2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Advanced Skills through Multiple Adversarial Motion Priors in Reinforcement Learning

Eric Vollenweider, Marko Bjelonic, Victor Klemm, Nikita Rudin, Joonho Lee and Marco Hutter

***Abstract*— Reinforcement learning (RL) has emerged as a powerful approach for locomotion control of highly articulated robotic systems. However, one major challenge is the tedious process of tuning the reward function to achieve the desired motion style. To address this issue, imitation learning approaches such as adversarial motion priors have been proposed, which encourage a pre-defined motion style. In this work, we present an approach to enhance the concept of adversarial motion prior-based RL, allowing for multiple, discretely switchable motion styles. Our approach demonstrates that multiple styles and skills can be learned simultaneously without significant performance differences, even in combination with motion data-free skills. We conducted several real-world experiments using a wheeled-legged robot to validate our approach. The experiments involved learning skills from existing RL controllers and trajectory optimization, such as ducking and walking, as well as novel skills, such as switching between a quadrupedal and humanoid configuration. For the latter skill, the robot was required to stand up, navigate on two wheels, and sit down. Instead of manually tuning the sit-down motion, we found that a reverse playback of the stand-up movement helped the robot discover feasible sit-down behaviors and avoided the need for tedious reward function tuning.**

I. INTRODUCTION

Reinforcement Learning (RL) has greatly impacted legged locomotion by producing robust policies capable of navigating challenging terrains in the real world [1]. With this progress, we believe that articulated robots can perform specialized motions similar to their natural counterparts. Our goal is to further push the limits of these robots by executing advanced skills, such as the *quadruped-humanoid transformer* shown in Fig.1, performed by our wheeled-legged robot [2]. To achieve these skills, we combine motion priors with RL.

A. Related Work

Executing specific behaviors for a real robot is a fundamental challenge in robotics and RL. For example, the computer animation community synthesizes life-like behaviors from human or animal demonstrations for their simulated agents. Boston Dynamic's real humanoid robot, *Atlas*, shows impressive dancing motions and backflips based on human

Fig. 1. Quadruped-humanoid transformer (https://youtu.be/kEdr0ARq48A) with a time-lapse from left to right of a stand-up and sit-down motion (top image), obstacle negotiation (middle image), and indoor navigation (bottom images). The former skill and the humanoid navigation on two legs are achieved through traditional RL training with a task reward formulation. Instead of tuning the sit-down skill, we can reverse the playback of the stand-up motion and use it as a motion prior that helps the robot discover feasible sit-down behaviors avoiding tedious reward function tuning.

This work was supported in part by armasuisse W&T and the Swiss National Science Foundation (SNF) through the National Centres of Competence in Research Robotics (NCCR Robotics) and Digital Fabrication (NCCR dfab). Besides, it has been conducted as part of ANYmal Research, a community to advance legged robotics.

All authors are with the Robotic Systems Lab, ETH Zürich, 8092 Zürich, Switzerland.

ericvol@microsoft.com, marko.bjelonic@mavt.ethz.ch, victor.klemm@mavt.ethz.ch, nikita.rudin@mavt.ethz.ch, joonho.lee@mavt.ethz.ch

motion animators. Similarly, our wheeled-legged robot can track motions from an offline trajectory optimization with an model predictive control (MPC) algorithm, as shown in our previous work [3]. Furthermore, motion optimizations, such as [4], [5], have the added benefit of producing physically plausible motions, which is favorable in computer graphics but vital in robot control. However, designing objective functions is usually exceptionally difficult. Furthermore, these tracking-based methods require carefully designed objective functions. When applied to more extensive and diverse motion libraries, these methods need heuristics to select the suitable motion prior to the scenario.

Data-driven strategies like [6] automate the imitation objective and mechanisms for motion selection based on adversarial imitation learning. This paper verifies that this imitation learning approach can be applied to physics-based models and not just computer animations. Gaussian processes [7], [8] can learn a low-dimensional motion embedding space generating suitable kinematic motions when provided with a relatively large amount of motion data. However, the approaches are not goal conditioned and can not leverage task-specific information.

Animation techniques [9]–[11] attempt to solve this by imitating/tracking motion clips. This is usually implemented with pose errors, requiring a motion clip selection and synchronizing the selected reference motion and the policy's movement. By using a phase variable as an additional input to the policy, the right frame in the motion data-set can be selected. It can be challenging to scale the number of motion clips with these approaches. Defining error metrics that generalize to a wide variety of motions is difficult.

Two alternative approaches are adversarial learning and student-teacher architectures [12]. The latter trains a teacher policy with privileged information such as perfect knowledge about the height map, friction coefficients, and ground contact forces. With that, the teacher can learn complex motions more easily. After the teacher's training, the student policy learns to reproduce the teacher's output using non-privileged observations and the robot's proprioceptive history. Hereby, a style transfer from teacher to student is happening. On the other hand, adversarial imitation learning techniques [13], [14] and more recently [6] build upon a different approach. The latter offers a discriminator-based learning strategy called Adversarial Motion Priors (AMP), which outsources the error-metrics, phase, and motion-clip selection to a discriminator which learns to distinguish between the policy's and motion data's state transitions. *AMP* does not require specific motion clips to be selected as tracking targets since the policy automatically chooses which style from the motion data to apply given a particular task. The method's limitation is that whenever multiple provided motion data-sets cover the same movements, the policy might either go for the more straightforward style to fulfill or find a hybrid motion similar to both motion clips. In other words, there is no option of actively choosing between multiple styles in single or multi-task settings. Furthermore, the task-reward still has to motivate the policy to execute a specific movement

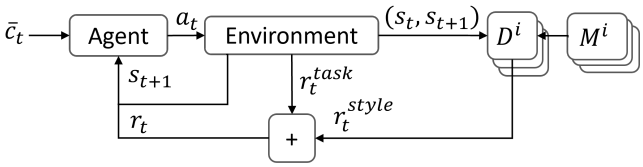

Fig. 2. Multi-AMP overview: The discriminator predicts a style reward s_t^{style} which is high if the policy's behavior is similar to the motions of the motion database M^i, by distinguishing between state transitions (s_t, s_{t+1}) of both sources. The style reward is added to the task reward, which finally leads to the policy fulfilling the task while applying the motion data's style.

because otherwise the policy might identify two states and oscillate between them. Generally, to our experience, it is not trivial to find task-reward formulations for complex and highly dynamic movements that do not conflict with the style reward provided by the discriminator.

B. Contribution

This paper introduces the Multi-AMP algorithm and applies it to our real wheeled-legged robot. Like its AMP predecessor [6], this approach automates the imitation objective and motion selection process without heuristics. Furthermore, our extension allows for the intentional switching of multiple different style objectives by changing flags in the command input. The approach can imitate motion priors from three different data sets, i.e., from existing RL controllers, trajectory optimization, and reverse stand-up motions. The latter enables the automatic discovery of feasible sit-down motions on the real robot without tedious reward function tuning. This permits exceptional skills with our wheeled-legged robot in Fig. 1, where the robot can switch between a quadruped and humanoid configuration. To the best of our knowledge, this is the first time such a highly dynamic skill is shown and also the first time that the AMP approach is verified on a real robot.

II. MULTIPLE ADVERSARIAL MOTION PRIORS

In this work, the goal is to train a policy π capable of executing multiple tasks, including styles extracted from n individual motion data-sets $M^i, i \in \{0, ..., n-1\}$ with the ability to actively switch between them. In contrast to tracking-based methods, the policy should not blindly follow specific motions but rather extract and apply the underlying characteristics of the movements while fulfilling its task.

Similar to the AMP algorithm [6], we split the reward calculation into two parts $r_t = r_t^{task} + r_t^{style}$. The task-reward is a description of *what* to do, e.g., velocity tracking, and the style-reward r_t^{style} defines *how* to do it, namely by extracting and applying the style of the motion data. While task rewards often have simple mathematical descriptions, the style reward is not trivial to calculate. In the following, we introduce *Multi-AMP*, a generalization of *AMP* which allows for switching of multiple different style-rewards, which constitutes the main theoretical contribution of this work.

A style reward motivates the agent to extract the motion data's style. We use an adversarial setup with n discriminators $D^i, i \in \{0, ..., n-1\}$. For every trained style

i, a roll-out buffer B_π^i collects the states of time-steps where the policy applies the i^{th} style, and another buffer M^i contains the motion data to that specific style. Each discriminator D^i learns to differentiate between descriptors built from a pair of consecutive states (s_t, s_{t+1}) sampled from M^i and B_π^i . Thus, every trainable style is defined by a tuple $\{D^i, B_\pi^i, M^i\}$. By avoiding any dependency on the source's actions, the pipeline can process data of sources with unknown actions, such as data from motion-tracking and character animation. The discriminator D^i learns to predict the difference between random samples of its motion database M^i , and the agent's transitions sampled from the style's roll-out buffer B_π^i by scoring them with $+1$ and -1, respectively. This behavior is encouraged by solving the least-squares problem [6] defined by

$$\begin{aligned} L^i = \; & \mathbb{E}_{d^{M^i}(s,s')} \left[(D^i(\phi(s), \phi(s')) - 1)^2 \right] \\ & + \mathbb{E}_{d^{B_\pi^i}(s,s')} \left[(D^i(\phi(s), \phi(s')) + 1)^2 \right] \\ & + \frac{w^{gp}}{2} \mathbb{E}_{d^{M^i}(s,s')} \left[\| \nabla_\phi D^i(\phi)|_{\phi=(\phi(s),\phi(s'))} \|^2 \right], \end{aligned} \tag{1}$$

where $d^{M^i}(s, s')$ and $d^{B_\pi^i}(s, s')$ denote the likelihoods of descriptors build from state-transitions (s, s') under the respective distributions and w^{gp} is a weighting constant.

The descriptors are built by concatenating the output of an arbitrary function $\phi(\cdot) : \mathbb{R}^{d_s} \mapsto \mathbb{R}^{d_d}$ for two consecutive states, whereby the design of ϕ decides which style information is extracted from the state-transitions, e.g., the robot's joint and torso position, velocity, etc. d_d and d_s denote the dimensionalities of descriptors and states respectively.

A. Task-reward

Our agents interact with the environment in a command-conditioned framework. During the policy's roll-out, the robot's state s_t is passed to the policy at every time-step t. This state s_t contains a command c_t sampled from a command distribution $p(c)$. As in the standard RL-cycle, after the policy $\pi(a_t|s_t)$ predicts an action a_t, the environment returns a new state s_{t+1} and a task-reward r_t^{task}, which rewards the policy for fulfilling commands c_t. For example, the task might be to achieve the desired body velocity sampled from a uniform distribution in x, y and yaw coordinates. The task is included in the policy's observation and essentially informs the agent what to do. The task reward depends on the performance of the policy with respect to the command $r_t^{task} = R(c_t, s_t, s_{t-1})$

B. Style-reward

During the policy's roll-out only one style is active at a time. The command c_t contained in the policy's observation is augmented with a one-hot-encoded style selector c_s, i.e., the elements of c_s are zero everywhere except at the index of the active style i. After the environment step, the latest state-transition (s_t, s_{t+1}) is used to construct the style-descriptor $d_t = [\phi(s_t), \phi(s_{t+1})] \in \mathbb{R}^{2d_d}$, which is mapped to a style-reward $r_t^{style} \in \mathbb{R}^+$ using the current style's discriminator D^i and the style-reward given by

$$r_t^{style} = -log \left(1 - \frac{1}{1 + \exp^{-D^i([\phi(s_t), \phi(s_{t+1})])}} \right). \tag{2}$$

C. Multi-AMP algorithm

The sum of the style and task rewards $r_t = r_t^{task} + r_t^{style}$ constitutes the overall reward, which can be used in any RL algorithm such as Proximal Policy Optimization (PPO) [15] or Soft Actor Critic (SAC) [16]. The state s_t is additionally stored in the style's roll-out buffer B_π^i to train the discriminator at the end of the epoch. The full approach is shown in the following algorithm:

Require: $M = \{M_i\}, |M| = n$ (n motion data-sets)
$\pi \leftarrow$ initialize policy
$V \leftarrow$ initialize Value function
$[\mathcal{B}] \leftarrow$ initialize n style replay buffers
$[D] \leftarrow$ initialize n discriminators
$\mathcal{R} \leftarrow$ initialize main replay buffers
while not done **do**
  for trajectory i = 1, ..., m **do**
    $\tau^i \leftarrow \{(c_t, c_s, s_t, a_t, r_t^G)_{t=0}^{T-1}, s_T, g\}$ roll-out with π
    $d \leftarrow$ style-index of τ^i (encoded in c_s)
    if M^d is not empty **then**
      for t = 0, ..., T-1 **do**
        $d_t \leftarrow D^d(\phi(s_t), \phi(s_{t+1}))$
        $r_t^{style} \leftarrow$ according to Eq. 2
        record r_t^{style} in τ^i
      end for
      store d_t in $\mathcal{B}^d$ and τ_i in $\mathcal{R}$
    end if
  end for
  for update step = 1, ..., $n_{updates}$ **do**
    for d = 0, ..., n **do**
      $b^{\mathcal{M}} \leftarrow$ sample batch of K transitions $\{s_j, s'_j\}_{j=1}^K$ from $\mathcal{M}^d$
      $b^{\pi} \leftarrow$ sample batch of K transitions $\{s_j, s'_j\}_{j=1}^K$ from $\mathcal{B}^d$
      update D^d according to Eq. 1
    end for
  end for
  update V and π (standard PPO step using $\mathcal{R}$)
end while

D. Data-free tasks

If no motion data is present for a desired task and it should nevertheless be trained alongside multiple styles extracted from motion data, Multi-AMP can be adapted slightly. While the policy learns the motion data free task, r_t^{style} is set to 0. Thereby, the data-free style is still treated as a valid style and present in the one-hot-encoded style-selector c_s, but the policy π is not guided by the style-reward.

III. EXPERIMENTAL RESULTS AND DISCUSSION

We implement and deploy the proposed Multi-AMP framework on our wheeled-legged robot in Fig. 1 with 16 DOF (degrees of freedom). The training environment consists of three tasks, two of which are supported by motion

TABLE I

TASK-REWARDS.

All tasks	formula	weight
r_τ	$\|\tau\|^2$	-0.0001
$r_{\dot{q}}$	$\|\dot{q}\|^2$	-0.0001
$r_{\ddot{q}}$	$\|\ddot{q}\|^2$	-0.0001
4-legged locomotion		
$r_{lin\ vel}$	$e^{\|\dot{x}_{target,\ xy} - \dot{x}\|^2/0.25}$	1.5
$r_{ang\ vel}$	$e^{\|\omega_{target,\ z} - \omega\|^2/0.25}$	1.5
Ducking		
r_{duck}	$e^{0.8*\|x_{goal} - x\|}$	2
Stand-up	see Tab. II	

TABLE II

REWARDS FOR AOW STANDING UP, SITTING DOWN, AND NAVIGATING WHILE STANDING

symbols	description	
$q^{robot} \in \mathbb{H}$	Robot base-frame rotation	
$p^{robot} \in \mathbb{R}^3$	Robot base-frame position	
q	Joint DOF positions (excl. wheels)	
q_{hl}	Hind-Leg DOF position	
α	$\angle$(robot-x axis, world z axis)	
f	Feet on ground (binary)	
s	Standing robots (binary)	
stand-up	formula	weight
r_α	$\frac{\pi/2-\alpha}{\pi/2}$	2
r_{height}	p_z^{robot}	3
r_{feet}	f	-2
r_{wheels}	$\sum \dot{q}^2_{front\ wheels} * (1-f)$	-0.003
$r_{shoulder}$	$\|q_{shoulder}\|^2$	-1
$r_{stand\ pose}$	$exp(-0.1 * \|q_{hl} - q_{0,\ hl}\|^2)$	1
sit-down		weight
$r_{un-stand}$	$max(\frac{\pi/2-\alpha}{\pi/2} * 3, 0)$	-3
$r_{sit-down}$	$\frac{min(\alpha, \pi/2)}{\pi/2}$	2.65
$r_{dof\ vel}$	$\|\dot{q}\|^2$	-0.015
$r_{dof\ pos}$	$\exp(-0.5 * \|q_0 - q\|^2) * \frac{\alpha}{\pi/2}$	3
navigation		weight
$r_{track\ lin}$	$exp(-4 * \|\dot{x}_{des} + \dot{p}^{robot}_{local,z}\|^2) * s$	2
$r_{track\ ang}$	$exp(-4 * \|\omega_{des} - \omega^{robot}_{local,x}\|^2) * s$	2

data, and one is a partly data-decoupled task, meaning that there are no style-rewards for a portion of the task. The first task is four-legged locomotion, the motion data of which was sourced from another RL policy (Fig. 3 top left). The second task is to duck under a table. The motion data for this style was generated by a trajectory optimization pipeline, which was deployed and tracked by an MPC controller [3] (Fig. 3 bottom left). The last task is partly data-decoupled. Here, the wheeled-legged robot learns to stand up on its hind legs followed by two-legged navigation (Fig. 4), before sitting down again. The sit-down training is supported by motion data as detailed in Section III-B. The task of standing up and sitting down is from now on referred to as the "stand-up task". A video available at https://youtu.be/kEdr0ARq48A showing the results accompanies this paper. We believe this task-selection shows that Multi-AMP can achieve dynamic skills, which require diverse sequences of motions to achieve, while partly covering the same command space.

The training environment of our Multi-AMP pipelines is implemented using the *Isaac Gym simulator* [17], [18], which allows for massively parallel simulation. We spawn 4096 environments in parallel to learn all three tasks simultaneously in a single neural network. The number of environments per task is weighted according to their approximate difficulty, e.g., $[1, 1, 5]$ in the case of the tasks described above (While important, the training is not very sensitive to this weighting). The state-transitions collected during the roll-outs of these environments are mapped using a function $\phi(s)$ such that it extracts the linear and angular base velocity, gravity direction in base frame, the base's height above ground, joint position and velocity, and finally the position of the wheels relative to the robot's base-frame, i.e., $\phi(s) = (\dot{x}_{base}, x_z, e_{base}, q, \dot{q}, x_{ee,base}) \in \mathbb{R}^{50}$. The task reward definitions for the three tasks are in Table I and II.

A. Experiments

Due to the problem of catastrophic forgetting [19]–[21], we learn these tasks in parallel. This section analyzes the task performance of each Multi-AMP policy compared to policies that exclusively learn a single task (baseline). The three tasks (standing up, ducking, and four-legged locomotion) are trained in different combinations, where ducking and walking are always learned with motion data and stand-up only partly with motion data during the sit-down phase:

1) Stand up only
2) Duck only
3) Walk only
4) Walking and standing up
5) Walking and ducking
6) Walking, ducking, and standing up

First, we compare the learning performance of the stand-up task between the models Nr. 1, 4, and 6. We normalize all rewards in Fig. 6 with the number of robots receiving the reward, making the plots comparable between the experiments. Fig. 6 shows important metrics of the stand-up learning progress. The figure shows that the policy does not compromise during the training of multiple tasks compared to single-task settings. The policy learning three tasks simultaneously (*3 styles* in Fig. 6) performs equally well while standing up and sitting down. While it takes the *3 style* policy a bit longer to reach the maximum rewards (see r_{stand} and $r_{stand\ track\ ang\ vel}$ at epoch 1000), the differences vanish after sufficiently long training times. In this case, it takes Multi-AMP about 300 epochs longer to reach the maximum task rewards compared to the single task policy.

The walking and ducking tasks perform similarly, with the specialized policies (model Nr. 2 and 3 in the aforementioned list) reaching a similar final performance compared to the others. All policies manage to extract the walking and ducking style such that no visible difference can be seen.

In summary, in this specific implementation of the environment and selection of tasks, Multi-AMP, while taking longer, learns to achieve all goals equally well as more specialized policies that learn fewer tasks.

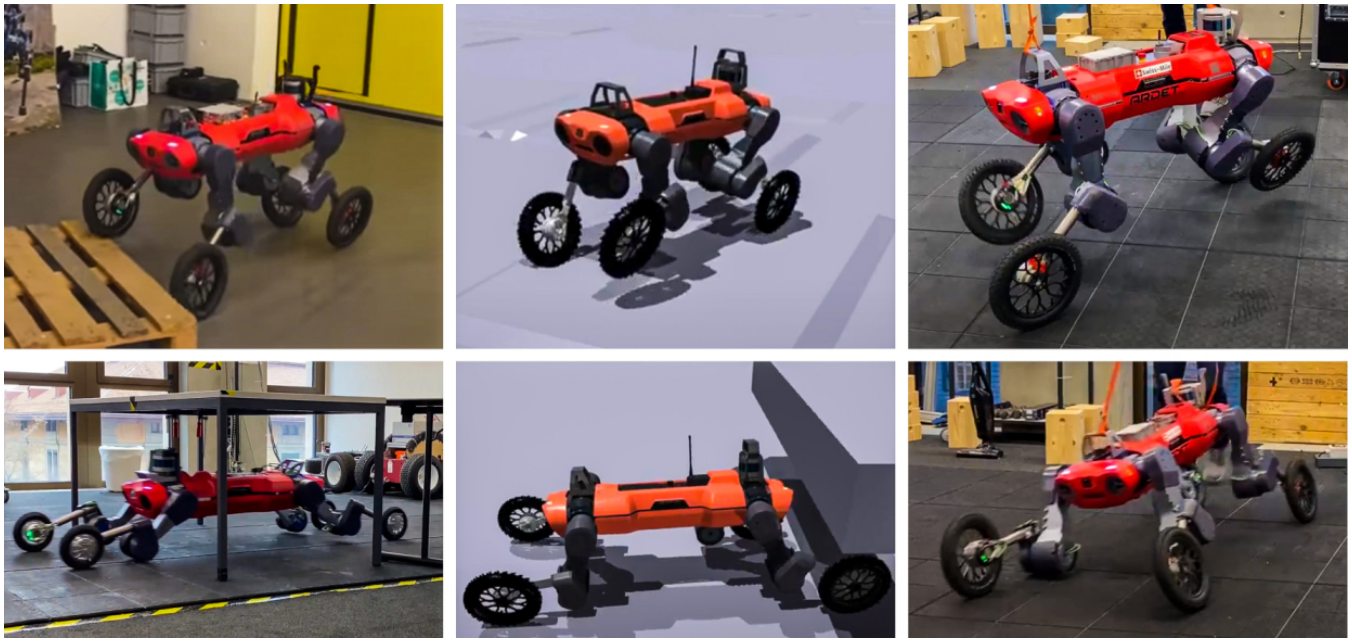

Fig. 3. Four-legged locomotion (top row) and ducking motion (bottom row) of the motion data source (left column), simulation training (center column), and final deployment on the real robot using Multi-AMP. The former task is trained with motion data from a different simulation environment and control approach, while the latter is trained with data from trajectory optimization [3].

Fig. 4. Stand up-sequence in simulation and on the real robot. The policy is able to stand up, navigate large distances on two legs, and finally sit down again using motion data from the reversed stand-up sequence.

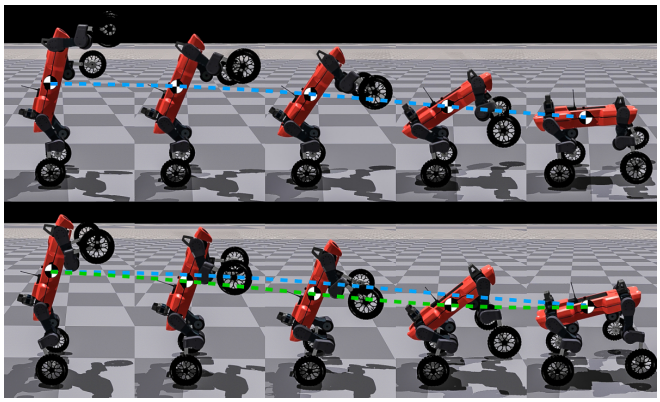

Fig. 5. Comparison of the sitting down motions. Top row: If the agent learns to sit down with task rewards only, it falls forward with extended front legs, which causes high impacts and leads to over-torque on the real robot. Marked in blue is the trajectory of the center of gravity of the base. Bottom row: When sitting down with task reward and style reward from the reversed stand-up sequence, the robot squats down to lower its center of gravity before tilting forward, thereby reducing the impact's magnitude. Marked in green is the trajectory of the center of gravity of the base.

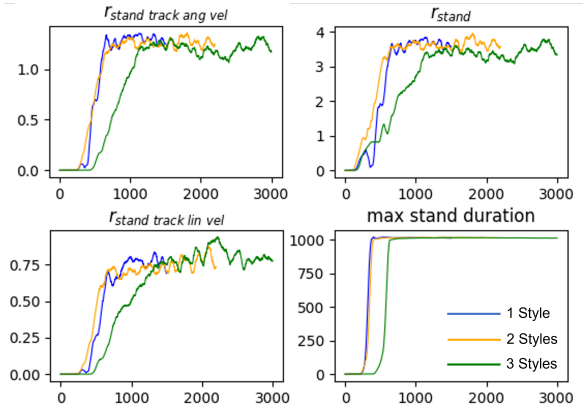

Fig. 6. Multi-AMP learning capability of the stand-up task. The horizontal axis denotes the number of epochs, and the vertical axis represents the rewards after post-processing for comparability. Furthermore, the maximum stand duration is plotted over the number of epochs.

B. Sit-down training

While the sit-down rewards presented in Table II work well in simulation, the policy's sit-down motions created high impulses in the real robot's knees, which exceeded the robot's safety torque threshold. To easily perform more gentle sit-down motions and avoid reward function tuning, we recorded the stand-up motion, reversed the motion data, and trained a policy using Multi-AMP. As this motion starts with a front end-effector velocity of 0 when lifting them off the ground, the reversed style should encourage low impact sit-down motions. In the Multi-AMP combination, one style contains the reversed motion data for sitting down, while the second style receives plain stand-up rewards. The result is a sit-down motion that uses its hind knees to lower the center of gravity before tilting the base and catching itself on four legs, as shown in Fig. 5. The agent receives zero task rewards for a predefined time after the command to sit down, avoiding task rewards that conflict with the sit-down motion-prior, since rewarding horizontal body orientation leads the agent to accelerate the sit-down, which breaks the style. After this buffer-time, the sit-down task-rewards become active. This allows the robot to sit down with its own speed and style, and guarantees non-conflicting rewards.

C. Remarks

Finding a balance between training the policy and the discriminators is vital during the Multi-AMP training process. Our observations show that fast or slow training of the discriminators relative to the policy hampers the policy's style training. In our current implementation, the number of discriminator and policy updates is fixed, which might not be an optimal strategy. Since the setup is very similar to Generative Adversarial Network (GAN), more ideas from [22] could be incorporated into Multi-AMP.

We use an actuator model for the leg joints to bridge the sim-to-real gap [23] while an actuator model is not needed for the velocity controlled wheels. Moreover, we apply strategies to increase the policy's robustness, such as rough terrain training (see rough terrain robustness in Fig. 1), random disturbances, and game inspired curriculum training [18]. The highly dynamic stand-up task is especially prone to these robustness measures, which we solve by introducing timed pushes and joint-velocity-based trajectory termination. The former identifies the most critical movement-phase and pushes the policy in the worst way. This increases the number of disturbances the policy experiences during these critical phases, rendering it more robust, which also helps with sim-to-real efforts. Furthermore, by terminating the trajectory if the joint velocity of any DOF exceeds the actuator's limits, the policy learns to keep a safety tolerance to these limits.

IV. CONCLUSIONS

This work introduces Multi-AMP, with which we automate the imitation objective and motion selection process of multiple motion priors without heuristics. Our experimental section shows that we can simultaneously learn different styles in a single policy. Furthermore, our approach can intentionally switch between them, whereby also data-free styles are supported. Various multi-style policies are successfully deployed on a wheeled-legged robot. To this end, we show different combinations of skills such as walking, ducking, standing up on the hind legs, navigating on two wheels, and sitting down on all four legs again. We avoid tedious reward function tuning by training the sit-down motions with motion data gained from reversing a stand-up recording. Furthermore, we note that similar performances as in the single-style case can be expected even when learning multiple styles simultaneously. We conclude that Multi-AMP and its predecessor AMP [6] are promising steps towards a possible future without style-reward function tuning in RL. However, even though less time is invested in tuning reward functions, more time is required to generate motion priors, which is in most cases not available for specific tasks.

To the best of our knowledge, this is the first time that a quadruped-humanoid transformation is shown on a real robot, challenging how we categorize multi-legged robots. Over the next few years, this skill will further expand the possibilities of wheeled quadrupeds by opening doors, grabbing packages, and many more use-cases.

REFERENCES

[1] 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, 2022.

[2] M. Bjelonic, R. Grandia, O. Harley, C. Galliard, S. Zimmermann, and M. Hutter, "Whole-Body MPC and Online Gait Sequence Generation for Wheeled-Legged Robots," in *under review for IEEE Int. Conf. on Robotics and Automation*, 2021.

[3] M. Bjelonic, R. Grandia, M. Geilinger, O. Harley, V. S. Medeiros, V. Pajovic, S. Edo, Jelavic Coros, and M. Hutter, "Complex motion decomposition: combining offline motion libraries with online MPC," *under review for The International Journal of Robotics Research*, 2022.

[4] M. H. Raibert and J. K. Hodgins, "Animation of dynamic legged locomotion," vol. 25, no. 4, 1991. [Online]. Available: https://doi.org/10.1145/127719.122755

[5] K. Wampler, Z. Popović, and J. Popović, "Generalizing locomotion style to new animals with inverse optimal regression," vol. 33, no. 4, 2014. [Online]. Available: https://doi.org/10.1145/2601097.2601192

[6] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–20, 2021.

[7] S. Levine, J. M. Wang, A. Haraux, Z. Popović, and V. Koltun, "Continuous character control with low-dimensional embeddings," vol. 31, no. 4, 2012. [Online]. Available: https://doi.org/10.1145/2185520.2185524

[8] Y. Ye and C. K. Liu, "Synthesis of responsive motion using a dynamic model," *Computer Graphics Forum*, vol. 29, no. 2, pp. 555–562. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01625.x

[9] V. B. Zordan and J. K. Hodgins, "Motion capture-driven simulations that hit and react." New York, NY, USA: Association for Computing Machinery, 2002. [Online]. Available: https://doi.org/10.1145/545261.545276

[10] N. Chentanez, M. Müller, M. Macklin, V. Makoviychuk, and S. Jeschke, "Physics-based motion capture imitation with deep reinforcement learning," 11 2018, pp. 1–10.

[11] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 143:1–143:14, July 2018. [Online]. Available: http://doi.acm.org/10.1145/3197517.3201311

[12] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, p. eabc5986, 2020. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abc5986

[13] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning." New York, NY, USA: Association for Computing Machinery, 2004. [Online]. Available: https://doi.org/10.1145/1015330.1015430

[14] J. Ho and S. Ermon, "Generative adversarial imitation learning," 2016.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[16] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018.

[17] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State, "Isaac gym: High performance GPU based physics simulation for robot learning," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. [Online]. Available: https://openreview.net/forum?id=fgFBtYgJQX_

[18] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," 2021.

[19] C. Atkinson, B. McCane, L. Szymanski, and A. V. Robins, "Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting," *CoRR*, vol. abs/1812.02464, 2018. [Online]. Available: http://arxiv.org/abs/1812.02464

[20] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017. [Online]. Available: https://www.pnas.org/content/114/13/3521

[21] P. Kaushik, A. Gain, A. Kortylewski, and A. L. Yuille, "Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping," *CoRR*, vol. abs/2102.11343, 2021. [Online]. Available: https://arxiv.org/abs/2102.11343

[22] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *CoRR*, vol. abs/1606.03498, 2016. [Online]. Available: http://arxiv.org/abs/1606.03498

[23] 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, p. eaau5872, 2019. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.aau5872