Vision-Guided Quadrupedal Locomotion in the Wild with Multi-Modal Delay Randomization

Chieko Sarah Imai[*1] **Minghao Zhang**[*2] **Yuchen Zhang**[*1] **Marcin Kierebiński**[1]
Ruihan Yang[1] **Yuzhe Qin**[1] **Xiaolong Wang**[1]

[1]UC San Diego [2]Tsinghua University

Abstract

Developing robust vision-guided controllers for quadrupedal robots in complex environments, with various obstacles, dynamical surroundings and uneven terrains, is very challenging. While Reinforcement Learning (RL) provides a promising paradigm for agile locomotion skills with vision inputs in simulation, it is still very challenging to deploy the RL policy in the real world. Our key insight is that aside from the discrepancy in the domain gap, in visual appearance between the simulation and the real world, the latency from the control pipeline is also a major cause of difficulty. In this paper, we propose *Multi-Modal Delay Randomization (MMDR)* to address this issue when training RL agents. Specifically, we simulate the latency of real hardware by using past observations, sampled with randomized periods, for both proprioception and vision. We train the RL policy for end-to-end control in a physical simulator without any predefined controller or reference motion, and directly deploy it on the real A1 quadruped robot running in the wild. We evaluate our method in different outdoor environments with complex terrains and obstacles. We demonstrate the robot can smoothly maneuver at a high speed, avoid the obstacles, and show significant improvement over the baselines. Our project page with videos is at `https://mehooz.github.io/mmdr-wild/`.

1 Introduction

Developing a robust controller for the quadrupedal robot to traverse complex, wild environments (Carlo et al., 2018; Di Carlo et al., 2018) is a challenging and important problem in robotics. Recent approaches show its wide range of applications such as delivery in the rugged terrains (Arena et al., 2006). With the advancement in robot learning for navigation and manipulation (Mirowski et al., 2017; Levine et al., 2016), Reinforcement Learning (RL) and Imitation Learning have also brought crucial improvements in legged locomotion (Peng et al., 2020; Hwangbo et al., 2019; Luo et al., 2020; Da et al., 2020) with proprioceptive state only. Specifically, we have witnessed impressive performance in the wild using RL incorporating robustness and adaptive ability (Lee et al., 2020; Kumar et al., 2021).

However, only given proprioceptive information, the *blind* RL controller addresses challenging scenarios by training with large-scale randomized environment parameters (Lee et al., 2020; Xie et al., 2020c). While this technique delivers promising results for maneuvering on uneven and unknown-material ground, it's insufficient for more complicated tasks like avoiding obstacles that are hard to step over, or estimating accurate foot placement positions for safety. To overcome these challenges, the robot needs to perceive surroundings and plan the path to move. Thus recent works (Escontrela et al., 2020; Jain et al., 2020; Yang et al., 2021) introduce visual perception to the robot by using a

[*]Equal contribution. Name ordered alphabetically.

Deep RL Workshop on Neural Information Processing Systems 2021.

depth camera or LiDAR. Moreover, proprioceptive states only provide information about current or prior terrain, while vision provides information about near-future topography for developing better stepping skills. By combining the information from both the proprioceptive states and visual inputs, the robot can maneuver in intricate, wild scenarios as Figure 1 demonstrates.

Nevertheless, learning with visual inputs introduces new challenges in Sim2Real transfer for RL. While previous methods focus on narrowing the domain gap in observations using domain randomization and augmentation techniques (Tobin et al., 2017; Laskin et al., 2020b; Hansen & Wang, 2021), the latency in hardware has not been well studied. Latency exists in all parts of the real robot control system. The sensors' transmission, the control algorithm's computation, and the robot's response all introduce latency for perception and control. This problem is further magnified when the control policy requires both high-dimensional vision and proprioceptive state inputs. First, the deficiency of on-board computing resources in a quadrupedal robot causes severe latency in processing the visual inputs; by the time the action is computed from the visual policy, the robot could have already walked a few steps ahead. Thus the policy will need to be robust to the visual delay and still perform correct decision making. Second, the proprioceptive state inputs are still provided to the network at a high frequency, which causes misalignment between the proprioceptive state and the visual observation. Solving the two problems above is the key to transfer vision-based RL policy trained in simulation to the real robot.

In this paper, we propose *Multi-Modal Delay Randomization (MMDR)*, an asynchronous delay randomization technique for improving the robustness of RL policies in real robots. Our RL policy takes both proprioceptive states and observed depth images as inputs, and outputs the target angle for each robot joint in an *end-to-end* manner. During training, instead of forwarding both observations with fixed temporal intervals, we provide randomized asynchronous multi-modal inputs to the network. Specifically, we maintain two buffers online for two types of inputs (one for visual observation, another one for proprioceptive states). Each buffer will store a stream from one type of recent observations. We independently sample a sub-sequence of observations from each buffer, and forward both types of inputs to the policy network. In this way, we can simulate the time discrepancy between proprioceptive and visual signals in the real robot. MMDR simulates the accumulated latency of the whole system across all parts at once. The quadrupedal robot trained with RL using MMDR is more robust to the delayed visual inputs and the misalignment between two types of observations for locomotion control.

We experiment with our proposed method on arduous, in-the-wild maneuvering tasks with a Unitree A1 quadruped robot (Unitree, 2018) as shown in Figure 1. These environments include static or moving obstacles of different sizes and shapes, uneven terrains, and changing lighting conditions. Results show that applying MMDR during training significantly improves the performance and robustness for real-world deployment of the learned RL policy. Our policy can even generalize to environments with unseen or moving obstacles, where the A1 robot plans and maneuvers smoothly through the obstacles at a high moving speed.

We highlight our main contributions as follows:

- We present Multi-Modal Delay Randomization (MMDR), which models the latency of multi-modal inputs for Sim2Real RL policy transferred for vision-guided quadrupedal locomotion control.
- To the best of our knowledge, this is the first work that allows an end-to-end RL trained quadrupedal robot to maneuver in the wild under visual guidance.

2 Related Work

Sim-to-real with Domain Randomization. To address the sim2real gap for policy deployment in the real-world, domain randomization introduces variability to different components of the simulated environments during training, including variant physical parameters (Mordatch et al., 2015; Peng et al., 2018b; Tan et al., 2018; Li et al., 2021), visual attributes (Tobin et al., 2017; Pinto et al., 2018; Andrychowicz et al., 2020), and perturbations (Andrychowicz et al., 2020). Thanks to its simplicity and effectiveness, domain randomization has been widely used in learning legged robot control to complete real-world tasks with diverse locomotion skills (Xie et al., 2020b,a; Peng et al., 2020; Kumar et al., 2021; Lee et al., 2020). However, how to better apply domain randomization to address

Figure 1: Our method enables quadruped robots to traverse complex environments with obstacles of different shapes in the wild. The learned locomotion policy obtains different agile locomotion skills like flexibly turning before barriers and stably walking on steep lawn, using reinforcement learning. The policy is trained in simulation with our multi-modal delay randomization and deployed in the real-world, unseen scenarios without any adaptation or fine-tuning.

the sim2real gap for learning vision-guided legged locomotion in the real world (like the multi-modal latency) has not been well-explored yet. In this work, we propose multi-modal delay randomization to model the sim2real gap beyond the observation discrepancies.

Learning-based Legged Locomotion. Controlling a legged robot has been studied by the robotics community for a long time. Control theory and trajectory optimization approaches have shown great results on legged locomotion control (Gehring et al., 2013; Carlo et al., 2018; Di Carlo et al., 2018; Carius et al., 2019; Ding et al., 2019; Bledt & Kim, 2020). However, these methods require in-depth knowledge about the environment and substantial manual efforts for parameter tuning. As an alternative, reinforcement learning provides an autonomous learning paradigm for legged locomotion skills from self-exploration in complex environments (Kohl & Stone, 2004; Luo et al., 2020; Peng et al., 2018a, 2020; Tan et al., 2018; Hwangbo et al., 2019; Lee et al., 2020; Xie et al., 2020c; Iscen et al., 2018). Despite the successful application of RL on legged robots, most RL approaches depend only on proprioceptive input. To utilize the rich visual information, recent works introduce visual observation to legged locomotion learning in end-to-end (Escontrela et al., 2020; Yang et al., 2021) or hierarchical manners (Jain et al., 2020). But visual-guided locomotion policies learned by these methods are still limited to simple and indoor environments. With our method, we can successfully apply the learned policies in the wild directly.

Reinforcement Learning with Vision. To enrich the perception of the agent, many works have studied RL with visual input in navigation (Sax et al., 2018; Faust et al., 2018; Wijmans et al., 2020), locomotion control (Laskin et al., 2020a; Hansen & Wang, 2021; Yarats et al., 2021; Hansen et al., 2021), and manipulation (Jain et al., 2019; Levine et al., 2016, 2018). To generalize visual perception in variant tasks, Hansen et al. (2021) extend representation learning for out-of-distribution environments. Our work is most related to previous work leveraging multi-modal input for locomotion control (Heess et al., 2017; Merel et al., 2020; Jain et al., 2020; Escontrela et al., 2020; Yang et al., 2021). Instead of running the robot in simulators or labs, we show the vision-guided quadruped robot can run smoothly and safely in the wild.

Learning with Delay. The latency of sensing and control recently draws attention from robotic learning community. Li et al. (2020) introduces the concept of streaming perception into computer vision, unlike previous work focusing on offline efficient (fast) inference (Redmon et al., 2016; Liu et al., 2016). In RL area, continuous-time RL (Doya, 2000) and MDP with delay (Ramstedt & Pal, 2019; Katsikopoulos & Engelbrecht, 2003) has been proposed to address constant system latency in simulation. Beyond simulation, the continuous-time algorithms are developed for concurrent control (Andersen et al., 2015; Xiao et al., 2020; Lutter et al., 2021) in real-world control tasks. For modeling delay, previous works (Tan et al., 2018; Yang et al., 2020) randomize the observation latency for proprioceptive state-only quadrupedal locomotion policies. When using information from different sources, multi-modal synchronization also brings new difficulties on stable control (Olson,

2010; Liu et al., 2021). In this work, we address the delay problem in learning-based vision-guided control with a simple yet effective randomization technique.

3 Delay Randomization in a Multi-Modal Control System

The Markov property is fundamental to most RL algorithms rely on (Puterman, 1994). However, the latency in real robot hardware jeopardizes the Markov property. This makes it difficult to train policies deployed in the real world. Worse still, in a multi-modal robot system that perceives information from multiple modalities, e.g. vision and proprioception, the latency of different sensors varies greatly. Thus, multiple sensor data fusion will become a non-trivial task in the real world, compared to the synchronized settings that are commonly used in RL literature (Ibarz et al., 2021), which do not face these difficulties. To tackle this challenge, we propose Multi-Modal Delay Randomization (MMDR), which explicitly models the latency from different sources in a real robot system, as shown in Figure 2. In this following section, we will first introduce the latency of a real robot in Section3.1, then discuss how we mitigate the latency issue via our MMDR.

3.1 Latency in Real Robot RL

In common RL settings, an agent perceives observations from the environment, computes the next step action based on the policy, then forwards this action to the environment. Most physical simulators (Coumans, 2015; Todorov et al., 2012; Koenig & Howard, 2004; Makoviychuk et al., 2021) assume a synchronized observation-action loop. In this synchronized setting, the environment will wait for the next action and stay unchanged before that. However, the real world is asynchronous and never waits for the policy. Any latency in the observation-action loop will influence the execution of the action, which in turn lowers the task performance. Figure 2 illustrates delays in multiple stages: (i) Transmission delay to receive sensory readings; (ii) Processing delay to compute the action from an observation, commonly the time required by the neural network; (iii) Actuation delay to execute the action via the motors. Among all three types of delays, the processing delay is the most significant. With a battery-based power supply and limited memory, the computational resources on-board often can not afford instant network inference, especially for high-dimensional visual input. By the time the control signals are applied, the robot's surroundings have already changed.

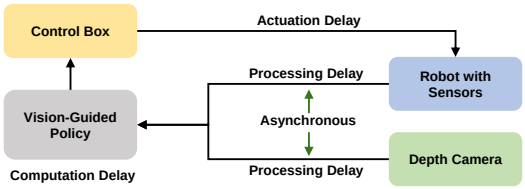


Figure 2: *Multiple sources of latency for a multi-modal control policy.* We take into account delays in perception processing, policy computation, and control execution.

3.2 Multi-Modal Delay Randomization

To simulate the same control flow as in the real world, MMDR provides randomized latency and asynchronous multi-modal observation in the simulation. We randomize the sampling of the proprioceptive state and visual observation separately. This allows us to utilize the domain-specific characteristics and simulate independent latency for different modalities.

We represent the proprioceptive state as an 84D vector, where each digit of the vector has specific physical meaning. To maintain the fidelity of the simulation for RL, the simulation frequency is generally several times higher than the control frequency of the robot. Our method reads the proprioceptive state at every simulation step and uses a queue with a fixed length to store historical observations from the near

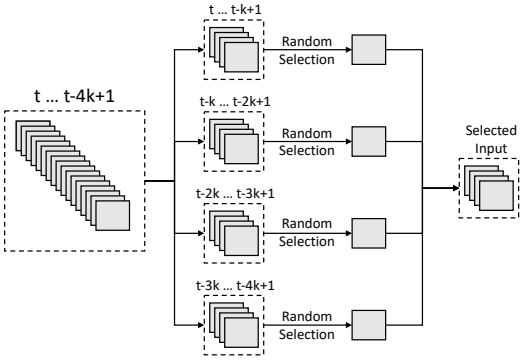
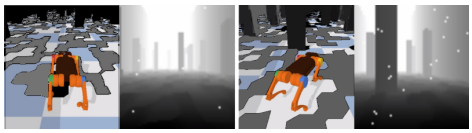


Figure 3: *Visual delay randomization.* We maintain a depth map buffer with a length of $4k$, where k is a hyper-parameter. From each consecutive k frames, we randomly select one frame and stack them as visual input in training.



(a) *Simulated Environments.* For each example, the environment is shown in the left, and the corresponding observation in the right. Aside from the randomized obstacles and terrain, we add randomly placed white spots in the visual observation to simulate the real observation from the depth camera.

(b) *Real Environments.* We evaluate the generalization ability of policies in real-world scenarios with obstacles of different shapes and sizes on a complex terrain. Due to the noisy reading of the depth camera, some depth values are missing in the depth map, which breaks the shape of an object in the visual observation.

Figure 4: Samples from the simulator and the real world.

past. We also assume the proprioceptive state is
changing smoothly in the real world. Therefore, we sample a proprioceptive delay for each episode. Then we use linear interpolation to calculate the delayed observation from two adjacent states in the entire buffer, with the sampled delay.

We represent the visual observation as a four-stacked depth image, in order to maintain temporal information. A straightforward way to simulate latency for visual observations is to use the same randomized delay as for the proprioceptive state during training. However, in the real world, depth cameras can provide much fewer observations (around 30 Hz) compared to proprioceptive sensors (around 1K Hz). In this case, the transition from one frame to the next is no longer smooth. To simulate the perception latency for visual observation at a lower frequency, we obtain the simulated visual observation at every control step and store the near past frames into a queue. As illustrated in Figure 3, we store the most recent $4k$ depth maps as our visual observation buffer, split the whole visual observation buffer into four sub-buffers, then sample one depth map from each sub-buffer to create the visual input with randomized latency. Though there are multiple kinds of latency as we discussed in 3.1, what really influences the real-world deployment is the accumulated latency across the different stages. Therefore, the separated randomized latency for proprioceptive state and visual observation is enough for the policy training.

4 Reinforcement Learning in Simulation

With our MMDR, we are able to simulate the real-world latency in simulators. We train locomotion polices in the simulation using RL, under different settings for real-world deployment and comparison.

4.1 Simulation Environment

We perform our simulation using PyBullet (Coumans & Bai, 2016–2021). We train agents to control a Unitree A1(Unitree, 2018) robot to maneuver in an environment with obstacles and a complex terrain. The simulated obstacles are cuboid rigid bodies with random positions and shapes, which remain static throughout the episode. To force the robot to learn how to walk on natural, uneven grounds, we create complex terrain with random height fields, as shown in Figure 4a.

In all of the experiments, we use the same observation space, action space and reward function across all environments.

Observation Space. The observation of the robot is made up of proprioceptive state and visual observation. The proprioceptive state consists of (i) 12D robot joint rotation, (ii) 4D IMU sensor reading (angle and angular velocity of roll and pitch), (iii) 12D last action executed by the policy. The proprioceptive input is an 84D vector containing three proprioceptive states. The visual input consists of four stacked depth images of shape 64×64. All the depth images come from the depth camera mounted on the head of the robot. To constrain the scale of the visual observation, depth values in visual observation are clipped to $[0.3, 10]$m.

Action Space. The action space for the policy is the target joint angle for each joint of the robot. Target angles are converted to motor torques using a default PD controller.

Rewards. In general, our reward function encourages the robot to safely move forward (along the x-axis in simulation) with a target speed (0.35m/s) while minimizing the energy cost. Specifically, the reward function is given by: $R = \alpha_{\text{moving}} R_{\text{moving}} + \alpha_{\text{safe}} R_{\text{safe}} + \alpha_{\text{energy}} R_{\text{energy}}$ where $\alpha_{\text{moving}} = 1, \alpha_{\text{safe}} = 0.005, \alpha_{\text{energy}} = 0.1$. The moving reward contains two terms. One encourages the robot to move forward at target speed and the other penalizes the speed along the z-axis to keep the robot walking smoothly. The safe reward is 1.0 for each step until the robot falls, in which case the episode terminates. The energy reward penalizes the energy consumption represented by the square of motor torques.

4.2 Network Architecture and Training Details.

We train all methods with 10M samples. We use PPOp Schulman et al. (2017) for policy training and utilize the generalized advantage estimator (GAE) (Schulman et al., 2016) to stabilize the training process (Peng et al., 2018a, 2020). We use a batch-size of 16384 and split it into 16 mini-batches. We use Adam optimizer (Kingma & Ba, 2015) with learning rate of 1e-4 for both policy and value networks. In our experiment, the policy network and value network share the same architecture as Figure 5. We use a 2-Layer MLP encoder for proprioceptive input and a 3-Layer CNN for visual input to get encoded features from both modalities. The encoded features from both modalities are 256D vectors. We then concatenate the encoded features to get a unified feature and feed it into the additional 2-Layer MLP to get the action distribution or value prediction. For better sample efficiency and stable training, the CNN for visual input is shared by the policy and the value function to capture a consistent visual representation.

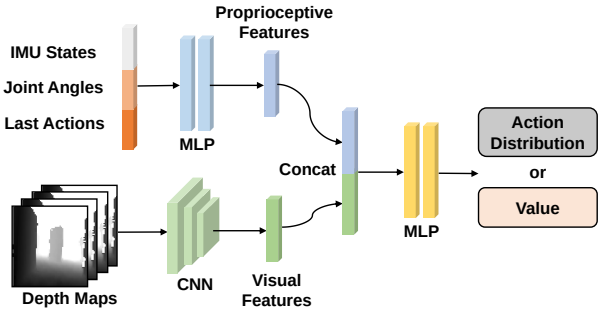


Figure 5: *Network Architecture.* We use separate encoders for multi-modal inputs to get domain-specific features and use a MLP to get the value or action distribution from concatenated features.

4.3 Domain Randomization

Besides the multi-modal latency, we leverage domain randomization on various physical parameters to narrow the sim2real gap in the observation space. We sample a group of values within specific ranges at the beginning of each episode. The randomized parameters and the corresponding ranges are listed in Table 1.

Parameters	Range
Mass ($\times$ default value)	[0.8, 1.2]
Motor Friction (Nms / rad)	[0.0, 0.05]
Motor Strength ($\times$ default value)	[0.8, 1.2]
Lateral Friction (Ns / m)	[0.5, 1.25]
Inertia ($\times$ default value)	[0.5, 1.5]
Proprioception Latency (s)	[0, 0.04]
Kp	[40, 90]
Kd	[0.4, 0.8]

Table 1: Variation of Environment and Robot Parameters.

Ambient light in the real world and the stereo nature of a depth camera result in occlusion in the depth image. Therefore, the original depth estimation from the camera is not available in certain visual regions of the depth map. We fill in all the missing values with the maximum depth (10m). As shown in Figure 4b, the shape of obstacles breaks because of these missing values, which makes the deployment of the visual-guided locomotion policy even more challenging. To simulate this phenomenon, we randomly sample 3-30 pixels in each depth map, and set the value of these pixels to the maximum depth. The modified depth map are visualized in Figure 4a.

4.4 Baselines

We compare MMDR with two baselines both in simulation and the real world. The No-Delay baseline is trained without delay randomization, and stacks the most recent 4 depth maps as visual

input. The Frame-Extract baseline stores the most recent $4k$ frames and stacks the first frames from every consecutive k frames as visual input to provide input with more temporal information. As shown in Figure 6, our method uses the same amount of temporal information as the Frame-Extract baseline, but models the real-world latency by performing random selection.

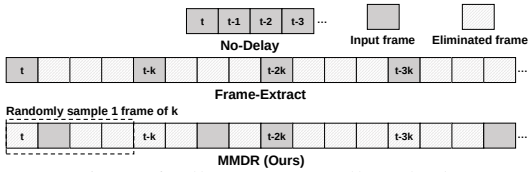


Figure 6: *Illustration of all methods.*

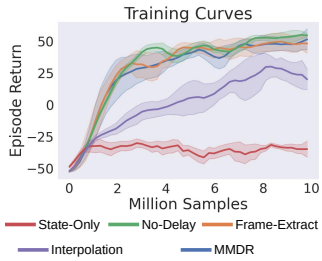
5 Experimental Results and Analysis

In our experiments, we evaluate the performance of policies using two metrics: (i) *Moving Distance*: the distance in the forward direction covered by the robot; (ii) *Collision Steps*: the number of robot steps during which the robot is in contact with an obstacle. The moving distance represents stability of the gait while the collision steps reflects safety. Please refer to our project page and supplementary video for visualizations.

5.1 Results in Simulation

We evaluate all methods in an environment with random delays varying from [0.04, 0.12]s. Aside from the static obstacles, we also compare the policies in environment with moving obstacles to evaluate the generalization ability in simulation.

Though MMDR introduces more uncertainty into the observation, the robot maintains comparable learning efficiency as shown in Figure 7. When tested in the same scenario with random delays, the policy learned with MMDR moves further with less collisions. This demonstrates the potential to better adapt to the real world.



	Moving Distance ↑
State-Only	$2.9_{\pm 0.5}$
No-Delay	$26.5_{\pm 5.0}$
Frame-Extract	$24.9_{\pm 3.1}$
Interpolation	$21.4_{\pm 2.7}$
Ours	$\mathbf{28.7}_{\pm \mathbf{7.7}}$

Figure 7: *Training curves and evaluation results of all settings.* All methods show similar sample efficiency in the environment that it's trained on, even if MMDR learns in a noisy environment. When evaluated in environment with random delay, our MMDR outperform all other baselines.

To test the generalization ability further, we include a modified scenario in which the obstacles will continuously move at a random speed and direction. Such a dynamic environment requires more accurate perception, so a slight delay may bring a very large impact like crashing. Table 2 shows our superiority on both traversing skill and safety, tested against an unseen dynamic environment. MMDR significantly increases the forward distance by nearly **100%**, and reduces the Collision Steps by **475%** to No-Delay and **340%** to Frame-Extract. We conjecture it is because asynchronously randomizing multi-modal delays provides data-augmentation for relative position and speed changes of surroundings. This allows the policy to be more adaptable to a dynamic environment during testing.

Ablation. We add two ablation studies to show the substance of vision and random selection. First, we train the blind policy in the same environment without visual input. We use a 4-Layer MLP for training, corresponding to State-Only in Figure 7. The State-Only agent learns almost nothing for such a complex task. Second, like how we model the proprioceptive state delay, we sample a delay t_d for each episode and use linear interpolation to calculate visual observation from two adjacent observations around $t - t_d$ at each step t (Interpolation in Figure 7). We compare MMDR with this smooth interpolation to show discrete sampling is more appropriate for vision. MMDR consistently outperforms them during training stages. We also find that though Interpolation does not perform well compared to both baselines during training, when evaluated on environment with moving obstacles, it generalizes better. It shows that when temporal information is crucial for decision-making, it's essential to take latency into consideration. We stop the two methods in simulator rather than real-world deployment due to the disadvantage in training phase.

(a) Box (b) Dense Box (c) Moving Box (d) Forest

Figure 8: Four Real world environments in our experiments.

	Moving Distance ↑	**Collision Steps** ↓
State-Only	$3.5_{\pm 1.3}$	$475.1_{\pm 261.3}$
No-Delay	$6.0_{\pm 1.7}$	$401.5_{\pm 118.0}$
Frame-Extract	$6.6_{\pm 2.5}$	$287.9_{\pm 264.5}$
Interpolation	$7.1_{\pm 0.7}$	$94.8_{\pm 12.3}$
Ours	$\mathbf{11.4_{\pm 1.9}}$	$\mathbf{84.4_{\pm 22.9}}$

Table 2: *Generalization evaluation in simulation with moving obstacles.* When evaluated in much more challenging environment with moving obstacles, MMDR not only performs significantly better than baselines in both metrics, but also is much more stable across different seeds.

5.2 Results in Real World

Given the performance of different methods in simulation, we conduct real-world experiments with these three methods: Ours, No-Delay, and Frame-Extract.

Robot Setup. In the real world, we perform evaluations using the Unitree A1 Robot (Unitree, 2018), a low-cost quadruped robot with 18 links and 12 degree of freedoms (3 for each leg). Our policy computes the target joint angles at 25 Hz, while the underlying PD controller computes the motor torques at 400 Hz. Kp and Kd are 40 and 0.6 respectively.

We first evaluate all the methods in scenarios similar to the training environments. We launch 9 trials on 3 seeds for each setting. The evaluation metric used in deployment is **Collision Count**, i.e. number of times robot hits the obstacle. This differs from Collision Steps used in simulation because in outdoors environments it is dangerous to let the robot continuously collide with objects. We place differently-sized boxes on a sloped lawn with different density. We refer to these environments as **Sparse box placement (Box.)** (as in Figure 8(a)) and **Dense box placement (Dense Box.)** (as in Figure 8(b)).

In these two environments, as demonstrated in Table 3, MMDR excels baselines by a large margin. MMDR moves twice as far as the other baselines do without colliding into any boxes in **Sparse box placement**, and maintains a similar performance when the obstacles become much denser. In contrast, Frame-Extract performs poorly in **Sparse box placement (Box.)** and the performance becomes worse as the density of boxes increases. We speculate that using visual observation covering longer time span makes the policy become more sensitive to the latency in the real world. This phenomenon suggests that modeling the real world latency is essential.

Beyond static boxes on a sloped lawn, we deploy policies in two more challenging scenarios: (i) **Moving box (Moving Box.)**: the box is moving slowly in the same sloped lawn as previous environments as Figure 8(c) shows; (ii) **Forest covered with branches and fallen leaves (Forest.)** as Figure 8(d) shows. These two tasks are more challenging than the static box environments, for the diverse lighting condition, complex terrains, and dynamic obstacles. We find that in the **Moving Box** environment, the improvement our method obtained over the baselines becomes larger. We conjecture that this happens because when the environment is dynamical, the latency in the multi-modal control system becomes more influential. So it gets harder to predict the movement of the obstacles and act accordingly with latency. This result also indicate that it's necessary to simulate latency for vision-guided locomotion policy.

For the Forest environment, the challenges mostly come from the unseen obstacles (trees), the branches, and leaves on the ground. In Forest environment, the policy trained with MMDR still obtains large improvement over the baselines in the moving distance, while having similar collision results. Stepping on branches or fallen leafs can cause unexpected state transition that don't happen

	Box.	Dense Box.	Moving Box.	Forest.
	Moving Distance ↑			
No-Delay	444.7±115.0	447.6±147.6	505.7±120.5	733.8±118.0
Frame-Extract	358.4±155.3	280.0±79.8	380.4±261.6	572.4±256.7
Ours	**859.9±271.4**	**641.2±49.9**	**973.0±148.4**	**992.5±335.0**
	Collision Count ↓			
No-Delay	0.0±0.0	1.0±0.27	0.33±0.27	0.22±0.16
Frame-Extract	0.56±0.42	1.21±0.47	0.44±0.62	0.22±0.32
Ours	**0.0±0.0**	**0.9±0.6**	**0.33±0.0**	**0.22±0.16**

Table 3: *Real world deployment performance.* MMDR significantly improve the maneuvering skills in complex wild environments for forward efficiency and safety.

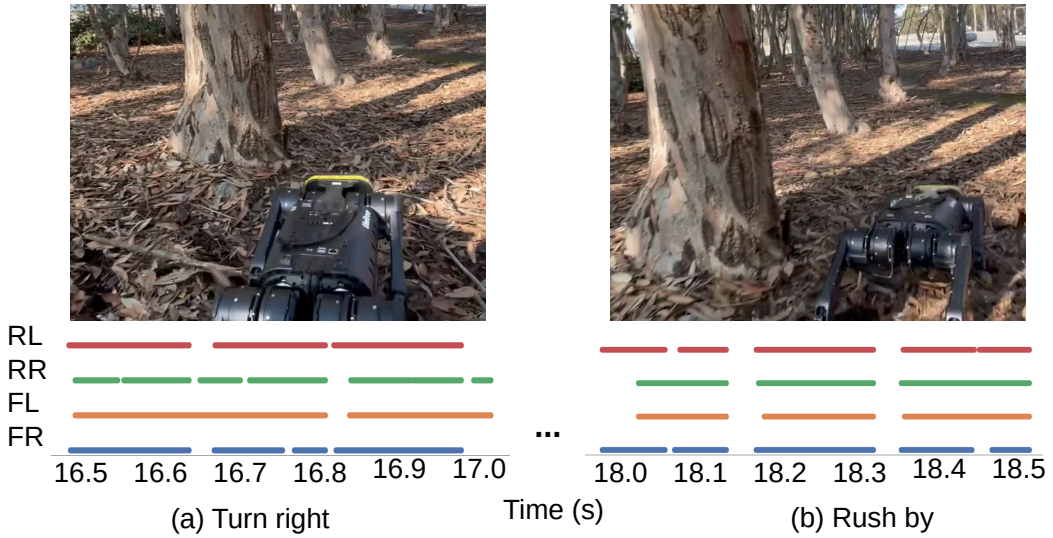


Figure 9: *Gait pattern analysis during key frames.* We plot the gait pattern according to contact forces to analyze the detailed locomotion. Lines denote stepping to the ground. During turning right, the front left leg continuously steps to support the twisting while two right legs move in higher frequency for reorienting. After adjusting the direction, robot turns to run faster in bouncing gait.

in the simulation. Thanks to our asynchronous randomization for multi-modal input, our method can learn policies that are more robust to unseen transitions.

To study the learned locomotion skills in more details, we visualize the positive (down) torque pattern during testing in Forest environment as shown in Figure 9. We selected two essential frames when the robot is turning around a tree. In Figure 9(a), the front left leg pushes the ground longer for power and the two right legs take small steps to change the body orientation. Once the robot faces to correct direction, it uses the bouncing gait for acceleration to pass the tree (Figure 9(b)). Such a complex locomotion sequence demonstrates that robot combines multi-modal information well to adapt to the complex environments.

6 Conclusion

Latency, as a crucial reality gap, exists in many parts of the robot control pipeline. We propose the Multi-modal Delay Randomization technique to address the latency issues in real-world deployment for vision-guided quadruped locomotion control. Our approach shows great advantages on two realistic metrics in both the simulator and real world. It also improves generalization and adaptation ability in unseen scenarios so as to help the robot pass through arbitrary barriers in real world. These results suggest MMDR can be universally applied not only to legged locomotion, but potentially also to many other visual robotic control tasks.

References

Thomas Timm Andersen, Heni Ben Amor, Nils Axel Andersen, and Ole Ravn. Measuring and modelling delays in robot manipulators for temporally precise control using machine learning. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pp. 168–175. IEEE, 2015. 3

Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Józefowicz, Bob McGrew, Jakub W. Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *Int. J. Robotics Res.*, 39(1), 2020. doi: 10.1177/0278364919887447. URL https://doi.org/10.1177/0278364919887447. 2

P. Arena, L. Fortuna, M. Frasca, L. Patané, and M. Pavone. Realization of a cnn-driven cockroach-inspired robot. *2006 IEEE International Symposium on Circuits and Systems*, pp. 4 pp.–, 2006. 1

Gerardo Bledt and Sangbae Kim. Extracting legged locomotion heuristics with regularized predictive control. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 406–412. IEEE, 2020. 3

Jan Carius, René Ranftl, Vladlen Koltun, and Marco Hutter. Trajectory optimization for legged robots with slipping motions. *IEEE Robotics and Automation Letters*, 4(3):3013–3020, 2019. doi: 10.1109/LRA.2019.2923967. 3

Jared Di Carlo, Patrick M. Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, pp. 1–9. IEEE, 2018. doi: 10.1109/IROS.2018.8594448. URL https://doi.org/10.1109/IROS.2018.8594448. 1, 3

Erwin Coumans. Bullet physics simulation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference, SIGGRAPH '15, Los Angeles, CA, USA, August 9-13, 2015, Courses*, pp. 7:1. ACM, 2015. doi: 10.1145/2776880.2792704. URL https://doi.org/10.1145/2776880.2792704. 4

Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021. 5

Xingye Da, Zhaoming Xie, David Hoeller, Byron Boots, Animashree Anandkumar, Yuke Zhu, Buck Babich, and Animesh Garg. Learning a contact-adaptive controller for robust, efficient legged locomotion. *ArXiv*, abs/2009.10019, 2020. 1

Jared Di Carlo, Patrick M Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–9. IEEE, 2018. 1, 3

Yanran Ding, Abhishek Pandala, and Hae-Won Park. Real-time model predictive control for versatile dynamic motions in quadrupedal robots. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8484–8490. IEEE, 2019. 3

Kenji Doya. Reinforcement learning in continuous time and space. *Neural Comput.*, 12(1): 219–245, 2000. doi: 10.1162/089976600300015961. URL https://doi.org/10.1162/089976600300015961. 3

Alejandro Escontrela, George Yu, Peng Xu, Atil Iscen, and Jie Tan. Zero-shot terrain generalization for visual locomotion policies, 2020. 1, 3

Aleksandra Faust, Kenneth Oslund, Oscar Ramirez, Anthony G. Francis, Lydia Tapia, Marek Fiser, and James Davidson. PRM-RL: long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pp. 5113–5120. IEEE, 2018. doi: 10.1109/ICRA.2018.8461096. URL https://doi.org/10.1109/ICRA.2018.8461096. 3

Christian Gehring, Stelian Coros, Marco Hutter, Michael Blösch, Mark A. Hoepflinger, and Roland Siegwart. Control of dynamic gaits for a quadrupedal robot. In *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10, 2013*, pp. 3287–3292. IEEE, 2013. doi: 10.1109/ICRA.2013.6631035. URL https://doi.org/10.1109/ICRA.2013.6631035. 3

Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *International Conference on Robotics and Automation*, 2021. 2, 3

Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation, 2021. 3

Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017. URL http://arxiv.org/abs/1707.02286. 3

Jemin Hwangbo, J. Lee, A. Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4, 2019. 1, 3

Julian Ibarz, Jie Tan, Chelsea Finn, Mrinal Kalakrishnan, Peter Pastor, and Sergey Levine. How to train your robot with deep reinforcement learning: lessons we have learned. *The International Journal of Robotics Research*, 40(4-5):698–721, 2021. 4

Atil Iscen, Ken Caluwaerts, Jie Tan, Tingnan Zhang, Erwin Coumans, Vikas Sindhwani, and Vincent Vanhoucke. Policies modulating trajectory generators. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings*, volume 87 of *Proceedings of Machine Learning Research*, pp. 916–926. PMLR, 2018. URL http://proceedings.mlr.press/v87/iscen18a.html. 3

Deepali Jain, Atil Iscen, and Ken Caluwaerts. From pixels to legs: Hierarchical learning of quadruped locomotion, 2020. 1, 3

Divye Jain, Andrew Li, Shivam Singhal, Aravind Rajeswaran, Vikash Kumar, and Emanuel Todorov. Learning deep visuomotor policies for dexterous hand manipulation. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3636–3643, 2019. doi: 10.1109/ICRA.2019.8794033. 3

Konstantinos V. Katsikopoulos and Sascha E. Engelbrecht. Markov decision processes with delays and asynchronous cost collection. *IEEE Trans. Autom. Control.*, 48(4):568–574, 2003. doi: 10.1109/TAC.2003.809799. URL https://doi.org/10.1109/TAC.2003.809799. 3

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980. 6

Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2149–2154, Sendai, Japan, Sep 2004. 4

Nate Kohl and Peter Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004, April 26 - May 1, 2004, New Orleans, LA, USA*, pp. 2619–2624. IEEE, 2004. doi: 10.1109/ROBOT.2004.1307456. URL https://doi.org/10.1109/ROBOT.2004.1307456. 3

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: rapid motor adaptation for legged robots. In Dylan A. Shell, Marc Toussaint, and M. Ani Hsieh (eds.), *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, 2021. doi: 10.15607/RSS.2021.XVII.011. URL https://doi.org/10.15607/RSS.2021.XVII.011. 1, 2

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/e615c82aba461681ade82da2da38004a-Abstract.html. 3

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in Neural Information Processing Systems*, 33, 2020b. 2

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47), 2020. 1, 2, 3

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17:39:1–39:40, 2016. URL http://jmlr.org/papers/v17/15-522.html. 1, 3

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *Int. J. Robotics Res.*, 37(4-5):421–436, 2018. doi: 10.1177/0278364917710318. URL https://doi.org/10.1177/0278364917710318. 3

Mengtian Li, Yuxiong Wang, and Deva Ramanan. Towards streaming perception. In *ECCV*, 2020. 3

Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. *CoRR*, abs/2103.14295, 2021. URL https://arxiv.org/abs/2103.14295. 2

Shaoshan Liu, Bo Yu, Yahui Liu, Kunai Zhang, Yisong Qiao, Thomas Yuang Li, Jie Tang, and Yuhao Zhu. The matter of time–a general and efficient system for precise sensor synchronization in robotic computing. *arXiv preprint arXiv:2103.16045*, 2021. 4

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Liu et al. (2016), pp. 21–37. ISBN 978-3-319-46447-3. URL http://dblp.uni-trier.de/db/conf/eccv/eccv2016-1.html#LiuAESRFB16. 3, 12

Y. Luo, Jonathan Hans Soeseno, T. Chen, and Wei-Chao Chen. Carl: Controllable agent with reinforcement learning for quadruped locomotion. *ArXiv*, abs/2005.03288, 2020. 1, 3

M. Lutter, S. Mannor, J. Peters, D. Fox, and A. Garg. Robust value iteration for continuous control tasks. In *Robotics: Science and Systems (RSS)*, July 2021. 3

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning. *CoRR*, abs/2108.10470, 2021. URL https://arxiv.org/abs/2108.10470. 4

Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Trans. Graph.*, 39(4):39, 2020. doi: 10.1145/3386569.3392474. URL https://doi.org/10.1145/3386569.3392474. 3

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJMGPrcle. 1

Igor Mordatch, Kendall Lowrey, and Emanuel Todorov. Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, pp. 5307–5314. IEEE, 2015. doi: 10.1109/IROS.2015.7354126. URL https://doi.org/10.1109/IROS.2015.7354126. 2

Edwin Olson. A passive solution to the sensor synchronization problem. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1059–1064. IEEE, 2010. 3

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37 (4):143:1–143:14, July 2018a. ISSN 0730-0301. doi: 10.1145/3197517.3201311. URL http://doi.acm.org/10.1145/3197517.3201311. 3, 6

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, pp. 1–8. IEEE, 2018b. doi: 10.1109/ICRA.2018.8460528. URL https://doi.org/10.1109/ICRA.2018.8460528. 2

Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. In *Robotics: Science and Systems*, 07 2020. doi: 10.15607/RSS.2020.XVI.064. 1, 2, 3, 6

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *RSS*, 2018. 2

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994. ISBN 0471619779. 4

Simon Ramstedt and Christopher Pal. Real-time reinforcement learning. *arXiv preprint arXiv:1911.04448*, 2019. 3

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 3

Alexander Sax, Bradley Emi, Amir R. Zamir, Leonidas J. Guibas, Silvio Savarese, and Jitendra Malik. Mid-level visual representations improve generalization and sample efficiency for learning visuomotor policies. 2018. 3

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1506.02438. 6

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347. 6

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In Hadas Kress-Gazit, Siddhartha S. Srinivasa, Tom Howard, and Nikolay Atanasov (eds.), *Robotics: Science and Systems XIV, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, June 26-30, 2018*, 2018. doi: 10.15607/RSS.2018.XIV.010. URL http://www.roboticsproceedings.org/rss14/p10.html. 2, 3

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017. 2

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109. URL https://doi.org/10.1109/IROS.2012.6386109. 4

Unitree. A1: More dexterity, more posibility, 2018. URL https://www.unitree.com/products/a1/. 2, 5, 8

Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: learning near-perfect pointgoal navigators from 2.5 billion frames. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1gX8C4YPr. 3

Ted Xiao, Eric Jang, Dmitry Kalashnikov, Sergey Levine, Julian Ibarz, Karol Hausman, and Alexander Herzog. Thinking while moving: Deep reinforcement learning with concurrent control. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJexHkSFPS. 3

Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonanthan Hurst, and Michiel Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In *Conference on Robot Learning*, pp. 317–329. PMLR, 2020a. 2

Zhaoming Xie, Xingye Da, Michiel van de Panne, Buck Babich, and Animesh Garg. Dynamics randomization revisited: A case study for quadrupedal locomotion. *CoRR*, abs/2011.02404, 2020b. URL https://arxiv.org/abs/2011.02404. 2

Zhaoming Xie, Xingye Da, Michiel van de Panne, Buck Babich, and Animesh Garg. Dynamics randomization revisited: A case study for quadrupedal locomotion. *CoRR*, abs/2011.02404, 2020c. URL https://arxiv.org/abs/2011.02404. 1, 3

Ruihan Yang, Minghao Zhang, Nicklas Hansen, Huazhe Xu, and Xiaolong Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers, 2021. 1, 3

Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pp. 1–10. PMLR, 2020. 3

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf. 3