Learning to Jump from Pixels

Anonymous Author(s)

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]. However, they perform blind locomotion and primarily rely on proprioception and robust control schemes to achieve sturdy locomotion in challenging conditions involving snow, thick vegetation, slippery mud, etc. The downside of not using visual observations is the inability to execute motions that require knowledge of the land surface in front of the robot or to plan ahead-of-time. 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 limits 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, climb over stairs, etc. But, 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 have overcome this limitation and plan directly from depth observations [4, 7]. These works search for stable footholds and use model-based controllers to plan safe trajectories. Often the model-based optimization makes several assumptions such as a fixed gait (e.g., the trot gait [7]) or three out of the four legs of the robot being in contact with the ground at every time step [5], etc., that results in conservative and non-agile locomotion.

The challenge in planning agile behaviors such as jumps for moving on discontinuous terrains is different and complementary to traversing on continuous and unstructured terrain. Executing the
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]. The
most impressive demonstration to date has been of MIT Cheetah 2 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 leverage the success of
deep reinforcement learning and predict joint torques directly from visual inputs. However, because
RL algorithms are data-inefficient, training must be performed in simulation. This leads to another
problem – the agent learns to exploit inaccuracies in the simulation to achieve high reward, but
such policies don’t transfer to the real system. These issues can be mitigated by planning in an
abstract action space encoding locomotion-specific priors instead of joint torques. Motivated by
this, one line of work uses a trajectory generation method known as PMTG [16]. PMTG with
domain randomization can successfully transfer simulated walking behaviors to the real world. 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 [17].

The sim-to-real problem is severely aggravated by the requirement of agility and operation from
visual inputs. Robustness is therefore of paramount importance! As a step towards vision-guided
agile locomotion, we present a general framework for obtaining agile behaviors without sacrificing
robustness. To evaluate such a system, we constructed 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 1). Some of these gaps are larger than the quadruped’s
body length and thus necessitate a jump. 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 [18] as the experimental platform and report results
both in simulation and the real world.

While no individual component of our framework is novel, the specific integration choices allow us
to achieve the best-of-both-worlds: flexibility of model-free learning and the robustness of model-
based trajectory optimization. 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 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-methods (e.g.,
PMTG [16]). 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, 19] 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 allows the robot to flexibly change when its leg contacts the ground and is key to crossing
a sequence of wide gaps. 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. The other possibility is to leverage the dynamics model of
the robot and solve for the joint torques using powerful trajectory optimization methods – a scheme
commonly known as whole-body-control (WBC) [20, 21, 22, 23]. Because whole-body controllers
operate at high frequency and are aided by lower-level PD control systems, they confer robustness.
One issue however is that a typical WBC tracks the robot’s center-of-mass (CoM) [20, 21] 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. Such a whole-body impulse controller
(WBIC) is well suited for highly dynamic motions [23].

We found that while it was possible to train the high-level policy from depth images, training using
the ground truth height map is more sample efficient and yields higher final performance (see Section
5.2). However, in real-world environments, the heightmap is typically either unavailable or must be
constructed in real-time [4] – a requirement that becomes extremely challenging as the speed of the
robot increases. We overcome this issue through a two-stage asymmetric behavioral cloning [1, 19]
where 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 \( T_t = [\mathbf{p}_b, \mathbf{p}_{\text{LF}}, \mathbf{p}_{\text{RF}}, \mathbf{p}_{\text{LR}}, \mathbf{p}_{\text{RR}}, \mathbf{C}] \in \mathbb{R}^{34} \times [0, 1]^4 \) where \( \mathbf{p}_b = [x, y, z, \alpha, \beta, \gamma] \in \mathbb{R}^6 \) is the robot body pose (position \((x, y, z)\) and euler angles \((\alpha, \beta, \gamma)\)). The terms \( \mathbf{p}_f = [p_{x,\text{RF}}, p_{y,\text{RF}}, p_{z,\text{RF}}, p_{x,\text{LF}}, p_{y,\text{LF}}, p_{z,\text{LF}}, p_{x,\text{LR}}, p_{y,\text{LR}}, p_{z,\text{LR}}, p_{x,\text{RR}}, p_{y,\text{RR}}, p_{z,\text{RR}}] \in \mathbb{R}^{12} \)
denote the position of the Right (R.), Left (L.) Front (F) and Rear (R.) feet respectively. \( \mathbf{C} = [I_{\text{C}}, I_{\text{LF}}, I_{\text{RF}}, I_{\text{LR}}, I_{\text{RR}]} \in [0, 1]^4 \) is the binary contact state of each foot, with \( I_{\text{C}} \) 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 \( \mathbf{a}_t = \pi(\mathbf{s}_t, \mathbf{o}_t, \mathbf{a}_{t-1}) \) 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 \( \pi \).

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
\( \mathbf{o}_t = \mathbf{E}_t \in \mathbb{R}^{48 \times 15} \) or a depth image \( \mathbf{o}_t = \mathbf{I}_t \in \mathbb{R}^{160 \times 120} \) 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:

$$C_{trot} = \begin{cases} [1, 0, 0, 1] & t < d/2 \mod d \\ [0, 1, 1, 0] & t \geq d/2 \mod d \end{cases}$$

$$C_{pronk} = \begin{cases} [1, 1, 1, 1] & t < d/2 \mod d \\ [0, 0, 0, 0] & t \geq d/2 \mod d \end{cases}$$

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

$$C_{flex} = \{ [fC(a_t[4])] \} \text{ at time } t$$

where \(fC\) 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 \(a_t\) is converted into the desired whole-body trajectory for the next \(H\) time steps as:

$$T_{des}^{T+H} = [p_b(a_{t-1}), p_b(a_t), \dot{p}_b(a_t), \ddot{p}_b(a_{t-1}), p_f^{raibert}, \dot{p}_f^{raibert}, \ddot{p}_f^{raibert}, C]$$

where the key quantity adapted by the policy is \(\dot{p}_b(a_t)\) (denoted \(\dot{x} = \dot{a}_t[0], \dot{y} = \dot{a}_t[1], \dot{z} = \dot{a}_t[2], \dot{\alpha} = 0, \dot{\beta} = 0, \dot{\gamma} = 0\) from which \(p_b(a_{t-1})\) and \(p_b(a_t)\) are fixed for dynamic consistency. \(p_f^{raibert}, \dot{p}_f^{raibert}, \ddot{p}_f^{raibert}\) are foot targets satisfying the Raibert Heuristic (see supplement).

**Whole-body Trajectory Tracking** is performed using the hybrid control scheme described in [23]. It is a high-frequency controller that solves a quadratic program (QP): \(q_{des} = QP(T_{des}, T)\) without access to terrain information. It is composed of three controllers operating hierarchically:

- **A Model Predictive Controller (MPC)** converts the future whole-body trajectory \(T_{des}^{T+H}\) into target ground reaction forces \(f_{t+H}\) 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 whole-body trajectory \(T_t\) and desired ground reaction forces \(f_t\) 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:

$$r_t = c_1(p_{b,x}^t - p_{b,x}^{t-1}) - c_2 \max(0, ||v_{b,x}^t||_2 - V_{thresh}) - c_3|\alpha_{t,x}^b| - c_4|\beta_{t,x}^b| - c_5|\gamma_{t,x}^b| - c_6|\dot{q}|$$

The first term rewards the forward progress \(p_{b,x}^t - p_{b,x}^{t-1}\), where \(p_{b,x}^t\) 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 \(v_{b,x}^t\) exceeds \(V_{thresh}\). The third, fourth, and fifth terms incentivize stability by penalizing the roll, pitch, and yaw of the body, denoted as \(\alpha_{t,x}^b, \beta_{t,x}^b, \gamma_{t,x}^b\). 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.

**Algorithm 1 Policies Modulating Whole-body Objectives**

1: \(t \leftarrow 0; a_{-1} \leftarrow 0\)
2: observe \(s_0, o_0\)
3: while not IS-TERMINAL(s_t) do
4: \(s_t, o_t \sim \pi(s_t|s_{t-1}, a_{t-1})\)
5: \(T_{t+H} \leftarrow T(a_t)\)
6: \(T_{des}^{t+H} = T_{R}(s_t, T_{t+H})\)
7: \(t \leftarrow t + 1\)
8: observe \(s_t, o_t\)
9: end while
3.1 Neural Network Training

**Network Architecture**  The policy $\pi_{\theta}(a_t | s_t, o_t, a_{t-1})$ is modeled using a deep recurrent neural network that includes a convolutional neural network (CNN) to process the high-dimensional terrain observation $o_t$. The output features of the perception module are concatenated with proprioceptive inputs $s_t$, previous action $a_{t-1}$, and a cyclic timing parameter [16] and passed through a sequence of fully connected layers to output a probability distribution over the next action $a_t$. 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.

**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 ($\theta$) are optimized using Proximal Policy Optimization (PPO) [24]. We use Adam optimizer [25] 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 ($\pi_E$) from ground truth height map. A second deployable policy ($\pi_{BC}$) is trained from depth inputs to mimic the expert policy. For this, we use a variant of Behavioral Cloning (BC) known as DAgger [26] to minimize the KL-divergence between the output action distribution of the imitating agent $\pi_{BC}(a | s)$ and the expert $\pi_E(a | s)$: $\min D_{KL}(\pi_E(a | s) || \pi_{BC}(a | s))$. 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 $\pi_{BC}$ as a recurrent neural network. Prior works have applied similar approaches to blind rough-terrain locomotion [1] and autonomous driving [19]. 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 [18], 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 [27] 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
5 Results

5.1 Dynamic Performance

By design, learning with trajectory generators 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- to 70-cm gaps), the visually informed policy learns to select a variable-bounding contact schedule
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 $W_{\text{max}}$, the evaluated gap width is $W_{\text{max}} - 5$.

<table>
<thead>
<tr>
<th>Input</th>
<th>T, 10cm</th>
<th>T, 20cm</th>
<th>P, 20cm</th>
<th>P, 30cm</th>
<th>VP, 30cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heightmap (MLP)</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.7</td>
<td>1.0</td>
</tr>
<tr>
<td>Depth Image (RNN)</td>
<td>0.6</td>
<td>0.3</td>
<td>0.9</td>
<td>0.9</td>
<td>0.7</td>
</tr>
<tr>
<td>Heightmap (MLP) $\rightarrow$ Depth Image (MLP)</td>
<td>1.0</td>
<td>0.9</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Heightmap (MLP) $\rightarrow$ Depth Image (RNN)</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>0.4</td>
<td>1.0</td>
</tr>
</tbody>
</table>

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 $W_{\text{max}}$ 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 E.2). 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 [18]. 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. 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. Although we do not yet use real-time terrain observations in the deployments shown, our integrated system runs fast enough to be applied for real-time visual control. Figure 1 illustrates successful deployment of our visually adaptive pronk across two fourteen-centimeter gaps.

6 Related Work

Model-free RL for locomotion is shown to benefit from acting over low-level control loops rather than raw commands [28]. 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 [29] 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 [29] to facilitate transfer to the real robot.

Model-based control for locomotion has achieved highly dynamic blind walking [30], running [23], 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, 31]. 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. [32] 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. [33, 34] 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. [35] 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, [35] 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 dynamics parameters such as the robot’s inertia, friction coefficient etc. we made use of a motion capture system to infer 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


