Learning to Jump from Pixels

Anonymous Author(s)
Affiliation
Address
email

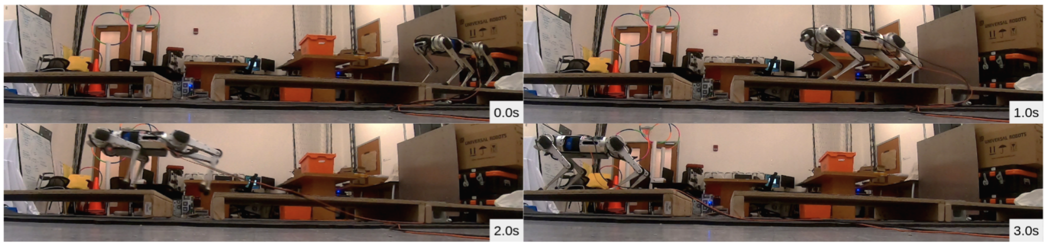


Figure 1: Mini Cheetah dynamically traverses a 23-centimeter gap.

Abstract: Today's robotic quadruped systems can robustly walk over a diverse range of natural but *continuous* terrains involving snow, rain, slip, rubble, etc. Locomotion on *discontinuous* terrains such as one with gaps or obstacles presents a complementary set of challenges. It becomes necessary to plan ahead using visual inputs and execute agile behaviors such as jumps to cross gaps. Such dynamic motion results in significant motion of onboard camera that introduces a new set of challenges for real-time visual processing. The need for agility and the operation from vision reinforce the need for robust control. We present a system that can, in real-time, process visual observations from an onboard RGBD camera to command a quadruped robot to jump over wide gaps. The proposed method brings together the flexibility of model-free learning and the robustness of model-based control. We evaluate performance both in simulation and in the real world.

Keywords: Locomotion, Vision, Hierarchical Control

1 Introduction

One of the grand challenges in robotics is to construct legged systems that can successfully navigate novel and complex landscapes. The *Spot* robot and the *ANYmal* robot are impressive in their ability to traverse a wide diversity of natural and man-made terrains [1]. During *blind* locomotion, such robots primarily rely on proprioception and robust control schemes to achieve sturdy walking in challenging conditions involving snow, thick vegetation, slippery mud, etc. The downside of not using visual observations is the inability to execute motions that anticipate the land surface in front of the robot. This is especially prohibitive on terrains with significant elevation discontinuities. For instance, crossing a wide gap requires the robot to jump, which cannot be initiated without knowing where and how wide the gap is. Without vision, even the most robust system would either step in the gap and fall or otherwise treat the gap as an obstacle and stop. Additionally, the inability to plan results in conservative behavior that is unable to achieve the energy efficiency or the speed that the hardware affords.

Vision-based legged locomotion holds the promise to overcome these challenges, and substantial progress has been made in this direction [2, 3, 4, 5, 6, 7, 8, 9]. The state-of-the-art systems can traverse uneven terrain, walk across gaps, and climb over stairs. Many of these systems assume access to the ground truth height-map of the terrain [2, 3, 5] which is generally not available for new terrains encountered by the robot. Several works overcome this limitation by performing online construction of a terrain heightmap from depth images [4, 7]. These works perform a rule-based search for stable footholds and use model-based controllers to plan safe trajectories. Often, the model used in planning makes simplifying assumptions such as fixed body trajectory [7] or restricted contact pattern [5]. These assumptions result in conservative and non-agile locomotion.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

Planning agile behaviors, such as jumps, on discontinuous terrain offers a different and complementary challenge to traversing continuous and unstructured terrain. Executing a jump requires planning the location of the jump, the force required to lift the body, and dealing with severe under-actuation during the flight phase. Prior work has demonstrated standing jumps in simulation [10, 11], on a real robot [7], and running jumps in simulation [12, 13, 14, 9, 8]. Prior work on the MIT Cheetah 2 achieved running and jumping over a single obstacle [15]. However, this system was heavily hand-engineered: it assumes straight-line motion, uses a specialized control scheme developed for four manually segmented phases of the jump, and the vision system was specialized for detecting specific obstacles. Further, the robot was constrained to a fixed gait. Consequently, this system is specific to jumping over one obstacle type, and substantial engineering effort would be required to extend agile locomotion to diverse terrains in the wild.

Operation in the wild requires a system architecture that can automatically generate a diverse set of agile behaviors directly from visual observations. One possibility is to apply deep reinforcement learning to predict joint torques directly from visual inputs. Previous work [9, 14, 16] trains locomotion RL algorithms in simulation, since a simulated environment can be easily randomized, requires little human maintenance, and can be conveniently parallelized for large-scale experience collection. However, there is a drawback to training in simulation: the agent learns to exploit inaccuracies in the simulation to achieve high reward, when such behavior fails to transfer to the real system. Some prior work aims to reduce simulator exploitation by encoding locomotion-specific priors into the agent's action space. One such line of work uses a trajectory generation method known as PMTG [17], which combined with domain randomization can successfully transfer simulated walking behaviors to the real world. However, it is well known that while greater randomization increases robustness and improves sim-to-real transfer, it also results in more conservative and thereby suboptimal policies [18].

The sim-to-real problem is severely aggravated by requirements of agility and operation from visual inputs. This motivates a different style and rigor of controller design that emphasizes transfer robustness. As a step towards vision-guided agile locomotion, we present a general framework for synthesizing adaptive, agile behaviors with the support of a low-level robust controller. To evaluate our proposed system and baselines, we construct a suite of gap-world environments that require a quadruped to cross a sequence of randomly placed gaps of varying widths using observations from an attached depth camera (see Figure A.1). While this environment is much simpler than "in-the-wild", traversing it successfully requires solving many of the core challenges in vision-guided agile locomotion. We use the MIT Mini Cheetah robot [19] as the experimental platform and report results both in simulation and the real world.

We contribute a novel analysis of design choices for integrating flexible model-free learning with robust model-based trajectory optimization in the context of visually guided locomotion across discontinuous terrain. The end result is a system that can (a) cross a sequence of wide gaps in real-time using depth observations from a body-mounted camera in the real world (i.e., jumping from pixels; Figure A.1); (b) requires no dynamics randomization for sim-to-real-transfer; (c) does not assume fixed gait; (d) achieves the theoretical limit of jump width with fixed gaits and even wider jumps with variable gaits and (e) outperforms existing state-of-the-art methods (e.g., PMTG [17]). Our framework relies on a combination of model-free RL for high-level planning, a hybrid model-based low-level controller, and asymmetric behavior cloning [1, 20] that we discuss in Section 2. Implementation details are provided in Section 3 and results in Section 5.

2 Method Overview

The mapping from depth images to joint torques is complex. To simplify the problem we make use of a hierarchical scheme where a high-level controller processes visual inputs to predict the desired trajectory of the robot's body and a blind low-level controller ensures that the predicted trajectory is tracked. This separation eases the task for both the controllers: the high-level is shielded from intricacies of joint-level actuation and the low-level is not required to reason about visual observations. Our choice of the action space for high-level controller is guided by the intuition that a wide range of agile behaviors can be generated by using a *variable gait schedule* and *commanding the body velocity* of the quadruped. A high forward velocity results in running, whereas different ratios of vertical and forward velocity can control the height and the span of a jump. The variable gait schedule allows the robot to change when its foot contacts the ground and thus further expands

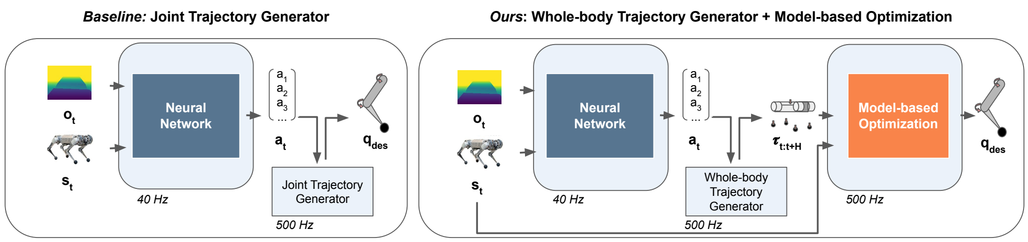


Figure 2: Control architecture of Jumping from Pixels and baseline.

the range of feasible contact locations and applied forces. We solve the problem of mapping depth observations to velocity and gait-schedule commands using model-free deep reinforcement learning.

To ensure that the robot tracks these commands, one possibility is to simultaneously train a low-level controller using RL that converts the high-level velocity and gait commands into joint torques. Such a scheme has two drawbacks: (i) sim-to-real transfer issues discussed in Section 1 and (ii) copious data requirement for training. Another possibility is to leverage an analytical model of the robot and solve for joint torques using trajectory optimization – a scheme commonly known as whole-body control (WBC). Whole-body control can promote robustness by optimizing for body stability at high frequency [21, 22, 23, 24]. One issue, however, is that a typical WBC tracks the robot's center-of-mass (CoM) [21, 22], which is infeasible during the flight phase of agile motion due to under-actuation of the robot's body. To overcome this issue, we leverage a prior hybrid control scheme built on the intuition that the changes in body velocity can be realized by modifying the forces applied by the robot's feet on the ground. This frees the controller from the requirement of faithfully tracking the CoM and instead tracks the contact timing and the ground forces applied by the feet. This approach, called whole-body impulse controller (WBIC), is well suited for highly dynamic motion [24].

The velocity and gait-schedule targets for WBIC are selected in our system by a learned policy. We found that while it was possible to train this high-level policy from depth images, training using the ground truth heightmap is more sample efficient and yields higher final performance (see Section 5.2). However, in real-world environments, such a heightmap is typically either unavailable or must be constructed in real-time [4], incurring latency, information loss, and engineering effort. We eschew heightmap construction using a two-stage *asymmetric behavioral cloning* [1, 20] approach, in which we first train an expert policy using height-map data and then distill it to a student policy that only uses depth images. More details are provided in Section 3.1.

The quadruped's *whole-body state* at time t is fully defined as $\mathcal{T}_t = [\mathbf{p}_\text{b}, \dot{\mathbf{p}}_\text{b}, \ddot{\mathbf{p}}_\text{b}, \mathbf{p}_\text{f}, \dot{\mathbf{p}}_\text{f}, \ddot{\mathbf{p}}_\text{f}, \mathbf{C}]_t \in \mathbb{R}^{54} \times [0,1]^4$ where $\mathbf{p}_\text{b} = [x, y, z, \alpha, \beta, \gamma] \in \mathbb{R}^6$ is the robot body pose (position (x, y, z) and euler angles (α, β, γ)). The terms $\mathbf{p}_\text{f} = [p_x^{\text{LF}}, p_y^{\text{LF}}, p_z^{\text{LF}}, p_x^{\text{RF}}, p_y^{\text{RF}}, p_z^{\text{RF}}, p_x^{\text{LR}}, p_y^{\text{LR}} p_z^{\text{LR}}, p_x^{\text{RR}}, p_y^{\text{RR}}, p_z^{\text{RR}}] \in \mathbb{R}^{12}$ denote the position of the Right (R_), Left (L_) Front (_F) and Rear (_R) feet respectively. $\mathbf{C} = [\mathbb{1}_C^{\text{LF}}, \mathbb{1}_C^{\text{RF}}, \mathbb{1}_C^{\text{LR}}, \mathbb{1}_C^{\text{RR}}] \in [0,1]^4$ is the binary contact state of each foot, with $\mathbb{1}_C^{\text{f}}$ taking a value of 1 if foot f is in contact with the ground and a value of zero otherwise.

3 Training the Jumping Policy

Let the high-level policy be $\mathbf{a}_t = \pi_\theta(\mathbf{s}_t, \mathbf{o}_t, \mathbf{a}_{t-1})$ where $\mathbf{a}_t$ is the action and $\mathbf{s}_t, \mathbf{o}_t$ denote the robot's internal state and the terrain observation respectively. The action at previous time-step is fed as input to encourage smoothness. We use a deep neural network to represent π.

Observation Space The proprioceptive state $\mathbf{s}_t \in \mathbb{R}^{34}$ consists of the robot body height $(\mathbb{R})$, orientation $(\mathbb{R}^3)$, linear velocity $(\mathbb{R}^3)$, and angular velocity $(\mathbb{R}^3)$, as well as the joint positions $(\mathbb{R}^{12})$ and velocities $(\mathbb{R}^{12})$. The terrain observation $\mathbf{o}_t$ is either a body-centered elevation map $\mathbf{o}_t = \mathbf{E}_t \in \mathbb{R}^{48\times15}$ or a depth image $\mathbf{o}_t = \mathbf{I}_t \in \mathbb{R}^{160\times120}$ from a body-mounted camera. Observations are normalized using the running mean and the standard deviation.

Action Space We train policies with either fixed or variable gait patterns. With fixed gait, $\mathbf{a}_t \in \mathbb{R}^4$ encodes the target body linear velocity and yaw velocity. By setting the velocity, we are essentially modulating the target acceleration. For computational efficiency, our low-level controller assumes that the target pitch and roll are near zero, and consequently, we exclude them from the high-level policy output. This assumption does not prevent our system from making agile jumps. For variable gaits, we additionally predict the contact schedule of the legs denoted by $\mathbf{C} \in [0,1]^4$. As an example, the contact schedule for *trot* and *pronk* gaits corresponds to:

$$\mathbf{C}_{trot} = \begin{cases} [1,0,0,1] & t < d/2 \mod d \\ [0,1,1,0] & t \geq d/2 \mod d \end{cases} \qquad \mathbf{C}_{pronk} = \begin{cases} [1,1,1,1] & t < d/2 \mod d \\ [0,0,0,0] & t \geq d/2 \mod d \end{cases}$$

where d is the gait cycle duration. In our experiments with fixed gaits, we set $d = 10$. For variable gaits, the contact schedule $\mathbf{C}(\mathbf{a}_t)$ is selected by the policy:

$$\mathbf{C}_{flex} = \left\{ [f_C(\mathbf{a}_t[4])] \quad \text{at time } t \right.$$

where f_C maps a discrete policy output to a contact state target. There are $2^4 = 16$ possible contact states for the four feet, so the associated policy output makes a new choice among 16 categories at each timestep in the fully gait-free case. We also train gait-free policies with fewer contact state options, such as the Variable Pronk which synchronizes the contacts of all feet.

The action $\mathbf{a}_t$ is converted into the *desired* whole-body trajectory for the next H time steps denoted as

$$\mathcal{T}^{des}_{t:t+H} = [\mathbf{p}_\text{b}(\mathbf{a}_t), \dot{\mathbf{p}}_\text{b}(\mathbf{a}_t), \ddot{\mathbf{p}}_\text{b}(\mathbf{a}_t), \mathbf{p}_\text{f}^\text{raibert}, \dot{\mathbf{p}}_\text{f}^\text{raibert}, \ddot{\mathbf{p}}_\text{f}^\text{raibert}, \mathbf{C}]$$

where the key quantity adapted by the policy is $\dot{\mathbf{p}}_\text{b}(\mathbf{a}_t) = [\dot{x} = \mathbf{a}_t[0], \dot{y} = \mathbf{a}_t[1], \dot{z} = \mathbf{a}_t[2], \dot{\alpha} = 0, \dot{\beta} = 0, \dot{\gamma} = \mathbf{a}_t[3]]$, from which $\mathbf{p}_\text{b}(\mathbf{a}_t)$ and $\ddot{\mathbf{p}}_\text{b}(\mathbf{a}_t)$ are fixed for dynamic consistency. $\mathbf{p}_\text{f}^\text{raibert}, \dot{\mathbf{p}}_\text{f}^\text{raibert}, \ddot{\mathbf{p}}_\text{f}^\text{raibert}$ are foot targets satisfying the Raibert Heuristic (see supplement).

Algorithm 1 Policies Modulating Whole-body Objectives

$t \leftarrow 0$; $\mathbf{a}_{-1} \leftarrow \mathbf{0}$
observe $\mathbf{s}_0$, $\mathbf{o}_0$
while not IS-TERMINAL($\mathbf{s}_t$) **do**
  sample $\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t | \mathbf{s}_t, \mathbf{o}_t, \mathbf{a}_{t-1})$
  $\mathcal{T}_{t+H} \leftarrow \mathcal{T}(\mathbf{a}_t)$
  TRACK-TRAJECTORY($\mathbf{s}_t, \mathcal{T}_{t:t+H}$)
  $t = t + 1$
  observe $\mathbf{s}_t$, $\mathbf{o}_t$
end while

Whole-body Trajectory Tracking is performed using the hybrid control scheme described in [24]. It is a high-frequency controller that solves a quadratic program (QP): $\mathbf{q}_{des} = \text{QP}(\mathcal{T}_{des}, \mathcal{T})$ without access to terrain information. It is composed of three controllers operating hierarchically:

- A *Model Predictive Controller (MPC)* converts the future whole-body trajectory $\mathcal{T}_{t:t+H}$ into target ground reaction forces $\mathbf{f}_{t:t+H}$ at contact for every foot at each timestep. MPC operates at **40 Hz**.
- A *Whole-Body Impulse Controller (WBIC)* finds the target position, velocity, and feedforward torque commands for all joints required to track the current step of the the whole-body trajectory $\mathcal{T}_t$ and desired ground reaction forces $\mathbf{f}_t$ computed by the MPC. WBIC operates at **500 Hz**.
- A *Proportional-Derivative Plus Feedforward Torque Controller* takes as input a target position, target velocity, and feedforward torque command for each of the robot joints, as well as an observation of each joint's current position and velocity. It computes and actuates a resulting output torque for each motor at **40 kHz**.

Rollout Procedure The iterative execution routine for our model-free planner and model-based controller is given by Algorithm 1. The low-level controller performs receding-horizon optimization of contact forces over horizon H, with the assumption that the future contact and pose targets taken into account for planning will not change. This motivates our design choice in the high-level policy to select the trajectory target H timesteps into the future. In our experiments, $H = 10$, the policy timestep is 0.036s, and the low-level controller timestep is $0.002s$.

Reward Function The reward r_t at time t is defined as:

$$r_t = c_1(p^b_{t,x} - p^b_{t-1,x}) - c_2 \max(0, ||v^b_t||_2 - V_{thresh}) - c_3|\alpha^b_t| - c_4|\beta^b_t| - c_5|\gamma^b_t| - c_6|\dot{q}|$$

The first term rewards the forward progress $p^b_{t,x} - p^b_{t-1,x}$, where $p^b_{t,x}$ is the projection of the body frame position at time t onto the x-axis in the world frame. The second term applies a soft safety constraint by penalizing when the body velocity v^b_t exceeds V_{thresh}. The third, fourth, and fifth terms incentivize stability by penalizing the roll, pitch, and yaw of the body, denoted as α^b_t, β^b_t. The sixth term rewards smooth motion by penalizing the joint velocity $\dot{q}$; in training with adaptive contact schedule, we found this term critical to promote exploration of lower-frequency gaits. $c_1, c_2, c_3, c_4, c_5, c_6$ are the coefficients of each reward term. In our experiments, $c_1 = 1.0, c_2 = 0.5, c_3 = 0.02, c_4 = 0.05, c_5 = 0.15, c_6 = 0.03, V_{thresh} = 1.0$m/s.

3.1 Neural Network Training

Network Architecture The policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t, \mathbf{o}_t, \mathbf{a}_{t-1})$ is modeled using a deep recurrent neural network that includes a convolutional neural network (CNN) to process the high-dimensional terrain observation $\mathbf{o}_t$. The output features of the perception module are concatenated with proprioceptive inputs $\mathbf{s}_t$, previous action $\mathbf{a}_{t-1}$, and a cyclic timing parameter [17] and passed through a sequence of fully connected layers to output a probability distribution over the next action $\mathbf{a}_t$. Figure 3 illustrates the architecture of the policy or actor network; during training, we also use a critic network, in which the final layer of the actor is replaced by a value prediction head.

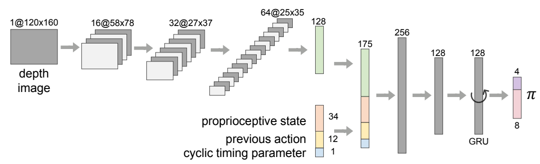


Figure 3: High-level gait prediction network

Initialization and Termination For each training episode, the robot is initialized in a standing pose on flat ground. The locations of gaps and their widths are randomized. An episode terminates if any of three terminal conditions are met: (1) the body height is less than 20 centimeters; (2) body roll or pitch exceeds 0.7 radians; or (3) a foot is placed in a gap. The maximum episode length is 500 steps, equivalent to 25 seconds of simulated locomotion.

Policy Optimization The parameters of the neural network (θ) are optimized using Proximal Policy Optimization (PPO) [25]. We use Adam optimizer [26] with learning rate 0.0003 and batch size 256. During training, 32 environments are simulated in parallel. We find that policies converge within 6000 training episodes, equivalent to 60 hours of simulated locomotion or 12 hours of computation.

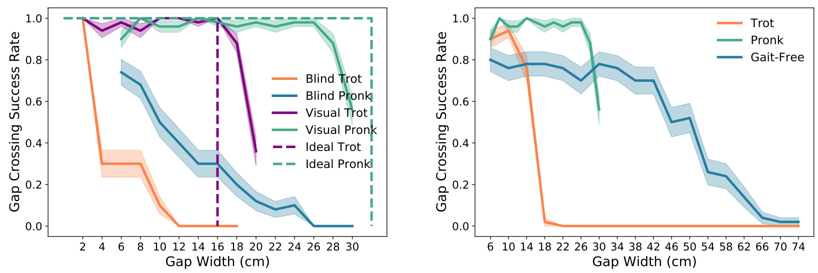
Asymmetric-Information Behavioral Cloning Learning directly from depth images is challenging because a front-facing depth camera can only provide information about the terrain in front of the robot, not the terrain underneath its feet, making the contact-relevant terrain partially observed. Furthermore, the depth image is dependent on the robot egomotion as well as the terrain shape, introducing noise. To address the challenge of learning a policy from the noisy, partial observations provided by a body-mounted depth camera, we use a two-stage approach that first trains an expert policy (π_{E}) from ground truth height map. A second *deployable* policy (π_{BC}) is trained from depth inputs to mimic the expert policy. For this, we use a variant of Behavioral Cloning (BC) known as DAgger [27] to minimize the KL-divergence between the output action distribution of the imitating agent $\pi_{\mathrm{BC}}(a|s)$ and the expert $\pi_{\mathrm{E}}(a|s)$: $\min D_{\mathrm{KL}}\big(\pi_{\mathrm{E}}(a|s)||\pi_{\mathrm{BC}}(a|s)\big)$. Because the depth images in our setup contain only a portion of the information in the heightmaps, it is necessary to integrate depth data over time. Therefore, we represent π_{BC} as a recurrent neural network. Prior works have applied similar approaches to blind rough-terrain locomotion [1] and autonomous driving [20]. Evaluation reported in Table 1 indicates that this cloned policy matches the performance of the expert trained from heightmaps, and substantially outperforms learning directly from depth images.

4 Experimental Setup

Hardware: We use the MIT Mini-Cheetah [19] , a 9kg electrically-actuated quadruped that stands 28cm tall with a body length of 38cm. A front-mounted Intel RealSense D435 camera provides real-time stereo depth data. The robot is also equipped with an onboard computer [7] that supports a hierarchical trajectory-tracking controller described in Section 2. Data from the depth camera is processed by an offboard computer that communicates the output of the high-level policy to the robot via an Ethernet cable. Treating proprioceptive state estimation as an orthogonal research direction to our work, we use a motion capture system to obtain accurate measurements of the robot body state.

Simulator: We trained our vision-conditioned policy using the PyBullet [28] simulator. In addition to simulating the robot dynamics, PyBullet simulates the frames of our mounted depth camera, calibrated from an accurate CAD model of our robot and from the sensor's known intrinsic parameters.

Gap World Environment: To evaluate the ability of our system to dynamically traverse discontinuous terrains, we define a test environment consisting of variable-width gaps and flat regions. The difficulty of traversing gap worlds depends on the proximity of gaps as well as gap width, with closer and wider gaps presenting a greater challenge to the controller. Our training dataset consists



(a) (b)

Figure 4: **(a)**:Performance comparison for blind and visually guided policies. Shaded regions indicate standard error. Ideal performance is derived from maximum stride length given velocity, foot placement, and contact schedule constraints. Note that while the theoretical limits are derived assuming zero yaw, the learned trotting controller learns to move with nonzero yaw, thus extending the foot placements further apart and beating the ideal. **(b)** A comparison of performance among policies trained with fixed contact schedule and adaptive contact schedule demonstrates the flexibility and dynamic range of our method.

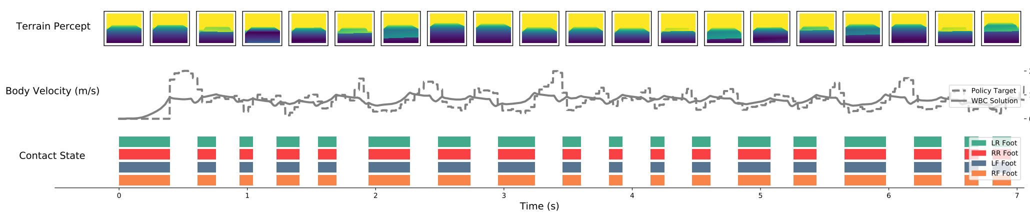


Figure 5: Adaptive contact schedule generated by our policy. Given a terrain observation (top), the policy modulates the body velocity (middle) and contact duration (bottom) to traverse 30cm gaps.

of randomly generated gaps with uniform random width between $W_{\text{min}} = 4$ and W_{max} centimeters, separated by flat segments of randomized width 0.5 to 2.0 meters. Our test dataset contains novel terrains drawn from the same distribution.

Policies Modulating Trajectory Generators (PMTG) [17] Baseline augments the action space of model-free RL using a parametric *trajectory generator* (TG) capable of producing cyclic leg motions. Given a timing parameter (t) that cycles between 0 and 1 and trajectory parameters ($\mathbf{a}$) – stride frequency, length etc., TG outputs joint position targets $\mathbf{q}_{des} = \text{TG}(t, \mathbf{a})$. The policy also directly predicts residuals ($\Delta\mathbf{q}_{des}$). The output command is therefore $\mathbf{q}_{des} + \Delta\mathbf{q}_{des}$.

5 Results

5.1 Dynamic Performance

By design, learning with a trajectory generator introduces constraints and biases into the resulting policies. This aids learning and enables robust behavior. However, this yields a concern: are the constraints and biases of the trajectory generator *too* rigid to accommodate useful locomotion strategies? Or do they serve to guide learning effectively without hindering final performance? Our results indicate the latter. In this section, we evaluate the flexibility and performance of our integrated perception and control approach. We find that our system is both high-performing under constraints and flexible when constraints are removed.

Optimality Under Constraints We train our framework to cross gaps up to the theoretical limit for constrained families of trotting and pronking gaits. Figure 4a reports the performance of our method for adaptive fixed-gait gap crossing in simulation. While the baseline fixed gaits without vision are capable of sometimes crossing gaps by chance, our visually-guided approach succeeds at above 90% of gap crossing attempts up to the theoretical limits derived in the supplementary material.

Unconstrained Range of Motion We relax all constraints on contact schedule and train a controller with a *vision-adaptive contact schedule* to cross wide gaps. Figure 4b reports the performance of our method for gait-adaptive gap crossing in simulation. When trained with extremely wide (40-

Table 1: Gap crossing success rate for RL policies (with Trotting (T), Pronking (P), or Variable-Timing Pronking (VP)) trained on various maximum gap widths with with height maps, depth images as input respectively, and the policy produced by behavioral cloning with and without recurrent architecture. For model trained with maximum gap width $W_{\max}$, the evaluated gap width is $W_{\max} - 5$.

Input	T, 10cm	T, 20cm	P, 20cm	P, 30cm	VP, 30cm
Heightmap (MLP)	1.0	1.0	1.0	0.7	1.0
Depth Image (RNN)	0.6	0.3	0.9	**0.9**	0.7
Heightmap (MLP) → Depth Image (MLP)	1.0	0.9	0.1	0.0	0.0
Heightmap (MLP) → Depth Image (RNN)	**1.0**	**1.0**	**1.0**	0.4	**1.0**

to 70-cm gaps), the visually informed policy learns to select a variable-bounding contact schedule which achieves superior performance to trotting and pronking for very large gaps. However, we note that the low-level controller may truly not support the motions generated in simulation for crossing such large gaps. When we restrict the maximum gap size to 40cm or less, a variable-timing pronking gait emerges in the gait-free controller. Figure 5 illustrates the variable contact timings and velocity modulation of the variable pronking controller in simulation.

Ease of Training Our method is capable of learning successful policies for multiple gaits and gap-world parameters with no specialized modification. In contrast, we found that the PMTG baseline was highly sensitive to the tuning of the reward and trajectory generator. We first tuned the trajectory generator, residual magnitudes, and reward function of PMTG for forward locomotion on flat ground; details and videos of the baseline can be found in the supplementary material. We found that these tuned parameters on flat ground were overly conservative for gap crossing tasks. However, an action space with large residuals can significantly override the predefined motion primitives, presenting an obstacle to learning realistic gap-crossing behaviors without any curriculum or specialized reward design [8]. Indeed, without any such special inclusions in our training, the baseline failed to learn any gap-crossing behavior when the maximum gap width $W_{\max}$ exceeded 15cm for trotting or 25cm for pronking as well as when gap separation was reduced to 0.5m in simulation.

5.2 Vision and Behavioral Cloning

Performance Table 1 illustrates that behavioral cloning offers an advantage over learning directly from depth images in many but not all cases. We find that learning from heightmaps + BC consistently achieves higher reward than learning directly from depth images. These results also demonstrate that the combination of behavioral cloning with a variable gait schedule is beneficial, with the cloned Variable Pronk achieving the highest performance for wide gaps of any depth-image policy.

Recurrent Architecture We ablate the recurrent architecture of the cloned policy, and note that policies with recurrent architecture consistently yield higher final performance than without, particularly for environments with larger gaps which require more dynamic motion (Table **??**). This suggests that the hidden state is helpful in forming a useful representation of unobserved terrain regions given the observation history.

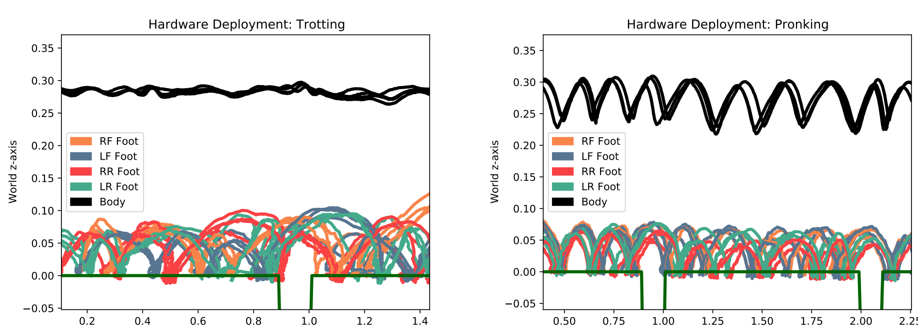


Figure 6: Motion capture data verifies the transfer of planned trajectories to the hardware system.

5.3 Sim-to-Real Transfer

We deploy vision-adaptive locomotion controllers trained with Jumping from Pixels on the MIT Mini Cheetah robot [19]. Figure 6 plots motion capture data from four deployments of the adaptive trotting controller (left) and three deployments of the adaptive pronking controller (right) using ground-truth state information and heightmap input. The relevant cross-section of the terrain surface is drawn in dark green. The consistency of foot placements and visible adaptive avoidance of the gap verify that our method trained in simulation produces terrain-appropriate behaviors which can cross the sim-to-real gap. Further, we successfully deploy student policies in fully real-time fashion, directly making use of depth images and an onboard state estimator. We report and analyze these results at https://sites.google.com/view/jumpingfrompixels.

6 Related Work

Model-free RL for locomotion is shown to benefit from acting over low-level control loops rather than raw commands [29]. Previous work in simulation [11, 13, 9] has applied model-free reinforcement learning to traversal of discontinuous terrains in simulation. [9] notably applied model-free RL to the problem of crossing stepping stones with physically simulated characters, but this method did not use realistic perception or take measures to promote sim-to-real transfer. Recent work on ANYmal [30] learns a model-free policy to predict joint position targets for a PD controller, and demonstrates better energy efficiency and higher maximum velocity than comparable model-optimization-based controllers. However, joint-space policies learned in simulation can still be unrobust and fail to cross the sim-to-real gap. Reward shaping, system identification, and domain randomization were used in [30] to facilitate transfer to the real robot.

Model-based control for locomotion has achieved highly dynamic blind walking [31], running [24], and jumping over obstacles [15] using known quadruped whole-body and centroidal dynamics. Other works have applied model-based control to terrain-aware navigation of a mapped environment, typically with complete information about the terrain [4, 32]. In general, control strategies based on known models are high-performing and robust where the state is known and the model is sufficiently accurate. In contrast, model-free controllers excel at incorporating unstructured or partially observed state information when large data is available.

Interfacing Model-based and Model-Free Methods. A previous line of work has leveraged model-free perception for foothold selection. [33] locally adapted foot placements to safe footholds predicted by a CNN. RLOC [6] similarly uses a learning-based online footstep planner in combination with a learning-modulated whole-body controller to perform terrain-aware locomotion. Unlike our method, [6] uses a complete terrain heightmap as observation, plans by targeting foot placements, and is limited to relatively conservative fixed walking and slow trotting gaits. On the other hand, concurrent work applies RL to modulate a model-based controller's target command without perception. [34, 35] demonstrated that using a model-free policy to choose contact schedules for a reduced-order model leads to the emergence of efficient gait transitions during blind flat-ground locomotion. [36] demonstrates the integration of a model-free high-level controller with a centroidal dynamics model. This framework deployed with a fixed trotting gait is demonstrated to achieve flat-ground and conservative terrain-aware locomotion. Unlike our work, [36] does not demonstrate gaits with flight phases or plan from realistic terrain observations.

7 Conclusion and Discussion

We have presented a vision-based hierarchical control framework capable of traversing discontinuous terrain with gaps. The combination of model-free high-level trajectory prediction and model-based low-level trajectory tracking enables us to simultaneously achieve high performance and robustness. Consequently, we demonstrate that the learnt behaviors cross the sim-to-real gap.

One aspect that prevents in-the-wild deployment is that the readings of onboard sensors for estimating the robot's internal state are noisy and insufficient to plan high-precision foot placement for crossing gaps. To focus on transfer of visual inputs and dynamic control in this work, we made use of a motion capture system to measure the robot's state. We believe that improving on-board state estimation by leveraging vision is a worthwhile, but a complementary direction of future research.

References

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

[2] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng. A control architecture for quadruped locomotion over rough terrain. In *2008 IEEE International Conference on Robotics and Automation*, pages 811–818. IEEE, 2008.

[3] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal. Learning, planning, and control for quadruped locomotion over challenging terrain. *The International Journal of Robotics Research*, 30(2):236–258, 2011.

[4] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter. Robust rough-terrain locomotion with a quadrupedal robot. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[5] V. Tsounis, M. Alge, J. Lee, F. Farbod, and M. Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. In *IEEE International Conference on Robotics and Automation*, 2020.

[6] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis. Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control. *arXiv preprint arXiv:2012.03094*, 2020.

[7] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim. Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot. In *IEEE International Conference on Robotics and Automation*, 2020.

[8] A. Iscen, G. Yu, A. Escontrela, D. Jain, J. Tan, and K. Caluwaerts. Learning agile locomotion skills with a mentor. In *2021 International Conference on Robotics and Automation (ICRA)*, 2021.

[9] Z. Xie, H. Y. Ling, N. H. Kim, and M. van de Panne. Allsteps: Curriculum-driven learning of stepping stone skills. *Computer Graphics Forum*, 39(8):213–224, 2020.

[10] H. C. Wong and D. E. Orin. Control of a quadruped standing jump over irregular terrain obstacles. *Autonomous Robots*, 1(2):111–129, 1995.

[11] G. Bellegarda and Q. Nguyen. Robust quadruped jumping via deep reinforcement learning. *arXiv preprint arXiv:2011.07089*, 2020.

[12] D. P. Krasny and D. E. Orin. Evolution of dynamic maneuvers in a 3d galloping quadruped robot. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1084–1089. IEEE, 2006.

[13] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. Van De Panne. Locomotion skills for simulated quadrupeds. *ACM Transactions on Graphics (TOG)*, 30(4):1–12, 2011.

[14] N. Heess, G. Wayne, Y. Tassa, T. Lillicrap, M. Riedmiller, and D. Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

[15] H. Park, P. Wensing, and S. Kim. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. In *Robotics: Science and Systems*, 2015.

[16] D. Jain, A. Iscen, and K. Caluwaerts. From pixels to legs: Hierarchical learning of quadruped locomotion. *arXiv preprint arXiv:2011.11722*, 2020.

[17] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke. Policies modulating trajectory generators. In *Conference on Robot Learning*, pages 916–926. PMLR, 2018.

[18] 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. *Robotics Science and Systems*, 2018.

[19] B. Katz, J. Di Carlo, and S. Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301. IEEE, 2019.

[20] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

[21] L. Righetti and S. Schaal. Quadratic programming for inverse dynamics with optimal distribution of contact forces. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 538–543. IEEE, 2012.

[22] D. Kim, J. Lee, J. Ahn, O. Campbell, H. Hwang, and L. Sentis. Computationally-robust and efficient prioritized whole-body controller with contact constraints. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[23] H.-W. Park, P. M. Wensing, and S. Kim. High-speed bounding with the mit cheetah 2: Control design and experiments. *The International Journal of Robotics Research*, 36(2):167–192, 2017.

[24] D. Kim, J. D. Carlo, B. Katz, G. Bledt, and S. Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *ArXiv*, abs/1909.06586, 2019.

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

[26] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[27] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[28] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation in robotics, games and machine learning, 2017.

[29] X. B. Peng and M. van de Panne. Learning locomotion skills using deeprl: Does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–13, 2017.

[30] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[31] C. D. Bellicoso, C. Gehring, J. Hwangbo, P. Fankhauser, and M. Hutter. Perception-less terrain adaptation through whole body control and hierarchical optimization. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 558–564. IEEE, 2016.

[32] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter. Multi-layered safety for legged robots via control barrier functions and model predictive control. *arXiv preprint arXiv:2011.00032*, 2020.

[33] O. A. V. Magana, V. Barasuol, M. Camurri, L. Franceschi, M. Focchi, M. Pontil, D. G. Caldwell, and C. Semini. Fast and continuous foothold adaptation for dynamic locomotion through cnns. *IEEE Robotics and Automation Letters*, 4(2):2140–2147, 2019.

[34] X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich, and A. Garg. Learning a contact-adaptive controller for robust, efficient legged locomotion. *arXiv preprint arXiv:2009.10019*, 2020.

[35] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots. Fast and efficient locomotion via learned gait transitions, 2021.

[36] Z. Xie, X. Da, B. Babich, A. Garg, and M. van de Panne. Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model, 2021.

[37] Anymal website. https://www.anybotics.com/anymal-legged-robot/. Accessed: 2021-2-26.