$ steps. This allows us to first learn good standing torques and then learn how to follow specific commands and to polish the motion.

C. Multi-head neural network

1) General idea: This method combines the presented position-based and torque-based approaches by using a multi-head neural network that outputs action samples $a_t \in \mathbb{R}^{24}$ of twice the size of the number of joints that are fed into a special controller consisting of a PD-controller and a mixing component (see Figure 2).

The doubled output allows us to interpret the samples produced by the first head as target joint residuals that get scaled and fed into the classical PD controller (see (8) in the appendix), where they get added to the nominal position and are used to compute the PD controller command torques $\tau_{\mathrm{cmd}}^{\mathrm{pd}}$. The samples produced by the second head get scaled and interpreted as direct command torques $\tau_{\mathrm{cmd}}^{\mathrm{torque}}$ as in the default torque method. We use the same scaling factors as before. The two resulting torque vectors are subsequently combined using a weighted sum where the weights can depend on the number t of training steps already passed:

$$\tau_{\mathrm{cmd}} = \theta_t^{\mathrm{pd}} \tau_{\mathrm{cmd}}^{\mathrm{pd}} + \theta_t^{\mathrm{torque}} \tau_{\mathrm{cmd}}^{\mathrm{torque}} \tag{7}$$

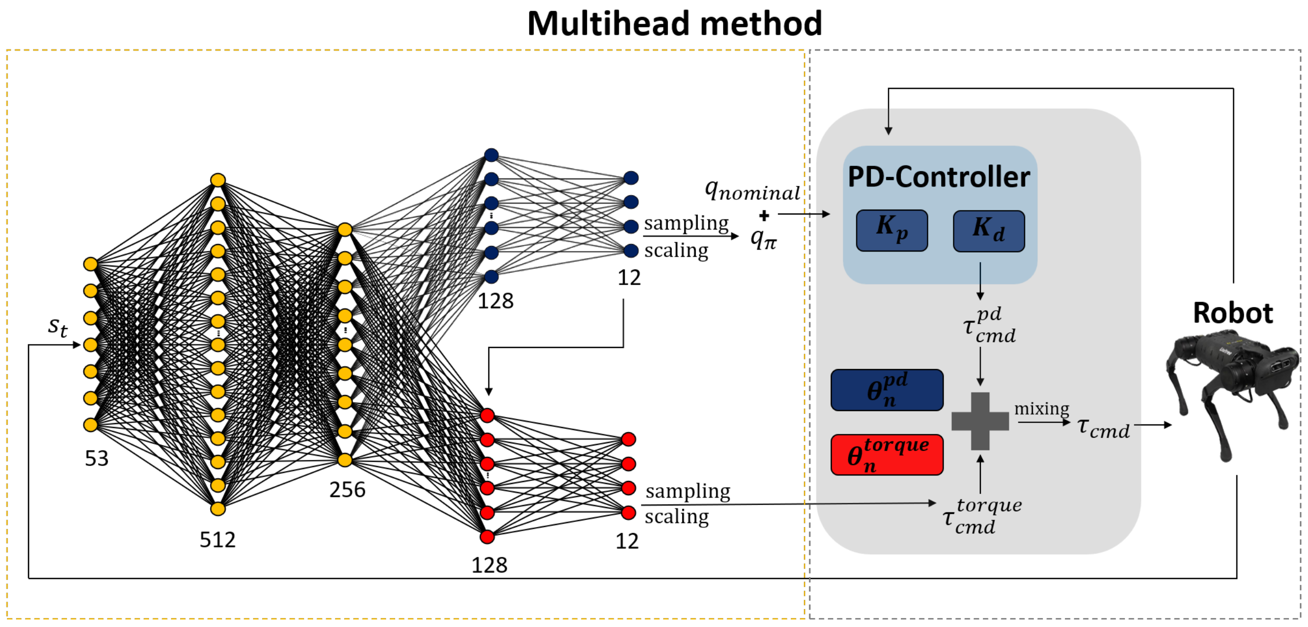


Fig. 2: The proposed multi-head framework for learning torque control. A neural network consisting of general layers (yellow) and one head for position-based output (blue) and one head for torque-based output (red) output two action vectors. The first one is interpreted as position residuals which are fed into a pd controller where a command torque $\tau^{\text{pd}}_{\text{cmd}}$ is computed. The second one is directly interpreted as torque command $\tau^{\text{torque}}_{\text{cmd}}$. Then both commands are subsequently combined using a weighted sum with time dependent weights and applied to the robot.

The primary goal of this method is to stabilize the training process in the early stages by initially placing a higher weight on the torques derived from the PD controller, which relies on the nominal joint positions. Over time, the influence of these PD-derived torques is gradually reduced, allowing the system to transition smoothly towards relying more on the torques generated directly in the neural network.

2) Architecture Details: For a fair comparison, the network architecture is designed similarly to a traditional position-based or torque based model and consists of two general layers with 512 and 256 units, respectively. The output of the second layer is first fed into the joint-position-head, which includes one hidden layer and outputs 12 joint position residuals. These joint position residuals, along with the output from the general layers, are then fed into the torque head. The torque head has the same structure as the joint-position-head, featuring a hidden layer that processes the combined input to produce the final torque outputs. This architecture allows a dual approach, where the joint-position-head focuses on generating target joint positions that stabilize the robot's stance, while the torque head refines these positions into precise torque commands.

The intuition behind using the position-based-output as additional torque head input is to embed a form of control mechanism directly within the neural network. This internalization allows the network to develop a deeper understanding of the relationship between joint positions and the required torques, leading to more precise and adaptive control strategies.

Additionally we aim to stabilize the later training stages, when the weight assigned to these PD-derived torques is gradually reduced, allowing the network to rely more on the torque outputs generated directly by the neural network. However, even as the influence of the PD torques decreases (i.e. when the PD torque weight approaches zero), using this approach the information learned by the network in generating these PD torques is not lost. Instead, this information continues to flow into the torque head, effectively embedding the principles of PD control within the neural network itself and ensures that the network benefits from the early-stage stability provided by PD control.

3) Application:: When using the multi-head approach, the crucial part lies in an adequate choice of the mixing coefficients θ^{pd}_t and θ^{torque}_t. For example, using constant coefficients, such as a low PD weight and a high torque weight from the outset, can result in the same issues as default torque control, leading to slow training progress by not taking advantage of the stabilizing influence of the PD controller early in training.

In our tests we used a weights curriculum similar to the track reward weights (1). We defined start-mixing and end-mixing weights as well a number of training steps t_{end} after which the final weights should be reached. We then adjusted the weights linearly towards the final weights based on the fraction of training time steps already passed and the curriculum t_{end}. In contrast to the reward weights we do not update the weights at every step, but select a frequency that determines the intervals in which the weights are updated. We always choose this frequency such that the updates do only take place between the rollouts and policy updates of PPO. For a more comprehensive discussion of the right choice of mixing weights we refer to the appendix II-E.

IV. EXPERIMENTAL SETUP AND RESULTS

A. Evaluation Setup

To compare the strengths and weaknesses of our various approaches, a comprehensive evaluation setup is established that allows us to compare performance, efficiency, and robustness of different model classes under consistent conditions.

The same set of Proximal Policy Optimization (PPO) hyperparameters is utilized for all experiments (see Table II in the appendix). All models operate at a standard frequency of 50 Hz, which is a typical control rate for such tasks. Hence, effects of the model architecture and training strategies are isolated and the influences of the hyperparameters, control timing or update rates are eliminated.

Given that the models were trained with different reward functions, we developed a separate evaluation environment to enable unbiased assessment. This environment is independent of the training process and has its own evaluation metrics. Those metrics consist of three tracking metrics (called track xy difference norm, track angular difference norm and track overall percentage), a cost of transport term and a collision force term. This combination enables a holistic assessment of each model's performance by evaluating not only its ability to achieve objectives (as indicated by the tracking rewards) but also its energy efficiency (measured by the dimensionless cost of transport) and safety (assessed through collision forces).

We evaluate every 128k training steps, i.e. after every fourth policy rollout in PPO. During the evaluations, we focus on the average policy predictions without considering the influence of the policy standard deviation which is used for exploration during the training. This way, the focus lies on the model's actual performance.

To assess the robustness of each model, we conduct four separate experiments for each model class. These experiments differ in their probability of occurrence and intensities of linear velocity perturbations. For the first experiment (called "Low perturbation") we use linear velocity pushes that lie in $[-0.3, 0.3]\,\mathrm{m\,s^{-1}}$. These perturbations occur with a probability of 0.002. For the second experiment ("High perturbation") the linear velocity pushes lie in $[-1, 1]\,\mathrm{m\,s^{-1}}$ and the probability remains the same. For the third experiment (High perturbation and frequency), the perturbation frequency is tripled to a probability of 0.006. In the fourth experiment (Doubled high perturbation and frequency), we again increase the perturbation frequency by doubling the probability to 0.012, but only in the evaluation environment. Thereby the models get tested in situations other than those they were trained in.

For each experiment, we use four runs per model that differ in their seeds. By doing so, we evaluate how well each model adapts to varying conditions, offering insights into their generalisability.

B. Baseline

We use a common position-based model as presented in section III and shown in appendix II-A as baseline. The position-based approach is a well-established benchmark in quadrupedal robot control and is widely used in robotic systems due to its simplicity and reliability, especially in ensuring basic stability and locomotion [1, 2, 21].

C. Gravity pre-training

Simulations reveal problems that a robot encounters when torques are computed using the gravity pre-training model (3) as described in subsubsection III-A.2. The robot does not consistently perform as expected. When initialized in a position where all four feet maintain ground contact, the robot typically remains upright. In some cases, the derived torques appear to be insufficient to fully support the robots' weight and one or two joints give way, causing the corresponding knee joints to touch the ground. If few feet are in ground contact at initialization, or ground contact cannot be maintained, abrupt, rapid and seemingly random joint movements occur, resembling the behaviour initially exhibited under the default torque policy. This issue arises due to the controller's difficulties in handling changes in contact state without the force control term, as explained in section III-A.4.

The gravity pre-trained policy network fails to learn completely. The mean torque norm for PD is approximately 250. For gravity pre-training, that value over 13.500 in the same step range as above. A torque norm of this magnitude suggests that all joints are consistently operating at the control range limit. One possible explanation for this behavior is that the supervised learning process was dominated by data from those runs characterized by random behavior and excessively high torques. The reward terms for action rate norm and acceleration norm similarly deviated from the values achieved by the other models by several orders of magnitude.

These norms have exceptionally high values and negative weights, resulting in high costs and thus in strictly negative reward terms. These are persistently clipped to zero and make learning impossible. Therefore, gravity pre-training is not included in any of the experiment figures as the policy was not able to learn.

D. Curriculum pre-training

For the final experiments we use a curriculum pre-training approach with $w^{\text{nominal}}_{\text{start}} = 20\Delta t$, $w^{\text{nominal}}_{\text{end}} = \Delta t$, $\gamma^{\text{nominal}} = 0.5$ and a curriculum length of $5 \cdot 10^6$ time steps, which turned out to be promising parameters.

E. Multi-head

In our first tests, we used mixing weights resulting in a weighted average, i.e. θ^{pd}_t and θ^{torque}_t with $\theta^{\text{pd}}_t + \theta^{\text{torque}}_t = 1$, creating a dynamic problem for both the PD and torque heads, as the weighted average shifts over time. Another approach is to maintain the torque weight constant while gradually reducing the PD weight. This method results in a more stable training environment since only the position-based control problem changes over time, leading to a less volatile learning process, but also slows down the training at the beginning as the torque influence is bigger than in the first approach. Nevertheless, we decided to use the latter approach for the evaluation experiments.

For our experiments, we use two different models derived from the multi-head approach. Both use a mixing curriculum of $50 \cdot 10^6$ steps, an update frequency of 64k steps, a constant torque weight $\theta_t^{\text{torque}} = 1$ and a PD-torque start weight of one. They only differ in their PD end weight.

The first models' end weight is zero. We call it "torque multi-head" because the final torque command contains only the torque signal produced by the torque head after the curriculum has ended. The second model is called "multi-head" as its weight ends at 0.1, meaning that this model still mixes two torque commands after the curriculum end.

F. Results

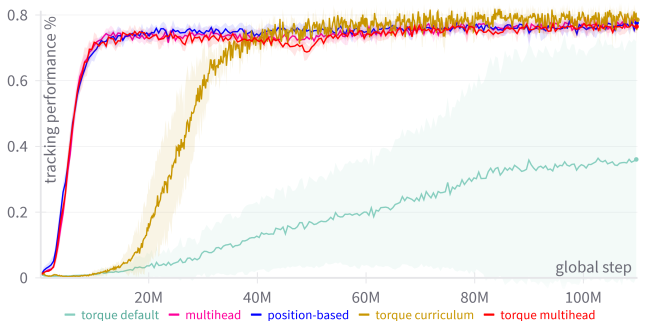


Fig. 3: Tracking learning curves of experiments with low domain perturbation. The default torque model (green) without any modifications shows significantly worse learning speed and performance than all other models.

The performance of our experiments, broken down by the five tracked metrics (see subsection IV-A), are summarized in Table I and also shown graphically in appendix III, figures 6 and 7. The dependence of the tracking performance on the number of steps is illustrated in Figure 3, furthermore appendix II-F contains figures that analyze walking height (Figure 5a) and the gradient norms used to update the critic network (Figure 5b).

Figure 3 shows the learning curves of torque default, position-based, torque curriculum, torque multi-head and multi-head as tracked in the least perturbation environment. It is representative for all conducted experiments and reveals the first interesting difference between the approaches. All tested approaches learn significantly faster than the default torque-based approach without additional constraints and dominate it in terms of tracking. The multi-head models' learning curves (red, purple) are similar to those of the position-based model (blue) but show slightly worse tracking performance while the mixing coefficients are changing. However, the performance of the multi-head models adapts quickly after the final weight ($t = 50 \cdot 10^6$) is reached, and gets a slight boost due to the now constant learning problem.

In contrast, the curriculum learns more slowly. This results from the curriculum itself, as this models' reward function is dominated by the nominal position reward in the beginning, because of which a standing position is learned first while locomotion becomes less important. After the curriculum has finished, optimizing the tracking performance still needs some time. This could result from the value function landscape being overestimated by the high rewards of the nominal position reward in the beginning. Therefore time is needed to update the value function and allow the policy to learn to move (see appendix II-F and Figure 5b for more details).

Comparison of the mean tracking performance percentages (first row of Table I) shows that the torque curriculum method performs best in the first two experiments. Thus, the nominal position reward contributes most to the improvement in locomotion performance when perturbation frequency is low. However, upon increase in the perturbation frequency, the model tracking performances converges to similar values for all models. Therefore, the tracking superiority of curriculum method is not robust to more frequent disturbances.

When comparing the individual tracking norms for experiments with higher perturbation, the position-based approach appears to be most stable as it has the lowest squared deviation in the xy-tracking velocities (second row), while the multi-head models seem to be more robust in following the target angular velocities (third row). However these are only minor differences that lie within the standard deviation and must therefore be interpreted with caution.

The illustrated experiments use collision forces as a passive term only, which means that the training metric does not contain any collision force term. However, further experiments that include a collision force penalty in the training reward lead to similar results. Analysis of the collision forces (fourth row) shows that the collision forces are the lowest for the position-based strategy, although those of the torque curriculum are close. This could be due to the fact that torque control uses a larger area of control space.

In contrast, differing results can be observed in transportation costs (fifth row), which are indirectly encoded in the training reward by a torque norm regularization. Both multi-head-based approaches perform significantly better than the position-based and curriculum method in terms of energy efficiency. For the low disturbance environments, the curriculum approach performs worst. For experiments with more frequent disturbances, the curriculum and PD methods perform equally bad and are clearly surpassed by the multi-head approaches. Overall, the costs of transport metric indicates that the multi-head models learned to follow specific commands using a mixing approach that results in lower torques and therefore less energy consumption.

Comparison of the multi-head models among each other shows that the multi-head torque model performs slightly

Metric	Model	Environment			
		Low perturbation	High perturbation	High perturbation & frequency	Doubled high perturbation & frequency
Track %	PD	0.7642	0.7518	0.7163	0.6765
	Torque multi-head	0.7624	0.7534	0.7159	0.6699
	Multi-head	0.7669	0.7511	0.7181	0.6739
	Torque curriculum	**0.7866**	**0.7700**	0.7197	0.6764
Track xy	PD	0.0324	0.0511	0.0933	0.1602
	Torque multi-head	0.0350	0.0556	0.0997	0.1685
	Multi-head	0.0344	0.0544	0.0970	0.1698
	Torque curriculum	0.0316	0.0534	0.0972	0.1697
Track angular	PD	0.0124	0.0122	0.0152	0.0171
	Torque multi-head	0.0110	0.0113	0.0138	0.0158
	Multi-head	0.0102	0.0113	0.0133	0.0156
	Torque curriculum	0.0097	0.0111	0.0143	0.0177
Collision forces	PD	**135.1**	**131.7**	**138.5**	**148.7**
	Torque multi-head	140.9	145.5	157.3	166.8
	Multi-head	139.0	142.0	153.5	160.1
	Torque curriculum	**136.0**	134.7	150.5	151.9
Cost of transport	PD	0.8316	0.8518	0.9728	1.0340
	Torque multi-head	**0.7187**	**0.7307**	**0.8380**	**0.8808**
	Multi-head	**0.6977**	**0.7209**	**0.8029**	**0.8611**
	Torque curriculum	0.8590	0.9112	0.9572	1.0373

TABLE I: Comparison of evaluation performance in different environments. For each setting we used four runs per method. The metrics shown are: overall tracking performance (track %), tracking of the linear and angular velocity of the base (track xy and track angular), cost of transport and strength of the collision forces. The values shown are the mean values of the four runs during all evaluations after the 90 millionth training step. Values in bold indicate significant superiority in the respective metric and experiment.

worse in all categories compared to the multi-head model whose PD-mixing weights end at 0.1. This suggests that the position-based torques continue to act as a stabiliser even after the curriculum ends. Another reason for this could be the slightly larger mixing weight updates of the multi-head torque model due to the same curriculum time, but lower end weight.

V. CONCLUSIONS AND FUTURE WORK

We have presented several possible modifications of a torque-based reinforcement learning policy, intended to improve it's overall performance compared to a baseline position-based policy, with special focus on learning speed, generalisability and robustness. In each approach, we addressed the problem that a torque learning policy struggles to find an initial standing position. We presented gravity pre-training as a model-based addition to the policy. Therein, initial standing shall be achieved by compensating the gravity force. While first results in data collection seemed promising, we have not yet been able to conduct experiments with the pre-trained policy due to problems in conserving the contact state.

In the curriculum pre-training, we modified the reward term to encourage the robot to assume a position close to the nominal position in early stages of training. This led to a slight improvement in tracking performance, which is not robust against more disturbances.

Finally, we constructed a two-headed neural network to combine both position-based and torque-based control, aiming to exploit the advantages of both strategies. It showed good angular tracking performance, but was especially successful in reducing the cost of transport.

In conclusion, we were able to apply a variety of methods with the capability to improve learning torque control. All tested models despite gravity pre-training were able to learn and beat unmodified torque control, especially with regard to the duration of their learning process. They show basic robustness against perturbations. However, only slight improvements in isolated aspects could be seen in comparison to the state-of-the art position-based control.

For future research, we suggest investigating a combination of the various approaches presented to achieve better results. Extending the duration of the learning process might provide interesting insights, especially for the curriculum-reward approach. Furthermore, the nominal position could be adjusted to a higher standing position that enables even faster training. For the gravity pre-training approach, future efforts should focus on implementing reaction force control to solve the problem of varying Jacobi matrix dimensions. In addition, physically implausible data should be eliminated through feature engineering and data cleaning. Regarding the multi-head approach, we propose exploring to run torque control and PD control at different frequencies.

Experiments in different environments could provide useful insights. Finally, all our work has been carried out in simulation. After improvements have been made, a sim-to-real transfer should be attempted.

References

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

[2] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, "Learning torque control for quadrupedal locomotion."

[3] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter, "Dynamic locomotion on slippery ground," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4170–4176, 2019.

[4] 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.

[5] D. Kim, G. Berseth, M. Schwartz, and J. Park, "Torque-based deep reinforcement learning for task-and-robot agnostic learning on bipedal robots using sim-to-real transfer," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6251–6258, 2023.

[6] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots."

[7] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *IROS Madrid 2018*, ([Piscataway, New Jersey]), pp. 1–9, IEEE, 2018?

[8] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots."

[9] M. Focchi, R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell, and C. Semini, "Heuristic planning for rough terrain locomotion in presence of external disturbances and variable perception quality," 2019.

[10] M. Camurri, M. Fallon, S. Bazeille, A. Radulescu, V. Barasuol, D. G. Caldwell, and C. Semini, "Probabilistic contact estimation and impact detection for state estimation of quadruped robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1023–1030, 2017.

[11] J. Hwangbo, C. D. Bellicoso, P. Fankhauser, and M. Hutter, "Probabilistic foot contact estimation by fusing information from dynamics and differential/forward kinematics," IEEE, 2016.

[12] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science robotics*, vol. 4, no. 26, 2019.

[13] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," Robotics: Science and Systems Foundation, 2019.

[14] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots."

[15] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, "Compliant quadruped locomotion over rough terrain," in *IEEE*, pp. 814–820, IEEE / Institute of Electrical and Electronics Engineers Incorporated, 2009.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms."

[17] N. Bohlinger and K. Dorer, "Rl-x: A deep reinforcement learning library (not only) for robocup."

[18] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym."

[19] 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* (I. Staff, ed.), ([Place of publication not identified]), pp. 5026–5033, IEEE, 2012.

[20] J. Park and O. Khatib, "Contact consistent control framework for humanoid robots," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*, pp. 1963–1969, IEEE / Institute of Electrical and Electronics Engineers Incorporated, 2006.

[21] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning."

[22] I. Exarchos, Y. Jiang, W. Yu, and C. K. Liu, "Policy transfer via kinematic domain randomization and adaptation."

[23] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world."

[24] B. Siciliano, L. Sciavicco, and L. Villani, *Robotics: Modelling, planning and control*. Advanced Textbooks in Control and Signal Processing, New York: Springer, 2009.

[25] M. Hutter, M. A. Hoepflinger, C. Gehring, M. Bloesch, C. D. Remy, and R. Siegwart, "Hybrid operational space control for compliant legged systems," in *Robotics* (N. Roy, P. Newman, and S. Srinivasa, eds.), pp. 129–136, Cambridge, Massachusetts: The MIT Press, 2013.

[26] M. Gor, P. Pathak, A. Samantaray, J. M. Yang, and S. W. Kwak, "Dynamic modeling and simulation of compliant legged quadruped robot," 2013.

Appendix I
Setup

A. PPO hyperparamers

The hyperparameters used for PPO are listed in Table II.

Learning rate	0.0001
Discount factor	0.99
Learning epochs per rollout	5
GAE-lambda	0.9
Clip range	0.1
Entropy coefficient	0.0005
Batch size	1000
Number of environments	16
Total timesteps	110000000
Steps per environment	2000
Maximum gradient norm	5
Start standard deviation	1
Evaluation frequency	128000
Number of evaluation environments	16
Evaluation number of steps per environment	2000

TABLE II: Hyperparameters for PPO.

B. Domain adaption

As shown in previous work [22, 23], domain randomization helps closing the sim-to-real-gap, makes the policy more generalizable and enables exploitation of simulator dynamics. Thus, as many aspects as possible are randomized.

Namely, we randomize the robots initial state (roll, pitch and yaw angles, joint position and velocities, as well as linear and angular velocities), the robot attributes (center of mass displacement, mass and inertia of the trunk, foot size and placement, joint friction, joint damping, joint stiffness), terrain coefficients (ground friction, damping and contact stiffness, gravity), and the control parameters (motor strength, joint offsets, action scaling factor and PD gains) if they are included in the respective approach. To simulate noisy sensor signals, we add noise to the observations of joint position and velocity, to the angular velocity of the trunk and to the gravity vector.

C. Rewards

Detailed information on the reward terms is shown in Table III.

Appendix II
Approaches for the implementation of torque control

A. Position-based control

The control function of a PD controller is

$$\tau_{\text{cmd}} = K_{\text{p}}\left(q_{\text{nominal}} + q_{\pi} - q_t\right) + K_{\text{d}}(-\dot{q}_t), \tag{8}$$

where q_{π} describes the scaled policy network sample, q_t the current joint positions and $\dot{q}_t$ the current joint velocities. K_{p} and K_{d} are the proportional and derivative gains of the PD controller. We use a proportional gain of 20, a derivative gain of 0.5 and an action scale of 0.25 as this value proved to form the residual output of the network into physically reasonable working ranges.

Figure 4 illustrates the common position-based approach to policy learning for quadrupedal locomotion.

B. Gravity pre-training: modelling the system dynamics

A well-founded description of the system dynamics, which is derived using Lagrange's principle, can be found in [24]. It can be used in various applications and is kept general, thus it is not limited to quadrupeds. In [25] an operation space control framework for quadrupeds is established, which also allows task prioritisation and provides a handling of contact forces. The bond graph method was used by [26] to derive a model. This model is able to describe the system dynamics in great detail, but places less emphasis on control frames. [15] uses inverse dynamics to predict the contact forces. It presents a compliant combination of a floating control based on inverse dynamics and a predictive force control as well as a trajectory planner.

C. Gravity pre-training: force control torque

The force control term (6) is responsible for maintaining the contact state. It uses the selection matrix S_{c} to select components of the contact force f_{c} that exceed the limits of $f_{\text{c}} \in [f_{\text{c,min}}, f_{\text{c,max}}]$, the selection matrix S_{k} to eliminate virtual joints and the Jacobian for contact positions and orientations

$$\dot{x}_{\text{c}} = J_{\text{c}}\dot{q}. \tag{9}$$

The other forces can be calculated as follows:

$$f_{\text{csd}} = \begin{cases} S_{\text{c}} f_{\text{c,min}} & \text{if } f_{\text{c}} < f_{\text{c,min}}, \\ S_{\text{c}} f_{\text{c,max}} & \text{if } f_{\text{c}} > f_{\text{c,max}}, \\ S_{\text{c}} f_{\text{c}} & \text{else.} \end{cases} \tag{10}$$

and

$$f_{\text{c,task+posture}} = \bar{J}_{\text{c}}^{\text{T}} S_{\text{k}}^{\text{T}} (\Gamma_{\text{task}} + \Gamma_{\text{posture}}) - \mu_{\text{c}} - p_{\text{c}}, \tag{11}$$

where μ_{c} and p_{c} are the projections of Coriolis, centrifugal and gravity forces into contact space.

D. Curriculum pre-training

One common choice to transform the deviation of joint positions from the nominal position into a reward that is larger when we are close to the nominal positions and lower if we are far away is the negative weighted Euclidean norm:

$$\hat{r}_t^{\text{nominal}} = -w_t^{\text{nominal}} \left\| q_t - q_{\text{nominal}} \right\|_2^2 .$$

However, one big problem of this choice is that it generates high cost in the beginning of training where the norm and the weight are high. Because of this, the sum of all terms becomes negative and therefore the overall reward becomes zero after taking the maximum with zero. This causes the gradients to be zero and no training occurs. Without taking the maximum, the reward is negative such that termination (i.e. falling) becomes a rewarding strategy (as describes in the reward section II-D). Therefore we need to find a strategy to keep this reward positive at all times, and in particular in the

Reward term	Expression	Weight
Base linear velocity tracking	$\exp\left(-\gamma^{\text{track}_{xy}} \left\lVert v_{xy}^{\star} - v_{xy,t} \right\rVert_2^2\right)$	$w_t^{\text{track}_{xy}} \cdot \Delta t$
Base angular velocity tracking	$\exp\left(-\gamma^{\text{track}_{\text{angular}}} \left\lVert v_{\text{angular}}^{\star} - v_{\text{angular},t} \right\rVert_2^2\right)$	$w_t^{\text{track}_{\text{angular}}} \cdot \Delta t$
Linear velocity	$\mathbf{v}_z^2$	$-2\Delta t$
Angular velocity	$\lVert \omega_{\text{xy}} \rVert_2^2$	$-0.05\Delta t$
Orientation	$\lVert \mathbf{g}_{\text{projected,xy}} \rVert_2^2$	$-0.2\Delta t$
Joint position limit	$\sum\left(\max(\mathbf{q} - \mathbf{q}_{\max}, 0)\right) - \sum\left(\min(\mathbf{q} - \mathbf{q}_{\min}, 0)\right)$	$-10\Delta t$
Joint Acceleration	$\lVert \frac{\dot{\mathbf{q}}_{\text{last}} - \dot{\mathbf{q}}}{\Delta t} \rVert_2^2$	$-2.5 \cdot 10^{-7}\Delta t$
Joint torque	$\lVert \tau \rVert_2^2$	$-2 \cdot 10^{-4}\Delta t$
Action rate	$\lVert \mathbf{a}_{\text{last}} - \mathbf{a} \rVert_2^2$	$-1 \cdot 10^{-6}\Delta t$
Collision	$\sum_{i=1}^{4}(c_{\text{calf,i}} + c_{\text{thigh,i}}) + c_{\text{trunk}}$ for $c_{\text{calf,i}}, c_{\text{thigh,i}}, c_{\text{trunk}} \in \{0,1\}$	$-1\Delta t$
Walking height	$(q_{\text{z}} - q_{\text{z,nominal}} - z_{\text{terrain}})^2$	$-30\Delta t$
Feet air time	$\sum_{i=1}^{4}(t_{\text{swing,i}} - 0.5)$	$-0.1\Delta t$
Symmetry	$(1 - f_{\text{fr}}) \cdot (1 - f_{\text{fl}}) + (1 - f_{\text{br}}) \cdot (1 - f_{\text{bl}}), f_{\text{ij}} \in F$ $f_{\text{ij}} = \begin{cases} 1, & \text{foot on ground} \\ 0, & \text{foot in air} \end{cases}$	$-0.5\Delta t$

TABLE III: Definitions of the used reward terms. Negative total rewards are clipped to zero. Δt is the inverse of the control frequency, in this case $0.02\,\text{s}$. The hyperparameters used for the tracking reward terms are $w_{\text{start}}^{\text{track}_{x,y}} = 4$, $w_{\text{end}}^{\text{track}_{x,y}} = 2$, $w_{\text{start}}^{\text{track}_{\text{angular}}} = 0.1$, $w_{\text{end}}^{\text{track}_{\text{angular}}} = 1$ and the tracking temperatures are $\gamma^{\text{track}_{xy}} = \gamma^{\text{track}_{\text{angular}}} = 4$. The curriculum starts at $t_{\text{start}} = 0$ and ends at $t_{\text{end}} = 10 \cdot 10^6$. F is the set of robot feet, where subscript $i \in f, l$ denotes front or back leg and subscript $j \in l, r$ denotes left or right leg. Positive weights encourage a certain behaviour while negative weights represent penalties.

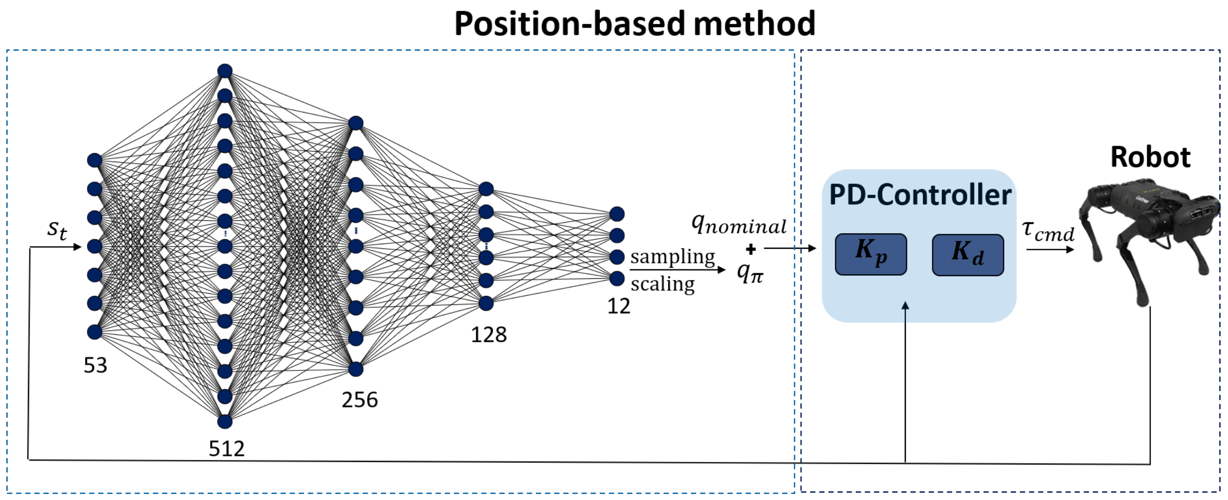


Fig. 4: The common position-based method has two main modules where a neural network outputs joint position residuals which are added up to a nominal position. The target joint positions are then converted into torque commands by a PD controller that can operate at a higher frequency than the network. The setup is used in the traditional position-based approach (section III).

beginning when the norm is the biggest. One choice could be to just add a big constant term c to the reward, e.g. by taking the maximal experimental achieved norm multiplied with the maximal weight:

$$\tilde{r}_t^{\text{nominal}} = \hat{r}_t^{\text{nominal}} + c.$$

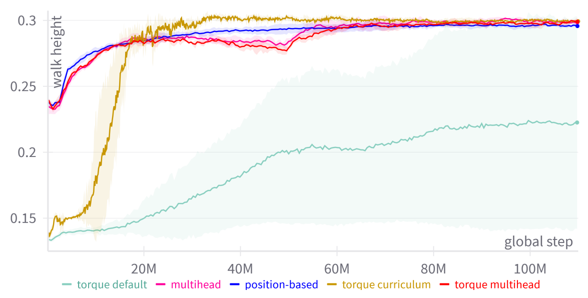
The approach we found most useful is to use an exponential reward term. This directly ensures a lower and upper bound of the reward:

$$r_t^{\text{nominal}} = w_t^{\text{nominal}} \exp\left(-\gamma^{\text{nominal}} \lVert q_t - q_{\text{nominal}} \rVert_2^2\right),$$

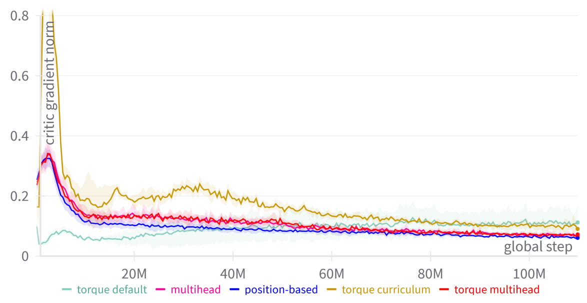
where γ^{nominal} denotes a suitable temperature.

E. Multi-head mixing coefficients

When using the multi-head approach and training with variable coefficients, one has to keep in mind that this creates a dynamic problem for both the PD and torque head, driving the network to adapt continuously. If not managed correctly, it can also introduce instability. Therefore, when employing variable coefficients, careful consideration must be given to the selection of the initial and final values, as well as the timing of when these final values are reached.



(a) Walk heights of experiments with least domain perturbation.



(b) Comparison of the norm of the gradients used to update the critic network.

Fig. 5: Additional metrics for the experiment with the least perturbation.

The updates of these coefficients must be smooth and gradual to ensure that the training process is stable and effective, but also every update means a new problem to the control mechanism. As a consequence, we should always choose a good trade-off between smooth updates and not change the problem too frequently.

Therefore setting the curriculum duration sufficiently long helps in relaxing this trade-off.

Moreover the curriculum duration should should not take up too much time of the total training length to allow the final model to learn and adapt effectively to the fixed final problem.

F. Additional aspects of learning speed

The main problem of the default torque approach is that determining nominal positions as in the position-based approach is difficult. This causes the learning to be slow and inefficient. The approaches we presented in this paper aim to stabilize the robot in the early stages of learning, accelerate initial training and improve the sample efficiency to learn following desired locomotion commands better. Figure 5a shows the different evaluated walking heights of our approaches compared to the position-based baseline and the default torque approach without any additional requirements. As it can be seen, all of our approaches need significantly less time to find an upright walking position than the default torque models which could be the reason that significantly speeds up the training.

In section subsection IV-F, we analyzed the learning curves of the torque default, position-based, torque curriculum, torque multi-head and multi-head, whereby we were able to confirm that indeed our tested approaches learn significantly faster than the default torque-based approach, but the curriculum models learn more slowly than the position-based and multi-head models.

A possible explanation for that could be the curriculum itself, as this models' reward function is dominated by the nominal position reward in the beginning where the weight of the nominal position reward term is high. As we can see in figure Figure 5b high nominal position reward causes a high rise of the critic gradient in the beginning. As a result the value function landscape of PPO seems to be overestimated compared to the latter reward function. After the curriculum has ended the algorithm needs time to update the value function into the direction of the new reward function that is now dominated by the tracking reward. This can be seen in the renewed rise of the value function gradient between the 10 millionth and 35 millionth training step, where at the same time the policy learns how to move into the desired directions (see Figure 3).

Appendix III
Results

To facilitate comparison of the results presented in table I, they were visualized as bar graphs in Figure 6 (low perturbation experiment and high perturbation experiment) and Figure 7 (high frequency experiments).

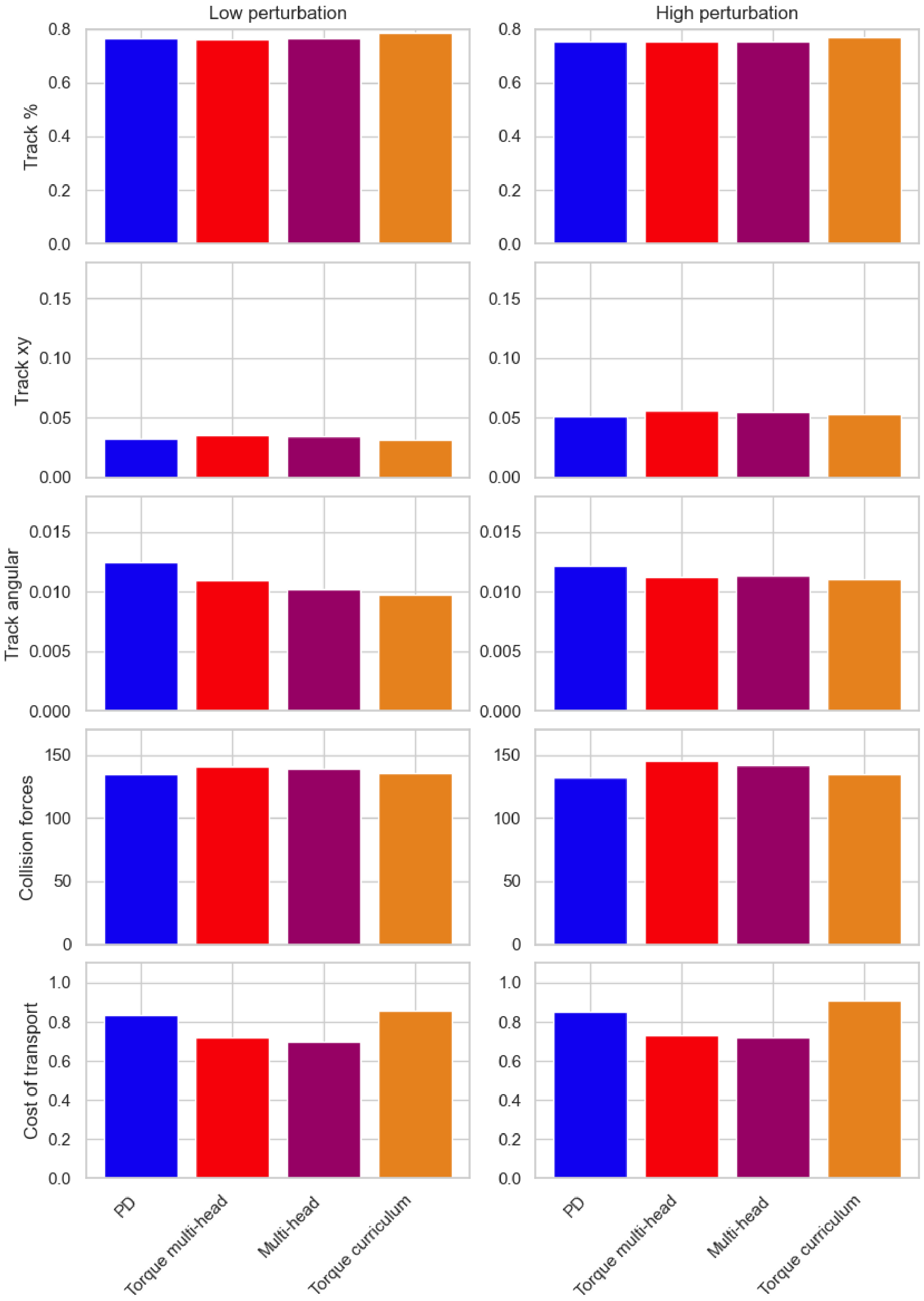


Fig. 6: Results for low and high perturbation from Table I as bar plots. For tracking performances, high values are desirable. For Collision forces and cost of transport, low values are desirable.

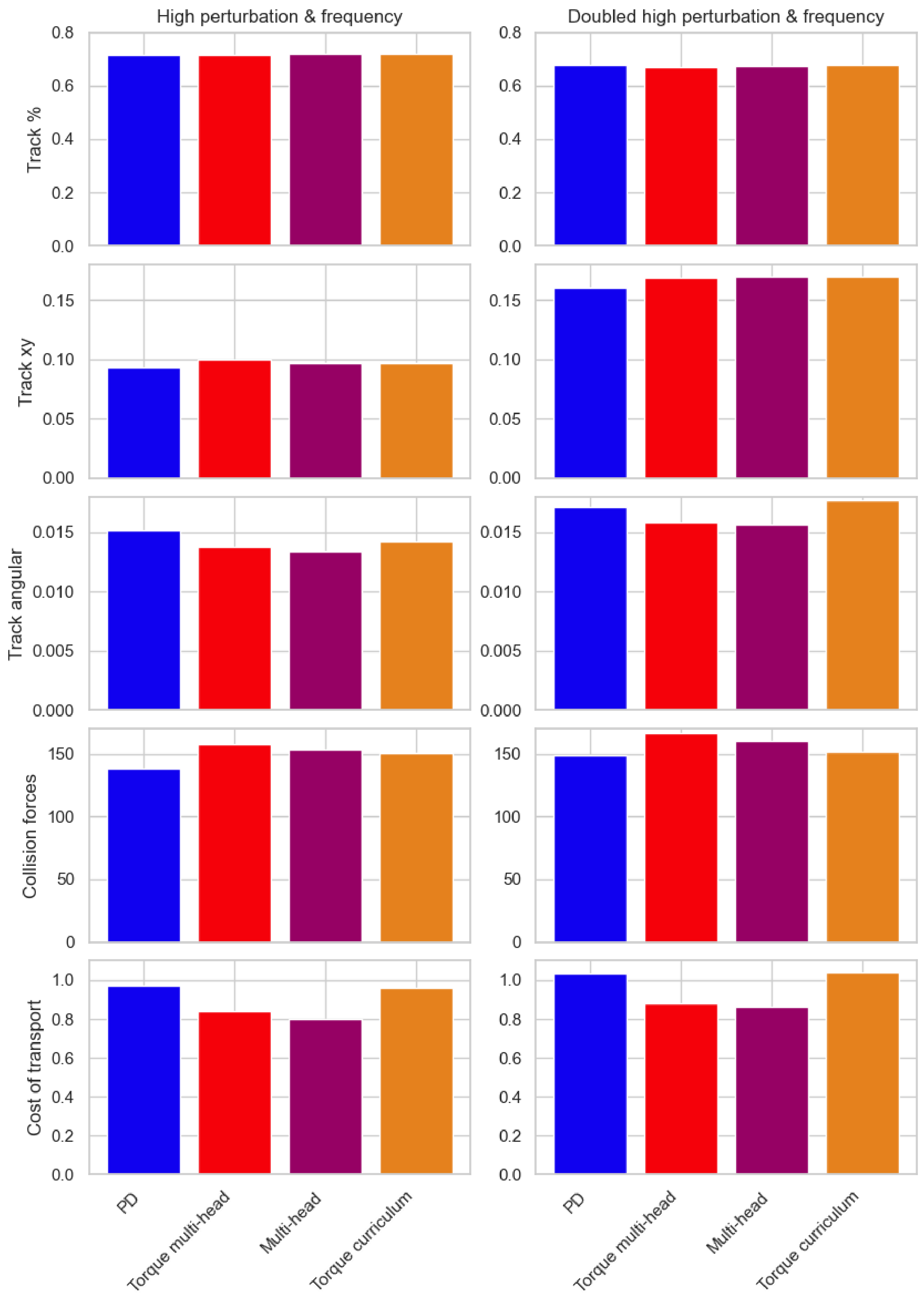


Fig. 7: Results for high perturbation and frequency as well as doubled high perturbation and frequency from Table I as bar plots. For tracking performances, high values are desirable. For Collision forces and cost of transport, low values are desirable.