Versatile Locomotion Skills for Hexapod Robots

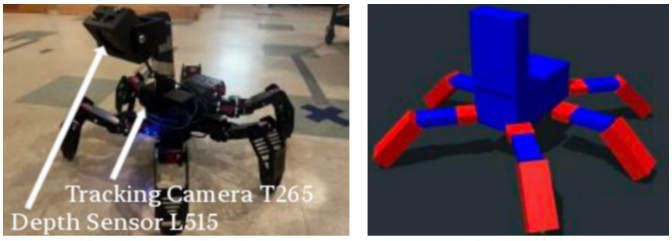
Tomson Qu[1], Dichen Li[1], Avideh Zakhor[1], Wenhao Yu[2], Tingnan Zhang[2]

***Abstract*— Hexapod robots are potentially suitable for carrying out tasks in cluttered environments since they are stable, compact, and light weight. They also have multi-joint legs and variable height bodies that make them good candidates for tasks such as stairs climbing and squeezing under objects in a typical home environment or an attic. Expanding on our previous work on joist climbing in attics, we train a legged hexapod equipped with a depth camera and visual inertial odometry (VIO) to perform three tasks: climbing stairs, avoiding obstacles, and squeezing under obstacles such as a table. Our policies are trained with simulation data only and can be deployed on low-cost hardware not requiring real-time joint state feedback. We train our model in a teacher-student model with 2 phases: In phase 1, we use reinforcement learning with access to privileged information such as height 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 pose captured by a tracking VIO camera. By manipulating available privileged information, constructing simulation terrains, and refining reward functions during phase 1 training, we are able to train the robots with skills that are robust in non-ideal physical environments. We demonstrate successful sim-to-real transfer and achieve high success rates across all three tasks in physical experiments.**

I. Introduction

Lightweight legged robots are ideal platforms for navigating inside cluttered home environments in which the robot has to get around big objects such as a refrigerators, squeeze under low objects such as couches and beds, and climb over staircases to get from one floor to the other. They can also be useful in rough environments such as attics, which are full of joists, and are uncomfortable and potentially dangerous for workers to do air sealing and vacuuming before they add insulation [1]. 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 they step on the sheetrock between two joists by mistake.

There has been a significant amount of work on quadrupeds and bipedal robot locomotion. In our prior work [2], we developed methods to enable hexapods to climb joists in harsh attic environment. In this work, we extend our prior work to stair climbing, obstacle avoidance and squeezing under objects for a hexapod robot. We focus on hexapods for two main reasons. First, hexapod robots can be more stable and lightweight than quadrupeds and humanoids of similar size. Secondly, bipedal or quadruped robots are often taller than hexapods and are therefore less suitable for traversing 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 on the order of thousands if not tens of thousands of dollars. For example, model predictive control methods such as [3] 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 [4] 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 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 [5]. In this paper, we propose an end-to-end learning-based perceptive controller for low-cost, sub-thousand-dollar hexapods to autonomously climb over staircases, avoid obstacles and squeeze under objects and demonstrate zero-shot sim-to-real transfer in real environments. In addition to attics, these skill sets are useful for robots to navigate inside homes full of furniture which the robot has to get around, and squeeze under obstacles and climb staircases to get from one floor to the other. Our robot is a $600 SpiderPi robot with no real time joint feedback manufactured by Hiwonder, shown in Figure 1, equipped with an Intel L515 depth camera and a T265 tracking camera with a customized camera mount. Similar to the approach proposed in [2], we use 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



(a) Physical Robot *(b) URDF*

Fig. 1: Physical and URDF of the robot. *(a)* The hexapod robot standing at the reset position - roughly 37 cm tall. *(b)* The URDF of the hexapod.

[1] EECS Department, University of California, Berkeley
[2] Google DeepMind

including body pose and egocentric depth images. Since optimal stair climbing, squeezing and obstacle avoidance motions are fundamentally different from walking, we train our controllers without human-defined prior gait knowledge, guiding the models to explore task-appropriate motions. Through extensive simulations and physical experiments we show our low cost robot successfully learn the following three skills and generalize across different terrains: (a) climb up and down staircases with as many as 15 steps including a landing pad ; (b) squeeze under objects that are low, regardless of whether the objects are long or short; (c) maneuver right and left around multiple obstacles without scarping or touching them and continue walking in the same direction it was headed before each obstacle. With proper design of terrains, reward functions, and choice of privileged information in simulation environment, our model is able to learn a variety of control skills.

II. Related Work

In this part, we focus on relevant prior works in two major areas: (1) locomotion control for hexapods, quadrupeds, and robots of other shapes; (2) reinforcement learning for stair climbing, obstacle avoidance, and squeezing.

A. locomotion control

Prior works have focused on specific control methods to achieve basic locomotion abilities in hexapod and quadruped robots. Researchers have used two-layer Central Pattern Generator (CPG) networks and posture control strategies based on Force Distribution and Compensation to enable robot locomotion across a variety of terrains [6]. In our previous work, we successfully enabled the hexapod to climb over joist terrains using a teacher-student model and actor/critic reinforcement learning strategy [2].

Deep Reinforcement Learning (DRL) has become a promising approach for developing autonomous and complex behaviors in real world systems. Many researchers choose to test the performance of their systems in simulation environments before applying them to real-world applications. However, the sim-to-real gap poses significant challenges in transferring simulated learning to real-world applications [7], [8]. Solutions such as system identification, domain randomization, domain adaptation, imitation learning, meta-learning, transfer learning, and knowledge distillation have shown promise in narrowing this gap, enabling more effective deployment of robotic systems in physical environments [7], [9]–[11]. Rizzardo proposed a sim-to-real technique that trains a Soft-Actor Critic agent together with a decoupled feature extractor and a latent-space dynamics model, enabling transferring without retraining or fine-tuning [8]. Tiboni introduced DROPO, a novel method for estimating domain randomization distributions for safe sim-to-real transfer [12].

Model-free reinforcement learning has emerged as a pivotal approach for achieving different locomotion tasks, such as hopping and crawling, in various environments, both terrestrial and aquatic, barriers or gaps [13]–[19]. Special network designs have been introduced to facilitate the training of the locomotion policy. [16] proposed learning low-level motion from a biological dog first and then learning high-level tasks in order to save training time. All these policies were combined into a single framework, allowing the robot to autonomously select and execute the appropriate policy. [18] used two-layer structure, a visual navigation layer to output the angular velocity commands and a visual locomotion layer to control quadruped robots to step over scattered ground to the destination. [19] proposed a hierarchical structure with a high-level vision policy and a low-level motion controller to enable a quadrupedal robot to traverse uneven environments.

B. Reinforcement learning

In order to show the efficacy of the policy employed on the robot, physical experiments are imperative. Traditional methods such as dynamic window approach (DWA) [20] are effective in avoiding obstacles robotics. Actor-critic reinforcement learning-based avoidance methods have been used to enable robots to avoid scattered obstacles [21], [22]. [23] proposed training ANYmal robots with reinforcement learning in simulation and deployed the policy to run in challenging natural environments. [24] presents an approach to teach quadruped robot to adapt and conquer unseen environments with a base policy and adaption module. Researchers introduced a general DRL framework for obstacle avoidance, and a manipulability index into the reward function in order to avoid joint singularity while executing tasks [25]. The FAM-HGNN framework, which relies on an attention mechanism within a heterogeneous graph neural network, presents a novel solution for the obstacle avoidance problem in RL. This approach surpasses the performance of both multi-layer perceptron-based and existing GNN-based RL methods [26].

With regards to squeezing, researchers enabled the reconfigurable robot RSTAR to squeeze through two adjacent obstacles, duck underneath an obstacle and climb over an obstacle using Q learning algorithm [27].

In tasks involving climbing stairs, both Deep Deterministic Policy Gradients (DDPG) and Trust Region Policy Optimization (TRPO) were evaluated, with the latter showing superior performance [28]. Researchers used sim-to-real RL to achieve climbing by only modifying an existing flat-terrain training framework to include stair-like terrain randomization, without any changes in reward function [29]. Researchers also performed experiments on enabling different articulated, tracked robots [30] and assistive robots [31] to climb on slopes or stairs using machine Learning techniques.

III. Methodology

We aim to train the robot to climb up/down stairs, avoid obstacles, and squeeze under objects. For each task, we train the corresponding policy in simulation using Isaac Gym, then directly deploy the trained policy onto the robot, which is equipped with an Intel RealSense Tracking Camera T265 for pose estimation and an L515 for depth estimation. Each task requires a different terrain construction, reward function,

Reward Term	Expression	Joist Climbing	Stair Climbing	Obstacle Avoidance	Obstacle Squeezing
Linear velocity in global x (forward)	clip(v_x, min = -0.4, max = 0.4)	1^2	1^2	1^2	1^2
Linear velocity in body y (left/right)	v_y^2	-1^1	-1^1	-1^1	-1^1
Global heading	θ^2	-3^1	0	0	0
Angular velocity: yaw	ω^2	-1^0	0	-1^0	-1^0
Ground impact	$\|f_t - f_{t-1}\|^2$	-1^{-1}	0	0	0
Collision penalty	$\mathbf{1}$\{coxa, femur, or base contacting terrain\}	-1^0	0	-3^0	-1^0
Action rate	$\|a_t - a_{t-1}\|^2$	-5^{-1}	-5^{-1}	-5^{-1}	-5^{-1}
Action magnitude	$\|a_t\|^2$	-1^{-2}	0	-1^{-2}	-1^{-2}
Torques	$\|\tau\|^2$	-1^{-3}	0	-1^{-3}	-1^{-2}
Joint acceleration	$\hat{\ddot{q}}^2 = \frac{\dot{q}_t - \dot{q}_{t-1}}{\Delta t}^2$	-1^{-5}	-1^{-5}	-1^{-5}	-1^{-5}
Joint limit penalty	clip($q_t - q_{\text{min}}$, max = 0) + clip($q_t - q_{\text{max}}$, min = 0)	-1^0	-1^0	-1^0	-1^0
End effector height	$\|z_{\text{end\_effector}}\|$	-1^{-1}	0	0	0
Global y deviation	$\|y_{current} - y_{start}\|^2$	0	-1^2	-1^0	-1^0
Distance to obstacle (front)	$f(H)$	0	0	-1^{-1}	0
Distance to obstacle (above)	$f(H)$	0	0	0	-1^{-1}

TABLE I: Reward function terms and their corresponding weights for different tasks. The definition of the top 12 reward terms are in [2].

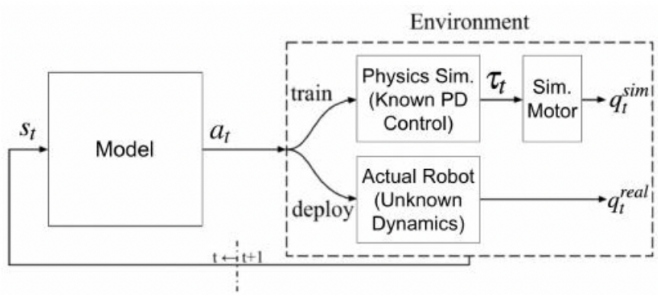
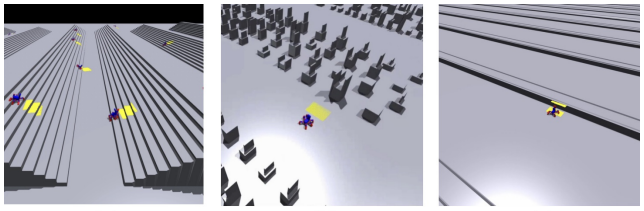
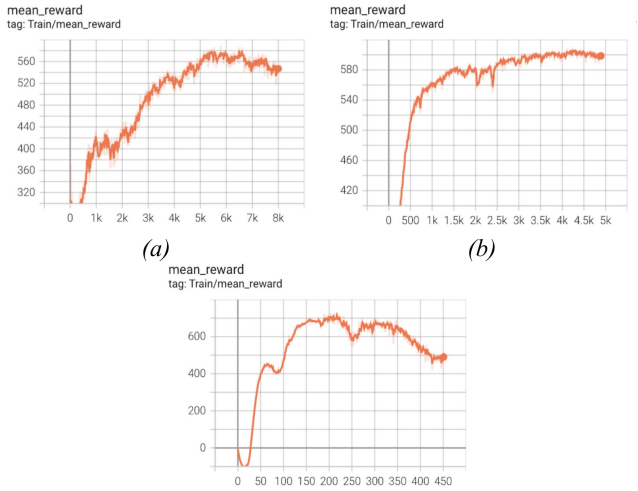


Fig. 2: High-level overview of training methodology [2]



(a) *(b)* *(c)*

Fig. 3: Simulation environment in Isaac Gym. *(a)* Stairs Climbing. *(b)* Obstacle Avoidance. *(c)* Squeezing under obstacles.



(a) *(b)* *(c)*

Fig. 4: Reward vs. episode convergence curve for tasks. *(a)* Stairs Climbing. *(b)* Obstacle avoidance. *(c)* Squeezing under obstacles.

and camera angle, as shown in Tables I and II. We adopt a 2-stage student-teacher model shown in Figure 2, similar to [2] in which the teacher policy is trained with privileged information such as joint feedback and height map shown as the yellow pad in Figure 3, and is transferred to a student policy that takes in pose estimation and depth images as visual input using supervised learning. In the teacher model training, even though different tasks are trained with different reward terms, sizes of height map, and terrain constructions, they are each distilled to a policy that takes depth map of size 320×240 as visual input. During training, we control our robot by directly predicting the joint angles of the robot and applying them to the corresponding joints. The angles range from [-120, 120] degrees. Each joint is initialized to a resting position provided by the manufacturer as shown in Figure 1.

The entire training process takes approximately 10 hours for the teacher policy and 5 hours for the student policy on an RTX TITAN GPU. The reward terms and weights for each of the four tasks, joist climbing, stairs climbing, obstacle avoidance, and squeezing, are shown in Table I. Different tasks require different number of training iterations to converge, as seen in the reward convergence curve shown in Figure 4. As seen, the squeezing task converges in 10 times fewer episodes than the other two tasks. During model deployment, we take the depth image and pose estimation from our visual odometry system as input to our policy and output the joint angles for each of the 18 joints. We then send signals to each servo to set them to desired angles.

The optimal camera angle for different tasks is shown in Table II. Other than squeezing, which requires the optical axis of the camera to be parallel to the horizon, all other tasks work well with a 30-degree downward-looking inclination. This is to be expected since the robot has to look "up" rather than "down" to see the obstacle above it, when it is squeezing under an object.

Height map size and location for different tasks are also shown in Table II. The latter refers to the distance of height map from the front of the robot. We have empirically found obstacle avoidance to need a larger height map to be further away from the robot than joist climbing and stair climbing.

A. *Stair climbing*

We trained the robot to climb up and down staircases according to the 2-step teacher-student method described earlier. We deployed curriculum training in order to help the robot climb more challenging staircases by first learning easier ones. In particular, we let the riser heights increase

Task	Camera Angle (degrees)	Height map size (m)	Height map location (m)
Joist Climbing	30	0.6×0.8	0.3
Stairs Climbing	30	0.6×0.8	0.3
Obstacle Avoidance	30	0.6×1.0	0.6
Squeezing under Obstacles	0	0.6×0.8	0

TABLE II: Optimal camera angle and height map for each task.

from 4.5 cm to 18 cm and the tread depth decrease from 30 cm to 18 cm as the level of difficulty increases in curriculum training. In addition, we randomized tread depth in a given staircase in simulation to improve generalization of our policy.

The weight of the reward term in the third row from the bottom in Table I is an order of magnitude larger for stair climbing than the other tasks. The main motivation for this is to keep the robot from unnecessarily deviating to the right or left as it climbs up a staircase. Intuitively this term minimizes the lateral deviation of the robot from the direction of the axis it was pointed before it starts climbing. We empirically found that the 30° tilt angle for the depth camera works well for both climbing up and down staircases.

B. Obstacle avoidance

Even though traditional methods such as DWA [20] work well in obstacle avoidance, since our eventual goal is to combine the different RL skills into one, we will also develop an RL based policy for obstacle avoidance. The most straightforward way to teach the robot to avoid collisions is by minimizing collisions between the robot's femur, coxa, and body during simulation, assigning large negative rewards when such collisions occur. This is shown as the "Collision Penalty" in Table I and indicates the number of collisions, whereby a collision is defined as an event with contact force larger than a certain force threshold. We have empirically found that such a method results in the robot scraping by and touching obstacles as it tries to avoid them. To circumvent that, we use a reward term shown in the one to the last row of Table I given by:

$$R(H) = -\sum_{i=1}^{M} \left(\sum_{j=1}^{N} h'_{ij} \cdot \overrightarrow{w_1}_j \right) \cdot \overrightarrow{w_2}_i$$

where $h'_{i,j}$ is the $(i,j)^{th}$ element of the binarized height map $M \times N$ matrix H' given by:

$$h'_{ij} = \begin{cases} 1 & \text{if } h_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

h_{ij} is the $(i,j)^{th}$ element of the $M \times N$ matrix height map matrix H, and $\overrightarrow{w_1}$ is an N dimensional weight vector of dimension N of a triangular shape given by:

$$\overrightarrow{w_1} = \left[1, 1 + \frac{2}{N}, 1 + \frac{4}{N}, \ldots, 2, \ldots, 1 + \frac{4}{N}, 1 + \frac{2}{N}, 1\right]$$

corresponding to the perpendicular distance of the points in the binarized height map to the axis parallel to the direction of the movement of the robot passing through its center. w_2 is an M dimensional weight vector of the shape of a ramp

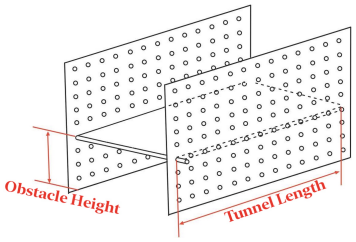


Fig. 5: Physical design of a squeezing environment.

ranging from 1 to 2, providing more weight to the obstacle points closer to the front of the robot given by:

$$\overrightarrow{w_2} = \left[1, 1 + \frac{1}{M}, 1 + \frac{2}{M}, \ldots, 2\right]$$

For the terrain in simulation, we included a variety of obstacle shapes in the map. We also note that if the obstacles are densely placed on the map at the beginning stages of the training, the policy likely converges to a local maximum reward and ends early. This is because of the sudden large penalty generated from distance to obstacle reward term. Thus, we incorporate curriculum training where the robot first learns to walk without any obstacles and then to avoid sparsely placed obstacles followed by a more dense obstacle terrain. The density ranges of each level of difficulty is given by $(2 \times Level/Total\_Levels) \times Density_{final}$, where the $Density_{final}$ is defined as the density of the obstacle spacing in the final difficulty level.

The obstacle avoidance functionality provided by the manufacturer of our robot uses ultrasound to sense obstacles and change direction accordingly, but it never goes back to its original orientation as seen in this video link. In contrast, we ideally want our obstacle avoidance functionality to have the robot revert back to its original orientation after it avoids a given obstacle. To achieve this, we use a combination of global y deviation term and a local y velocity term where the y axis is perpendicular to the direction the robot is headed and moving along before it reacts to an obstacle. Global y deviation term, shown in third to the last row of Table I, indicates the distance between the robot and the y axis. The local y velocity term shown in row 2 refers to the square velocity along the y direction. The latter term prevents the robot from going back to the original direction of motion too soon due to not having the obstacle in its field of view after the initial turn.

C. Squeezing under obstacles

Squeezing under objects is an important skill to learn in traversing environments with low objects such as beds, couches, and tables. We mainly care about two aspects in the

design of the squeezing terrain; obstacle height and 'tunnel length' as defined in Figure 5. We aim to make the robot learn such a skill and generalize to obstacles of varied shapes and 'tunnels' with varied lengths.

Since Isaac Gym does not support floating terrains and requires everything to be grounded, we modified the code to construct terrains on vertices 'in the air'. In the simulation environment, we created a floating obstacle above the basic terrain shown in Figure 3(c) to mimic the effect of a physical table, bed, or couch. The height map used in phase 1 of the training in simulation, takes on the ground level height when there is no obstacle above and the obstacle's height otherwise. We then define a single reward term, shown in the last row of Table I, to maximize the distance between the robot's body to the obstacle when there's an obstacle above and maximize the distance between the robot's body to the ground otherwise. Concretely, the reward is defined as follows:

$$R(H) = -\sum_{i=1}^{N}\left(\sum_{j=1}^{M} h''_{ij}\cdot\overrightarrow{w_{3j}}\right)\cdot\overrightarrow{w_{4i}}$$

where h''_{ij} is a function of the height map h_{ij} and the distance of the base robot to the ground, b, as follows:

$$h''_{ij} = \begin{cases} 1 & \text{if } (h_{ij}-b) > 0 \\ -2\cdot|h_{ij}-b| & \text{otherwise} \end{cases}$$

$\overrightarrow{w_3}$ is a N dimensional vector of all ones, and $\overrightarrow{w_4}$ is an M dimensional ramp vector ranging from 1 to 2 to assign a larger weight to the obstacle points closer to the front of the robot given by:

$$\overrightarrow{w_4} = \left[1, 1+\frac{1}{N}, 1+\frac{2}{N}, \ldots, 2\right]$$

We use curriculum learning so that the robot first learns to walk and then squeeze under the obstacles. The level of difficulty increases as we decrease the object distance to the ground from 37 to 35 to 33 and to 31 cm. The obstacle itself can have variable height in the vertical direction as seen in Figures 8(a) and 8(b), and variable length or width. The longer the obstacle is, the longer the robot has to squeeze to avoid hitting it while it squeezes underneath. These three parameters are randomly chosen across all levels since they do not reflect the difficulty of the task. We also randomize the origin so that in simulation the robot reaches the obstacle after walking different distances. Finally, the optical axis of the depth camera is parallel to the ground so that it can clearly see the beginning and the end of the "tunnel" created by the obstacle.

IV. Physical Experiments

We run the policies trained in simulation on a Raspberry Pi processor placed on the physical robot. We conducted experiments on all three tasks to verify and characterize the performance of our policies in real-world environments. For the staircase, we used three sets on the campus of U.C. Berkeley and for the other two tasks we constructed several terrains to evaluate the performance of the robot by measuring its success rate in completing tasks.

(a) Cory stairs; 13.5, 38.1 *(b) Soda stairs; 9.2, 45.7*

(c) Sutardja stairs; 14.0, 38.1

Fig. 6: Experiments Scene for stairs climbing. The first number is the riser height and the second number is the tread depth both in centimeters.

A. Stair climbing

We chose three staircases on the U.C. Berkeley campus, as shown in Figure 6, to evaluate the robot's performance in both climbing up and descending tasks. In Figure 5, Cory Hall and Soda Hall each had 7 steps and Sutardjai Hall had 8 steps. At the start of each experiment, the robot is placed 20 cm in the front of the stairs and is reset to its default standing position shown in Figure 1. We interrupt only if the robot falls over or is stuck on a stair.

We evaluated performance by counting the number of steps the robot could complete in each trial. We then average that number across 10 trials as shown in Table III. As seen, Soda Hall has the highest success rate because it has the lowest rise and the highest tread. Few of the climbing up failure cases resulted from the robot falling over. Others resulted from the robot getting stuck on the last stair for too long, where it believes it is walking on a plane ground as the camera image is not showing any stairs. For climbing down stairs, the failure cases come from the robot losing control and falling down. We present a video link showing the robot climbing two sets of stairs separated by a platform, with 8 and 7 steps in the first and second sets respectively. As seen in the video, the robot is able to traverse in a relatively straight line in the center without too much lateral shift to right or left. The video also shows the robot climbing down stairs with our policy compared to a baseline walking policy where it is trained to walk on a flat terrain environment. As expected, applying the "walk" policy to the staircase results in the robot crashing down the stairs.

Method	Climbing Up (Stairs completed/total stairs)			Climbing Down (Stairs completed/total stairs)		
	Cory Stairs	Soda Stairs	Sudardja	Cory Stairs	Soda Stairs	Sudardja
Perceptive (Ours)	6.0/7.0	7.0/7.0	7.6/8.0	6.6/7.0	7.0/7.0	7.5/8.0

TABLE III: Stair climbing performance.

Method	Single Box			A Box and A Bag			Single Person Standing			Single Person Moving		
	success	scrape	collision	success	scrape	collision	success	scrape	collision	success	scrape	collision
Perceptive (Ours)	7/10	2/10	1/10	9/10	1/10	0/10	7/10	2/10	1/10	8/10	1/10	1/10

TABLE IV: Object avoidance performance.

Fig. 7: Obstacle Avoidance Environment.

B. Avoiding obstacles

We constructed a variety of obstacle scenarios to evaluate the robot's obstacle avoidance performance. We test the robot's generalization ability to detect arbitrary shapes of obstacles and to avoid robustly. We design a total of four environments. The first one consists of a single box-shaped obstacle placed in front of the robot. The second consists of two obstacles: a box-shaped obstacle placed in front and a deformable plastic bag placed behind it. The third consists of a person standing/sitting in front of the robot. The fourth consists of a person walking towards the robot. We test each of the environments 10 times, with our results in Table IV. We group the outcome into three cases: success, scrape, and collision. Success means the robot can navigate around the obstacle without any collision, and scrape means the robot is able to go around the obstacle but with some leg scraping the objects/person, and collide means the robot runs into the object. For the failure cases, the robot often detects the objects, but reacts too slowly with the back leg scraping the objects after the front part successfully avoids it. In success cases, the robot exhibits avoidance behavior both to the left and right depending on its distance to each side of the obstacle. This indicates our policy did not merely memorize to avoid objects by always veering to one direction. After passing the obstacle, the robot rotates back to its original orientation and keeps moving forward. Finally, we did a comprehensive test on the robot's ability to avoid all six of our tested's obstacles in a row as shown in Figure 7 and visualized in this video link.

C. Squeezing under obstacles

The squeezing setup shown in Figure 8, aims to mimic obstacles such as a couch, a bed, or a table. At the beginning of the experiment, the robot is positioned 30 cm in front of

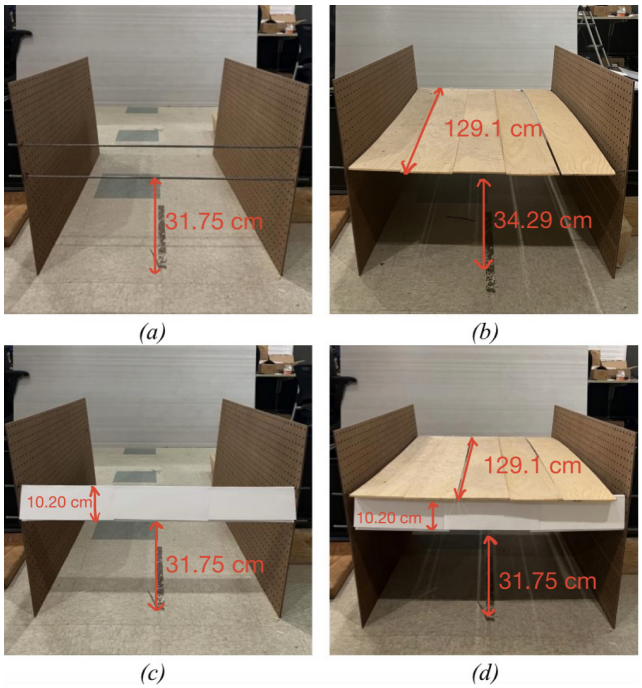


Fig. 8: Experiments Scene for Squeezing. *(a)* Metal rod as obstacle without tunnel. *(b)* Lengthy tunnel. *(c)* Paper block as obstacle without tunnel. *(d)* Paper block as obstacle with tunnel.

the obstacle. It is tasked to squeeze under the object and walk in the squeezed mode until it gets out of the "tunnel" created by the overhead obstacle at which time it is expected to raise its body to the default height and continue walking. The experiment is considered a success if the robot is able to cross the tunnel without any collisions[1]. Our experimental set up for squeezing is shown in Figure 8 with the dimensions superimposed on the pictures. In Figures 8(a), 8(c) and 8(d) there are a pair of metal rods that are 4" or 10.20 cm apart. The end of the rods are plugged into two boards with holes on a one inch grid. In Figures 8(c) and 8(d) there are paper pieces connecting the two rods, creating a different depth image from Figure 8(a) during inference and hence testing the generalizability of our policy. Finally, there is a wood board in Figures 8(b) and 8(d) creating a tunnel of length 129.1 cm to ensure the robot can be in the squeeze position for an extended time. As a reference, the robot is 28.75

[1]Collisions are visually detected and defined as events where contact between the robot and obstacles results in a trajectory change.

Fig. 9: Squeezing under two consecutive obstacles.

cm high when laying flat and 37 cm high when standing in the reset standing position shown in Figure 1. While we acknowledge that the height of the robot in the squeezing position is still too large, our main goal is to demonstrate the concept of teaching the robot to squeeze when needed. For each of the four settings shown in Figure 8, we repeat 10 to 20 trials and present the success rate in Table V.

Terrain	Success Rate	% Success
Metal rod w/o tunnel (a)	17/20	85%
Lengthy tunnel (b)	10/10	100%
Block w/o tunnel (c)	18/20	90%
Block with tunnel (d)	9/10	90%

TABLE V: Squeezing performance with corresponding construction in Figure 8.

As seen, the success rate is high[2]. While the robot can instantaneously squeeze as low as 31.75 cm to go under a thin metal rod, it cannot sustain that height for an extended distance of say 129 cm without hitting the tunnel roof. However, it can sustain the squeeze posture for a 34.29 cm tunnel of Figure 8(b). The support boards on the side have grid hole spacing of an inch; as such we can only move the rods one inch at a time in the vertical direction, resulting in a "jump" from 31.75 to 34.29 cm height. We speculate that the robot can successfully go through a tunnel of height 33 cm without scraping the top.

In one experiment as shown in Figure 9 where there are two sets of obstacles of height 31.75 cm and no tunnel in between, we notice that the robot squeezes under the first obstacle, rises up shortly during the middle, squeezes again under the second obstacle upon sensing it, and finally stands up to walk away. We present a video demonstration of this experiment in this link. As seen in the video, the "belly" of the robot comes close to the ground as it squeezes under the obstacles and raises up afterwards. The plot in Figure 10 shows the change in the height of the robot's base relative to the ground, as captured in simulation.

[2]We have empirically found that the failure rate for all three tasks increases when the battery is low and the supplied voltage is around 10 rather than the nominal 12 volts.

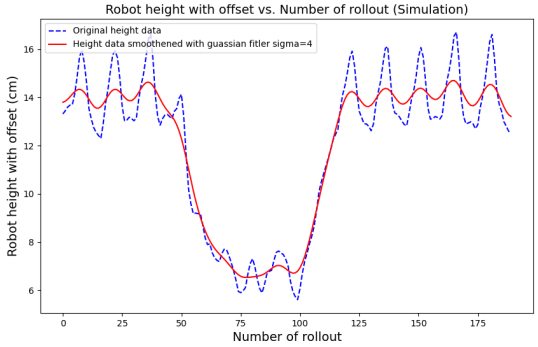


Fig. 10: Height of robot body vs. rollout step in Isaac Gym.

V. Limitations

One limitation of our approach is the requirement for different camera angles for climbing stairs and squeezing under objects. Switching policies on the go would thus present a bottleneck. Also, for squeezing under objects, although we have demonstrated the ability to generalize and to detect different shapes of objects and lengths of tunnels, the overall height of the system limits its performance to squeeze under extremely short obstacles such as a couch. The hexapod itself is quite short but the payload with the depth camera is too tall in the current system and limits the size of the objects it can squeeze under. In future work, we aim to incorporate an adjustable camera setup that extends upwards when the robot walks and downwards when it squeezes. We also need to change the camera angle between 0 and 30 degrees in the process.

VI. Conclusions

In this work, we trained a hexapod robot to climb up/down stairs, avoid obstacles, and squeeze under objects using only a depth camera and a visual inertial odometry sensor. We demonstrated the robot's ability to perform these tasks effectively through rigorous physical experiments. Future work consists of creating one universal policy combining all three tasks.

References

[1] A. Zakhor, E. Lathrop, A. Austin, J. Zang, M. Tang, A. Roychowdry, W. Naing, M. K. Beauden, and Y. Li. [Online]. Available: https://www-video.eecs.berkeley.edu/papers/avz/revised-icra-2022.pdf

[2] Z. Zang, M. Kawawa-Beaudan, W. Yu, T. Zhang, and A. Zakhor, "Perceptive hexapod legged locomotion for climbing joist environments," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 2738–2745.

[3] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," 2018.

[4] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," 2022.

[5] K. Paramaguru, "Oscar pistorius: The blade runner makes olympic history," Jul 2012. [Online]. Available: https://olympics.time.com/2012/07/05/oscar-pistorius-the-blade-runner-makes-olympic-history/

[6] W. Ouyang, H. Chi, J. Pang, W. Liang, and Q. Ren, "Adaptive Locomotion Control of a Hexapod Robot via Bio-Inspired Learning," *Frontiers in Neurorobotics*, vol. 15, p. 627157, Jan. 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnbot.2021.627157/full

[7] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, vol. 40, no. 4-5, pp. 698–721, Apr. 2021. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364920987859

[8] C. Rizzardo, F. Chen, and D. Caldwell, "Sim-to-real via latent prediction: Transferring visual non-prehensile manipulation policies," *Frontiers in Robotics and AI*, vol. 9, p. 1067502, Jan. 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2022.1067502/full

[9] J. Hua, L. Zeng, G. Li, and Z. Ju, "Learning for a Robot: Deep Reinforcement Learning, Imitation Learning, Transfer Learning," *Sensors*, vol. 21, no. 4, p. 1278, Feb. 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/4/1278

[10] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots," May 2018, arXiv:1804.10332 [cs]. [Online]. Available: http://arxiv.org/abs/1804.10332

[11] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. Canberra, ACT, Australia: IEEE, Dec. 2020, pp. 737–744. [Online]. Available: https://ieeexplore.ieee.org/document/9308468/

[12] G. Tiboni, K. Arndt, and V. Kyrki, "DROPO: Sim-to-real transfer with offline domain randomization," *Robotics and Autonomous Systems*, vol. 166, p. 104432, Aug. 2023. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0921889023000714

[13] S. Song, Kidziński, X. B. Peng, C. Ong, J. Hicks, S. Levine, C. G. Atkeson, and S. L. Delp, "Deep reinforcement learning for modeling human locomotion control in neuromechanical simulation," *Journal of NeuroEngineering and Rehabilitation*, vol. 18, no. 1, p. 126, Aug. 2021. [Online]. Available: https://jneuroengrehab.biomedcentral.com/articles/10.1186/s12984-021-00919-y

[14] J. Shi, T. Dear, and S. D. Kelly, "Deep Reinforcement Learning for Snake Robot Locomotion," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9688–9695, 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2405896320333772

[15] M. Bogdanovic, M. Khadiv, and L. Righetti, "Model-free reinforcement learning for robust locomotion using demonstrations from trajectory optimization," *Frontiers in Robotics and AI*, vol. 9, p. 854212, Aug. 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2022.854212/full

[16] Q. Wan, A. Luo, Y. Meng, C. Zhang, W. Chi, S. Zhang, Y. Liu, Q. Zhu, S. Kong, and J. Yu, "Learning and Reusing Quadruped Robot Movement Skills from Biological Dogs for Higher-Level Tasks," *Sensors*, vol. 24, no. 1, p. 28, Dec. 2023. [Online]. Available: https://www.mdpi.com/1424-8220/24/1/28

[17] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot Parkour Learning," Sep. 2023, arXiv:2309.05665 [cs]. [Online]. Available: http://arxiv.org/abs/2309.05665

[18] S. Kareer, N. Yokoyama, D. Batra, S. Ha, and J. Truong, "ViNL: Visual Navigation and Locomotion Over Obstacles," Oct. 2023, arXiv:2210.14791 [cs]. [Online]. Available: http://arxiv.org/abs/2210.14791

[19] W. Yu, D. Jain, A. Escontrela, A. Iscen, P. Xu, E. Coumans, S. Ha, J. Tan, and T. Zhang, "Visual-Locomotion: Learning to Walk on Complex Terrains with Vision."

[20] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[21] J. Choi, G. Lee, and C. Lee, "Reinforcement learning-based dynamic obstacle avoidance and integration of path planning," *Intelligent Service Robotics*, vol. 14, no. 5, pp. 663–677, Nov. 2021. [Online]. Available: https://link.springer.com/10.1007/s11370-021-00387-2

[22] F. Hart and O. Okhrin, "Enhanced method for reinforcement learning based dynamic obstacle avoidance by assessment of collision risk," *Neurocomputing*, vol. 568, p. 127097, Feb. 2024. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925231223012201

[23] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, Oct. 2020. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abc5986

[24] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," 2021.

[25] Y. Shen, Q. Jia, Z. Huang, R. Wang, J. Fei, and G. Chen, "Reinforcement Learning-Based Reactive Obstacle Avoidance Method for Redundant Manipulators," *Entropy*, vol. 24, no. 2, p. 279, Feb. 2022. [Online]. Available: https://www.mdpi.com/1099-4300/24/2/279

[26] F. Zhang, C. Xuan, and H.-K. Lam, "An obstacle avoidance-specific reinforcement learning method based on fuzzy attention mechanism and heterogeneous graph neural networks," *Engineering Applications of Artificial Intelligence*, vol. 130, p. 107764, Apr. 2024. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0952197623019486

[27] L. Yehezkel, S. Berman, and D. Zarrouk, "Overcoming Obstacles With a Reconfigurable Robot Using Reinforcement Learning," *IEEE Access*, vol. 8, pp. 217 541–217 553, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9272763/

[28] U. Garg, "Virtual Robot Climbing using Reinforcement Learning," Master of Science, San Jose State University, San Jose, CA, USA, Dec. 2018. [Online]. Available: https://scholarworks.sjsu.edu/etd_projects/658

[29] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst, "Blind Bipedal Stair Traversal via Sim-to-Real Reinforcement Learning," in *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, Jul. 2021. [Online]. Available: http://www.roboticsproceedings.org/rss17/p061.pdf

[30] A. Mitriakov, P. Papadakis, J. Kerdreux, and S. Garlatti, "Reinforcement Learning Based, Staircase Negotiation Learning: Simulation and Transfer to Reality for Articulated Tracked Robots," *IEEE Robotics & Automation Magazine*, vol. 28, no. 4, pp. 10–20, Dec. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9583665/

[31] A. Mitriakov, P. Papadakis, S. Mai Nguyen, and S. Garlatti, "Staircase Traversal via Reinforcement Learning for Active Reconfiguration of Assistive Robots," in *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. Glasgow, United Kingdom: IEEE, Jul. 2020, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/9177581/