Robot-to-Robot transfer for Quadrupedal Locomotion

Nicholas Ioannidis
Department of Computer Science
University of British Columbia
Vancouver, Canada

Xindong Lin
Department of Computer Science
University of British Columbia
Vancouver, Canada

Matthew Tang
Department of Computer Science
University of British Columbia
Vancouver, Canada

***Abstract*—Deep Reinforcement Learning (DRL) has significantly advanced the field of robot control, empowering robots to learn how to perform dynamic tasks by using simulated environments. Core to these algorithms is the reward function which takes considerable effort to design, is task-specific, and under-constraint due to the high dimensionality of the motion. This often results in robots learning implausible and over-energetic motions that are nontransferable to real robots. Imitation learning is successful in addressing this problem. However, the reference motion trajectory in imitation learning is both expensive and hard to collect. To bridge this data gap, we show in this work that one can utilize available data from open-source robotic platforms to learn a policy that imitates its motion on new robots. To show the feasibility of this approach, we aim to train a policy for Solo-8(1), an 8 DoF robot, using trajectories generated(2) from Unitree Go1, a 12 DoF robot. We later show that this framework can generalize to other quadruped robots, and finally, we propose a novel PCA reward term that captures global similarities between reference and generated motion.**

I. Introduction

The field of robotics has truly benefited from the advances of Deep Reinforcement Learning (RL). Traditional methods such as Model Predictive Control typically rely on crude approximations of the robotic hardware struggle to perform in more dynamic environments and/or challenging terrains. RL instead has benefited from advances in physics simulators and deep learning. The physics simulators allow the RL agent to generate cheap data while exploring state space while parameterizing the control policy with a lightweight neural network architecture making the model capable of implicitly learning the dynamics of any environment.

An important challenge with RL remains to be the design of the reward function. The reward function is typically hand-engineered making it tedious to tune, additionally expressing a task as a reward function is not always simple, it can suffer from sparse feedback, or result in degenerate behaviors. Addressing this issue imitation-based approaches (3; 4) use simplified or task-relevant trajectories of the robot and reward the agent proportionally to how close the learned motion resembles the reference. With recent collective efforts on creating generalized datasets(5) for different robotic platforms, we show in this work that imitation-based approaches can be very powerful when paired with RL and can learn locomotion tasks across different robot morphologies. We focus our work on transferring motions(2) performed from the Unitree GO1, a 12 Degree of Freedom(DoF) quadruped robot, to the Solo8, an 8-DoF quadruped. We later show that this framework can generalize to other quadrupeds. Finally, we propose a novel reward term using PCAs as a way to capture global correlations between the motions.

II. Related Work

Developing controllers for high-dimensional robot motions has been studied for many decades. Many early methods focused on trajectory optimization algorithms to achieve realistic motions but the solutions were highly constrained to specific tasks and were not well generalized to different environments. Recently, many deep Reinforcement Learning (RL) methods have shown state-of-the-art performance on the problem of robot control and have allowed researchers to perform more dynamic tasks in highly non-linear systems.

However, many of these RL methods work well in only simulations and are difficult to transfer to real robots due to the aggressive and overly energetic behaviors learned from under-specified reward functions. One fundamental problem in designing reward functions that work well in real robots is that a significant amount of engineering effort is required so that dense and meaningful feedback can be provided to the agent. This makes the reproducibility of learned motions and tasks very challenging among different research labs especially when the variations of the robotic hardware are not trivial.

Some more recent work attempted to simplify the reward function by learning a control policy that imitates a reference motion generated via a simplified model of the robot through trajectory optimization (3) or by using partial demonstrations (6).(4) uses the Adversarial Motion Priors (AMP) framework (7) that uses an imitation reward via a GAN (8) which discriminates trajectories generated by the policy and by Mocap data from dog motions.

III. Method

We first introduce our framework which follows closely previous imitation-based approaches(3; 9) and consists of 3 main components **1)** the input reference motion generated from

(2) which will be used to imitate on the Solo8 robot, **2)** the feedback-controller consisting of the RL policy optimized via PPO(10) and a PD controller and **3)** the simulated environment and robots for which we use the Raisim physics simulator(11). A diagram with all the components of our project can be found in Fig. 1.

A. Imitation-based Reinforcement Learning

Given some reference trajectory, the goal of our RL training framework is to learn a closed-loop feedback controller for some quadruped robot that imitates the reference motion while also satisfying some constraints. To do this the learnt policy conditioned on the state of the robot and operating at 50Hz outputs a correction to the reference motion such that:

$$[\hat{q}_t, \hat{\dot{q}}_t] = [q_{r_t}, \dot{q}_{r_t}] + \pi_\phi(s_t) \tag{1}$$

where $[\hat{q}_t, \hat{\dot{q}}_t]$ are the target joint angles and velocities, $[q_{r_t}, \dot{q}_{r_t}]$ the joint angles and velocities from the reference motion, and $pi(s_t)$ the correction from the policy at timestep t. The output torque is then set using a PD controller operating at $1kHz$ such that:

$$u = k_p(\hat{q} - q) + k_d(\hat{\dot{q}} - \dot{q}) \tag{2}$$

where k_p, k_d are the PD gains and $q, \dot{q}$ the observed joint angles and velocities.

B. State Representation & Reward Function

The input state vector of the policy for a timestep t is defined as:

$$s_t = (o, q, \hat{q}, \phi) \tag{3}$$

where o is the orientation of the robot, $q, \hat{q}$ the joint angles and velocities, and ϕ is the phase of the reference motion represented as $\phi = (\cos(2\pi i/T), \sin(2\pi i/T))$ where i is the index of the current step from the reference motion and T the total number of steps in the reference motion. It is worth noting that although more privileged information could be used as part of the state such as (foot contact, body position, etc) we want the policy to be transferable to a physical robot, and hence it's important to condition the policy only on information that would typically be available, which includes proprioceptive information from an IMU and encoders.

The reward function consists of two main components (1) a tracking reward r_{tr} that evaluates how close the generated motion follows the reference motion and (2) a regularization reward r_c that nudges the policy to meet some physical constrain. The tracking reward is defined as:

$$r_{tr} = w_o r_o + w_j r_j + w_f r_f \tag{4}$$

where r_o, r_j, r_f represent at timestep t how well the generated motion tracks the orientation, joint angles, and foot contacts of the reference data, w here are hyperparameters representing weights. Similarly, the regularization reward is defined as:

$$r_c = w_a r_a + w_t r_t \tag{5}$$

where r_a, r_t represent action difference, and resulted torque which nudges the learned policy to output smooth actions while minimizing torque. Each reward term r_x with the exception of the foot reward r_f has the form of:

$$r_x = \exp(-\frac{||\hat{x} - x||^2}{2\sigma_x^2}) \tag{6}$$

where $\hat{x}$ is the target value and x the observed value. For the foot reward r_f if the generated motion at timestep t follows the contact pattern of the reference motion then $r_f = 1$ otherwise $r_f = 0$. The total reward is $r = r_{tr} + r_c$. Note that the reward function here nudges the policy to imitate the trajectory locally. We later propose a reward term using PCA to capture global similarities between the generated trajectory and reference motion.

C. Dataset

We use a recently released large-scale robotics dataset, Open X-Embodiment (5), for our experiment. This dataset contains real robot trajectories from over 22 robots performing a total of 160266 tasks. For our work, we are only interested in the quadruped data released by the robotics lab from the University of Toronto (2), which uses the Unitree Go1 — a 12-DoF quadruped robot. This dataset contains over 20 unique trajectories for the Go1 robot (e.g. trot forward/backward, bound forward/backward, etc.). Each trajectory in the dataset contains body linear/angular velocities, joint angles, and joint angular velocities. Specifically, the raw data contains 30 columns of features: 6D body velocities (linear + angular), 12D joint angles, and 12D joint velocities. However, one immediate challenge with the dataset is that certain state information is not provided, specifically the body position/orientation and the foot contacts. This is important since the optimization algorithm can end up converging to some degenerate solution that still satisfies the available states.

D. Data Processing/Retargeting

To collect this privileged information, we fed the existing open-loop Go1 data to RaiSim and recorded the information such as 4D body orientations in quaternion, and binary contact point information for the four feet. In total, the Go1 data after being processed has 38 columns of features. For the robots that have a lower degree of freedom, we will directly delete the corresponding joint data. For example, for Solo8, we will delete the shoulder joint angles and velocities, which leads to a 30D feature space.

E. PCA motion loss

Imitation learning often provides a reward on a per-time-step basis based on joint angle motion and velocity. A motion, however, happens over many timesteps. Hence, the granularity of per-step rewards can fail to capture the quality of a motion. Inspired by recent works that utilize video networks (12) (8) to evaluate a motion holistically over many image frames, we propose to provide an overall reward on the motion using a PCA instead due to the lack of data.

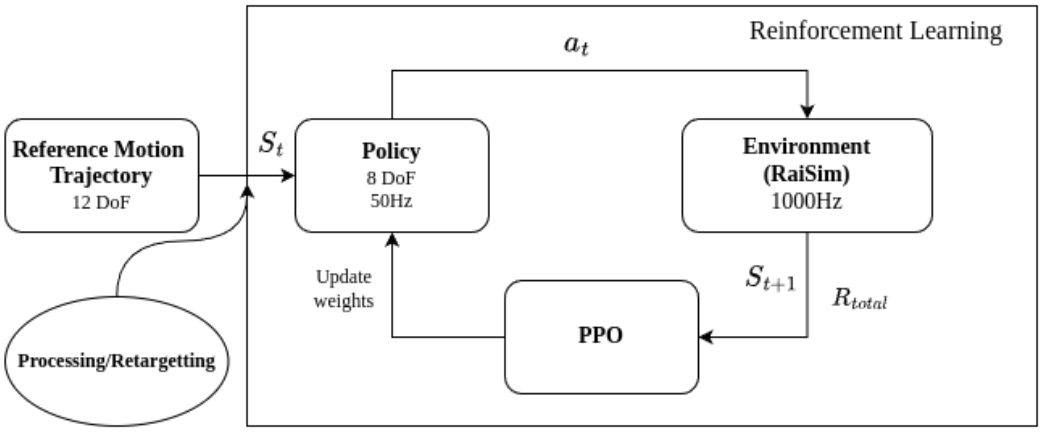

Fig. 1. **Our Imitation Learning Framework:** The reference motion is first processed to acquire privileged information and retargeted to the correct degree of freedom. We then pass the data to a standard deep reinforcement learning loop where the policy predicts the differences of the target motion parameters (eg. joint angle/velocity) between the ground truth and learning trajectory. The PD controller (RaiSim) in the environment will generate the hardware-level instruction to apply those motion parameters to reach a new state and then a reward is computed for the new state based on our customized reward function. After receiving the feedback from the environment, the PPO will update the policy parameters accordingly.

In detail, given a single reference trajectory of the go1 robot, we split the trajectory into blocks of size k time step. We do this for all the reference go1 trajectories to obtain a database of motions of size k. Utilizing this database, we can pretrain a PCA that learns a motion latent space. At training time, for every k timesteps, we retrieve the sample trajectory from the current policy as well as the reference trajectory and project them into the learned motion space in the PCA. We can derive a reward by measuring the distance between the sample and reference trajectory with

$$r_{PCA} = \exp(-\frac{||v_{\text{sample}} - v_{\text{ref}}||^2}{2\sigma^2_{PCA}}) \tag{7}$$

Where v_{sample} and v_{ref} are the projection of the sample trajectory and reference trajectory in the motion space respectively. The information computed from this reward is propagated by adding it to the per-step reward over those k steps.

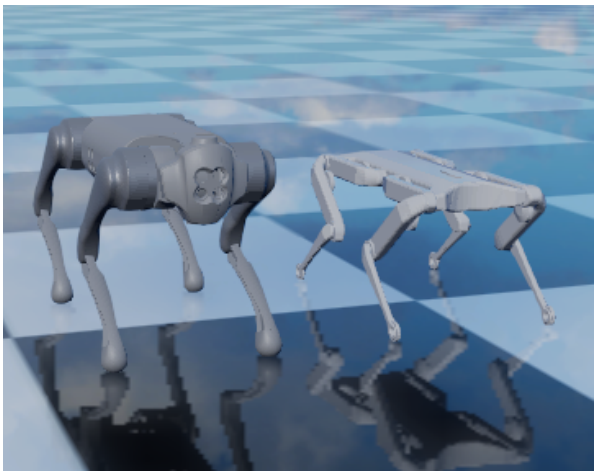

Fig. 2. Unitree Go1 (left), Solo8 (right)

IV. Experiments

A. Evaluation

For evaluation in all experiments, we have qualitative results with videos only. The reason is two fold. Firstly, due to the nature of our problem, we do not have ground truth motions to evaluate on since the policy is learned for a robot that lacks data. Secondly, the reward values themselves are not meaningful in RL as it is used only to measure convergence. Hence, we only have qualitative evaluation.

B. Go1 to Solo8 transfer

As depicted in Fig.1, we would like to use deep reinforcement learning (DRL) to improve the motions that lack feedback in the open loop. The first target in our experiments is an 8-DoF quadruped robot Solo8 (See Fig.2). Compared to Go1, it does not have a degree of freedom on its shoulder. We performed DRL on the data processed by the RaiSim simulator on several trot and bound motions in different directions (forward and backward).

The policy consists of two fully connected layers with 128 neurons each. The training contains 4000 epochs for each motion, which promised that every policy would converge to minima. In our case, we disabled the random starting point function in Opt-mimic to let the motions stay complete during training and inferencing. Although this slows down the training process a bit, the overall training time is still acceptable, which takes approximately 20 minutes per motion on an NVIDIA GTX1060 GPU. For the PD controller values, we use $k_p = 3, k_d = 0.3$ as they are common values for Solo8. As mentioned in Section IIIB, we used a similar reward function as in Opt-mimic, however, considering the target trajectory is fully open-looped, the position values will be somewhat meaningless to learn because different robots have different heights, leg lengths, etc. Thus, we gave the weight

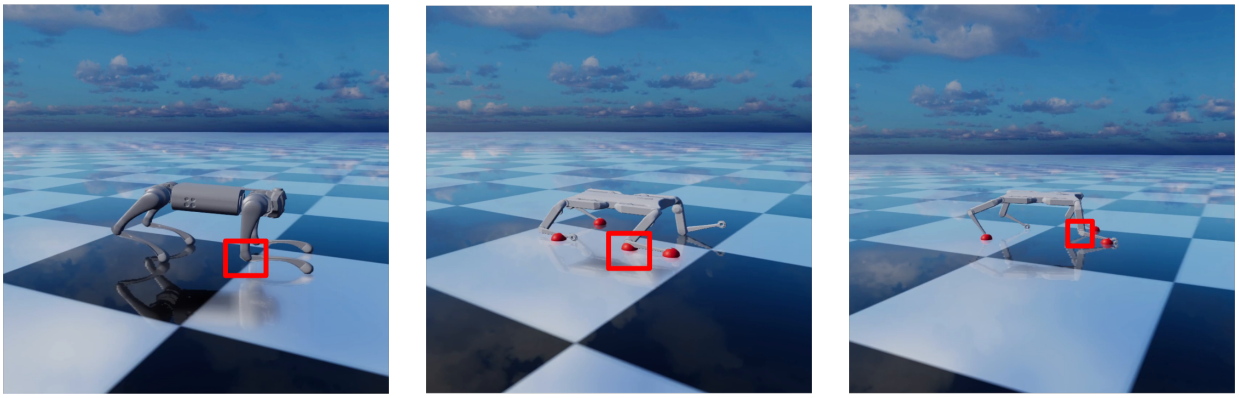

Fig. 3. From left to right: Go1 reference trajectory ran open loop on the Go1 robot, reference trajectory ran Open Loop on Solo8, Generated motion for Solo8 after training via RL. Red markers indicate the collision points of the robot and the ground. Notice from the bounding box that when directly applying the data Open Loop other body parts end up colliding during the motion, but after training this issue does not occur.

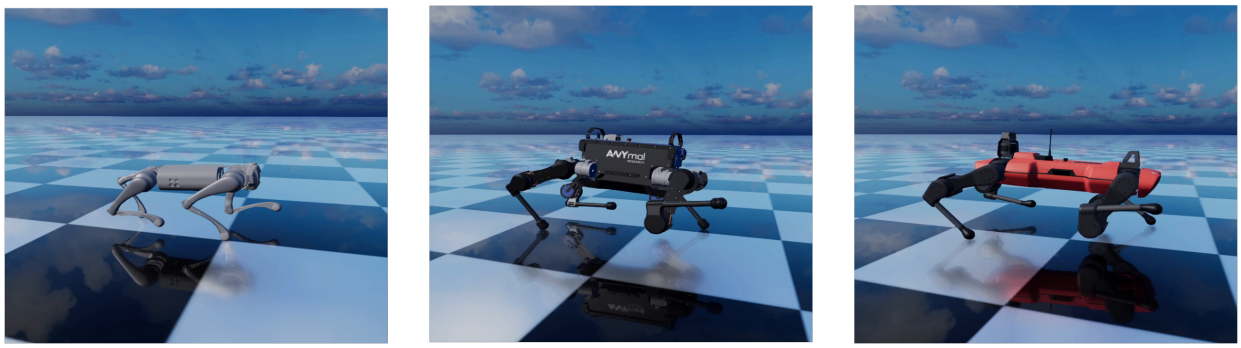

Fig. 4. Transferring a trot forward motion task on other quadrupeds. From left to right: Go1 (original) , ANYmal, ANYmal_c.

of the position reward to foot contacts, which makes more sense in representing the pattern of the target motion. The final set of reward weights used are $w_{\text{orientation}} = 0.2, w_{\text{joint}} = 0.2, w_{\text{action diff}} = 0.2, w_{\text{max torque}} = 0.1, w_{\text{foot contact}} = 0.3$. Then we compared the results as follows. We use open loop as our baseline for all experiments.

As can be seen from Fig.3, the baseline in the middle (open loop trot backward motion) performs very poorly in the simulator. The knees are touching the ground while trotting, which deviates a lot from the original motion either on the real Go1 or the Go1 in the simulator. After the DRL, the policy corrects the motions and now the trotting backward motion behaves as desired. For the complete motion comparison, please see the videos attached.

C. Go1 to other quadruped robots transfer

We also conduct experiments to evaluate the generalizability of DRL to other robots such as ANYmal and ANYmal_c. The experiment setup for the new robots is approximately the same except for the data retargeting and the PD control values. Since both ANYmal and ANYmal_c are 12-DoF robots, no data masking is needed and all 38 columns are used. The PD control values for these two robots are increased to $k_p = 200, k_d = 20$ due to different robot morphology. Fig.4 is a screenshot of the result. For the full results, please see our attached video.

V. Conclusion

In this work, we have shown our imitation framework was able to learn how to generate trajectories for the Solo8 robot similar to the motions of some reference data from the Unitree Go1 while also satisfying constraints such as reduced collision of undesired parts. This significantly improves the motion and makes it feasible to transfer from Go1 to other robots. We also showed that with minimal adjustments this framework can be applied to other quadruped robots. Finally, we propose a novel reward term using PCAs to capture global similarities between trajectories that we hope to integrate in future work.

A. Limitations

One important limitation of this work is that our framework is bottlenecked by readily available quadruped data and even for the data that is available it can sometimes suffer from discrepancies due to the Sim-to-Real gap. Additionally, certain motions can not transfer between robots, especially in the case where we are transferring a motion from a 12-DoF to an 8-DoF, or even if they can transfer they might not represent the optimal way that the robot could perform it. Further, due to time constraints, we were not able to transfer the policies on

the physical robot and see if our approach is robust to the Sim-to-Real gap. Finally, due to the nature of the problem, we do not have quantitative measures that can evaluate the quality of the transfer as there are no ground truth motion for the motion-transferred robot.

B. Future works

We hope to continue this work and integrate the PCA reward term into our framework, additionally, we would like to experiment with stitching different reference motions together and learning one policy to imitate it. Other interesting research directions would be conditioning the policy on robot morphologies such as mass, limp lengths, etc, and learning one general policy across different robots. Finally, it would be interesting to explore in the future different ways to generate relevant motions for robotics, for example, diffusion models have seen great success in generating unseen images by learning many different distributions and it would be interesting to see a similar approach in robotics via generating reference trajectories.

REFERENCES

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

[2] Y. Tang, W. Yu, J. Tan, H. Zen, A. Faust, and T. Harada, "Saytap: Language to quadrupedal locomotion," *arXiv preprint arXiv:2306.07580*, 2023.

[3] Y. Fuchioka, Z. Xie, and M. Van de Panne, "Opt-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5092–5098.

[4] A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel, "Adversarial motion priors make good substitutes for complex reward functions," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 25–32.

[5] O. X.-E. Collaboration, A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, A. Raffin, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Ichter, C. Lu, C. Xu, C. Finn, C. Xu, C. Chi, C. Huang, C. Chan, C. Pan, C. Fu, C. Devin, D. Driess, D. Pathak, D. Shah, D. Büchler, D. Kalashnikov, D. Sadigh, E. Johns, F. Ceola, F. Xia, F. Stulp, G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Schiavi, H. Su, H.-S. Fang, H. Shi, H. B. Amor, H. I. Christensen, H. Furuta, H. Walke, H. Fang, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Kim, J. Schneider, J. Hsu, J. Bohg, J. Bingham, J. Wu, J. Wu, J. Luo, J. Gu, J. Tan, J. Oh, J. Malik, J. Tompson, J. Yang, J. J. Lim, J. Silvério, J. Han, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Zhang, K. Majd, K. Rana, K. Srinivasan, L. Y. Chen, L. Pinto, L. Tan, L. Ott, L. Lee, M. Tomizuka, M. Du, M. Ahn, M. Zhang, M. Ding, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, P. R. Sanketi, P. Wohlhart, P. Xu, P. Sermanet, P. Sundaresan, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Martín-Martín, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Moore, S. Bahl, S. Dass, S. Song, S. Xu, S. Haldar, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Dasari, S. Belkhale, T. Osa, T. Harada, T. Matsushima, T. Xiao, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Wang, X. Zhu, X. Li, Y. Lu, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y. hua Wu, Y. Tang, Y. Zhu, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Xu, and Z. J. Cui, "Open X-Embodiment: Robotic learning datasets and RT-X models," https://robotics-transformer-x.github.io, 2023.

[6] C. Li, M. Vlastelica, S. Blaes, J. Frey, F. Grimminger, and G. Martius, "Learning agile skills via adversarial imitation of rough partial demonstrations," in *Conference on Robot Learning*. PMLR, 2023, pp. 342–352.

[7] 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.

[8] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE signal processing magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[9] X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

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

[11] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018. [Online]. Available: www.raisim.com

[12] S. A. Sontakke, J. Zhang, S. M. R. Arnold, K. Pertsch, E. Bıyık, D. Sadigh, C. Finn, and L. Itti, "Roboclip: One demonstration is enough to learn robot policies," 2023.