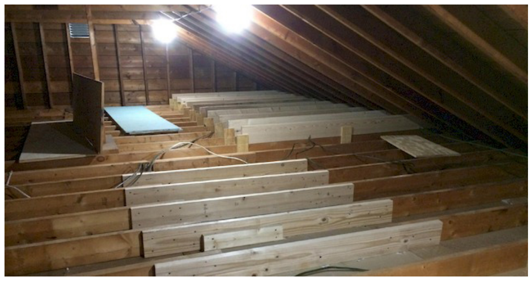
Perceptive Hexapod Legged Locomotion for Climbing Joist Environments

Zixian Zang[1], Maxime Kawawa-Beaudan[1], Wenhao Yu[2], Tingnan Zhang[2], Avideh Zakhor[1]

***Abstract*— Attics are one of the largest sources of energy loss in residential homes, but they are uncomfortable and dangerous for human workers to conduct air sealing and insulation. Hexapod robots are potentially suitable for carrying out those tasks in tight attic spaces since they are stable, compact, and lightweight. For hexapods to succeed in these tasks, they must be able to navigate inside tight attic spaces of single-family residential homes in the U.S., which typically contain rows of approximately 6 or 8-inch tall joists placed 16 inches apart from each other. Climbing over such obstacles is challenging for autonomous robotics systems. In this work, we develop a perceptive walking model for legged hexapods that can traverse over terrain with random joist structures using egocentric vision. Our method can be used on low-cost hardware not requiring real-time joint state feedback. We train our model in a teacher-student fashion with 2 phases: In phase 1, we use reinforcement learning with access to privileged information such as local elevation maps and joint feedback. In phase 2, we use supervised learning to distill the model into one with access to only onboard observations, consisting of egocentric depth images and robot orientation captured by a tracking camera. We demonstrate zero-shot sim-to-real transfer on a Hiwonder[1] SpiderPi robot, equipped with a depth camera onboard, climbing over joist courses we construct to simulate the environment in the field. Our proposed method achieves nearly 100% success rate climbing over the test courses, significantly outperforming the model without perception and the controller provided by the manufacturer.**


(a) Unfinished attic

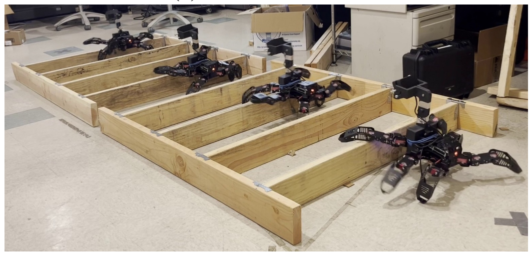

(b) Time-lapse photo of SpiderPi climbing over joists

Fig. 1: (a): Photo of an unfurnished attic with joists. (b): Time-lapse photo capturing the SpiderPi robot climbing over 8 joists using the method proposed in our work.

I. Introduction

Attics are one of the largest sources of energy loss in residential homes. A typical unfinished attic is shown in Fig. 1(a). A substantial reduction in home energy costs and its carbon footprint can be achieved through attic air sealing and insulation. However, it is usually difficult, uncomfortable, and potentially dangerous for workers to carry out these tasks. For example, since attics typically consist of multiple rows of joist structures, a human worker could easily fall through the attic floor and get seriously injured if he or she steps on the sheetrock between two joists by mistake. Workers may also need protective suits to protect themselves from toxic substances during vacuuming and spray foaming, which are common tasks for air sealing and insulation.

Lightweight legged robots are ideal platforms for navigating inside attics to carry out various tasks such as air-sealing and vacuuming. To do so, we need to enable legged robots to traverse in environments with dense and high joists. Since most single-family residential home attics in the U.S. contain approximately 6 or 8-inch high joists that are 16 inches apart, it is important for legged robots to be able to autonomously climb such joists inside attics.

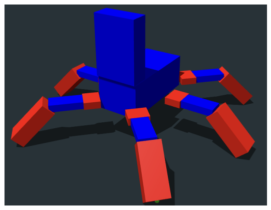
There has been a significant amount of work on quadrupeds and bipedal robot locomotion. However, in this work, we focus on using hexapods for joist climbing for two main reasons. First, hexapod robots are by design more stable and lightweight than quadrupeds and humanoids of similar size. Secondly, bipedal or quadruped robots are often taller than hexapods from the ground and are therefore less suitable for traversing within tight spaces such as the corners of attics.

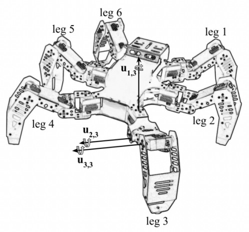
To facilitate the practical usage of robots in the retrofit business, it is important for legged locomotion controllers to work with low-cost hardware. However, most existing legged locomotion systems require high-end robots capable of real-time sensing of joint states, which could ultimately result in expensive hardware. For example, model predictive control methods such as [6] require powerful computation resources and real-time joint feedback from expensive robot platforms and often compromise real-time performance when incorporating more complex dynamics. Data-driven methods such as [2] can work with limited computation resources and are robust to a variety of perception failures but need fast joint state feedback. Many low-cost robots are not equipped with powerful onboard computation or real-time feedback such as joint torque and angle that are accessible on more

[1]UC Berkeley, [2]Robotics at Google

expensive platforms. Meanwhile, humans without leg sensing feedback when equipped with prosthetics can walk and even participate in competitive sports with only egocentric visual perception and a sense of body orientation [25].

(a) Physical Robot


(b) URDF



(c) Joint Structure

Fig. 2: (a): The SpiderPi hexapod robot used in our work, with Intel L515 and T265 mounted onboard. (b): URDF model of SpiderPi robot we create for training. (c): Joint structure of SpiderPi hexapod.

In this paper, we propose an end-to-end learning-based perceptive controller for low-cost, sub-thousand-dollar hexapods to autonomously climb over joists and demonstrate zero-shot sim-to-real transfer on joist terrain with configurations similar to the ones in typical attics. Our robot is a \$600 SpiderPi robot manufactured by Hiwonder, shown in Fig. 2(a), equipped with an Intel L515 depth camera and a T265 tracking camera with a customized camera mount. The robot does not have real-time joint feedback. The whole system including the sensors costs less than \$1500. We propose a two-stage teacher-student training procedure to learn models that can work without real-time joint feedback: the first stage involves Reinforcement Learning (RL) with access to privileged observations and the second stage uses supervised learning to distill the model using only onboard observations including body orientation and egocentric depth images. Since optimal joist climbing motions are fundamentally different from walking, we train our controllers without human-defined prior gait knowledge, guiding the models to explore task-appropriate motions. We compare our method to a few baselines including end-to-end training of a blind model and a hand-designed controller. Our proposed method significantly outperforms both by achieving a much higher success rate climbing over joists and higher speed. In addition, we study the design of the privileged information during phase 1 training and find that it is critical to include information such as joint feedback even though it is not available onboard. A time-lapse photo of using our proposed method to control the robot climbing over 8 joists with 6-inch height and 16-inch spacing is presented in Fig. 1(b).

II. RELATED WORK

A. *Hexapods Legged Locomotion on Challenging Terrain*

Hexapod robots traversing challenging terrain have been studied for decades. In 1990, [19] proposed a set of rules to control and navigate a hexapod robot. In [12], Frankhauser *et al.* developed a ROS library for navigating a hexapod or quadruped using elevation map. Shortly after that, Frankhauser *et al.* demonstrated solid results by deploying their method on a quadruped in [10]. [6] proposes a method to control hexapods on unstructured terrain using combination of exteroceptive and proprioceptive terrain characterization. Faigl *et al.* [9] use only position feedback to let a hexapod walk on uneven terrain. [21] proposes a teleoperator for hexapods under environmental perturbations. [8] improves passability of gait selection on sparse foothold environments using a Monte Carlo tree search algorithm. [30] proposes an adaptive hexapod controller using a force sensing pipeline. However, these methods are only capable of managing explicitly modeled uncertainties and therefore are not suitable for tackling uncommon terrains. To overcome this, data-driven methods are proposed by Li *et al.* in [22] and Qiu *et al.* in [26]. [4] enables hexapods to traverse rough terrain in simulation using binary contact sensors on feet endpoints. [20] uses RL to train a central pattern generator with spiking neural networks. However, there is no existing work showing a hexapod robot traversing challenging terrains using a learned controller. Our work adopts end-to-end learning without any human-defined gait prior knowledge, allowing the agent to be deployed on inexpensive hexapods and to perform reliably on joist structures.

B. *Legged Locomotion with Perception*

For a legged robot to efficiently climb high obstacles such as joists, it must be equipped with a perception system to identify and step over the obstacles. To incorporate perception in legged locomotion, most previous model-based methods first convert raw input from depth cameras or LiDARs into other forms such as local elevation maps; they then use elevation maps to evaluate local areas for foothold feasibility [28], [15], [24], [11], [3], [18], [16], [27] or traversability [7], [13]. These methods assume perfect conversion from raw sensor data to a desired form such as an elevation map. Conversion to elevation map is computation intensive, requiring expensive and heavy compute engines that must be carried by the robot. Recent data-driven approaches map from perception input directly to actions. Specifically, [29] proposes a method for quadrupeds to walk on complex terrain by predicting actions directly from depth captured onboard and executing actions with a model predictive controller. [14] trains an end-to-end controller that maps from depth images to motor commands and demonstrates the robot traversing various terrains in simulation. [23] proposes a method to produce high-level commands for jumping over gaps. In [2], Agarwal *et al.* use egocentric depth to directly predict joint angles. However, to work on actual robots, these methods need high-frequency state feedback provided by

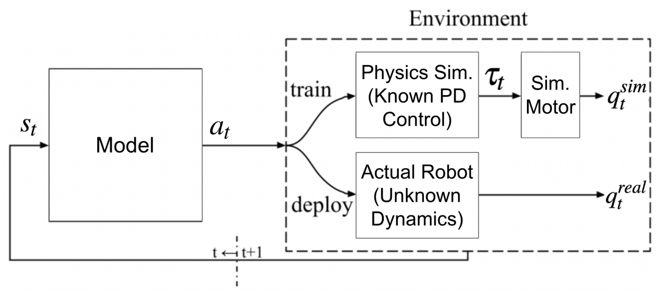


Fig. 3: High-level overview of our method.

costly hardware. In contrast, in this work, we incorporate visual perception on low-cost hardware without access to real-time joint states.

III. OVERALL APPROACH

In this section, we present our overall approach, including the choice of hardware, training strategy, and ways to retrieve the robot model used for training. We use a \$600 SpiderPi hexapod robot, shown in Fig. 2(a), which has a standing height of approximately 15cm without our custom-designed camera mount, 30cm with the mount, and weights around 2.4 kilograms. The onboard computer on our robot is a Raspberry Pi 4B with 4GB of RAM, 4 CPU cores, and 1.5 GHz clock rate. As shown in Fig. 2(c), the robot has six legs with three degrees of freedom per leg, each actuated by a LX-224HV three-port bus servo. Each servo provides a maximum of 1.96 Nm of torque and 5.24rads of angular velocity.

Even though we train fully in simulation, there are no models to simulate the motors reliably since our robot uses actuators with unknown dynamics. Unlike actuators in high-end robots that allow the reading and writing of Proportional Derivative (PD) controller gains, PD control is performed at the circuit level on LX-224HV by modulating motor voltage with unknown gains that cannot be modified. As a result, we use position control, which is physically less accurate. As depicted in Fig.3, during training, the model outputs desired joint angles a_t which are converted to joint torques τ_t and fed into motors in the simulation engine. When deployed on hardware, the joint angles are directly used as angle targets for servo motors on the actual robot with unknown dynamics.

We train our model using a teacher-student training scheme with two phases to target zero-shot sim-to-real transfer. The first phase is trained with access to privileged observations using reinforcement learning and the second phase is trained with only access to observations onboard. There are two reasons for having two training phases: First, depth rendering with current simulation significantly slows down simulation speed. Specifically, reinforcement learning takes a large number of samples to converge, so it is impractical to directly train perceptive models with depth sensing using reinforcement learning. Secondly, unlike actuators used on robots in high-end robots that have real-time joint feedback including position, velocity, and even torque, our servo positions are written to and read from joints through a shared serial port. This prohibits parallel access, so all joints are read and written to in serial, with wait times in between to clear the bus. One loop reading the position of all the joints can take multiple seconds, making it impossible to use them as real-time inputs to a controller. Similar to the case in [17], training without joint position feedback from scratch yields suboptimal performance, so we have to include joint feedback in phase 1 and distill it in phase 2. As a result, the model trained in phase 1 has access to joint angle feedback, body orientation, and elevation maps sampled around the robot. In phase 2, a student model with depth images as perception input learns from the model trained with elevation maps in phase 1 and takes body orientation as the only source of proprioception feedback. As we show later, models trained with no joint feedback in phase 1 achieve much lower reward than our proposed method.

State Element	Constituents	Units	Noise Range
Base attitude	o_t={roll, pitch, yaw}	Radians	$[-0.1, 0.1]$
Joint positions*	$q_t \in \mathbb{R}^{18}$	Radians	$[-0.01, 0.01]$
Previous actions	$a_{t-1} \in \mathbb{R}^{18}$	Radians	$[0,0]$
Elevation Map*	$h_t \in \mathbb{R}^{221}$	Meters	$[0.1, 0.1]$
Depth Image	$d_t \in \mathbb{R}^{320*240}$	Meters	$[0.1, 0.1]$

TABLE I: Constituent elements of the state. * denotes privileged observations that are only available during phase 1 training.

Since there is no available 3D model of our robot, we measure the length and weight of each leg segment and approximate the robot segments using boxes to build a URDF model for training in simulation, as shown in Fig. 2(b).

IV. METHOD

In this section, we describe our training method in more detail. We follow a teacher-student training scheme with two phases, with reinforcement and supervised learning as the first and second phases respectively. Since we directly deploy models trained from phase 2 on actual hardware, we incorporate several techniques to ensure the performance of zero-shot sim-to-real transfer.

A. Training Setup

We conduct training in IsaacGym simulator [5]. We first train on an easier rough terrain with randomly generated low joist courses, shown in Fig. 4(a), arranged to give rise to a learning curriculum similar to [5]. Then we fine-tune the model on a terrain that closely mimics the joists pattern in the field, as shown in Fig. 4(b).

We leverage parallel simulation training for our policies [5]. The state space is listed in Table I with corresponding units. Uniform random noises with listed ranges are added to the elements. At each time step, the perception input p_t, either an elevation map or a depth image, is first encoded by the perception head E_p into perception feature f_t:

$$f_t = E_p(p_t) \tag{1}$$

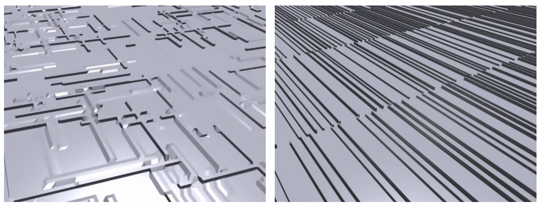


Fig. 4: Simulated terrains for training. (a): easy terrain. (b): difficult terrain.

Then the state $s_t = [x_t, f_t]$, resulting from the proprioception information x_t and perception feature f_t, are fed into the observation encoder E_o to produce a latent vector z_t:

$$z_t = E_o(s_t) \tag{2}$$

Then the encoded feature z_t is passed to the policy π that predicts the joint angles a_t:

$$a_t = \pi(z_t) \tag{3}$$

Our observation encoder E_o is a 1-layer LSTM with hidden size 64 and a CNN or MLP as perception head to extract features from perception input. The recurrent nature of the observation encoder enables the model to remember past observations and incorporate salient information into z_t, making it possible to traverse challenging terrain using proprioception and learn to overcome obstacles captured by egocentric vision. Our policy is an actor-critic with both actor and critic as a 3-layer MLP with hidden layers of size 128, 64, and 32. Exponential linear unit activation is used between the layers. We train the model in simulation with frequency of 25Hz. The simulation time step is set to 0.005 second and the simulation steps 8 times per model iteration. Even though the lightweight nature of our robot is useful in our application, its low inertial links require small time steps in simulation and hence long training times. Therefore, we have found it necessary to increase the number of substeps of the physics engine per simulation step from 1 to 4 to simulate physics accurately.

B. Phase 1: Reinforcement Learning

In phase 1, we train the model using the open-source implementation of Proximal Policy Optimization from [5]. The elevation map sampled around the robot is used as perception input in phase 1. As mentioned in [2], a critical step is to find the optimal elevation map configuration such that the elevation map does not contain information that cannot be inferred from the depth images used in phase 2 training. To achieve this, we start with a rough estimation of the ground area that the depth images on the actual robot capture. With the hardware configuration shown in Fig. 2, we consider a scenario with the robot standing still and the camera pointing at 30 degrees downward with FOV provided by L515 intrinsic information, and compute the projection area of the camera's view based on geometry. We then search elevation map configurations close to the rough projection area and use the one with the lowest phase 2 loss. This approach has been shown in [2] to result in optimal performance. The final elevation map is sampled from a 0.6m × 0.8m grid that is 0.3m in front of the robot with a sample spacing of 0.05m, resulting in a 13 × 17 elevation map.

Our reward is the weighted sum of the elements in Table II. This reward is shaped over multiple experiment cycles, beginning with only the linear velocity in body x term to incentivize forward motion. The linear velocity in body y and angular velocity yaw terms disincentivize sideways motion and turning to keep the agent on a straight line. Ground impact is added after initial real-world rollouts reveal the agent's tendency to let its feet collide harshly with the ground, damaging the hardware. The action rate, action magnitude, torque, and joint acceleration penalties are all added as regularizers to constrain the actions to behave in ways that are deployable on real hardware, instead of learning on-off control schemes that would burn out the motors or command impossible joint positions. We also incorporate the end effector height penalty since we observe the trained policies tend to raise the robot's end effctors high up in the air.

A one-layer MLP is used in phase 1 as perception head E_p of the observation encoder to encode elevation map h_t. During phase 1, proprioception information x_t consists of base orientation o_t, joint angles q_t, and the previous action a_{t-1}. Therefore, the state s_t has the following form:

$$s_t = [o_t, q_t, a_{t-1}, E_p(h_t)] \tag{4}$$

Models are first trained with the easier terrain shown in Fig. 4(a) and then tuned on the terrain with high and dense joists shown in Fig. 4(b), which is more difficult but more closely mimics the environment in the field. On each terrain, we use 4,000 robots simulated in parallel and train for 8,000 iterations, with 24 timesteps for all agents per iteration. 96,000 transition samples are created during each iteration and are split into 4 mini-batches with sizes of 24,000 each. To ensure the robustness and stability of the learned gait, we randomize the friction coefficient of each robot in the range [0.5, 1.25], and apply pushes in random directions to each robot's base mass every 8 seconds.

C. Phase 2: Supervised Learning for Sim-to-Real

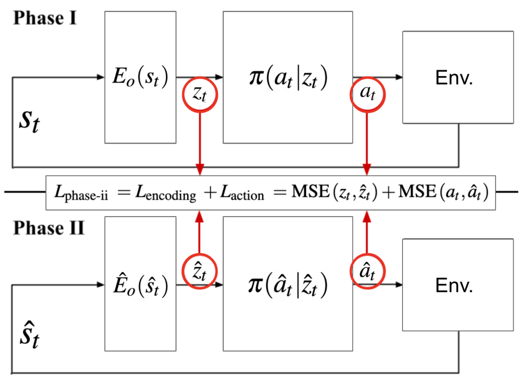
In phase 2, we use supervised learning to train the model with access to only proprioception and perception available onboard, supervised by a teacher model trained in phase 1. For the student model during phase 2 training, proprioception information $\hat{x}_t$ consists of base attitude o_t, previous action a_{t-1}, and egocentric depth image d_t as perception input. The state $\hat{s}_t$ for the student model then has the following form:

$$\hat{s}_t = [o_t, a_{t-1}, E_p(d_t)] \tag{5}$$

Only the observation encoder and its perception head are trained from scratch, with the policy being initialized with the trained policy weight from phase 1. Fig. 5 shows the phase 2 training process. We supervise the student encoder's $\hat{z}_t$ with the teacher encoder's z_t, and student policy's $\hat{a}_t$ with the teacher policy's a_t. In other words, we train the student

Reward Term	Expression	Weight
Linear velocity in global x	$\text{clip}(v_x, \min=-0.4, \max=0.4)$	1e2
Linear velocity in body y	v_y^2	-1e1
Global heading	θ^2	-3e1
Angular velocity: yaw	ω_z^2	-1e0
Ground impact	$\|f_t - f_{t-1}\|^2$	-1e-1
Collision penalty	$\mathbb{1}\{\text{coxa, femur, or base contacting terrain}\}$	-1e0
Action rate	$\|a_t - a_{t-1}\|^2$	·-5e-1
Action magnitude	$\|a_t\|^2$	-1e-2
Torques	$\|\tau\|^2$	-1e-3
Joint acceleration	$\hat{\ddot{q}}^2 = \frac{\dot{q}_t - \dot{q}_{t-1}}{\Delta t}^2$	-1e-5
Joint limit penalty	$\text{clip}(q_{\min} - q_t, \max=0) + \text{clip}(q_t - q_{\max}, \min=0)$	-1e0
End effector height	$\left\|z_{end\_effector}\right\|$	-1e1

TABLE II: Constituent elements of the multi-objective reward during phase 1 training.



Phase I
$E_o(s_t)$
z_t
$\pi(a_t|z_t)$
a_t
Env.
s_t
$L_{\text{phase-ii}} = L_{\text{encoding}} + L_{\text{action}} = \text{MSE}(z_t, \hat{z}_t) + \text{MSE}(a_t, \hat{a}_t)$
Phase II
$\hat{E}_o(\hat{s}_t)$
$\hat{z}_t$
$\pi(\hat{a}_t|\hat{z}_t)$
$\hat{a}_t$
Env.
$\hat{s}_t$

Fig. 5: Block diagram for phase 2 training.

encoder and policy to recover the same z_t and a_t, using the more limited observation set, by optimizing the following loss, as demonstrated in Fig. 5:

$$L_{\text{phase-ii}} = \text{MSE}(z_t, \hat{z}_t) + \text{MSE}(a_t, \hat{a}_t) \tag{6}$$

We also need to consider the sim-to-real gap of depth perception in order to successfully deploy the model trained in phase 2 on hardware. Depth images retrieved from Isaacgym are nearly perfectly simulated, which is not the case for our actual depth sensor. Since our robot is close to the ground and the distance between the depth camera and the ground is not too large, in order to obtain the best depth quality for near objects, we operate the onboard depth camera in low ambient light mode. By doing so however, far-away pixels often end up with invalid measurements. Moreover, since we do not have an exact 3D model of the robot and we approximate segments of the robot with boxes when the robot legs are within the view of the depth camera, there is a domain gap between the simulation and actual depth images.

To address the issue of front legs being inadvertently captured in the actual captured depth images, we mask out all the areas that could possibly be occupied by the legs with all physically possible joint angle combinations. The masked-out areas are presented as the white areas in Fig. 6(a). An overlay image of the leg mask on an egocentric depth image with the leg in view is shown in Fig. 6(b), with part of the robot leg captured by the depth camera is shown in the lower right corner and the highlighted area at the bottom corners being masked out.

To deal with the pixels in actual depth images that are invalid due to range, we randomly mask out pixels in simulated depth images based on the following probability:

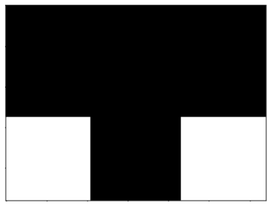
$$p_{invalid} = \frac{exp(min(d_{bound}, d))}{exp(k)} \tag{7}$$

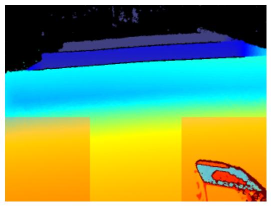
In the above expression, k is set to be the approximate upper bound of depth values for the majority of pixels on actual hardware. d_{bound} exists to make sure the probability of a pixel being masked out does not exceed 1, since we do not wish to mask out all pixels beyond a certain distance, $d_{bound} < k$. An example depth image captured in simulation is visualized in Fig. 6(c). The image with pixels being randomly masked out according to their depth values is shown in 6(d), with black pixels corresponding to the pixels that are masked out. To determine an appropriate range for k, we walked the robot around with the depth camera mounted using the built-in Hiwonder controller and plot the distribution of valid pixel values in multiple randomly selected captured depth images, shown in Fig. 6(e), and identified the range of k to be 3 based on collected depth data. We then tune d_{bound} to yield the best zero-shot sim-to-real performance. We find $d_{bound} = 2$ to result in the best performance.

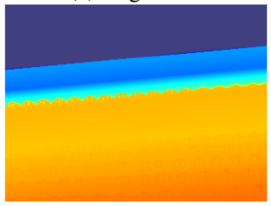
Phase 2 training is only conducted on the more difficult joist terrain environment to emphasize the trained model's ability to adapt to the environment in the field. Due to the compute intensity of depth retrieval in simulation, we only simulate 100 robots in parallel. We train the model for 2,000 iterations using Adam optimizer and learning rate of 5e-4.

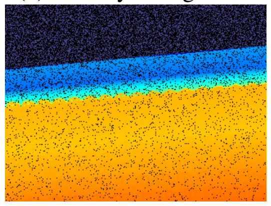
V. EXPERIMENTS

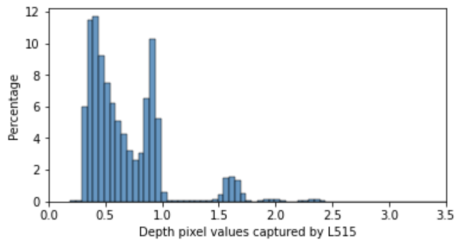
To evaluate the performance of our method, we deploy our proposed method, a blind version of our method, and the controller provided by the manufacturer, on a SpiderPi robot and have it climb over joist courses we have constructed to mimic attic environments. We also conduct ablation studies to investigate our design choices for sim-to-real transfer.


(a) Leg Mask


(b) Overlay of leg mask


(c) Example depth image


(d) Depth-based mask



(e) Depth Distribution on L515

Fig. 6: Techniques used in depth image processing. (a): Mask used to avoid the legs being captured. (b): Overlay of the leg mask on an egocentric depth image with the leg in view, with the highlighted area at the bottom corners masked out. Part of the robot leg captured by depth camera is visible at the lower right corner. (c): An example depth image captured in simulation. (d): Depth image in (c) with pixels randomly masked out according to depth value. (e): Distribution of depth value captured by L515 mounted on the robot.

A. Test Setup

To evaluate the robot's ability to traverse joists similar to the ones in attics, we construct a testing ground using wood joists with 2-inch thickness and 6-inch height. We construct a 4-joist and a 8-joist courses. Joists are arranged in a pattern with 16-inch center-to-center gap. The 8-joist course is shown in Fig. 1(b). We test the walking controllers by having the robot traverse in the direction that is perpendicular to the joists for multiple trials. The time limit to complete each trial is set to be 30 seconds for the 4-joist course and 60 seconds for the 8-joist course. The performance of each method is evaluated using two metrics: the average number of joists the robot successfully climbs over in each trial, and the average time it takes for the robot to climb over each joist. Success is defined as the entire body and all the legs of the robot passing over the joist. The time interval between passing the previous joist to passing the current joist is taken as the time to climb the current joist. We only interrupt a trial if the robot falls over and cannot recover on its own.

B. Performance on Actual Hardware

For comparison, we also train a blind model and use it without the depth camera. The training process of the blind model is the same as the perceptive policy, except that it does not have access to any perception information, without a perception head on the observation encoder. The blind model also takes body orientation as proprioception, so only the T265 tracking camera is attached to the robot when using the blind model. We compare the performance of our perceptive model to both the blind model and the walking controller provided by HiWonder. All models are deployed on a low-cost Raspberry Pi 4B with 4GB of RAM. Experimental results are presented in Table III. When tested on the 4-joist course over 10 trails, the perceptive model achieves a 100% success rate climbing over all joists within the given time, and uses around 3.7 seconds to climb over each joist. When tested on the 8-joist course, the perceptive model climbs over 7.8 joists per trail on average, and uses around 3.4 seconds to climb over each joist. In the only failed trial, the robot makes it very close to the end. It loses balance as it pitches up and falls over. An example video of the robot traversing over the 8-joist course is shown in this video link.

The blind model manages to climb over 1.8 joists in each trial on average and uses slightly more time than the perceptive model to climb over each joist. However, the blind model learns gaits with high impact and with its front legs raised high in the air even when there are no joists present in front of the robot. This is because it does not have access to any source of perception to interpret the appropriate action magnitude. The controller from Hiwonder is not capable of climbing over any joists over all the trials.

To further test the robustness of our proposed method, we test the robot approaching and climbing over the 4-joist course at a 45-degree angle for 10 trials. We find that it successfully climbs over the joist course in 9 of the 10 trials, as shown in this video.

We also test the proposed perceptive model with a 12-inch high joist course to determine its behavior when encountering obstacles that are physically too high to climb over. We find that the robot stops moving once it gets close to the joists. This is expected since the depth camera view is fully blocked by the high obstacle, and such a scenario is seldom encountered during training.

C. Ablation Studies

1) Depth Processing Techniques: In previous sections, we presented two depth image preprocessing techniques to enable zero-shot sim-to-real transfer. To evaluate the impact of these techniques, we conduct an ablation study where we compare the performance of the perceptive model trained with depth images preprocessed using (a) both leg masking and random invalid pixel, (b) without leg masking, (c) without random invalid pixels, and (d) without both. When deploying a model trained without leg masking on hardware, we do not apply leg masking to depth images to match the depth processing used during training.

	4 joists		8 joists	
Method	Average #Joists	Average Time / Joist (s)	Average #Joists	Average Time / Joist (s)
Perceptive (Ours)	**4**	**3.7**	**7.8**	**3.4**
Blind	1.8	3.8	1.9	4.1
HiWonder Controller	0	N/A	0	N/A

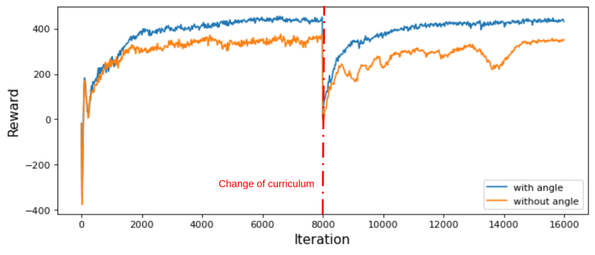
TABLE III: The average number of joists that each model successfully climbs over, and the average time for models to climb over each joist, evaluated on the proposed perceptive model, the blind model, and the controller provided by Hiwonder. The Hiwonder Controller cannot climb over any joist, therefore does not have a valid measurement of time per joist.

	4 joists		8 joists	
Method	Average #Joists	Average Time / Joist (s)	Average #Joists	Average Time / Joist (s)
Perceptive	4	3.7	7.8	3.4
w/o leg mask	0	N/A	0	N/A
w/o random invalid	1.3	6	1.5	6.1
w/o both	0	N/A	0	N/A

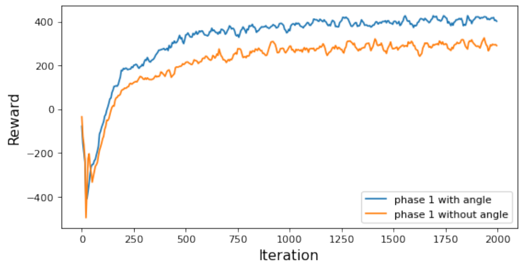
TABLE IV: The average number of joists that each model successfully climbs over, and the average time for models to climb over each joist, evaluated on the perceptive models trained with (a) both leg masking and random invalid depth pixels, (b) without leg masking, (c) without random invalid depth pixels and (d) without both. The models trained without leg masking do not successfully climb over any joists, therefore do not have a valid measurement of time per joist.

Experimental results are presented in Table IV. We observe that models trained and deployed without leg masking behave abnormally as soon as legs are captured by depth camera during climbing, and are not capable of climbing over any joists. This is likely due to the domain gap presented between the depth image of the actual robot legs and the legs in the approximate robot model. The model using only leg masking but not random invalid pixels makes some progress and climbs over joists, but gets stuck or falls over quite often. It is able to climb around 1.3 joists before it fails and takes around 6 seconds on average to successfully climb over each joist. Also, it never finishes a single trail. This proves the effectiveness of our proposed depth-based pixel masking technique to reduce the domain gap between depth images in simulation and captured by the actual L515 sensor.

2) Phase 1 without Joint Feedback: One of the main reasons we divide training into 2 phases is to learn an effective expert policy with access to privileged state feedback. To prove the necessity of having access to joint state feedback in phase 1, we compare training performances between models with and without access to joint angle feedback, as shown in Fig. 7. As seen in Fig. 7(a), the drop at the middle is due to curriculum change - the first 8000 iterations of phase 1 training are conducted on lower joist terrain shown in Fig. 4(a) and the second 8,000 iterations on higher joist terrain in Fig. 4(b). During phase 1, the model with access to joint angle feedback significantly outperforms the one without access to joint angle feedback. During phase 2 training, the student model without access to the joint angle is able to reach a similar reward as the teacher model with access to the joint angle. Student model learning from the teacher model without access to joint angle yields relatively low reward. This demonstrates the necessity to include critical privileged joint angle feedback in phase 1 training in order to optimize the model's zero-shot sim-to-real performance.



(a) Phase 1 reward with vs without joint angle



(b) Phase 2 reward

Fig. 7: (a): Phase 1 reward curves of models with and without access to joint angle feedback. (b): Phase 2 reward curves corresponding to the two models.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we presented an end-to-end approach to learning a perceptive walking model for low-cost hexapods, to climb over high and dense joist structures. We also show zero-shot sim-to-real deployment of models on hardware in a constructed test environment that closely matches the proposed use case of our method in attics. Our method significantly outperforms the baseline methods in terms of both the number of joists it climbs over within given time and the average time it takes to climb over each joist. As shown in our ablation studies, the success of our method's sim-to-real transfer is credited to the effectiveness of our proposed depth image preprocessing techniques and including critical privileged information during phase 1 training.

Even though our robot's compactness and low height allow it to traverse tight spaces, the depth camera mounted elevates the center of mass of the entire system. There are certain instances where the robot falls likely due to the instability related to an elevated center of mass. In the future, we would like to further improve the stability of our system by changing the design of our camera mount to lower the center of mass. This might entail learning a visual legged locomotion controller with a lower egocentric view.

Using a L515 LiDAR instead of a stereo camera as the perception sensor allows our robot to operate in dark environments. However, the T265 tracking RGB camera often loses track of the robot's orientation in the dark. This can be improved by switching our choice of pose sensor from a tracking camera to IMU.

Although being out of the scope of this work, we wish to explore letting the robot climb staircases or get around joists or other obstacles that are too large to climb. Eventually, we would like to teach the hexapod to perform high-level path planning in cluttered environments such as attics.

References

[1] Hiwonder. `https://hiwonder.hk/`.

[2] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision, 2022.

[3] Ayush Agrawal, Shuxiao Chen, Akshara Rai, and Koushil Sreenath. Vision-aided dynamic quadrupedal locomotion on discrete terrain using motion libraries, 2021.

[4] Teymur Azayev and Karel Zimmerman. Blind hexapod locomotion in complex terrain with gait adaptation using deep reinforcement learning and classification. *Journal of Intelligent & Robotic Systems*, 99, 09 2020.

[5] Filip Bjelonic, Joonho Lee, Philip Arm, Dhionis Sako, Davide Tateo, Jan Peters, and Marco Hutter. Learning-based design and control for quadrupedal robots with parallel-elastic actuators. *CoRR*, abs/2301.03509, 2023.

[6] Marko Bjelonic, Navinda Kottege, Timon Homberger, Paulo Borges, Philipp Beckerle, and Margarita Chli. Weaver: Hexapod robot for autonomous navigation on unstructured terrain. *Journal of Field Robotics*, 35:1063–1079, 06 2018.

[7] R. Omar Chavez-Garcia, Jérôme Guzzi, Luca Maria Gambardella, and Alessandro Giusti. Learning ground traversability from simulations. *IEEE Robotics and Automation Letters*, 3:1695–1702, 2017.

[8] Liang Ding, Peng Xu, Haibo Gao, Zhikai Wang, Ruyi Zhou, Zhaopei Gong, and Guangjun Liu. Fault tolerant free gait and footstep planning for hexapod robot based on monte-carlo tree, 2020.

[9] Jan Faigl and Petr Čížek. Adaptive locomotion control of hexapod walking robot for traversing rough terrains with position feedback only. *Robotics and Autonomous Systems*, 116, 03 2019.

[10] Peter Fankhauser, Marko Bjelonic, C. Dario Bellicoso, Takahiro Miki, and Marco Hutter. Robust rough-terrain locomotion with a quadrupedal robot. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5761–5768, 2018.

[11] Peter Fankhauser, Marko Bjelonic, C. Dario Bellicoso, Takahiro Miki, and Marco Hutter. Robust rough-terrain locomotion with a quadrupedal robot. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5761–5768, 2018.

[12] Péter Fankhauser and Marco Hutter. A universal grid map library: Implementation and use case for rough terrain navigation. 2016.

[13] Siddhant Gangapurwala, Mathieu Geisert, Romeo Orsolino, Maurice Fallon, and Ioannis Havoutis. Real-time trajectory adaptation for quadrupedal locomotion using deep reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5973–5979, 2021.

[14] Deepali Jain, Atil Iscen, and Ken Caluwaerts. From pixels to legs: Hierarchical learning of quadruped locomotion. In *Conference on Robot Learning*, 2020.

[15] Fabian Jenelten, Takahiro Miki, Aravind E Vijayan, Marko Bjelonic, and Marco Hutter. Perceptive locomotion in rough terrain – online foothold optimization. *IEEE Robotics and Automation Letters*, 5(4):5370–5376, 2020.

[16] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, and Stefan Schaal. Learning locomotion over rough terrain using terrain templates. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 167–172, 2009.

[17] Maxime Kawawa-Beaudan. Learning to walk: Legged hexapod locomotion from simulation to the real world, 2022.

[18] J. Zico Kolter, Michael P. Rodgers, and A. Ng. A control architecture for quadruped locomotion over rough terrain. *2008 IEEE International Conference on Robotics and Automation*, pages 811–818, 2008.

[19] S.H. Kwak and R.B. McGhee. Rule-based motion coordination for a hexapod walking machine. *Advanced Robotics*, 4(3):263–282, 1989.

[20] Ashwin Sanjay Lele, Yan Fang, Justin Ting, and Arijit Raychowdhury. Learning to walk: Spike based reinforcement learning for hexapod robot central pattern generation. In *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 208–212, 2020.

[21] Jiayu Li, Bo You, Liang Ding, Xiaoyang Yu, Weihua Li, Tianyong Zhang, and Haibo Gao. Dual-master/single-slave haptic teleoperation system for semiautonomous bilateral control of hexapod robot subject to deformable rough terrain. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(4):2435–2449, 2022.

[22] Tianyu Li, Nathan Lambert, Roberto Calandra, Franziska Meier, and Akshara Rai. Learning generalizable locomotion skills with hierarchical reinforcement learning, 2019.

[23] Gabriel B. Margolis, Tao Chen, Kartik Paigwar, Xiang Fu, Donghyun Kim, Sangbae Kim, and Pulkit Agrawal. Learning to jump from pixels, 2021.

[24] Carlos Mastalli, Ioannis Havoutis, Alexander Winkler, Darwin Caldwell, and Claudio Semini. On-line and on-board planning and perception for quadrupedal locomotion. 05 2015.

[25] Kharunya Paramaguru. Oscar pistorius: The blade runner makes olympic history, Jul 2012.

[26] Zhiying Qiu, Wu Wei, and Xiongding Liu. Adaptive gait generation for hexapod robots based on reinforcement learning and hierarchical framework. *Actuators*, 12(2), 2023.

[27] Martin Wermelinger, Péter Fankhauser, Remo Diethelm, Philipp Krüsi, Roland Y. Siegwart, and Marco Hutter. Navigation planning for legged robots in challenging terrain. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1184–1189, 2016.

[28] Martin Wermelinger, Péter Fankhauser, Remo Diethelm, Philipp Krüsi, Roland Siegwart, and Marco Hutter. Navigation planning for legged robots in challenging terrain. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1184–1189, 2016.

[29] Wenhao Yu, Deepali Jain, Alejandro Escontrela, Atil Iscen, Peng Xu, Erwin Coumans, Sehoon Ha, Jie Tan, and Tingnan Zhang. Visual-locomotion: Learning to walk on complex terrains with vision. In *5th Annual Conference on Robot Learning*, 2021.

[30] He Zhang, Rui Wu, Changle Li, Xizhe Zang, Xuehe Zhang, Hongzhe Jin, and Jie Zhao. A force-sensing system on legs for biomimetic hexapod robots interacting with unstructured terrain. *Sensors*, 17(7), 2017.