Words into Action: Learning Diverse Humanoid Robot Behaviors using Language Guided Iterative Motion Refinement

K. Niranjan Kumar
Georgia Institute of Technology

Irfan Essa
Georgia Institute of Technology

Sehoon Ha
Georgia Institute of Technology

Abstract: Humanoid robots are well suited for human habitats due to their morphological similarity, but developing controllers for them is a challenging task that involves multiple sub-problems, such as control, planning and perception. In this paper, we introduce a method to simplify controller design by enabling users to train and fine-tune robot control policies using natural language commands. We first learn a neural network policy that generates behaviors given a natural language command, such as "walk forward", by combining Large Language Models (LLMs), motion retargeting, and motion imitation. Based on the synthesized motion, we iteratively fine-tune by updating the text prompt and querying LLMs to find the best checkpoint associated with the closest motion in history. We validate our approach using a simulated Digit humanoid robot and demonstrate learning of diverse motions, such as walking, hopping, and kicking, without the burden of complex reward engineering. In addition, we show that our iterative refinement enables us to learn $3\times$ times faster than a naive formulation that learns from scratch. See video of results here - https://www.kniranjankumar.com/words_into_action/

1 INTRODUCTION

Humanoid robots have long fascinated both scientists and science fiction writers. Recent advancements have led to controllers that enable these robots to execute complex maneuvers like jumping, hopping, and even front-flips. However, the traditional approach to designing these controllers involves labor-intensive engineering: creating a robot model, planning trajectories, and optimizing a cost function. This process must be repeated for each new motion, making it an unscalable solution for controllers that need to adapt to changing environments and skill sets [1, 2]. An alternative is to use model-free deep Reinforcement Learning (DRL) techniques that learn control policies by maximizing a reward function through extensive simulation data. While these techniques have shown promise in solving complex control problems, they come with their own challenges, particularly in reward engineering. Designing a reward function that captures the nuances of desired behaviors for high-DOF robots is no small feat.

We take inspiration from the recent advances in the control of virtual humanoids by imitating the given reference motion [3, 4], which we refer to as motion imitation. This approach enables efficient learning of a variety of motor skills by offering a unified task definition of imitating the corresponding motions. Recent studies have explored using motion capture data as a reference for learned policies, showing promising results in quadruped robot locomotion [5, 6].

Building on these insights, our work focuses on learning joint-level control policies for humanoid robots directly from language commands, significantly simplifying the reward engineering process. Our method allows for iterative fine-tuning of policies through interactive language commands,

offering precise control over the robot's behaviors. We achieve this by first generating human motion based on language commands, retargeting them to a humanoid robot, and then learning control policies using a motion imitation approach. We then develop language-based policy refinement, a process which allows users to adjust the learned behavior by identifying the closest checkpoint with a large language model and fine-tune the policy starting from it.

We demonstrate a wide range of behaviors for the humanoid robot Digit using our framework. From simple language prompts, our framework learns diverse behaviors such as hopping, stepping to the side, and kicking. In addition, we provide evidence supporting the benefits of iterative human-guided policy refinement approach, which offers $3\times$ better sample efficiency.

2 RELATED WORK

2.1 Robot learning for legged robots

A natural solution to building robots adapted to the human living environment, is to adopt a morphology similar to human form. However, designing controllers for humanoid robots that are versatile and robust to external perturbations is a challenging problem that has been studied extensively over the last few decades. A classical approach is to develop a dynamics model for the robot and then use it to develop controllers that plan and execute control actions, optimizing a specified objective. These models [7, 8] span from simple approximations that reduce the robot to a linear inverted pendulum [9, 10, 11, 12, 13] to more sophisticated alternatives that consider the entire dynamics of the humanoid [1, 2] or some combination of the two strategies [14]. While simpler models make the problem tractable from a computational standpoint, they sacrifice exploiting the full capabilities of the robot dynamics and constrain themselves to motion feasible on the simple model. Modeling the full dynamics of the robot, on the other hand, is difficult and time-consuming and may not be applicable for real-time robot control.

In recent years, learning-based approaches, particularly deep reinforcement learning (DRL) [15] have gained increasing attention from the humanoid robot control community, following impressive results on quadruped [16, 17, 18] and bipedal robots [19, 20]. Some work incorporates learning within other model-based frameworks to improve robustness and adaptability. For example, in [21] a learned policy generates actuator trajectories that are then tracked using a feedback regulator designed with the robot model. In a related work [22], a learned policy acts as a foot-trajectory modulator while a low-level gait controller regulates the torso and ankle orientation. A recent line of work takes an end-to-end learning-based framework to develop humanoid robot controllers [23, 24]. Unlike model-based approaches, learning based techniques do not require solving for the control output at every time-step using the robot dynamics model, which makes them suitable for high-frequency real-time control. These policies can be trained in massively parallelized simulations [25] and transferred directly to the real-world without additional fine-tuning. While learning-based approaches offer a powerful alternative to model-based control, it is often challenging to design and tune reward functions to accomplish a given task. Recent work, [5, 26] leverages expert demonstrations to ground the space of behaviors that emerge from the policy by imposing a motion prior.

2.2 Human motion generation

Generating human motion has a wide range of applications from animation [27] to modeling human behaviors for human robot interaction. With the abundance of human motion capture datasets like HumanML3D [28] researchers have been able to create generative model for human motion [29]. With the advent of Large Language Models (LLMs), using text to generate human motion has gained considerable interest within the research community. Given the ease of prompting and guiding the generation process, LLMs offer a promising strategy to control synthesis.

However, the generated motion from these methods is not grounded in physics and often demonstrates foot sliding, jitters or self-collisions. So a line of work focuses on learning control policies for virtual agents in a physics simulation to imitate motion capture trajectories while being physi-

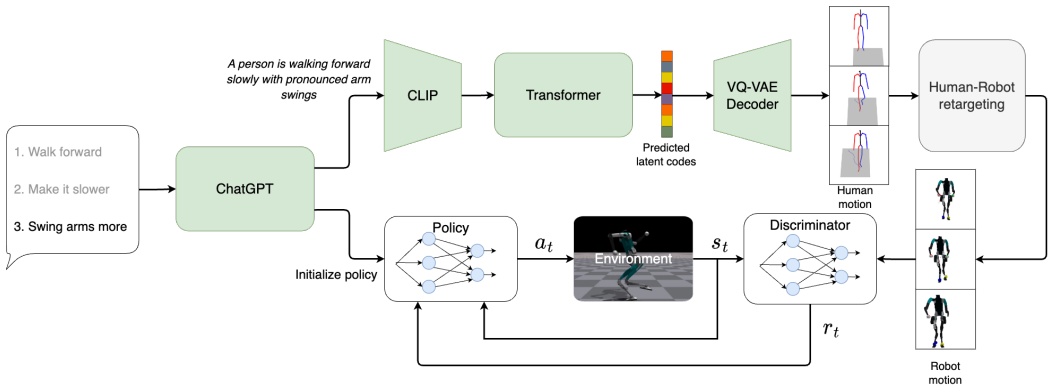


Figure 1: Overview of our proposed approach. Given a language instruction our framework outputs a learned control policy for the corresponding behavior.

cally plausible [3, 4]. In this work, we build upon these ideas to use human motion as a reference to guide on a humanoid robot control policies.

2.3 LLMs for Robot Control

LLMs have demonstrated human-level reasoning capabilities across a wide range of domains ranging from code-generation [30], multi-step reasoning [31, 32, 33] to prompt-engineering [34]. One limitation of LLMs is the lack of grounding in the real world. Robotics offers a promising approach to ground LLMs into the real world. They have been rapidly adopted by the robotics research community in the past few months. With LLMs reaching human-level reasoning, they have the potential to significantly augment a robot's capabilities. They have been shown to generate long-horizon plans [35], understand and process complex cluttered scenes [36], and explain the reasoning behind decisions a robot takes [37]. In this work, we use LLMs to generate human motion trajectories to build prompts based on interactive human instructions and to guide model initialization to reduce training time.

3 LEARNING POLICIES FROM LANGUAGE PROMPTS

The first step is to learn control policies from the language prompts given by users. It consists of three main components:

1. Human motion generation from text input
2. Motion retargeting from human to robot
3. Training a control policy to imitate retargeted motion

We provide an overview of our framework in Fig. 1. In the following section, we discuss each of these components.

3.1 Human motion generation from text input

Human motion generation has a wide range of applications from animation to behavior modeling. A natural modality to direct the generation of motion is through textual descriptions. Recently, there has been rapid progress in generating diverse high-quality human motion trajectories from language descriptions. Most of these approaches leverage large-scale annotated datasets of motion-capture data to train generative models to map language input to a distribution of possible human trajectories.

In this work, we follow T2M-GPT [29] to generate human trajectories from a language description. We first train a VQ-VAE that builds a discrete latent representation of the human motion space.

By selecting different parameters in the latent space, we have a handle on the nature of the motion generated. For a motion sequence $X = [x_1, x_2, \cdots, x_T]$ where $x_t \in \mathbb{R}^d$, T is the number of frames in the motion sequence, d is the dimension of the motion, the goal is to learn a codebook $C = \{c_k\}_{k=1}^{K}$ of size K with $c_k \in \mathbb{R}^{d_c}$ where d_c is the dimension of the codes. To accomplish this, we train an autoencoder type architecture where an encoder E computes the latent variable $Z = E(X)$ from X and a decoder D reconstructs the motion from the latent variable. In contrast to a traditional variational autoencoder, the latent variable is discrete, defined as $Z = [z_1, z_2, \cdots, z_{T/l}]$ where $z_i \in \mathbb{R}^{d_c}$ and l is the temporal downsampling rate. The encoded Z is then quantized to $\hat{Z}$ by finding the closest element in C.

$$\hat{z_i} = \underset{c_k \in C}{\arg\min} \|z_i - c_k\|_2 \tag{1}$$

The VQ-VAE objective is then to minimize the following:

$$\mathcal{L} = \mathcal{L}_{re} + \mathcal{L}_{embed} + \beta \mathcal{L}_{commit} \tag{2}$$

where,

$$\mathcal{L}_{embed} = \left\| Z - sg[\hat{Z}] \right\|_2, \mathcal{L}_{commit} = \left\| sg[Z] - \hat{Z} \right\|_2 \tag{3}$$

$\mathcal{L}_{embed}$ minimizes the embedding loss so that the predicted Z is close to the elements in the codebook and $\mathcal{L}_{commit}$ updates the codebook to better fit the encoded values. β is a hyper-parameter to control the relative impact of these terms on the final loss. The reconstruction loss $\mathcal{L}_{re}$ is an L_1 smooth loss between the ground truth and the predicted position and velocity of the motion.

Given an expressive enough motion embedding, we can generate any arbitrary motion sequence by auto-regressively generating a series of codebook indices. Including a text condition c provides a handle on the generated codebook entries. This process is modeled using transformer architecture and trained on the HumanML3D dataset [28]. Please refer to T2M-GPT [29] for more details about training and architecture.

3.2 Motion retargeting from human to robot

Given a human motion trajectory generated from the previous step, we want to imitate it using a learned control policy. Due to differences in the skeletons, we cannot directly map the joint angles. We instead track just the end-effector locations i.e., 2 hands + 2 legs. We design an Inverse Kinematics (IK) objective that minimizes error between the reference relative end-effector positions and the robot end-effector positions. Given the list of reference positions x_{H} and robot joint angles q_{D} the IK objective is defined as follows:

$$q^* = \underset{q}{\arg\min} \|x_H - T_{\mathrm{FK}}(q)\|_2 + \lambda \mathcal{C}_{\mathrm{f}}(q) \tag{4}$$

$T_{\mathrm{FK}}(q)$ is the forward kinematics function of the robot, C_{f} is a term that captures the feasibility of a pose and λ is a scaling factor. C_{f} in our case simulates rod constraints to mimic the 4-bar linkage present in the Digit robot and ensures that change in q over consecutive timesteps in minimal.

We solve this optimization for every timestep in the trajectory using Sequential Least Squares Programming. To ensure the temporal smoothness, we initialize the variables with the solution of the previous timestep. While the trajectory we get at the end is kinematically feasible, we cannot guarantee its success when subjected to robot dynamics and physics. Hence, we train a neural network policy to track these reference poses while being embedded in a physics simulator. In the next section, we discuss our policy learning framework.

3.3 Training control policy to imitate retargeted motion

Given a reference trajectory of the robot, we want to train a control policy that imitates it while being dynamically feasible. It should be noted that some of these states might not be physically realistic on the robot. We would like our approach to ignore these states and focus on parts of the trajectory

that can be reliably tracked instead. Motion imitation approaches offer a promising solution to this problem. Rather than forcing the policy to reach every segment in the reference, motion imitation approaches typically only require that the trajectories generated from the policy resemble those found in the reference. Therefore, the infeasible segments can be ignored as the policy is trained. Thus, the *mode collapse* problem that often plagues generative models works to our advantage here, eliminating the need to manually curate a set of dynamically feasible motions for the robot.

We use Adversarial Motion Priors (AMP) [3] to train neural network policies for joint-level control of the robot. Given a dataset of state transitions $\mathcal{M}$, a discriminator network $D(s, s')$ is trained to predict if the transition is from the learned policy π or from $\mathcal{M}$. This follows an architecture similar to GAIL [38] but with state transitions (s, s') instead of state-action pairs (s, a).

$$\arg\min_{D} - \mathbb{E}_{d^{\mathcal{M}}(\mathrm{s},\mathrm{s}')} \left[\log \left(D\left(\mathrm{~s},\ \mathrm{s}'\right)\right)\right] - \mathbb{E}_{d^{\pi}(\mathrm{s},\mathrm{s}')} \left[\log \left(1 - D\left(\mathrm{~s},\ \mathrm{s}'\right)\right)\right]. \tag{5}$$

The discriminator is then used to compute reward during policy training.

$$r\left(s_t, s_{t+1}\right) = \max[0, 1 - 0.25(D(s_t, s_{t+1}) - 1)^2] \tag{6}$$

The neural network policy is learned with proximal-policy optimization(PPO) [39] using a value function network and a gradient penalty to stabilize training. See AMP [3] for network architecture and details about the training process.

4 LANGUAGE-GUIDED ITERATIVE POLICY REFINEMENT

Training control policies for high-DOF robots involves careful reward engineering to tune the style of the behaviors that emerge, requiring expert knowledge about the impact of different reward terms on the final policy. In this section, we seek to simplify this process by conditioning the behaviors generated on natural language commands, iteratively adjusting the trained policy. We leverage the general-purpose language understanding and reasoning capacity of LLMs to achieve this goal. Given an input command update, the LLM is assigned two tasks:

1. Create/update the instruction given to T2M-GPT to incorporate the update
2. Initialize the policy with the closest previously trained model (if applicable) to improve sample efficiency.

We prompt a conversational LLM (ChatGPT-4) to behave as a prompt generator and policy initalizer using the following prompt:

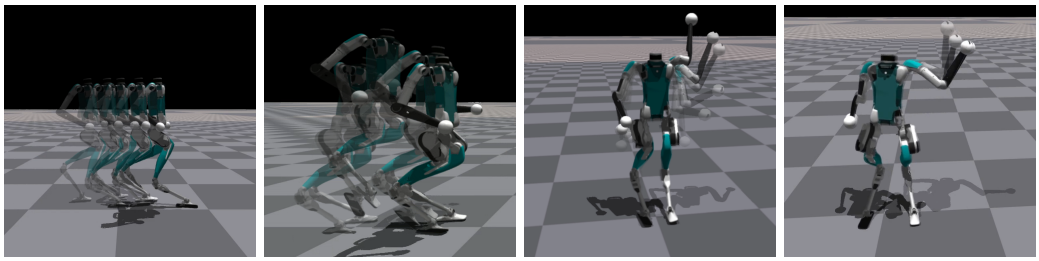
Your role is to generate prompts for a motion model. You have to continuously create/update the original prompt based on user input "command" and recover a motion "closest_prompt_in_history" from a history of generated prompts "motion_history" that resemble the generated prompt. If the motions in the history are completely unrelated to the generated prompt, return "None". Your prompts describe the actions of a person. Examples of prompts:

1. *A person is walking forward*
2. *A person is hopping forward then turning around and hopping back to the start.*

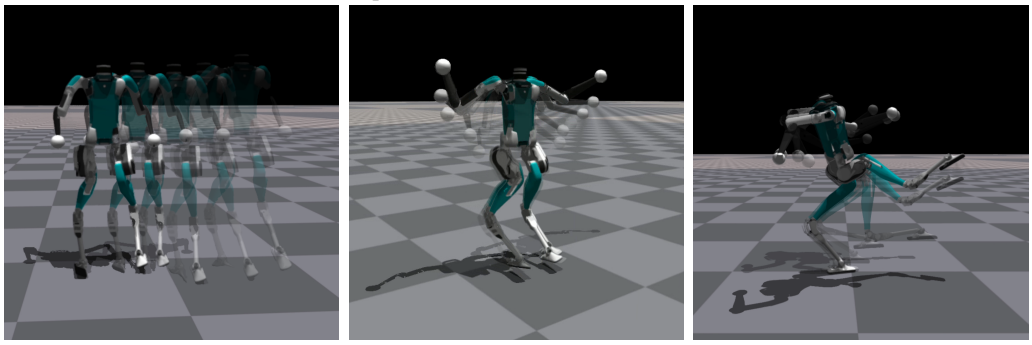
The user commands will have the format: ⟨*command*⟩
You should return :

- ⟨*prompt*⟩
- ⟨*closest_prompt_in_history*⟩
- ⟨*motion_history*⟩

"motion_history"is a list of the motion history that includes the generated"prompt". "closest_prompt_in_history" can be "None" if none of the motions are similar enough to the prompt. For example, jumping is very different from walking. But walking slow is similar to walking fast.Wait for the next user input.



(a) Walk forward (b) Hop forward (c) Raise hand (d) Wave to a friend



(e) Step to the side gradually (f) Celebrate (g) Kick slowly

Figure 2: Motion frames demonstrating the skills learned with our approach

Given new instructions conversational LLM automatically updates the prompt given to T2M-GPT and returns the closest model to initialize the policy with, if it exists. We present the behaviors trained using our approach in the following section.

5 EXPERIMENTS

In our work, we train control policies for the humanoid robot Digit from Agility robotics. Digit has "bird-legs" (digitigrades) which differ significantly from human legs. The robot walks on its toes instead of its foot. We show that our approach is immune to such differences in morphology and can learn a diverse set of joint-level control policies for Digit. We first define the underlying Markov Decision Process (MDP) for our control policy, then describe the simulation setup and results.

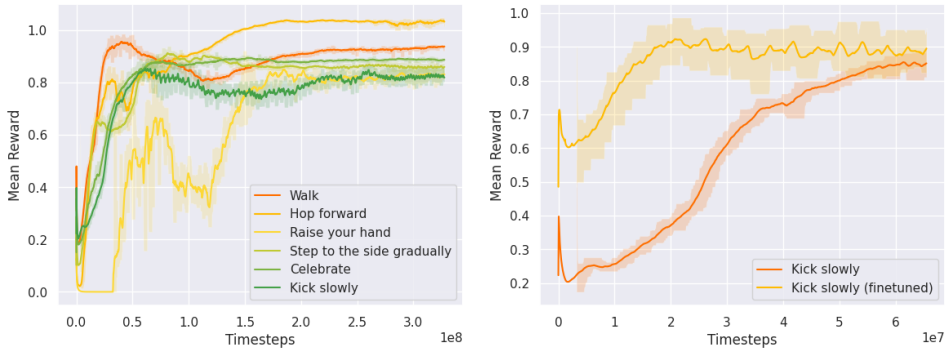
5.1 Problem Formulation

Learning control policies for a robot can be modeled as a Markov Decision Process (MDP) where the actions taken by the robot moves it from one state to another. In this work, we define the MDP underlying the robot control problem as follows:

- **States:** The state is a 69 dimensional vector formed by the concatenation of the root height (1), root orientation in normal-tangent encoding (6), root velocity (3), root angular velocity (3), joint positions (22), joint velocities (22) and end-effector positions (4×3).
- **Actions:** The action is a 22 dimensional vector of joint angle positions that are sent to a PD controller to calculate the control torque.
- **Rewards:** We use a single reward term calculated using Eq. 6. However, more task specific rewards can be added if required.

5.2 Simulation

We use IsaacGym [25] to train all our control policies on multiple parallelized simulated robots. We simulate 4096 Digit robots in parallel to train our neural networks on a single NVIDIA Titan



(a) Reward curves for the different skills in our approach

(b) Iteratively fine-tuning a skill from similar previous skills reduces training time

Figure 3: Reward curves for the different behaviors trained using our approach

X GPU. However, IsaacGym does not support closed-loop chains which are necessary to model the 4-bar linkages in Digit's legs. To handle this, we simulate virtual springs that mimic the physics of the rods, similar to previous work [23]. To ensure that the policies learned are robust, we perform domain randomization by adding Gaussian noise to observation, gravity, and actions taken by the robot with standard deviations $0.02, 0.4, 0.02$, respectively. We train each of our policies for a total of $330M$ steps, taking ≈ 3 hours of training time.

5.3 Results

5.3.1 Learned Skills

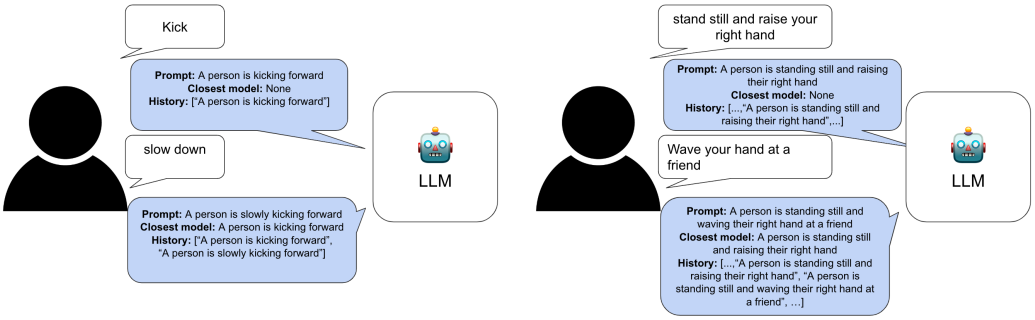
Using our framework, we train a diverse set of skills for Digit. We list the behaviors generated using our approach below:

1. Walk forward - a locomotion policy that makes the robot walk forward while swinging its arms
2. Hop forward - a policy for that makes the robot hop forward continuously
3. Raise your hand - a policy that makes the robot stand in place and balance while lifting its hand
4. Wave to a friend - a policy that makes the robot wave its hand to say hello
5. Step to the side gradually - a policy that makes the robot step to the side orthogonal to its heading direction.
6. Celebrate - a policy that makes the robot act like a cheerleader
7. Kick slowly - a policy that makes the robot balance on one leg as it lifts the other leg up and kicks

We present motion frames of these skills, along with the intermediate human motion generated in Fig. 2. The reward curve for all our policies converge resulting in motions that recreate feasible portions of the reference (See Fig. 3a). Our approach can handle abstract commands and generate meaningful policies that demonstrate the intent behind the command. When instructed to "Celebrate", our framework generates a cheerleader's routine, hopping and swinging its arms as shown in Fig. 2.

5.3.2 Human guided iterative refinement

In addition to generating behaviors given a language instruction, our framework allows us to interactively fine-tune the robot motion. We show examples of interactive refinement in Fig. 4b. Notice



(a) Iterative refinement to generate feasible behavior (b) Iterative refinement to reuse learned skills

Figure 4: Examples of human guided policy refinement using our framework

that as we interact with our framework, the generated command is continuously updated to account for user feedback and the policy is initialized automatically using previously learned behaviors to minimize training iterations. We demonstrate two examples of iterative skill refinement below:

1. We tell our framework to make the robot "Kick". The LLM does not have any previously learned skill it can reuse, so it initializes the model from scratch. But as the policy trains we notice that the behavior is too sudden and is not feasible on the robot morphology. We tell our framework to "Slow down". The LLM updates the command prompt given to T2M-GPT to kick slowly, realizes that we have already learned how to kick and initializes the policy with pre-trained weights. A motion frame for the trained skill is shown in Fig. 2g
2. We tell our framework to make the robot "Wave to a friend" after learning a set of skills. Our framework realizes that we have already learned how to "Stand still and raise your right hand up" (Fig. 2c), and initializes the policy with pre-trained weights. A motion frame for the trained skill is shown in Fig. 2d

In Fig. 3b we show the reward curves for training a "Kick slowly" skill from scratch, vs fine-tuning from a previously learned "Kick" skill. Training using our approach takes significantly fewer steps (≈ 40M fewer in this case) compared to training from scratch.

6 CONCLUSION

In this work, we presented preliminary results on a framework that generates control policies for dynamic and agile behaviors on a humanoid robot from language instruction. We demonstrated the behaviors learned by our approach on 6 different instructions and discussed an iterative human-in-the-loop refinement approach to fine-tune the learned behaviors. For future work, we are interested in building a robot motion embedding that directly translates language commands into control actions, without requiring us to retrain a new policy for every instruction. We would also like to deploy our policy on a real Digit robot to test the robustness of our learned policy against environmental perturbations and the sim-to-real gap.

References

[1] M. Posa, S. Kuindersma, and R. Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1366–1373. IEEE, 2016.

[2] B. Henze, A. Dietrich, and C. Ott. An approach to combine balancing with hierarchical whole-body control for legged humanoid robots. *IEEE Robotics and Automation Letters*, 1(2):700–707, 2015.

[3] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. 40(4), jul 2021. ISSN 0730-0301. doi:10.1145/3450626.3459670. URL https://doi.org/10.1145/3450626.3459670.

[4] 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)*, 37(4):1–14, 2018.

[5] 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. *arXiv preprint arXiv:2203.15103*, 2022.

[6] A. Klipfel, N. Sontakke, R. Liu, and S. Ha. Learning a single policy for diverse behaviors on a quadrupedal robot using scalable motion imitation. *arXiv preprint arXiv:2303.15331*, 2023.

[7] J. W. Grizzle, C. Chevallereau, R. W. Sinnet, and A. D. Ames. Models, feedback control, and open problems of 3d bipedal robotic walking. *Automatica*, 50(8):1955–1988, 2014.

[8] Y. Hurmuzlu, F. Génot, and B. Brogliato. Modeling, stability and control of biped robots—a general framework. *Automatica*, 40(10):1647–1664, 2004.

[9] M. H. Raibert. Legged robots. *Communications of the ACM*, 29(6):499–514, 1986.

[10] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *2003 IEEE international conference on robotics and automation (Cat. No. 03CH37422)*, volume 2, pages 1620–1626. IEEE, 2003.

[11] J. Yamaguchi, E. Soga, S. Inoue, and A. Takanishi. Development of a bipedal humanoid robot-control method of whole body cooperative dynamic biped walking. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 1, pages 368–374. IEEE, 1999.

[12] T. Sugihara, Y. Nakamura, and H. Inoue. Real-time humanoid motion generation through zmp manipulation based on inverted pendulum control. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1404–1409. IEEE, 2002.

[13] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of honda humanoid robot. In *Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146)*, volume 2, pages 1321–1326. IEEE, 1998.

[14] H. Dai, A. Valenzuela, and R. Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 295–302. IEEE, 2014.

[15] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[16] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[17] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[18] K. N. Kumar, I. Essa, and S. Ha. Cascaded compositional residual learning for complex interactive behaviors. *IEEE Robotics and Automation Letters*, 2023.

[19] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2811–2817. IEEE, 2021.

[20] A. Kumar, Z. Li, J. Zeng, D. Pathak, K. Sreenath, and J. Malik. Adapting rapid motor adaptation for bipedal robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1161–1168. IEEE, 2022.

[21] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid. Reinforcement learning-based cascade motion policy design for robust 3d bipedal locomotion. *IEEE Access*, 10:20135–20148, 2022.

[22] L. Krishna, G. A. Castillo, U. A. Mishra, A. Hereid, and S. Kolathaya. Linear policies are sufficient to realize robust bipedal walking on challenging terrains. *IEEE Robotics and Automation Letters*, 7(2):2047–2054, 2022.

[23] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Learning humanoid locomotion with transformers. *arXiv preprint arXiv:2303.03381*, 2023.

[24] Y. Guo, Z. Jiang, Y.-J. Wang, J. Gao, and J. Chen. Decentralized motor skill learning for complex robotic systems. *arXiv preprint arXiv:2306.17411*, 2023.

[25] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[26] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[27] J. Lee, J. Chai, P. S. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 491–500, 2002.

[28] C. Guo, S. Zou, X. Zuo, S. Wang, W. Ji, X. Li, and L. Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5152–5161, 2022.

[29] J. Zhang, Y. Zhang, X. Cun, S. Huang, Y. Zhang, H. Zhao, H. Lu, and X. Shen. T2m-gpt: Generating human motion from textual descriptions with discrete representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[30] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[31] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[32] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[33] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023.

[34] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

[35] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[36] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[37] X. Zhang, Y. Guo, S. Stepputtis, K. Sycara, and J. Campbell. Explaining agent behavior with large language models. *arXiv preprint arXiv:2309.10346*, 2023.

[38] J. Ho and S. Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.