Generalizable Beam Walking for Legged Robots with Reinforcement Learning

Xilun Zhang
Department of Mechanical Engineering
Carnegie Mellon University
xilunz@andrew.cmu.edu

Changyi Lin
Department of Mechanical Engineering
Carnegie Mellon University
changyil@andrew.cmu.edu

Zongyuan Wu
Department of Mechanical Engineering
Carnegie Mellon University
zongyuaw@andrew.cmu.edu

Jian Chen
Department of Electrical and Computer Engineering
Carnegie Mellon University
jianc2@andrew.cmu.edu

Abstract: In our study, we explore the complexities of beam walking in robotics, a significant test in assessing mobility, closely paralleling the evaluations used for humans and animals. This task demands a sophisticated visuomotor coordination, balancing real-time body dynamics and environmental feedback. From previous works, robotic beam walking has relied on either specific world frame locations or external loading on the robot. However, our research draws inspiration from the latest advancements in robot parkour and visual servoing systems, leading us to develop a learning-based methodology that utilizes depth images for navigation, thus eliminating the need for additional loads. Inspired by direct collocation and curriculum learning, we propose a three-phase Reinforcement Learning(RL) pipeline. This pipeline also includes a pre-trained feature extractor, which transforms depth images into interpretable state vectors. This transformation is crucial for the robot's effective interaction with its environment. We applied this method to a quadruped robot, enabling it to autonomously navigate a 10 cm wide and 2-meter long beam, linking two platforms. The system proved effective even with different starting positions and orientations. This success demonstrates the potential of our RL training pipeline and designed perception system in advancing robotic mobility and autonomy, especially in complex tasks like beam walking.

Keywords: Agile Locomotion, Visuomotor Control, Reinforcement Learning

1 Introduction

This paper introduces a learning-based framework for robot beam walking, aimed at enhancing the dynamic agility of robots. Animals naturally adjust their gait across varying terrains, with beam walking being a notable skill facilitated by their 3 Degree of Freedom (DoF) hip joints[1]. This skill involves a complex interplay of perception and action[2], demanding accurate self-body state awareness and environmental estimation. In robotics, replicating such adaptive motion can greatly improve a robot's versatility in unfamiliar environments. Previous methods for robotic locomotion control have primarily used model-based approaches[3, 4, 5], focusing on joint controllers and requiring detailed system dynamics modeling. These methods often lack perception integration, rely heavily on world frame location data, and suffer from inadequate state estimation accuracy. Additionally, they typically require extensive tuning of control parameters and additional robot components. Recent advances in robot parkour[6, 7, 8, 9] have introduced various gaits to enhance robot agility, such as high climbs and long jumps. However, the flexibility in leg and foot positioning remains underexplored, potentially limiting action feasibility in certain terrains. Still,

7th Conference on Robot Learning (CoRL 2023), Atlanta, USA.

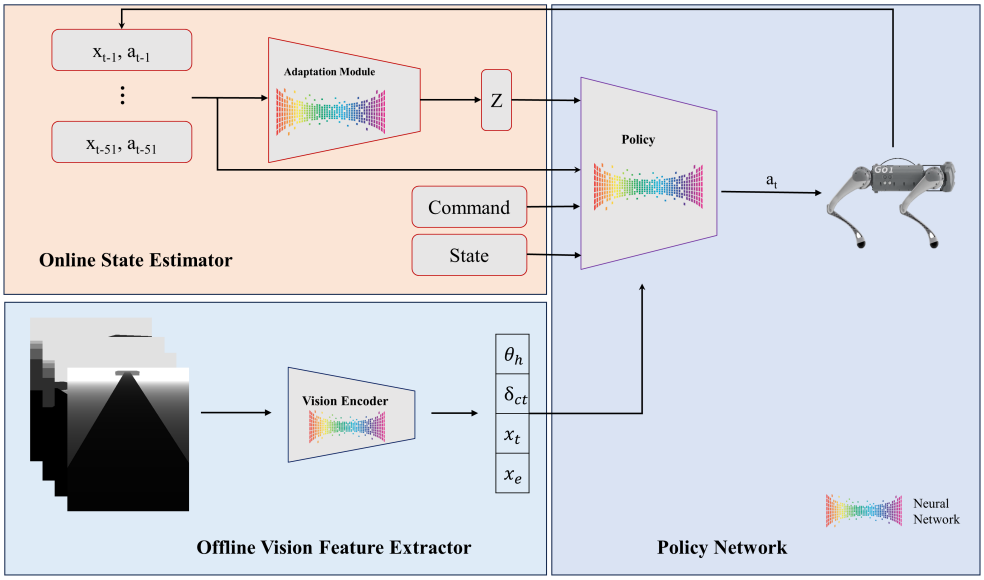

Figure 1: This figure introduces the algorithm pipeline of robot beam-walk learning. **Online State Estimator**: It takes the history state-action pairs to estimate the latent environment parameters. Then, pass the latent parameter Z into Policy Network along with history data, command, and proprioception states. **Offline Vision Feature Extractor**: This pre-trained perception module extracts the beam and platform features from raw depth images. The vision encoder maps the input depth images to interpretable 4D state vectors. **Policy Network**: Takes state estimator and feature extractor results as network input, and output the current action a_t. Afterwards, passing the a_t and s_t into training buffer.

learning-based approaches[2, 10] have shown promise in providing additional agility for quadruped robots, especially intergrated with egocentric vision inputs.

Our proposed framework for robot beam walking leverages the advantage of learning-based approach to boost the agility of locomotion tasks. It processes an embedded vector of depth image and robot intrinsic data to determine optimal joint positioning. While previous work[3, 11, 5] focused on how to balance the robot under fixed gait, we are focusing on proposing solutions to accomplish such task in a more interpretable and generalizable manner. The training process is divided into three phases: walking policy training, beam-walking policy training with soft dynamics constraints, and fine-tuning with hard dynamics constraints. Initially, a simple walking policy is trained without constraints. Next, the robot is trained on a beam placed on the ground, introducing a penalty for stepping off the designated width. The beam width is gradually decreased during training. Finally, the beam is elevated, and the training focuses on maintaining balance and adherence to the beam. Our reward function considers walking velocity, balance, and robot dynamics. Unlike other tasks[10, 12, 13, 14] that use direct depth image inputs, we convert the depth image into a 4-dimensional state vector for policy network input. Other than passing all the image data points into policy network, we built a pre-trained perception module that maps input images to useful environment interaction parameters following a supervised learning manner. More than increasing the training efficiency of the whole pipeline, such implementation also provide more intuitive solutions during during decision-making process.

The main contributions of this paper includes:

- **A three-stage RL method for training highly dynamic and agile agent**. This pipeline is designed to enhance the mobility of robots, particularly in terrains requiring high-level bal-

ance skills. The method allows for the effective training of agents to perform dynamically and agilely in challenging environments.

- **Generalization to different width and robot initialization pose**. Robot can operate on various width of beams, meanwhile it could adapt to different initialization point and heading angles in a zero-shot manner.
- **Customized 4D perception input**. We have introduced a state-based input strategy for visual input system to streamline and enhance the efficiency of RL training. By encoding raw images into 4-dimensional state vectors, we provide the policy network with crucial environmental interaction parameters. This approach not only simplifies the training process but also yields more intuitive and interpretable results, significantly advancing the field of robot perception and interaction.

2 Related Work

Enhanced Balance for Legged Robots. There have been some prior work on using optimal control method to enhance the balance of quadrupeds robot without extra load [4], which requires access to the world frame location. In [15], the authors balance a quadraped in a two-leg under-actuated pose. However, none of the previous work have hardware deployment of continuous locomotion on a narrow beam. There are some previous work that equip a reactive wheels on biped to improve walking efficiency [16, 17] and stabilize during high-speed running [5]. [3] equipped two reactive wheels at the back of quadruped robot, and has a hardware demonstration. However, the extra on-body load restrict the other mobility of the robot. More than reactive wheels, [18, 19] proposes to use tails to balance the robot, still, the issue caused by extra on-body load exists.

Learning-based Agile Locomotion. Learning-based agile locomotion is rapidly advancing beyond traditional model-based control methods[2, 20, 21], which used predefined controllers and elevation maps [22, 23, 24, 25] for legged robots. Researchers are now focusing on learning-based techniques that offer greater adaptability across varied environments [26, 27, 28, 29]. Innovations include unified locomotion policies [30] and game-inspired training for robots [31], with innovation of learning algorithm in the field of robot parkour, demonstrating robust handling of inaccuracies and advanced skill learning. [8, 9].

Vision-based Locomotion for Legged Robots. In robot perception, [14] develop an approach using differentiable planners and Vision Transformers for path planning in quadruped robots, utilizing 2D maps and obstacles as inputs. [13] focus on vision-based planning for legged robots, separating the process into foothold selection and pose adaptation. [12] introduce VP-Nav, a method that integrates low-level locomotion and high-level navigation with vision and proprioception for robust navigation in complex terrains. [10] showcase the first end-to-end locomotion system using egocentric vision for navigating stairs, curbs, and gaps. Finally, [32] present ViTAL, a strategy for selecting optimal footholds and robot poses for successful navigation.

3 Robot Beam-walk Learning

We aim to develop a beam-walking algorithm that leverages onboard depth imagery and preconceived inputs to guide the joint positioning of an economical robot, enabling it to execute beam-walking maneuvers. Diverging from traditional model-based methods proposed by Lee et al. [3], our learning-based approach capitalizes on visual data and enhanced mobility. The initial step involves distilling pertinent information from raw images to serve as the state input for the policy network. Owing to the distinctive tasks on beam-walking, we can describe depth imagery through a 4D state vector, encompassing the heading angle (θ_h), cross-track distance (δ_{ct}), and distances from the robot to both the beam tip (x_t) and end (x_e). Drawing inspiration from the robotic parkour training methodology outlined by [9], we proposed a three-phase training framework to inculcate beam-walking proficiency, as depicted in Figure 3.

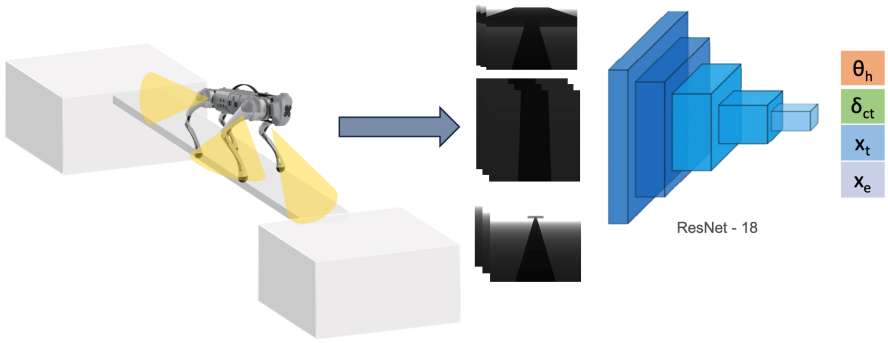

Figure 2: This figure illustrates the perception feature extraction pipeline. The depth images are collected through three ob-body cameras, placed at the front, back, bottom-side of the robot separately. Each fusion of images would give one set of states, i.e heading angle θ_h, cross-track distance δ_{ct}, distance to beam tip x_t, distance to beam end x_e, which then will be passed into policy network as part of the input.

The first phase focuses on teaching the walking policy sans gait constraints or feet distance penalties; the primary objective is to propel the robot forward along its heading. Progression to the second phase, beam-walking with soft dynamic constraints, is contingent upon the walking motion achieving a pre-set reward benchmark. Here, penalties ensue if the robot's feet transgress predetermined zones, with these 'soft beam' regions being dynamically configured to foster incremental gait refinement and balance. The final stage of RL fine-tuning elevates the robot above ground level onto a realistic dynamic platform, where falling from the beam precipitates episode termination and reduced rewards. Throughout the training, we employ straightforward, intuitive reward functions, minimizing the dependence on elaborate reward shaping or in-depth system knowledge. The training strategy, showcased in Figure 1, streamlines the learning process, diminishing the necessity for complex reward structures and extensive prior system comprehension.

3.1 Perception Module Learning

Different from conventional robot locomotion approaches, where they either need to have access to the world-frame location[3, 4] or directly pass the raw image as input[9, 8], we propose a method that maps the on-body depth camera data to a 4D state vector that represents the robot-beam interaction parameter. The raw depth images will be captured in real-time from three on-body cameras, i.e one front camera, one back camera, and one camera equipped at the bottom part of the robot body. The encoded state vectors include the information of robot heading angle θ_h, robot-beam cross-track distance δ_{ct}, robot-to-beam-tip distance x_t, robot-to-beam-end distance x_e. Perception module is illustrated in Figure 2. θ_h represents the angle between beam and robot heading direction in radians. δ_{ct} represents the projection of robot center of mass on the center of beam. The usage of x_t and x_e are essential, since we have to distinguish the robot current location state with respect to the platform and beam. The limit of x_t and x_e are based on the on-body camera coverage range d_{camera}. In our setting, the coverage range $d_{camera} = 0.7m$ The distance state x_t, x_e are given as follows:

$$(-d_{camera}, -d_{camera}) \leq (x_t, x_e) \leq (d_{camera}, d_{camera})$$

The usage of two state to represent the on-beam status check is required to model such diverse scenarios. For example, if the robot front legs are on the beam where the back legs are on the platform, the observed distance state x_t will be a positive value which less than d_{camera} since part of its body surpass the beam tip, and x_e will equals to negative value that equals to the max range of camera detection, i.e d_{camera}, due to none of the camera detect the beam end within the camera FOV. Another cases of robot-in-the-middle-of-beam will give $x_t = -d_{camera}$ and $x_e = -d_{camera}$, since none of the camera can detect neither the beam tip nor the beam end.

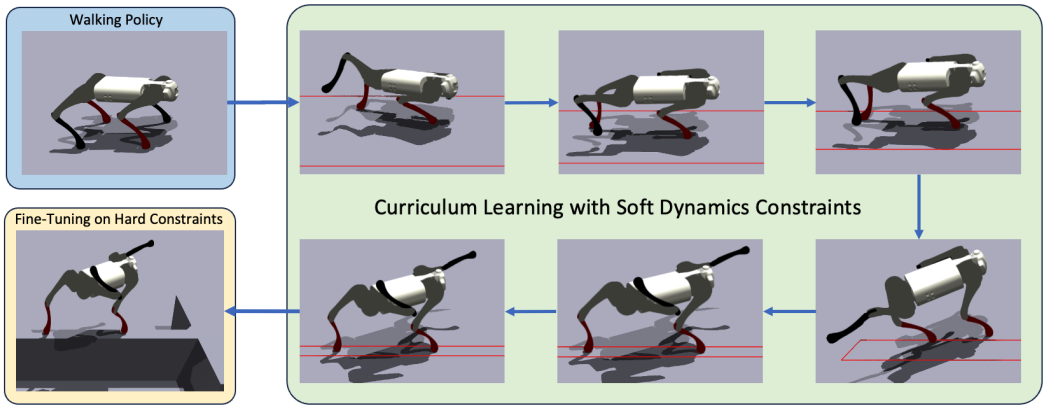

Figure 3: The training pipeline contains three phases. It starts with training the walking policy on ground without constraints, and then trains the agent using soft dynamics constraint with an automatic curriculum. Finally, as the robot could perform well under soft constraints, the policy will be fine-tuned on real beam with hard dynamics constraints

3.2 State-based RL Training

Given the input depth images will be mapped into 4D state vectors, the training pipeline have been simplified to pure state-based RL formulations. Considering future sim-to-real implementations, we have randomized the initial positions of the robot around the center point of the standing platform. The initialization of robot position consists four parameters, i.e $x^{init}, y^{init}, z^{init}, \theta_h^{init}$. We randomize the placement of x^{init}, y^{init} within the range of $(-0.1, 0.1)$ meters, and the initial heading angle θ_h^{init} within the range of $(-0.1, 0.1)$ radians, while keeping z^{init} constant. With the consideration of exploring more different gait to walk pass the beam, we implemented some simple reward, and try to discourage some feet touch the ground. To train an adaptive policy, we used the privileged state information to get the four robot-beam interaction states as the policy input. More than interaction states, the policy input also consists proprioception $s_t^{proprio} \in \mathbb{R}^{45}$, previous 50 steps of actions $a_{t-1\ldots t-50} \in \mathbb{R}^{12\times 50}$, and command scalars $s_t^{cmd} \in \mathbb{R}^5$. The commend values and tracking perforamnces are illustrated in Figure 5. The policy network then outputs the target joint positions $a_t \in \mathbb{R}^{12}$.

Reward Design. Reward Design composes three sections, including r_{dyn}, r_{mech}, and $r_{beamwalk}$. r_{dyn} includes dynamic properties of the robot such as velocity tracking, body height regulation, and orientation check. r_{mech} contains the mechanical properties of the systems, for example, torques and position limit regulations. $r_{beamwalk}$ consists rewards on restricting robot beam walking skills such as feet distance, and etc. Each component in the reward function have different weights, detailed reward scales and description can be found in Table 1.

Walking Policy Training. We trained the walking policy without any penalties on the foot width or constraints. The reward formulation here is

$$r_{total} = r_{dyn} + r_{mech}$$

where the r_{dyn} regulates target velocity and expected body height, and r_{mech} regulate joint space displacement and motion smoothness. The training has been done through Proximal Policy Optimization. Once the reward function converges, the constraints on beam walk will be added.

Curriculum Learning with Soft Dynamics Constraints. Instead of regulate the robot feet into narrow spaces directly, we propose a automatic curriculum that decrease the soft beam width as performance increases. The regulated beam width started with 40 cm and gradually decrease to 3cm. From the demonstration in Figure 3, we could observe that the robot tried to find the best policy that would fit the current width requirements, which boosts the training efficiency significantly. The reward function for soft dynamics constraint training have the additional beamwalk penalties, i.e

$$r_{total} = r_{dyn} + r_{mech} + r_{beamwalk}$$

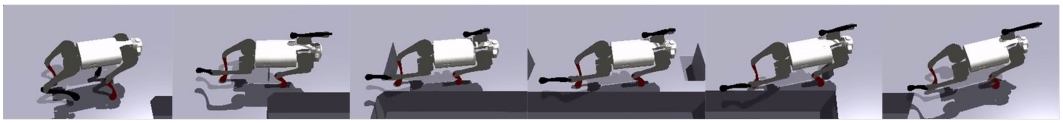

Figure 4: The beam walking policy we train enables the robot to walk stably through a narrow beam.

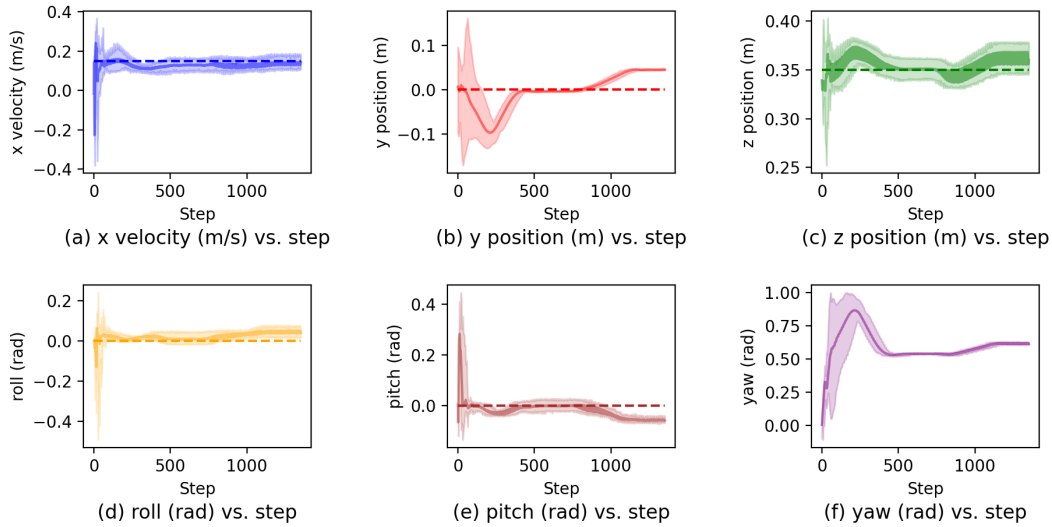

Figure 5: Pose and velocity tracking of the robot during beam walking. The velocity v_x, the positions p_y and p_z, and the orientations θ_{roll} and θ_{pitch} are highly aligned with the desired commands (dashed lines) when the robot is walking within the beam. The robot is able regulate the highly variable initial states to stable ones before entering the beam, especially the widely distributed p_z and θ_{yaw}.

During PPO training, the episode will not terminate if the agent step outside of the designed region, rather would receive penalties based on the distance between feet and beam center. Similar to phase 1, we will transfer to next phase when the reward function has converged in the most narrow cases.
Fine-tuning on Hard Dynamics Constraint. In the fine-tuning stage, we maintain the same reward structure because the criteria remain consistent with the earlier phase. However, we elevate the platform and the beam, introducing a new challenge: the robot must now balance on the elevated beam while maintaining a specified velocity, as illustrated in Figure 3. A misstep resulting in the robot stepping off the beam will cause a fall and consequently terminate the training episode, which then leads to low rewards.

4 Experiments

Configuration and Training. To train the mentioned policies, we choose Issac Gym [33] that is highly GPU-parallel as the simulator. The terrain for both the standard walking policy and the beam walking policy under soft dynamics constraints is a long platform with a width of $1.5m$ without height lift off the ground, which basically guides the robot to walk forward. For fine-tuning in the hard dynamics-constraint environment, we built a bridge beam to connect two platforms that have the same widths as the previous terrain. The bridge beam is slightly wider than Unitree Go1's foot, the widths of which are $10cm$ and $4cm$ respectively. As shown in Figure 4, after training the robot for $15,000$ iterations in 5096 parallel environments, the robot is able to traverse through the bridge beam stably. Next, We will present some quantitative analysis on the robot beam-walking policy performance.

Pose and Velocity Tracking. During beam walking, the robot is expected to follow a predetermined body state for enhanced stability and customizability. We collect 100 beam walking trajectories with the trained policy and visualize six critical parameters of body state in Figure 5, including velocity

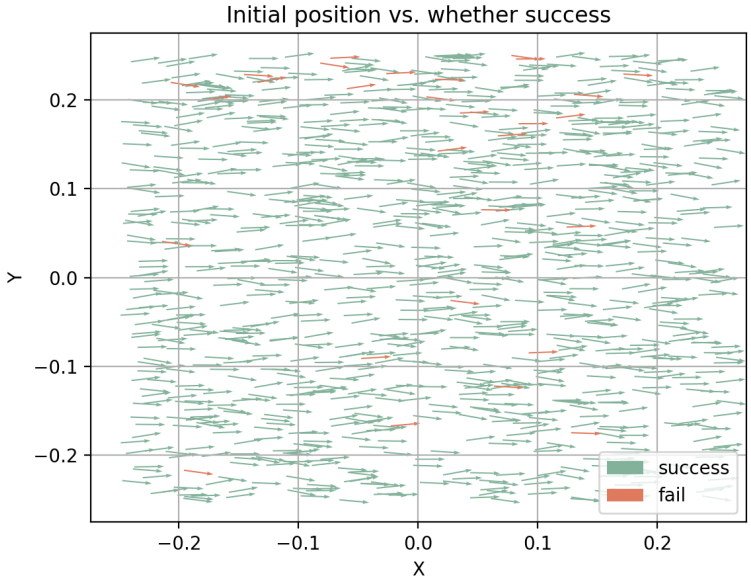

Figure 6: Initialization distribution of the robot during test. During training, the robot is initialized within the position of $[-0.1m, 0.1m]$ in both x and y direction, and with the yaw angle of $[-0.1rad, 0.1rad]$. During test, the robot is widely initialized within the position of $[-0.25m, 0.25m]$ in both x and y direction, and with the yaw angle of $[-0.25rad, 0.25rad]$. The start point of an arrow is the initial position of the robot, while the direction of an arrow points to the heading direction of the robot. The robot achieves to walk through the beam with a success rate of 97.2% under such out-of-distribution initialization.

in x-axis v_x, positions in y-axis p_y and z-axis p_z, and orientations θ_{roll}, θ_{pitch} and θ_{yaw}. As shown in Figure 5, the positions p_y and p_z and the orientations θ_{roll} and θ_{pitch} are highly aligned with the desired zero commands, meaning that the robot can keep the body stable, although the actions of the feet are highly dynamic. This would be extremely important if the robot is performing tasks that require high stability of its body, such as carrying fragile objects while walking through the beam. Furthermore, the robot is able to follow the desired velocity along the beam direction, demonstrating the potential to track a customized velocity if it is observed as a command. In addition, the last figure of Figure 5 shows that our trained beam walking policy enables the robot to regulate its body, even though the initial pose is widely configured.

Generalization to Initial Pose. In real world, it is very difficult to precisely place the robot to a specific initial pose relative to the beam. Therefore, the robot should be able to regulate itself before entering the beam under different initial poses. To achieve this, during training, the robot is randomly initialized in the position within $[-0.1m, 0.1m]$ in both the x and y directions, and within the yaw angle of $[-0.1rad, 0.1rad]$. In order to test the policy's generalization to out-of-distribution initial poses, we collect 1000 beam walking trajectories with the x and y positions range of $[-0.25m, 0.25m]$ and the yaw angle range of $[-0.25rad, 0.25rad]$. To our surprise, the robot performs beam walking very well with a success rate of 97.2% under such variable initial pose, which is vividly depicted in Figure 6 with the green arrows representing the initial body states of the successful beam walking cases and the red arrows representing those of the failed cases. We also notice that the initial poses that cause a failed beam walking generally have a relatively larger yaw angle, meaning that the robot tends to have better adaption ability to the initial positions.

Feet Trajectory on the Beam. For the beam walking task, the narrower the beam, the harder it is for the robot. Although the width $(10cm)$ of the beam is strictly narrow for the robot, we expect the robot to pass through a narrower beam with the trained beam walking policy. Therefore, we add a

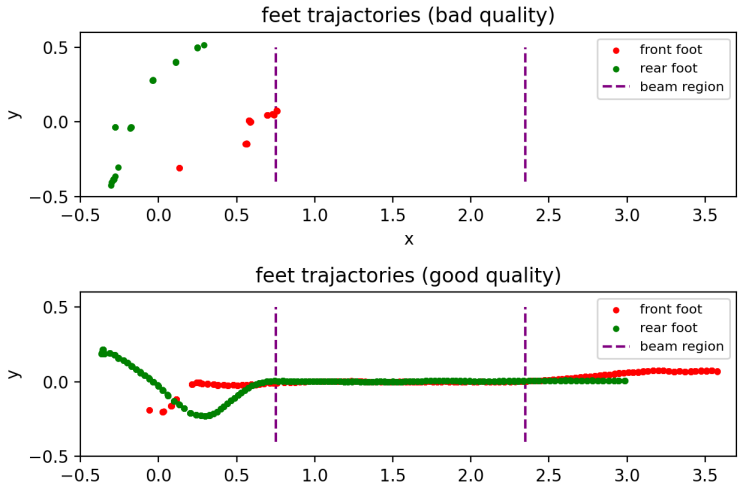

Figure 7: Feet trajectory of the robot during beam walking. The upper trajectory is a failed case where the robot is initialized with too large yaw angle relative to the beam, while the bottom trajectory is a successful one. The red points represent the positions of the front foot while the green points represent those of the rear foot. In the successful case, both of the feet are compactly tapping between the center of the beam.

reward to encourage the robot feet to lie in the center of the beam during training. Two trajectories of the robot feet sampled from 1000 trajectories mentioned above are shown in Figure 7. From the visualized positions of the feet within the beam region, the feet do lie very close to the center of the beam. Quantitatively analyzing the distance between the feet and the center of the beam for all the successful cases of the 1000 trajectories, the mean is $0.205cm$ and 0.341, while the std is 0.143 and 0.213 for the front foot and the rear foot respectively, which implies that the robot has the potential to walk through a beam with a width that is only $1cm$ wider than the robot foot.

5 Conclusion and Future Plans

In this paper, we present a three-stage RL training framework that incorporates a pre-trained feature extractor, enabling a quadruped robot to master beam-walking under extremely narrow conditions. Notably, our trained model adopts a beam-width-independent policy, which allows for immediate adaptation to beams of varying widths without additional training. The integration of a specialized pre-trained perception module significantly clarifies and simplifies the RL training process within the visuomotor architecture. Our findings reveal that the agent can navigate exceedingly narrow beams proficiently, regardless of the initial heading angle and position. We have conducted comprehensive evaluations on the agent's performance, focusing on the trajectories of the robot's body and feet.

Currently, our experiments are conducted entirely within a simulated environment. Although we have addressed numerous sim-to-real transfer concerns, there are still challenging aspects that require enhancement. For instance, the precision needed for beam-walking demands centimeter-level accuracy in beam estimation, which is difficult to achieve without meticulous calibration or advanced processing. Beyond conducting real-robot experiments, our future goal is to expand the applicability of our work to curved beams, thereby aligning more closely with real-world scenarios.

Acknowledgments

This is a course project for 16-831 Introduction to Robot Learning at Carnegie Mellon University. We appreciate all the feedback from Prof. Deepak Pathak and TAs.

References

[1] J. Yamaguchi, S. Inoue, D. Nishino, and A. Takanishi. Development of a bipedal humanoid robot having antagonistic driven joints and three dof trunk. In *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*, volume 1, pages 96–101 vol.1, 1998. doi:10.1109/IROS.1998.724603.

[2] R. Yang, M. Zhang, N. Hansen, H. Xu, and X. Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers, 2022.

[3] C.-Y. Lee, S. Yang, B. Bokser, and Z. Manchester. Enhanced balance for legged robots using reaction wheels. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9980–9987, 2023. doi:10.1109/ICRA48891.2023.10160833.

[4] C. Gonzalez, V. Barasuol, M. Frigerio, R. Featherstone, D. G. Caldwell, and C. Semini. Line walking and balancing for legged robots with point feet. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3649–3656, 2020. doi:10.1109/IROS45743.2020.9341743.

[5] J. Park, J. Lee, J. Lee, K.-S. Kim, and S. Kim. Raptor: Fast bipedal running and active tail stabilization. In *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 215–215, 2014. doi:10.1109/URAI.2014.7057424.

[6] L. Smith, J. C. Kew, T. Li, L. Luu, X. B. Peng, S. Ha, J. Tan, and S. Levine. Learning and adapting agile locomotion skills by transferring experience. *arXiv preprint arXiv:2304.09834*, 2023.

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

[8] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots, 2023.

[9] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning, 2023.

[10] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged locomotion in challenging terrains using egocentric vision, 2022.

[11] S. Yang, H. Choset, and Z. Manchester. Online kinematic calibration for legged robots. *IEEE Robotics and Automation Letters*, 7(3):8178–8185, 2022. doi:10.1109/LRA.2022.3186501.

[12] Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, and D. Pathak. Coupling vision and proprioception for navigation of legged robots, 2022.

[13] S. Fahmi, V. Barasuol, D. Esteban, O. Villarreal, and C. Semini. Vital: Vision-based terrain-aware locomotion for legged robots. *IEEE Transactions on Robotics*, 39(2):885–904, Apr. 2023. ISSN 1941-0468. doi:10.1109/tro.2022.3222958. URL `http://dx.doi.org/10.1109/TRO.2022.3222958`.

[14] J. Liu, S. Lyu, D. Hadjivelichkov, V. Modugno, and D. Kanoulas. Vit-a*: Legged robot path planning using vision transformer a*, 2023.

[15] M. Chignoli and P. M. Wensing. Variational-based optimal control of underactuated balancing for dynamic quadrupeds. *IEEE Access*, 8:49785–49797, 2020. doi:10.1109/ACCESS.2020.2980446.

[16] T. L. Brown and J. P. Schmiedeler. Reaction wheel actuation for improving planar biped walking efficiency. *IEEE Transactions on Robotics*, 32(5):1290–1297, 2016. doi:10.1109/TRO.2016.2593484.

[17] T. L. Brown and J. P. Schmiedeler. Energetic effects of reaction wheel actuation on underactuated biped robot walking. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2576–2581, 2014. doi:10.1109/ICRA.2014.6907228.

[18] T. Libby, A. M. Johnson, E. Chang-Siu, R. J. Full, and D. E. Koditschek. Comparative design, scaling, and control of appendages for inertial reorientation. *IEEE Transactions on Robotics*, 32(6):1380–1398, 2016. doi:10.1109/TRO.2016.2597316.

[19] A. Patel and M. Braae. Rapid turning at high-speed: Inspirations from the cheetah's tail. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5506–5511, 2013. doi:10.1109/IROS.2013.6697154.

[20] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *IROS*, 2016.

[21] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *IROS*, 2018.

[22] P. Fankhauser, M. Bloesch, and M. Hutter. Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters*, 3(4):3019–3026, 2018.

[23] Y. Pan, X. Xu, Y. Wang, X. Ding, and R. Xiong. Gpu accelerated real-time traversability mapping. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 734–740. IEEE, 2019. doi:10.1109/ROBIO49542.2019.8961816.

[24] A. Kleiner and C. Dornhege. Real-time localization and elevation mapping within urban search and rescue scenarios. *Journal of Field Robotics*, 24(8-9):723–745, 2007.

[25] I.-S. Kweon, M. Hebert, E. Krotkov, and T. Kanade. Terrain mapping for a roving planetary explorer. In *IEEE International Conference on Robotics and Automation*, pages 997–1002. IEEE, 1989.

[26] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *Robotics: Science and Systems*, 2021.

[27] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. In *ICRA*, 2022.

[28] Z. Fu, A. Kumar, J. Malik, and D. Pathak. Minimizing energy consumption leads to the emergence of gaits in legged robots. In *Conference on Robot Learning (CoRL)*, 2021.

[29] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha. Learning fast adaptation with meta strategy optimization. *RA-L*, 2020.

[30] G. B. Margolis and P. Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior, 2022.

[31] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning, 2022.

[32] S. Fahmi. On terrain-aware locomotion for legged robots, 2022.

[33] 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, 2021.

6 Appendix

Table 1: This table provide the detailed description of all the reward components with corresponding scales.

Reward Class	Reward Function	Reward Scale	Description
r_{mech}	track_x_vel	0.5	Track x-direction velocity in world frame
	move_forward_x	0.5	Track hand-calculated velocity in world frame
	lin_vel_z	-0.02	Penalize z-axis linear velocity
	ang_vel_xy	-0.001	Penalize xy-axes angular velocity
	body_height	-1.0e-2	Penalize robot body height if $>$0.05
	orientation	-1.0	Penalize non-flat base orientation
r_dyn	dof_pos	-1.0e-5	Penalize if the joint actions are too differ from initial pos
	torques	-5.0e-5	Penalize high joint torques
	dof_vel	-1e-4	Penalize high joint velocity
	dof_acc	-5.0e-8	Penalize high joint acceleration
	foot_slip	-5.0e-2	Penalize hard contact
	collision	-1.0e2	Penalize self-collisions
	action_rate	-1.0e-3	Penalize action change rate
	dof_pos_limits	-10.0	Penalize close-to-limit joint position
	action_smoothness_1	-5e-2	Penalize high action rate except for step1
	action_smoothness_2	-5e-2	Penalize high action rate except for step1 and step2
$r\_beamwalk$	feet_inside_beam	-2.0	Penalize if robot step outside of beam range
	feet_near_center	-8.0	Penalize the distance between robot foot and beam center
	swing_contact	-0.4	Penalize if selected feet have contact with the ground
	y_distance	-0.2	Penalize if robot center is far from beam center