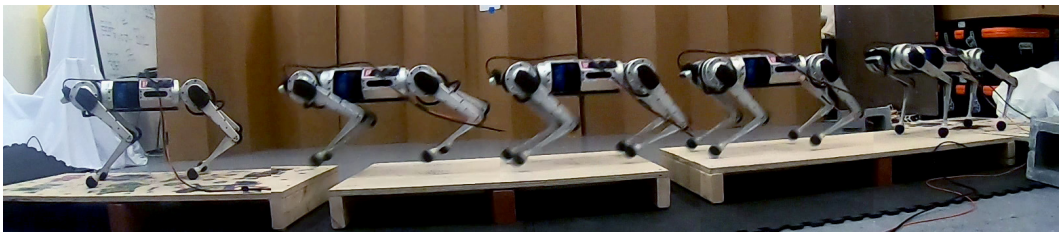


Figure A.1: Our learned visually adaptive pronk is deployed successfully on the MIT Mini Cheetah.

A Low-Level Controller Details

Complete information about the Model-Predictive Controller and Whole-Body Impulse Controller used in this work may be found in [21]. A simplified description of the control loop is given in Algorithm 2. The convex Model-Predictive Controller solves for target ground reaction forces over the planning horizon given the target whole-body trajectory and current state. The whole-body impulse controller then tracks these ground reaction forces at high frequency.

Algorithm 2 Trajectory Tracking Control Loop

function TRACK-TRAJECTORY($\mathbf{s}_t, X_{t:t+H}$)
  $f_r^{\text{target}} = \text{MPC}(\mathbf{s}_t, X_{t:t+H})$
  for $i \in [1..N_{WBIC}]$ **do**
    $q_{des}, \dot{q}_{des}, \tau_{ff} = \text{WBIC}(\mathbf{s}_t, X_t, f_r^{\text{target}})$
    SET-PD-TORQUE($q_{des}, \dot{q}_{des}, \ddot{q}_{ff}$)
    STEP-SYSTEM
  end for
end function

B Training Environment Details

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.

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 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 $\alpha^b_t, \beta^b_t, \gamma^b_t$. The sixth term rewards smooth motion by minimizing $\dot{q}$. In training with variable and unconstrained gaits, we found this term critical to promote exploration of lower-frequency gaits. The parameters in the reward term are set to: $c_1 = 1.0, c_2 = 0.5, c_3 = 0.02, c_4 = 0.05, c_5 = 0.15, c_6 = 0.03, V_{thresh} =$ 1.0m/s.

C Derivation of Theoretical Gap-Crossing Limits

C.1 Raibert Heuristic

Given the velocity of the base and the duration of the next contact, the Raibert Heuristic selects foot placements such that each leg's lever angle of incidence on the ground is equal to its angle of

departure. This foot placement follows the formula

$$p_{\text{symmetry}} = \frac{\Delta t_d^{l_i}}{2} v + k(v - v^{\text{cmd}})$$

where $\Delta t_d^{l_i}$ is the duration of the next placement of foot i, v is the estimated robot body velocity, v^{cmd} is the commanded robot velocity, and k is a tunable gain term.

C.2 Theoretical Limit on Fixed-Gait Gap Crossing

A quadruped stepping at a fixed cycle frequency f and moving at velocity $v = v^{cmd}$ will locomote a distance of $\frac{v}{f}$ each gait cycle, and so the nominal foot placement for any given foot under the Raibert heuristic will advance by a distance of $\frac{v}{f}$. In the pronking gait, wherein all legs contact the ground simultaneously, this places the upper limit on gap crossing at $\frac{v}{f}$. For a trotting gait, wherein pairs of diagonal legs meet the ground in alternating timing, this limit is reduced by half to $\frac{v}{2f}$. The Mini Cheetah nominally trots at a frequency of $f = 3$Hz, theoretically limiting its maximal gap crossing at a velocity of $v = 1$m/s to 33cm in the pronking case, or 17cm in the trotting case. Table C.1 lists derived limits for a few example gait frequencies, body velocities, and gaits. In practice, the popular ANYmal C by ANYbiotics, over twice as long and 5 times as massive as the Mini Cheetah, is rated by its manufacturers to cross gaps of up to 25cm.

Gait Cycle Frequency	*Body Velocity*	Trotting	Pronking
$3Hz$	$0.5m/s$	$8.3cm$	$16.7cm$
$3Hz$	$1.0m/s$	$16.7cm$	$33.3cm$
$4Hz$	$0.5m/s$	$6.3cm$	$12.5cm$
$4Hz$	$1.0m/s$	$12.5m$	$25.0cm$

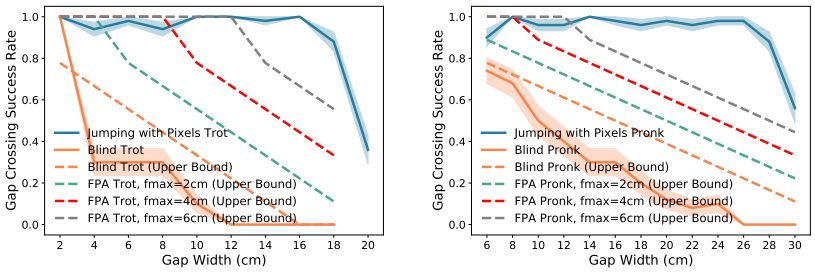
Table C.1: Theoretical upper limits on the longitudinal distance between foot placements for trotting, pronking, and bounding gaits.

C.3 Upper Bound on Gap Crossing Probability for Fixed Gait

Adhering to the Raibert heuristic with constant velocity and gait yields a distance d between footsteps of $d = \frac{v}{2f}$ for trotting and $d = \frac{v}{f}$ for pronking, as the analysis above shows. If we assume that a gap of width h is randomly positioned in the robot's path, the probability that any single foot steps into the gap is $\frac{h}{d}$. The probability that a single foot avoids the gap is then $1 - \frac{h}{d}$. The probability that *none* of the four feet step into the gap is less than or equal to the probability that any one foot avoids the gap. Thus, the probability of avoiding a randomly placed gap of width h for a blind controller at constant velocity is upper bounded at $1 - \frac{2fh}{v}$ for trotting and $1 - \frac{fh}{v}$ for pronking. Intuitively, the upper bound gap crossing probability decays linearly from one for a gap of width 0 to zero for a gap of width d.

C.4 Upper Bound on Gap Crossing Probability with Local Foothold Adaptation

The baseline controller of [3] performs locomotion with fixed gait at fixed velocity. However, if a foothold is detected to be unsafe, a local grid search is performed for the nearest safe foothold within some maximum displacement. Assume the maximum displacement Δ. Then, the probability of failure to cross a gap is the probability that a foot steps in the gap at least distance Δ from the edge. This results in single foot failure probability of $\frac{h-2\Delta}{d}$ for gap of width h and distance between footsteps d. The probability of avoiding a randomly placed gap of width h with maximum foot adaptation Δ is therefore upper bounded at $1 - \frac{2f(h-2\Delta)}{v}$ for trotting and $1 - \frac{2f(h-2\Delta)}{v}$ for pronking. Intuitively, the upper bound gap crossing probability decays linearly from one for a gap of width 2Δ to zero for a gap of width $d + 2\Delta$.



(a) (b)

Figure D.2: Performance comparison to the local foot placement adaptation (FPA) baseline [3]. Our method outperforms the best possible baseline performance across the feasible range of gap sizes for trotting **(a)** and pronking **(b)**.

D Local Foothold Adaptation Baseline

Local Foothold Adaptation Baseline [3] commands a constant velocity and contact pattern to a whole-body impulse controller, and adjusts foot placement locations locally by applying a safety heuristic to a terrain heightmap and searching for the nearest safe location to the nominal foothold. In our evaluation, we assume that this method has privileged access to the true terrain heightmap.

Figure D.2 presents a comparison between the performance achieved by our method (Jumping with Pixels) and the theoretical performance limits derived for rule-based foot placement adaptation (FPA) [3, 11] as described in C.4. Our method outperforms FPA across a range of maximum foot displacement values $f_{max} \in [2\text{cm}, 4\text{cm}, 6\text{cm}]$. Performance improvement is greater for wide gaps. Prior work [3] which implemented FPA on the same robot we use applied a maximum foothold adaptation of 4cm. We additionally note that foot placement adaptation is complementary to the velocity and contact schedule adaptation of our approach. We expect that future work combining these techniques will combine the performance improvement of each.

E PMTG Baseline

Policies Modulating Trajectory Generators (PMTG) [23] 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 $\mathbf{q}_{des} + \Delta\mathbf{q}_{des}$.

The policy of our baseline controller [23] given by Algorithm 3 has identical neural network architecture, proprioceptive inputs $\mathbf{s}_t$ and high-dimensional terrain observation $\mathbf{o}_t$ as described in Section 3.1. As in [1], we modeled the trajectory generator (TG) for each leg in the *Cartesian* space instead in the joint space. The frame of reference of the TG is attached to the hip joint of each leg with z-axis and x-axis of the frame in parallel with the base frame. The policy regulates the frequency f of the TG and outputs target foot position residuals $\Delta\mathbf{p}_{\text{f},t} \in \mathbb{R}^{12}$ at every time step $t = 0.025s$. These residuals are then added to the output of TG given by Algorithm 4 to calculate target foot positions $\mathbf{p}_{\text{f},t} \in \mathbb{R}^{12}$. The initial phases ϕ_0 for each leg is decided w.r.t. type of gait. Finally, the target joint positions q_{des} are computed from $\mathbf{p}_{\text{f},t}$ using an analytical inverse kinematics (IK) and tracked using a PD controller.

Algorithm 3 PMTG Controller

$t \leftarrow 0$; $\mathbf{a}_{t-1} \leftarrow \mathbf{0}$; $\phi_0 \leftarrow [0, \pi, 0, \pi]$
observe $\mathbf{s}_0$, $\mathbf{o}_0$
while not IS-TERMINAL($\mathbf{s}_t$) **do**
    sample $\mathbf{a}_t \sim \pi_\theta(\mathbf{a}_t|\mathbf{s}_t, \mathbf{o}_t, \mathbf{a}_{t-1}, \phi_t)$
    $\Delta\mathbf{p}_{\mathrm{f},t} \leftarrow \mathbf{a}_t[0:12]$; $f \leftarrow \mathbf{a}_t[12]$;
    $\mathbf{p}_{\mathrm{f},t} \leftarrow \mathrm{TG}(\phi_t) + \Delta\mathbf{p}_{\mathrm{f},t}$
    $q_{des} \leftarrow \mathrm{IK}(\mathbf{p}_{\mathrm{f},t})$
    SET-PD-TORQUE($q_{des}, q_t, 0, \dot{q}_t$)
    $t \leftarrow t + 1$
    $\phi_t \leftarrow (\phi_0 + 2\pi f * t)(mod 2\pi)$
    observe $\mathbf{s}_t$, $\mathbf{o}_t$
end while

Algorithm 4 Trajectory Generator **TG**(ϕ)

$i \leftarrow 0$; $h \leftarrow 0.17$; $d \leftarrow 0.08$; $\mathbf{p}_{\mathrm{f}} \leftarrow \{\}$
while $i < 4$ **do**
    $k \leftarrow 2(\phi[i] - \pi)/\pi$
    $p_x \leftarrow d * cos(\phi_i)$; $p_y \leftarrow 0$
    if $k \in [0, 1]$ **then**
        $p_z \leftarrow h(-2k^3 + 3k^2) - 0.28$
    else if $k \in (1, 2]$ **then**
        $p_z \leftarrow h(2k^3 - 9k^2 + 12k - 4) - 0.28$
    else
        $p_z \leftarrow -0.28$
    end if
    $\mathbf{p}_{\mathrm{f}}$.APPEND(p_x, p_y, p_z)
    $i \leftarrow i + 1$
end while
return $\mathbf{p}_{\mathrm{f}}$

The **reward function** for training the baseline controller is defined as $r_t = c_1 * r_{dx} + c_2 * r_{v_{thres}} + c_3 * r_r + c_4 * r_p + c_5 * r_y + c_6 * r_{GC} - c_7 * p_{GA}$, where $c_{1,2,3...7}$ are the coefficients of each reward terms respectively. The individual terms are defined as follows:

- Forward Distance Reward (r_{dx}) : This term maximizes the forward distance moved by the robot in x-direction. $r_{dx} = p^b_{t,x} - p^b_{t-1,x}$
- Velocity Threshold Reward ($r_{v_{thres}}$): This term penalizes if the robot body velocity $||v^b_t||_2$ exceeds threshold velocity V_{thresh}.

$$r_{v_{thres}} = \begin{cases} -1 + exp(-8(||v^b_t||_2 - V_{thresh})^2) & ||v^b_t||_2 > V_{thresh} \\ 0 & otherwise, \end{cases}$$

- Orientation Reward (r_r, r_p, r_y) : This term incentivizes stability of the robot, maintaining zero roll, pitch and yaw.

$$r_{r,p,y} = exp(-40(\alpha^b_t, \beta^b_t, \gamma^b_t)^2)$$

- Gap Crossing Reward (r_{GC}): Unlike [4], which intensively shape the reward near the gap, we considered a binary reward(0/500) which incentivizes if the robot body successfully reaches the other side of the gap.
- Gap Avoidance Penalty (p_{GA}): This term penalizes and terminates the environment when number of gaps crossed is 0 after 180 time-steps.

We found that Gap Crossing Reward (r_{GC}) and Gap Avoidance Penalty (p_{GA}) were critical for baseline policy as opposed to our proposed controller. Figure E.3 shows that the PMTG baseline without these reward terms learns to trot in place and avoids crossing any gaps of width 10cm, while the PMTG baseline with reward shaping is able to demonstrate a gap crossing behaviour. We find that learning with our method of whole-body trajectory modulation achieves higher performance in this environment and converges more rapidly than the baseline.

F PMTG Sim-to-Real Results

To verify our implementation of PMTG used for baseline comparison, we trained and deployed a simple forward walking policy on flat ground. The results of deployment are illustrated in Figure F.6.

We trained trotting and pronking policies in simulation to cross gaps of different width. We found that parameters such as TG frequency f and residual Δp_x are mainly responsible for exploration over large gaps. We performed an experiment with policies trained over different range of these parameters as follow:

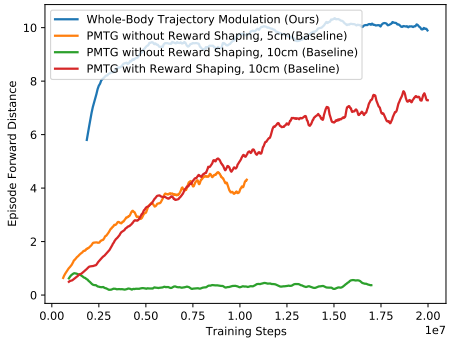


Figure E.3: PMTG does not learn to trot across 10cm gaps without shaped reward. Our method with whole-body low-level controller converges rapidly without shaped reward for gap crossing.

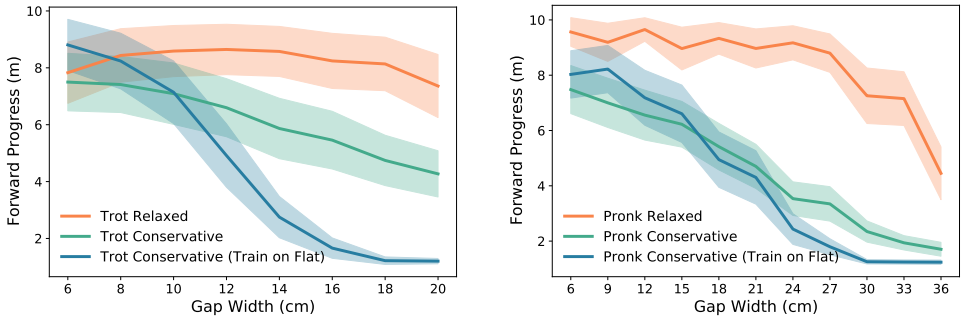


Figure E.4: A conservative trajectory generator inhibits gap crossing capability in PMTG, particularly for pronking gait. Relaxed trajectory generators are more successful, but Figure E.5 suggests they tend to exploit the simulator.

- **Trot/Pronk Relaxed** : Policies trained with $\Delta p_x \in [0cm, 10cm]$ and $f \in [3Hz, 5Hz]$
- **Trot/Pronk Conservative** : Policies trained with $\Delta p_x \in [0m, 7cm]$ and $f \in [3Hz, 4Hz]$
- **Trot/Pronk Conservative (Train on Flat)** : Policies trained with $\Delta p_x \in [0cm, 4cm]$ and $f \in [2.5Hz, 3.5Hz]$. These are the parameter ranges which was used to train a policy deployed on the flat-ground as shown in Fig. F.6

Figure E.4 shows the performance of policies trained with each parameter range. We note that only the relaxed policy is able to learn successful gap crossing behavior for large gaps, particularly for the pronking gait. Figure E.5 illustrates simulated body and foot trajectories of the converged gap-crossing policies trained with each set of TG parameter ranges. We observe that although policies trained with relaxed ranges achieve improved gap-crossing performance, their trajectories include foot dragging, high footswings, and unrealistic velocity changes. The policies with relaxed TG use larger residual commands than those trained with conservative TG, which enables them to discover these unrealistic patterns of simulator exploitation.

G Depth Image Preprocessing

The depth images produces by the depth sensor used in this work contain imperfections that are not modeled in simulation. To mitigate this, we apply standard preprocessing techniques before passing each depth image to the controller:

1. **Downsampling**: For computational efficiency, we downsample each image from its original resolution of 480×360 to 160×120 using nearest-neighbor interpolation.

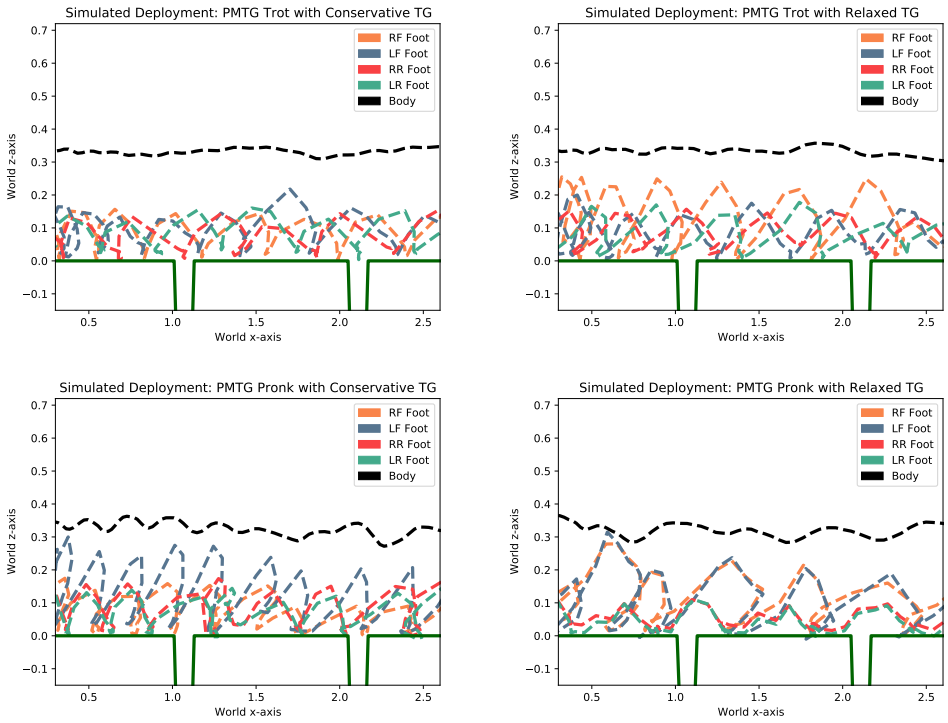
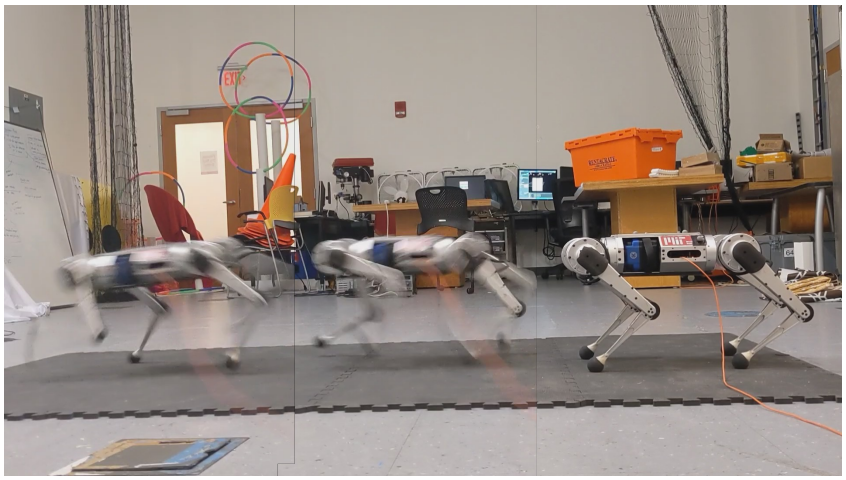


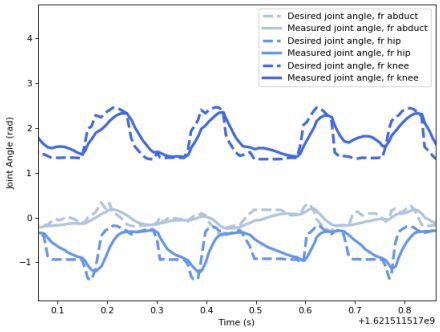
Figure E.5: Relaxing the ranges of trajectory generator parameters and residuals in PMTG improves performance, but results in large residual commands and motions unsuitable for sim-to-real.

2. **Invalid Band Crop**: Since our depth sensor is steroscopic, there is an ”invalid band” of variable size on one side of the image. In this band, some objects in the scene are only detected by one camera, preventing their distance from being accurately estimated. In the case of our system, we found that cropping the left side of the image by 20 pixels (after downsampling) avoided the invalid band while retaining sufficient terrain information to perform the task.
3. **Depth Filter**: We clip the points in the depth image to the range [0.1m, 1.0m].
4. **Hole-Filling Filter**: We apply a hole-filling filter from the `pyrealsense2` package to generally reduce noise artifacts in the image.

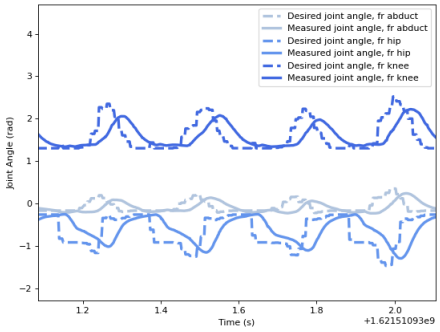
Figure G.7 provides example depth images before and after preprocessing.



(a) Three frames of locomotion captured during deployment.

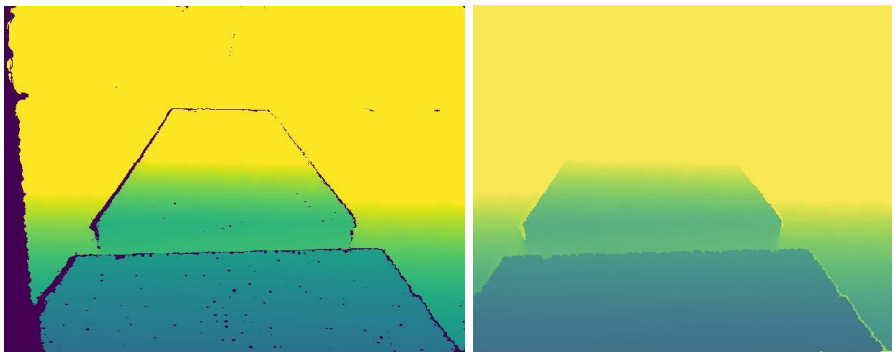


(b) Joint angle tracking in simulation.



(c) Joint angle tracking in deployment.

Figure F.6: Baseline deployment of learned forward locomotion policy using PMTG architecture.



(a) (b)

Figure G.7: Example of initial depth image **(a)** and corresponding image after preprocessing **(b)**.