Student-Teacher Reinforcement Learning with the UnitreeA1 Quadruped

Oliver Grein[1], Oliver Griess[1], Keagan Holmes[1], Nico Bohlinger

***Abstract*— This paper presents an investigation into student-teacher reinforcement learning, a well-established framework in reinforcement learning, for applications in quadrupedal locomotion. Two separate approaches utilizing the reduced computational requirement of training students are highlighted: torque control, which lets the student learn to provide torque directly without the need for a PD-Controller; and depth image transfer learning, showing the possibility for a student to learn to use a depth camera from only ground-truth height samples. Using MuJoCo to simulate the Unitree A1 quadrupedal robot, we demonstrate the commonly observed improved training efficiency and robustness compared to traditional methods. Students using torque control successfully achieved performance similar to the teacher used to train them and outperformed torque control models trained using only base reinforcement learning algorithms at low frequencies (50-200Hz). At higher frequencies, the torque students managed to outperform positional teachers by a wide margin. Information about additional experiments will be added if relevant.**

I. INTRODUCTION

The problem of legged locomotion has proven itself to be a longstanding research domain for reinforcement learning with promising applications. Unlike their wheeled counterparts, legged robots are able to traverse extremely challenging terrain. However, with this ability comes many difficult challenges, such as balance, dynamic stability, and coordination among multiple limbs. To tackle this complexity, current state-of-the-art systems mostly use model-based or learning-based approaches; these systems can be categorized into positional control and torque control. While torque control has been shown to increase robustness and action-tracking performance during real-world deployment, it has also led to significantly increased training times [1]. Student-teacher reinforcement learning, first proposed in 2013 [2], involves one already-trained agent—the *Teacher*—giving "advice" to a currently learning *Student*. As the learning paradigm of student-teacher reinforcement learning has recently gained increased attention [3], [4], [5], this work is evidenced as being highly relevant in the field of robotic locomotion.

[1]Intelligent Autonomous Systems Group, Department of Computer Science, Technische Universität Darmstadt, Karolinenplatz 5, 64289 Darmstadt, Germany {oliver.grein, olivermaximilian.griess, keagan.holmes}@stud.tu-darmstadt.de

This framework aims to tackle various sim-to-real challenges by greatly increasing the robustness of the agent through training the student for the same task as the teacher with **1)** slightly altered environments, **2)** different observations, and/or **3)** different rewards, while also offering greatly reduced computational requirements after an initial teacher has been trained. This work aims to explore the applicability of the student-teacher framework with the novel approach of training a teacher with position control then a student with torque actions, in order to leverage the respective advantages of each method and to balance out the slower learning associated with torque control. Another experiment explores how well a student can learn to use a depth camera on the front of the robot to avoid obstacles and navigate challenging terrain, without any explicit reward given for the behavior. This experiment was selected to gain insight into a model's ability to infer information within this framework, especially in the case that a teacher is given privileged information about the environment, such as ground-truth height samples around the robot, while a student must learn to understand this information through the more limited observations it has available.

II. PRELIMINARIES

A. Unitree A1

A simulated version of the quadrupedal Unitree A1 robot was selected for these experiments due to its robust design and the extensive research that has already been completed using this robot as a testbed. The Unitree A1 is highly mobile, enabling diverse locomotion tasks and motor adaptation experiments, and comes with several built-in sensors and cameras, allowing for diverse experiments across many domains[2]. Its modular design allows for seamless integration of additional sensors, facilitating the implementation of advanced learning algorithms and adaptive control strategies. Previous papers offer additional insight into strengths and challenges, as well as provide a good starting point for selecting weights for the reward functions. Some of these papers are highlighted in more detail in Section III.

[2]Exact specifications can be found at https://www.unitree.com/a1/

B. MuJoCo

For simulating the environment, the MuJoCo physics engine [6] was used—a state-of-the-art solution selected for its high-fidelity physics engine and flexibility in modeling complex dynamic systems. MuJoCo provides an intuitive, well-documented API that allows for (relatively) straightforward implementations of observation and domain randomization functions. In this setup, agents can undergo training and evaluation under diverse conditions, closely mirroring real-world situations. Furthermore, reproducibility and scalability are ensured, as the simulator is entirely deterministic, aiding in algorithmic refinement.

The environment used for training, except where otherwise noted, involves placing the robot on a flat plane and rewarding the agent for walking according to the provided goal velocities. Further details of the reward function are shown in the appendix in Table VII. Many variables are included in the various domain-randomization functions in the codebase, including the initial height of the robot, friction coefficients, and joint stiffness, to name a few. The agents were generally trained with three different seeds.

C. Jax

For implementing the learning algorithms and neural network models, the JAX/Flax framework was utilized [7] - a cutting-edge toolset chosen for its efficiency on modern hardware accelerators for machine learning models. JAX enables highly optimized numerical computations and provides the option of using Flax, a flexible, high-performance neural network library built on top of JAX.

JAX's just-in-time compilation via XLA (Accelerated Linear Algebra) enables up to 12 times more simulation steps per second, significantly improving efficiency compared to the previously used PyTorch implementation.

D. Proximal Policy Optimization

In this work, Proximal Policy Optimization (PPO) has been deployed as the training algorithm for the teacher [8]. PPO stands as the state-of-the-art in reinforcement learning algorithms, known for its efficacy in handling continuous action spaces. The algorithm first estimates the policy gradient based on the previous and current states, then ascends it. Changes larger than a specified value are reduced to this amount to avoid large changes that could destroy the learned policy. The student uses a similar setup, modified to encourage the student to mimic the actions of the teacher.

E. Student-Teacher Reinforcement Learning

In the student-teacher setup, a teacher is first trained to output actions based on the state of the simulation using the PPO algorithm. The student is trained after a sufficient teacher has been developed, with an additional loss function that encourages it to minimize the difference between its own actions and the teacher's proposed "ideal" actions in each state. For real-world applications, the teacher can be trained using privileged information that is only available in the simulated environment, such as exact data about the friction between the feet and ground or ground-truth height samples around the robot. This data is not made available to the student, who must instead infer this information from attainable data. This approach leads to a student that is highly robust and performs exceptionally well in unseen environments; the resulting student can additionally be fine-tuned to further improve performance. Another benefit is that training the student is highly efficient, requiring only a fraction of the time regular training would take to achieve convergence.

III. Related Work

The student-teacher framework was first applied to reinforcement learning in 2013 by Torrey and Taylor [2], who proposed the basic framework with the additional constraint that a teacher has a limited budget for the number of times it can advise the student. Two key findings from their work were very encouraging:

1) Teaching can improve student learning even when agents have different learning algorithms or state representations.
2) Students can benefit from advice even from teachers with less inherent ability, and then go on to outperform their teachers.

Regarding the locomotion task, Miki et al. showed that a policy trained purely in simulation can effectively traverse challenging mountain terrain [4]. Their setup consisted of two steps: firstly, a teacher was trained in simulation with privileged information, including ground-truth height samples around the robot, ground friction, and any introduced disturbances. Next, a student policy was trained to **1)** mimic the teacher's actions without the privileged information, and **2)** predict the unobserved information. The reward function accounted for both the behavior cloning loss and the information reconstruction loss. To highlight the robustness of this setup, the authors mention that the robot "... can continue walking even when mapping fails or the sensors are physically broken", due to the data during training having significant noise, gaps, and/or bias.

Around the same time, Agarwal et al. [9] used a similar student-teacher method with privileged height samples to traverse urban environments, including staircases, stepping stones, and sizable gaps. This study showed that the framework can be comfortably applied to these situations and that hardware onboard

the robots is also a limiting factor for many tasks, rather than just the ability to train a usable policy.

Though outside the scope of our experiments, another research direction around the student-teacher framework involves adding further optimizations or heuristics to the setup in order to allow the student to learn more quickly. One such paper proposed rewarding the teacher for offering good advice to the student [10], which showed a significant improvement in the initial rate at which a student was able to learn. Further research suggested training a teacher with the goal of selecting the most appropriate batch of data to train the student network on, given its state [11], and also showed marginal improvements in training times and resource requirements.

IV. EXPERIMENTS

A. Student-Teacher Setup

The process begins with training a teacher using all available data, followed by a student trained with non-privileged information to act as a control. After this, further experiments can continue, and various students using torque control are trained on the same flat ground.

TABLE I
BASE HYPERPARAMETERS

Parameter	Value
total_timesteps	300,000,000
nr_steps	2,000
nr_envs	16
nr_epochs	5
start_learning_rate	0.0001
start_learning_rate	0.0
policy_nr_hidden_units	“512_256_128”
clip_range	0.1
critic_nr_hidden_units	“512_256_128”
entropy_coef	0.0005
environment_timestep_delta"	0.001
gae_lambda	0.9
gamma	0.99

1) Teacher Setup: The teacher was trained with the PPO algorithm, considering different factors as rewards, such as goal velocity tracking and action rate. The full list of rewards is specified in Table VII in the appendix. The teacher was given the following input features, which we will call the ”privileged information”: the true state of the simulation (including the velocity and position for each joint and the trunk of the robot), the goal velocities (x-velocity, y-velocity, and yaw-velocity), the ground forces calculated by the simulation, the projected gravity vector, the action the agent performed in the previous state, the reward observation, the terrain observation, and the domain randomization observation. Satisfying results were achieved after testing only a few different configurations of the critic and policy network. The teacher was trained to use a control frequency of 50Hz.

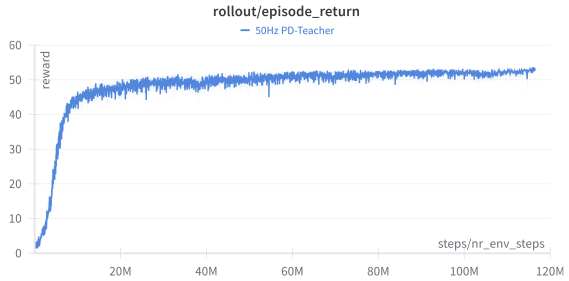

Fig. 1. The mean and min/max reward curve of the PD-Teacher (50Hz control frequency) during a 24-hour training run, showing its learning process. The x-axis displays the training steps (number of environment steps), while the y-axis shows the reward

Figure 1 shows that the teacher performed well after 10M steps but only fully converged after around 60M to 80M steps. The shape of this reward curve has since been shown to be very typical for the learning process of the agent: the agent first learns to stabilize itself, then quickly learns to walk in a way that earns a higher reward than remaining stationary, which occurs within the first few million steps. After this, the agent starts collecting rewards rapidly as it falls less and less and begins to achieve other reward factors, such as the goal velocities. After the first 20M steps, the teacher’s agent has learned to walk following the goal velocities but continues to fine-tune its movements to maximize reward; the gait of the agent undergoes no significant changes thereafter.

2) Student Setup: Every student - whether positional, torque, or with other modifications - and regardless of their frequency was trained using the same teacher trainer at 50Hz with positional control. The student received identical input features as the teacher, but the domain randomization observation was masked to increase robustness for real-world scenarios with limited information access (non-privileged information). Rather than directly maximizing the reward function, the student’s objective was to minimize the disparity between its actions and those of the privileged teacher. Identical hyperparameters were used for both the student and teacher to rule out any performance discrepancies due to hyperparameters. The comparison of actions was calculated using the mean squared error as can be seen below. as denotes the actions of the student while at denotes the actions of the teacher.

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(as_i - at_i)^2$$

To later enable fine-tuning via PPO for the student, a second term was added to the loss function to also train a critic network for the student. cs denotes the prediction of the student’s critic, while ct denotes the

teacher's critic prediction. The full loss function was defined as follows:

$$MSE = \frac{1}{n}\sum_{i=0}^{n}(as_i - at_i)^2 + (cs_i - ct_i)^2$$

B. Student Torque Control Setup

The goal of this experiment was to train a student that directly outputs motor torques instead of target positions. Previously, both the student and the teacher were trained to output positional information, which was then fed to a PD-Controller to compute the respective torques. Instead of directly using the teacher's output as feedback, the outputs were first converted to torques before being fed to the student for the loss function. The conversion function is denoted as $conv$ in the following formula.

$$MSE_{torque} = \frac{1}{n}\sum_{i=0}^{n}(as_i - conv(at_i))^2 + (cs_i - ct_i)^2$$

When training students with frequencies higher than the teacher's original frequency of 50Hz, the converted actions of the teacher were used multiple times instead of using any form of interpolation. Multiple versions of sampling the teacher at higher frequencies were tested via hyperparameter optimization using Optuna [12], but using the teacher at its original frequency yielded the best results (see Figure 2). The plot shows the average reward curve during hyperparameter optimization for different sampling frequencies of the teacher. The parameter "teacher action stretch factor" denotes how much the original action of the teacher needs to be "stretched" to match the frequency of the student. If the factor is 1, the action is fully "stretched" to match the student's frequency; for example, even if the student operates at 1000Hz, the teacher would be sampled every 50Hz. The student would then receive the same teacher sample for the loss function 20 times before the teacher is sampled again. Conversely, a teacher action stretch factor of 0.5 would mean a sampling frequency of 500Hz for the teacher, reusing the same sample 2 times.

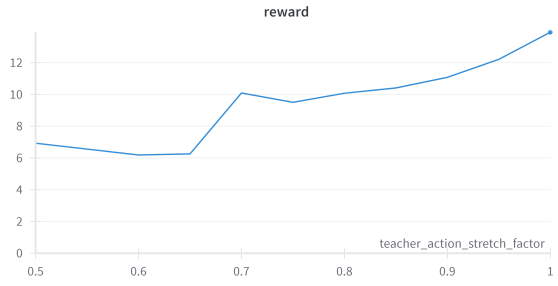

Fig. 2. The amount of steps the teacher action was stretched for and the respective reward during hyperparameter optimization. The student trained was running on 1000Hz. The x-axis displays the action stretch factor, while the y-axis shows the reward

For the final training of torque students, the same hyperparameters used during the teacher's training were applied. The training process was repeated multiple times for different control frequencies: the first experiment was run at the teacher's native frequency (50Hz) and was then increased by 50Hz for each subsequent experiment, up to the simulation frequency of 200Hz. At this point, both the simulation frequency and the control frequency of the agent were increased to 500Hz and 1000Hz, respectively.

C. Additional Experiments

To further test the limits of student-teacher learning and to better understand its benefits and drawbacks, two additional experiments were conducted:

- Speed curriculum experiments: Testing students and teachers at multiple different speeds
- Knowledge Distillation experiments: Testing students and teachers with smaller networks

1) Speed Curriculum: To further investigate the benefits of student-teacher reinforcement learning at different frequencies, a more challenging task in the form of a speed curriculum was implemented. Every time the agent reaches the predetermined reward threshold of 40 over the last 25 rollouts during training, the maximum speed commands are increased. This approach helps the agent converge to a maximum speed that it can learn with the current setup. First, a set of teachers was trained using the described curriculum to walk forward only. Two different training approaches were then used for the torque students: one based on torque control and the other on PD control. In the first approach, the students were trained using teachers who had already undergone training with the speed curriculum. Subsequently, the students underwent additional fine-tuning with the speed curriculum. Due to time constraints, fine-tuning was only applied to the positional students and a single torque control model.

2) Knowledge Distillation: Student-teacher learning has been utilized in various domains for knowledge distillation and model compression. This experiment investigates the potential of knowledge distillation for quadrupedal locomotion. This is particularly relevant for torque control, as inference times could become a bottleneck for control frequency in real-world robots with limited computational power. To compare the performance of student-teacher learning with PPO, a set of teachers and a set of students with different model architectures (256-256; 128-128; 64-64; 32-32; 16-16) were trained. Both the teachers and the students were trained at 50Hz using PD control to ensure their performance was comparable.

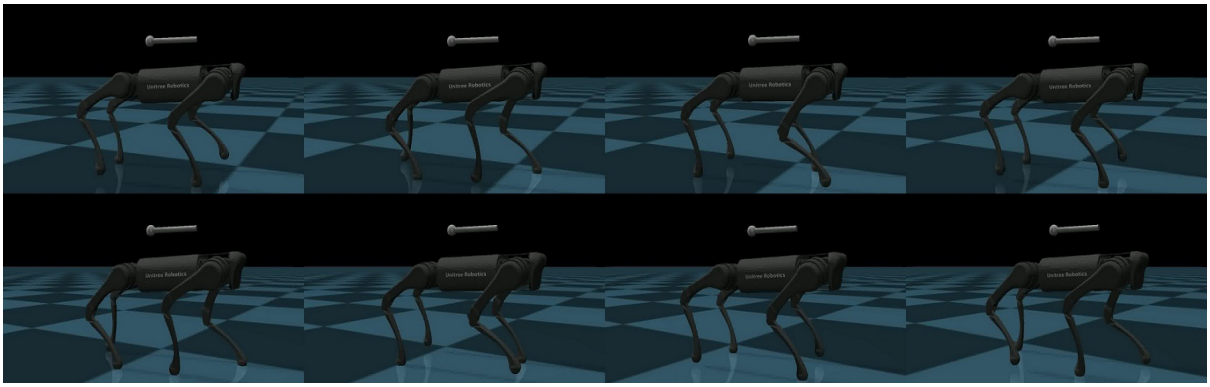

Fig. 3. A robot with a good gait. This one takes medium-sized steps but has a clean and stable gait. The sequence shows the robot walking without encountering any issues.

V. EVALUATION

A. Hardware

For training all the agents, a significant number of CPU hours were required. Most models were trained on a CPU cluster fitted with `AMD Ryzen 9 3950X 16-Core` processors. Though exact statistics were not tracked, a conservative estimate would be 12,000 hours of compute time on all 16 cores of one CPU. As previously mentioned, one of the biggest constraints of reinforcement learning is how expensive training the models is, especially with limited options to optimize reward functions and hyperparameters.

B. Evaluation Criteria

1) Test Environment: To get a final evaluation on the performance of an agent a seperate test environment was used to score the agent over 10 episodes. The test environment itself behaves the same as the test environment, but the key difference is that during training with PPO, the actions of the agent are sampled from a probability distribution for exploration. During testing only the mean action from that distribution is used, simulating the deployment of the agent, resulting in better rewards during training for agents that are trained via PPO (teachers, fine-tuned agents). Therefore especially when comparing the numeric evaluation criteria between students and teachers a test environment is used.

C. Eye-Test

When evaluating a model's performance, solely relying on rewards can be enough to get a rough estimate, but it is necessary to give a model the "eye test" to get the full picture. For this, a model is set into a rendered version of the environment and evaluated with both random and manually chosen goal velocities and rotations. The gait of the agent is then manually observed and assessed with respect to how natural the gait is and if there may be any flaws that are not penalized by the reward function. The results of this "eye test" are classified with respect to the following 4 categories:

Bad gait: A model that at most manages to take a few steps forward before collapsing. The episode reward is less than ten. A model like this is unusable and indicates that there was an issue with the training setup or the agent is simply not able to learn the given task in with the used learning framework.

Mediocre gait: A model that struggles with changing directions rapidly and frequently falls when randomizing movement directions. When walking straight it stumbles occasionally but always catches itself. Step size varies between small and medium and rewards fall into a range of ten to thirty. Achieving a model like this with a novel approach like high-frequency torque control. Some more fine-tuning is needed.

Good gait: A model that only falls when switching to a high velocity in the opposite direction. It stumbles occasionally but manages to catch itself in most cases. It takes medium-sized steps and has an episode reward of twenty to fifty. Only very minor imperfections can be spotted when the model is walking straight. A model like this is what the experiments aim for since it is good enough and can be further improved with fine-tuning.

Perfect gait: A model that never falls and only stumbles under extreme circumstances. It takes medium-sized to big steps and usually has an episode reward of fifty or higher. When walking straight it does not make any mistakes. A model like this would be ideal but is very hard to achieve.

D. Basic Teacher

Teachers were trained at frequencies between 50Hz and 1000Hz. Teachers trained at 500Hz or higher struggled to learn a smooth gait and can be categorized as mediocre to bad. While the 500Hz agent was able to move around and follow the goal velocities, though in rare cases falling and with very small steps,

the 1000Hz agent only learned to stablize itself in weird positions, sometimes follow the goal velocities, but mostly falling when trying to move. The 200Hz teacher reaches rewards of over 40 and a good gait after around 20M steps. Teachers with control frequencies under 200Hz performed similarly to each other with slightly increased time to convergence with increasing frequency. All of them reach rewards close to 50 and a perfect gait. The results show the increase in complexity due to higher control frequencies.

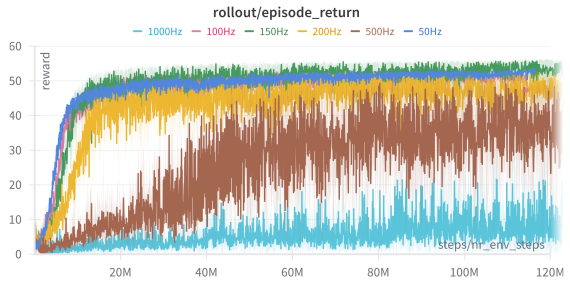

Fig. 4. The mean and min/max (shades) reward curve for various PD-Teachers on different control frequencies, during a 24-hour training run, showing their learning process. The x-axis displays the training steps (number of environment steps), while the y-axis shows the reward

E. Basic Student

When training positional students at different control frequencies, it becomes apparent that the training via student-teacher learning results in a faster learning curve, especially in higher control frequencies (Figure 5). Furthermore a significant improvement of the gait and the reward for the 500Hz and 1000Hz agents can be observed. While there was a significant dropoff in performance for the PD-Teachers, the learning curve for the PD-Students only shows a very small decrease in performance when compared to the lower frequency agents. Table IV and table III show a full comparison of the PD-Teachers and PD-Students based on their performance in the test environment.

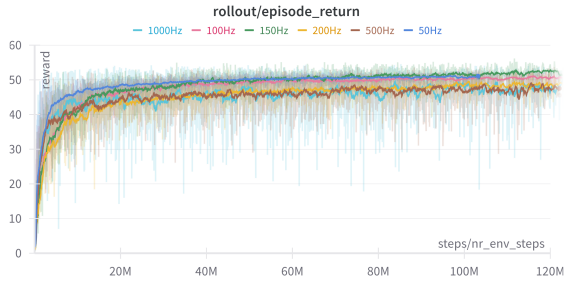

Fig. 5. The PD-control student's reward curve during a 24-hour training run, showing its learning process. The x-axis displays the number of environment steps, while the y-axis shows the reward. The reward curve is smoothed with a factor of 0.8 to better visualize the learning process.

TABLE II
PERFORMANCE PD-TEACHERS DURING 10 EPISODES IN TEST ENVIRONMENT

Control Frequency	Avg. Reward	Reward Span	Eye-Test
50Hz	54.9	53-56	perfect
100Hz	54.5	54-56	perfect
150Hz	57.6	56-59	perfect
200Hz	45.3	18-56	good
500Hz	50.8	37-55	mediocre
1000Hz	10.3	1-38	perfect

TABLE III
PERFORMANCE PD-STUDENTS DURING 10 EPISODES IN TEST ENVIRONMENT

Control Frequency	Avg. Reward	Reward Span	Eye-Test
50Hz	51.9	50-53	perfect
100Hz	52.7	50-55	perfect
150Hz	54.9	53-57	perfect
200Hz	51.8	49-54	perfect
500Hz	52.4	50-54	perfect
1000Hz	53.1	49-65	good

F. Torque Student

The students displayed a very similar smoothed learning curve, with all converging around a reward of approximately 40. The agents showed the most improvement in the first 20 million steps, with full convergence occurring around 120 million steps. Visual inspection revealed that all students had good to perfect gaits, although lower control frequencies tended to make the agents more unstable. This instability is reflected in the reward curves such as 50Hz and 100Hz. In contrast, students with control frequencies of 500Hz or 1000Hz exhibited a very stable reward curve and behavior during testing.

The benefits of student-teacher learning do not become apparent here since the torque student at this point is only trained to imitate the PD-teacher's actions.

TABLE IV
PERFORMANCE TORQUE-STUDENTS DURING 10 EPISODES IN TEST ENVIRONMENT

Control Frequency	Avg. Reward	Reward Span	Eye-Test
50Hz	44.2	38-50	mediocre
100Hz	48.9	43-51	good
150Hz	49.9	53-52	good
200Hz	51.3	47-53	perfect
500Hz	51.2	46-53	good
1000Hz	53	48-56	perfect

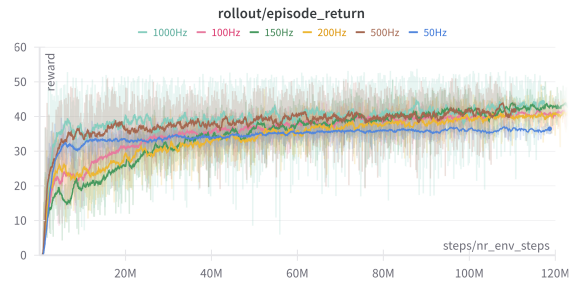

Fig. 6. The torque student's reward curves during a 24-hour training run, on different control frequencies (50Hz, 100Hz, 150Hz, 200Hz, 500Hz, 1000Hz), showing its learning process. The x-axis displays the number of environment steps, while the y-axis shows the reward. Smoothing factor: 0.8

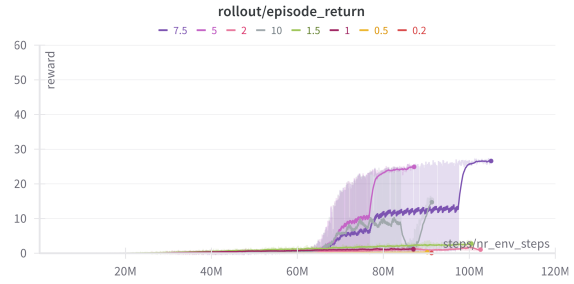

Fig. 7. The torque teacher's reward curve for directly learning torque control (50Hz control frequency) on various standard deviations (ranging from 0.2 to 10). Curves are smoothed for better visualization (0.8 factor)

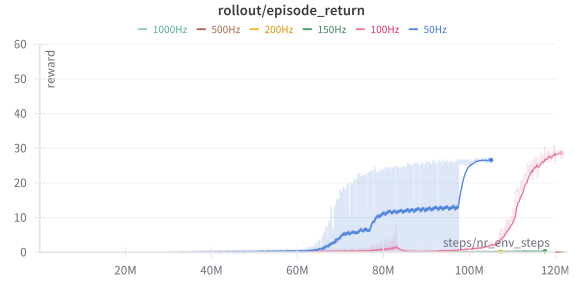

Fig. 8. The torque teacher's reward curve for directly learning torque control (7.5 standard deviation) on different control frequencies (50Hz, 100Hz, 150Hz, 200Hz, 500Hz, 1000Hz). Curves are smoothed for better visualization (0.8 factor)

G. Additional Experiments

1) Speed Curriculum: Figure 9 illustrates how the maximum speed the student has to achieve increases over time. Due to the reward threshold imposed by the curriculum, the teacher needs to reach a reward of 40 to increase the speed in one of its 16 environments. This leads to the teacher being able to reliably being able to achieve the speeds that the curriculum is setting. Using a lower reward lead to the teacher "cheating" by just standing still when a command was to difficult and still managing to get a small reward that managed to push it over the lower thresholds occasionally.

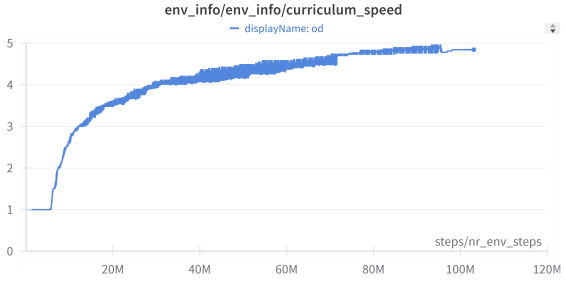

Fig. 9. The teacher's max curriculum speed curves during a 24-hour training run. The x-axis displays the number of environment steps, while the y-axis shows the maximum speed the agent is tasked with.

Training students using this teacher and the same curriculum worked mostly fine, but some configuration issues led to the high frequency (500,1000hz) not being trained successfully. 15 and 16 show, that pd students were able to reach a lot higher curriculum speed in their training runs than torque students did. This difference is more extreme than in the previous section since the best torque students outperformed by a factor greater than 2 when compared to the best pos student. The torque students' performance increases with increased frequency while the pos students perform better at lower frequencies.

Due to time constraints and a lack of space on the cluster we only managed to fine-tune the positional students and a single (3 seeds at 1000hz) torque student. Finetuning the torque student did not lead to any increase in speed. It made the model worse. Due to the limited amount of time and space on the cluster better hyperparameters for torque finetuning could not be found in time. The PD students however greatly benefited from finetuning across the board. This led to the finetuned 50hz student even outperforming the original teacher. The maximum speed the models were able to reach under test conditions can be seen in Table VIII in the Appendix.

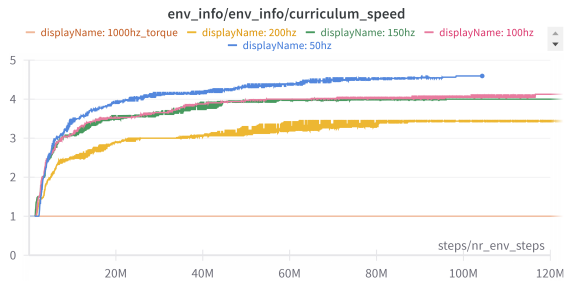

Fig. 10. All finetuning runs for the max curriculum speed curves during a 24-hour training run. The x-axis displays the number of environment steps, while the y-axis shows the maximum speed the agent is tasked with.

2) Knowledge Distillation: The Knowledge Distillation experiments show mixed results. As illustrated in Figure 11, PD-control students converge significantly faster. Although the convergence time increased as the neural network architecture became smaller, the PD-students still took roughly half the time of the respective PD-teachers to converge. However, as the network size decreased, the stability of the rewards for the PD-students slightly decreased, while the rewards for the PD-teachers remained stable. Except for the 16-16 architecture, both approaches achieved similar maximum reward convergence. For the 16-16 architecture, however, the PD-teacher significantly outperformed the PD-student, which is due to one of the three seeds that had significantly worse performance for that students than the other two. Visual inspection and scoring in the test environment, also confirm the observations made from the reward curve and can be seen in table V and table VI.

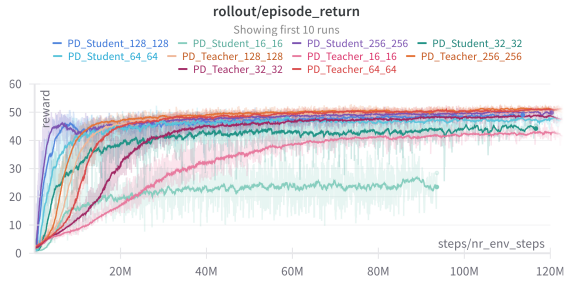

Fig. 11. The PD-student's/teacher's reward curves for different network architectures, during a 24-hour training run, showing their learning process. The x-axis displays the number of environment steps, while the y-axis shows the reward.

TABLE V

KNOWLEDGE DISTILLATION: PERFORMANCE PD-TEACHERS DURING 10 EPISODES IN TEST ENVIRONMENT

Architecture	Avg. Reward	Reward Span	Eye-Test
16-16	49.8	45-52	good
32-32	53.2	52-55	perfect
64-64	54.2	53-55	perfect
128-128	54	53-55	perfect
256-256	54.6	53-55	perfect

TABLE VI

KNOWLEDGE DISTILLATION: PERFORMANCE PD-STUDENTS DURING 10 EPISODES IN TEST ENVIRONMENT

Architecture	Avg. Reward	Reward Span	Eye-Test
16-16	44.6 (good seed)	16-53 (good seed)	bad/good[1]
32-32	49.6	45-54	good
64-64	49.9	48-52	perfect
128-128	50.7	49-53	perfect
256-256	51.2	49-53	perfect

VI. DISCUSSION AND FUTURE WORK

The work conducted so far supports the claim that the student-teacher framework is advantageous for similar (not just identical) tasks, however it is yet to be researched quantitatively how different these domains can be while the framework is still beneficial. Due to the common ground that all locomotive tasks have - that is, the movement of the limbs and torques/positions of the feet relative to the ground - it is very likely that initializing an agent with advice from a teacher with knowledge about how the robot moves will be beneficial to initially learning any locomotion task, regardless of environmental factors such as the terrain or external perturbations. For domains that are vastly different, it might still be advantageous to start with a very strong weight for the teacher's advice then reduce it after the agent has learned how actions influence the observations in the joint space, eventually eliminating this reward such that the student can effectively learn for its own task.

a) Torque Control Student: The novel approach of training students that directly control the robot's torque instead of only providing positions that are converted by a PD-Controller was a success. Multiple students of different frequencies were trained. The best of these students managed to get close to the quality of the teachers' gait without any fine-tuning. And even managed to outperform both teachers and positional students at high Frequencies (500Hz+). Furthermore, the experiments outlined that combining torque control with student-teacher learning could counteract the longer training times of torque control without a drop-off in performance.

[1] seed-dependent

b) Computer Vision Student: Additional experiments were conducted with an agent that was given observations from a depth camera placed on the front of the robot, in hopes of showing that the student-teacher framework can offer major advantages in terms of training times, as rendering the depth image in the simulator drastically reduces the achievable number of simulation steps per second. Due to issues with training adequate teachers, these results are excluded from this paper. Experiments were conducted on difficult terrain, including an exceptionally rough terrain and an inverted pyramid, such that the advantage of the additional depth observation and privileged height samples around the robot would be most valuable. The experimental setup involves training four teachers, with and without a depth camera and with and without the privileged height samples, then training students based off of these teachers with and without a depth camera. The resulting training times can then be compared to show the advantages of the framework on efficiency, as well as the maximum step (difficulty) height achieved on the inverted pyramid environment to gauge the robustness of each student, especially compared to its teacher.

c) Additional Experiments: Using student-teacher learning for training students at higher speeds than just 1 m/s was a success. The PD student managed to outperform its teacher after finetuning. Due to time limitations, torque students could not be fully evaluated with finetuning so they fell behind. Increased frequency did not help PD control models deal with the higher speeds, but the torque model seems to benefit from the increase in control frequency.

The Knowledge Distillation experiments show mixed results. PD-control students converged significantly, converging in about half the time of their PD-teachers. However, as network sizes decreased, the PD-students showed slightly less reward stability, while PD-teachers remained more consistent. Both approaches achieved similar maximum rewards, except for the 16-16 architecture, where a PD-student underperformed due to seed variance. Visual inspections confirmed these findings, though the lack of finetuning for torque students due to time constraints may have affected the overall results.

A. Future Work

Any ongoing experiment that remains unfinished when the final report is due will be added to this section, as well as thoughts about future work that came up during the additional experiments.

a) sim-to-real application: Since all the findings of this study have been based on simulations a verification of the results by deploying the learned policies on a real robot should be done. This way the robustness of the torque control student in a real-world scenario could be investigated similar to Chen et al. [1] in their study.

b) Teaching on a Budget: Currently the teacher's advice is available to the student throughout the training process, however it would be interesting to recreate the student-teacher setup in Torrey and Taylor's original paper [2] where the teacher is only able to advise the student a limited number of times, to see if similar results are produced from this more complex task and get more quantitative values for how much a teacher's advice can improve a student's learning. Four different advising schemes were highlighted, detailed below:

Early Advising The teacher simply uses its entire budget in the initial steps.

Importance Advising States are evaluated based on an importance metric, and advice is only offered if the metric is above a threshold.

Mistake Correcting Advice is given when the student's action is different from the teacher's and the importance metric is above a threshold.

Predictive Advising An additional classifier is trained to predict the student's actions, which is used by the teacher to decide when to offer advice, instead of just observing the student's past actions. Different predicted and ideal actions, as well as a high enough importance are both prerequisites for giving advice.

The paper found that Mistake Correcting generally offered the greatest improvement, however all variations led to much faster convergence.

VII. CONCLUSIONS

In conclusion, this work highlights both the theoretical and practical advantages of student-teacher reinforcement learning, particularly in reducing training time and enhancing model robustness. The experiments demonstrate that learning under less ideal conditions, such as with torque control, can be significantly improved by leveraging the supervised nature of the student's loss function and achieving faster convergence. The findings underscore the potential of student-teacher learning in robotic tasks and reveal that at higher frequencies, combining torque control with student-teacher learning can produce effective students. Any additional relevant results from ongoing experiments will be included in the final report.

APPENDIX

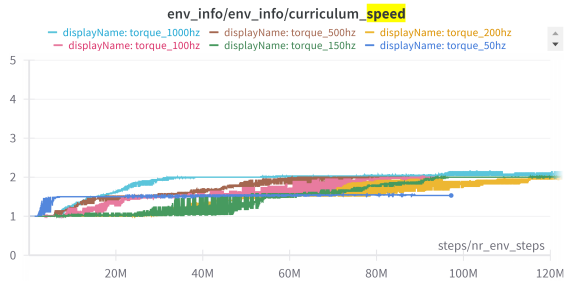

Fig. 15. The torque Students max curriculum speed curves during a 24-hour training run. The x-axis displays the number of environment steps, while the y-axis shows the maximum speed the agent is tasked with.

TABLE VII

REWARD FUNCTION PARAMETERS

Name in the Codebase	**What is Rewarded**
`track_xy_vel_cmd`	Forward movement
`track_yaw_vel_cmd`	Alignment with the command velocity
`linear_velocity`	Matching the linear command velocity
`angular_velocity`	Matching the angular command velocity
`joint_position_limit`	Keeping the joints within a specified range
`torque`	Using less torque
`acceleration`	Limiting changes in velocity
`velocity`	Limiting the joint velocity
`action_rate`	Limiting changes to the current action, compared to the previous
`collision`	Fewer bodies in contact with the terrain
`base_height`	Maintaining a specified height above the ground
`air_time`	Keeping each foot off the ground

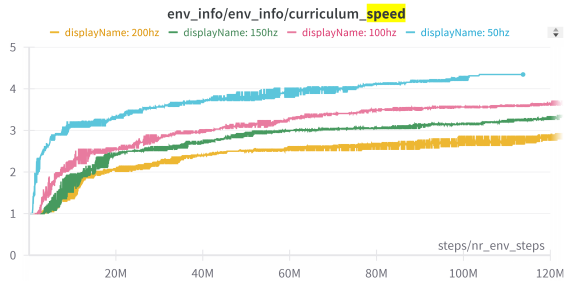

Fig. 16. The pd Students max curriculum speed curves during a 24-hour training run. The x-axis displays the number of environment steps, while the y-axis shows the maximum speed the agent is tasked with.

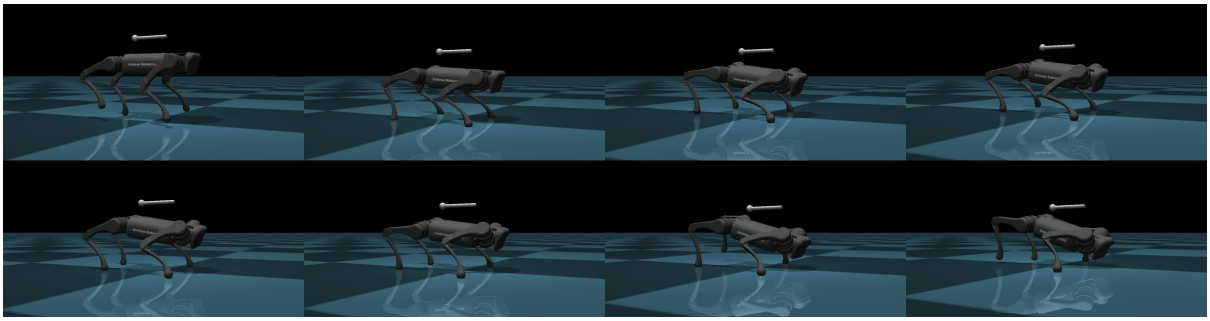

Fig. 12. A robot with a bad gait. This one stumbles and falls within taking its first few steps. The sequence shows the robot stumbling and falling.

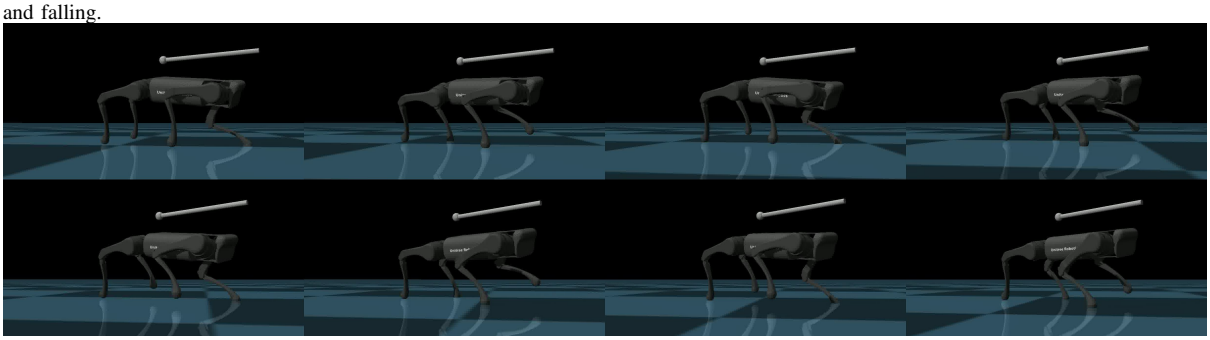

Fig. 13. A robot with a mediocre gait. This one takes big steps but struggles to maintain its direction. The sequence shows the moment where it stumbles but manages to catch itself. This leaves it facing a slightly different direction.

Fig. 14. The crouched position learned by the teacher with the depth image.

TABLE VIII

MAX SPEED ALL MODELS COULD ACHIEVE WITH A GAIT THAT WAS AT LEAST GOOD.

Model	**Max Speed Back**	**Max Speed Forward**
Teacher	-5	4.8
pd50Hz	-4.6	4.5
pd100Hz	-4.5	4.4
pd150Hz	-4.4	4.2
pd200Hz	-4.3	4
torque50Hz	-2	1.4
torque100Hz	-2.3	1.9
torque150Hz	-2.5	2.1
torque200Hz	-2	1.7
torque500Hz	-2.2	2.1
torque1000Hz	-3.2	2.4
pdfinetuned50Hz	-6	6
pdfinetuned100Hz	-4.7	4.3
pdfinetuned150Hz	-4.5	4.2
pdfinetuned200Hz	-4	3.4
torquefinetuned1000Hz	-1.8	1.5

ACKNOWLEDGMENT

Thanks to Nico Bohlinger for providing the initial codebase and for his continuous support during the course of this research.

$$L_{\odot} = \frac{-1}{I \times T^2} \sum_{t=1}^{T} \sum_{q=1}^{T} \sum_{i=1}^{N} \log \left(\frac{\exp(\langle \mu_i(t), \mu_i(q) \rangle / \tau)}{\sum_{j=1}^{N} \exp(\langle \mu_i(t), \mu_j(q) \rangle / \tau)} \right)$$

References

[1] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath, *Learning torque control for quadrupedal locomotion*, 2023. arXiv: 2203.05194 [cs.RO].

[2] L. Torrey and M. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 1053–1060.

[3] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, Oct. 2020, ISSN: 2470-9476. DOI: 10.1126/scirobotics.abc5986 [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abc5986

[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, eabk2822, 2022.

[5] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, *Rapid locomotion via reinforcement learning*, 2022. arXiv: 2205.02824 [cs.RO].

[6] 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. DOI: 10.1109/IROS.2012.6386109

[7] J. Bradbury et al., *JAX: Composable transformations of Python+NumPy programs*, version 0.3.13, 2018. [Online]. Available: http://github.com/google/jax

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[9] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on Robot Learning*, PMLR, 2023, pp. 403–415.

[10] M. Zimmer, P. Viappiani, and P. Weng, "Teacher-student framework: A reinforcement learning approach," in *AAMAS Workshop Autonomous Robots and Multirobot Systems*, 2014.

[11] R. El-Bouri, D. Eyre, P. Watkinson, T. Zhu, and D. Clifton, "Student-teacher curriculum learning via reinforcement learning: Predicting hospital inpatient admission location," in *International Conference on Machine Learning*, PMLR, 2020, pp. 2848–2857.

[12] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, *Optuna: A next-generation hyperparameter optimization framework*, 2019. arXiv: 1907.10902 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1907.10902