Perceptive Pedipulation with Local Obstacle Avoidance

Jonas Stolle, Philip Arm, Mayank Mittal, Marco Hutter

***Abstract*— Pedipulation leverages the feet of legged robots for mobile manipulation, eliminating the need for dedicated robotic arms. While previous works have showcased blind and task-specific pedipulation skills, they fail to account for static and dynamic obstacles in the environment. To address this limitation, we introduce a reinforcement learning-based approach to train a whole-body obstacle-aware policy that tracks foot position commands while simultaneously avoiding obstacles. Despite training the policy in only five different static scenarios in simulation, we show that it generalizes to unknown environments with different numbers and types of obstacles. We analyze the performance of our method through a set of simulation experiments and successfully deploy the learned policy on the ANYmal quadruped, demonstrating its capability to follow foot commands while navigating around static and dynamic obstacles.**

I. Introduction

Robotic manipulation skills have rapidly improved in recent years, powered by learning-based methods and increased computational resources. Legged mobile manipulation, in particular, is becoming increasingly feasible, by combining recent advances in robust quadrupedal locomotion [1], [2] with dedicated manipulator arms [3], [4]. However, robotic arms increase the mass and complexity of the system and reduce the capacity for additional payloads.

Recent works have explored *pedipulation* – manipulation using a quadrupedal robot's foot – to overcome these limitations [5]–[7]. These works use reinforcement learning (RL) to track position commands for the pedipulating foot with a whole-body behavior. They successfully demonstrate skills such as pushing a button, opening a door, and lifting objects using the foot as an end-effector. However, these methods primarily rely on proprioceptive information and do not consider obstacles in the environment. Thus, the learned behaviors are unable to avoid unwanted collisions.

A potential approach to address this limitation is to develop a hierarchical architecture. In this setup, a learned blind pedipulation policy could serve as the low-level controller, while a human operator, classical path planner, or RL planner [8]–[10] could act as a high-level obstacle-aware planner, that provides collision-free trajectories for the foot end-effector. However, since the high-level planner can only command the foot positions and not control the entire system, collisions may still occur with the other parts of the robot during command tracking by the blind policy. Implementing large safety margins around the robot's collision bodies can mitigate this issue, but it makes applications in confined spaces infeasible.

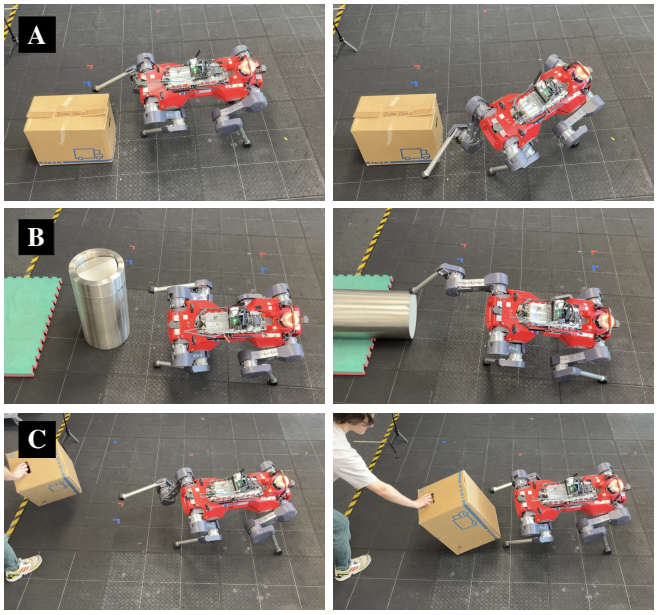

Fig. 1. Our perceptive pedipulation controller avoids obstacles with the pedipulating foot of a quadruped robot (A). It can also react to and avoid dynamic obstacles (B). By toggling the contact switch, the robot can push obstacles out of the way on command (C).

Thus, instead of a hierarchical design, this work proposes an end-to-end approach to learn a perceptive pedipulation policy. This policy retains the simple interface from prior works [5]–[7], requiring only a foot position command while extending the system's capability to simultaneously perform whole-body obstacle avoidance. We train this policy entirely in simulation using simple arrangements of cuboidal obstacles and providing appropriate collision rewards to learn the desired collision avoidance behavior. In addition, we introduce a switch to allow or disallow contacts with the pedipulating foot. With this contact switch, the policy can either avoid obstacles entirely or perform pedipulation tasks while avoiding obstacles with the rest of the body, retaining the capabilities from our previous work [5].

The key contributions of this work are as follows:

- We design an RL-based perceptive pedipulation controller that tracks foot position commands while avoiding obstacles in the robot's vicinity (Fig. 1).
- We show that our controller's obstacle-avoidance be-

All authors are with ETH Zurich, Robotics Systems Lab; Leonhardstrasse 21, 8092 Zurich, Switzerland. M. Mittal is also with NVIDIA. Contact: {parm, mittalma}@ethz.ch

This research was supported by the Swiss National Science Foundation through the National Centre of Competence in Digital Fabrication (NCCR dfab). This project has received funding through ESA contract nos. 4000137333/22/NL/AT and 4000135310/21/NL/PA/pt. This work has been conducted as part of ANYmal Research, a community to advance legged robotics.

havior generalizes to unseen geometries and effectively avoids dynamic obstacles despite being trained solely on a few scenarios with non-moving obstacles.
- We introduce a contact switch for the pedipulating foot, enabling both whole-body obstacle avoidance and pedipulation capabilities while avoiding obstacles with the remaining parts of the robot.

II. Related Work

Pedipulation

While dedicated robotic arms can be added to quadrupeds for manipulation [3], [4], [11], [12], they increase the mass and complexity of the system and reduce the capacity for additional payloads. Quadrupeds without a separate manipulation arm can still exhibit manipulation capabilities, for instance, object reorientation [13] and whole-body object repositioning [14].

For general and precise manipulation tasks, recent works have looked into pedipulation - the manipulation of objects using a robot's foot [5]–[7], [15]. These works specify a desired position for the pedipulating foot and control it with learning-based methods in a whole-body fashion. This approach provides a simple control interface for a human operator or a high-level planner to perform various tasks. However, the learned pedipulation policy is limited to obstacle-free environments, as it relies only on the proprioceptive information. In this work, we aim to account for obstacles in the robot's workspace by incorporating perceptive observations.

Similar to the prior works, we also choose RL to learn pedipulation, as we want to avoid specifying gait patterns that are typically required for online model-based methods [16], [17]. While model-based methods can solve locomotion tasks successfully, in the context of pedipulation, where the robot has to stand and locomote on three legs, providing the gait patterns can be difficult and unintuitive. Learning-based methods can help overcome this limitation, as the foot positions are unconstrained, and adaptive behaviors can emerge during training [18].

Obstacle Avoidance

While many classical approaches exist for obstacle avoidance, they are hard to apply to systems with high degrees of freedom (DoF). Grid-based methods suffer from exploding computational costs for increasing DoFs, while sampling-based methods cannot guarantee solutions [19]. Model-based approaches can find collision-free whole-body trajectories for high DoF systems [12] but are hard to apply to our problem because we want to avoid pre-specifying the gait sequence.

Recently, researchers are also investigating learning-based methods for obstacle avoidance [20]. Miki *et al.* [1] use perception to enable collision-free locomotion in confined spaces by training a high-level policy that commands the base pose and using a low-level policy to track this command. As motivated in Sec. I, this decoupling is limiting in the case of pedipulation. Honerkamp *et al.* [21] learn obstacle-avoiding manipulation skills for a wheeled-mobile base with an attached manipulator arm. In their work, the learned policy outputs the end-effector and base velocity commands, which are converted to the desired joint positions through inverse kinematics (IK). While it is possible to follow a similar approach that separates the system into a three-legged mobile base and the pedipulating leg, using IK reduces the whole-body reachability, as shown in [5].

III. Method

This work builds on our previous work [5], where we used RL to train a foot position tracking policy and showed successful sim-to-real transfer. We extend this work by incorporating perception and obstacles during training to enable automatic collision avoidance. Additionally, we incorporate a boolean switch to control the obstacle avoidance behavior of the pedipulating foot. The following section details the training process, simulation environment, and our Markov Decision Process (MDP) design for RL training.

A. Training Overview

We train our RL policy in three stages (Fig. 2):

1) Initially, the agent learns obstacle-free pedipulation with a performance-based curriculum on the command space (Fig. 2-A and Fig. 2-B), following our previous work [5].
2) Once the command space curriculum has converged, we introduce obstacles (Sec. III-B.2) and collision-avoidance rewards (Sec. III-E).
3) Finally, to overcome the sim-to-real perception gap, we add noise to the perceptive observations, emulating the noise introduced by the perception pipeline used during deployment on the ANYmal quadruped (Sec. III-F).

We separate Stages 1 and 2 to speed up training, as adding obstacle avoidance from the start makes the already hard problem of obstacle-free pedipulation more difficult and slows down training progress.

Explicitly modeling the different types of obstacles encountered in real life is infeasible due to their number and varying geometries. We hypothesize that with a small set of obstacles arranged in a well-constructed set of obstacle scenarios, the agent can learn a generalizable obstacle-avoiding behavior. To this end, we train the policy in parallel on all five scenarios by randomly selecting one scenario for each of the 4096 environments.

We add Stage 3 to improve the sim-to-real transfer, as the elevation map on the robot is subject to noise, outliers, and occlusions.

B. Simulation Environment

We design the simulation environment in NVIDIA Isaac Lab [22]. The simulation scene comprises rigid bodies for obstacles and the articulated rigid robot body.

1) Obstacles: We use cuboids as obstacles, with size and mass randomization according to Table I.

Previous works integrate obstacles into the static terrain [10], [23], which is computationally efficient as it avoids the computation of the obstacle dynamics. In the context

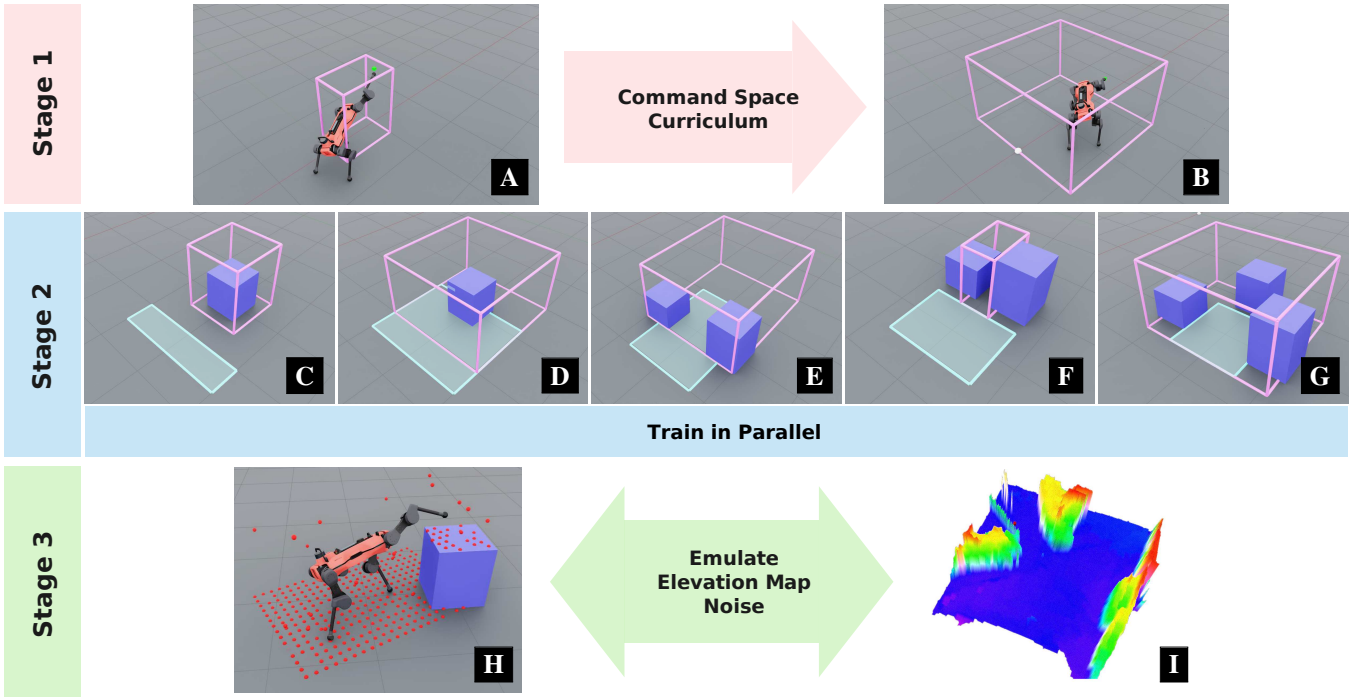

Fig. 2. High-level overview of the training process. The command spaces are denoted in pink, and the spawn spaces are denoted in cyan. In Stage 1, we learn obstacle-free pedipulation following [5]. Stage 2 introduces obstacles arranged in five scenarios, where the policy learns obstacle avoidance. Stage 3 increases the noise on the perceptive observations (H) to enable dealing with the noise and artifacts in the mapping pipeline during deployment (I).

TABLE I

SIZE AND MASS RANDOMIZATION OF THE THREE OBSTACLES USED DURING TRAINING. $\mathcal{U}$ DENOTES A UNIFORM DISTRIBUTION.

Property	Obstacle 1	Obstacles 2,3
Width	$\mathcal{U}(0.6m, 1.0m)$	$0.6m$
Length	$0.6m$	$0.6m$
Height	$\mathcal{U}(0.5m, 1.2m)$	$\mathcal{U}(0.5m, 1.2m)$
Mass	$\mathcal{U}(10kg, 30kg)$	$\mathcal{U}(10kg, 30kg)$

of pedipulation, however, where moving obstacles should be possible when desired (Sec. III-D), having only static obstacles would prevent the policy from learning interactive behaviors. Thus, we use movable obstacles, which increase training times but provide realistic collision dynamics.

2) Obstacle Scenarios: To encourage the policy to learn generalizable obstacle avoidance, we construct five obstacle scenarios (Fig. 2, Stage 2). We construct two scenarios using a single obstacle (Table I, Obstacle 1), focusing on close and far-range obstacle avoidance (Fig. 2-C and Fig. 2-D, respectively). Our third scenario (Fig. 2-E) has two obstacles (2 and 3) wide enough for the robot to spawn in between, which encourages the policy to act conservatively when constrained on both sides. Similarly, in the next environment (Fig. 2-F), we use two obstacles (2 and 3) to create a tight gap, into which the robot can only reach with its foot. Finally, using all three obstacles, we construct a scenario that features two gaps to reach through and position the obstacles such that there is enough space for the robot to turn on the spot to reach commands behind it (Fig. 2-G).

3) Robot: In the context of obstacle avoidance, it is important to choose an appropriate robot collision model that balances accuracy and computational efficiency (Fig. 4). The feet, which frequently encounter obstacles, are modeled accurately, while the base, which should rarely see collisions, is approximated with a cuboid. Conservative collision bodies for the knees and hips help the policy avoid self-collisions and challenging poses, such as reaching between its legs instead of rotating the base to track foot position commands behind the robot. Finally, we choose the right front foot (RF_foot) to perform pedipulation in this work.

4) Initial Spawning Location: We randomize the robot's spawn position near obstacles to increase the likelihood of the policy having to avoid obstacles given random commands. We find a flat patch large enough to spawn within, to ensure the robot does not spawn in collision with obstacles. To this end, we sample a random position and orientation from the spawn space and verify that the outline of a 1.1m by 0.8m rectangle at that position and orientation is sufficiently flat. If a sampled position is invalid, we resample until the position is valid. This approach allows for scenarios where the robot spawns between obstacles (e.g. Fig. 2-E).

C. Observations

We choose proprioceptive observations that are available on the hardware through the state estimator, similar to the prior work [5]. The observations include the actions $\mathbf{q}^*$, which we interpret as joint position offsets relative to the default joint positions.

In addition, we provide perceptive observations in the form of a 2.5D height scan (Fig. 3) previously used in locomotion

TABLE II

OBSERVATION TERMS SUMMARY. TERMS EXPRESSED IN THE ROBOT BASE FRAME CARRY THE SUBSCRIPT $\mathcal{B}$. THE HEIGHT SCAN IS A STACKED VECTOR OF HEIGHT VALUES h SAMPLED FROM AN l BY k GRID AND CLIPPED TO THE RANGE $[0.0, 1.0]$.

Observation Term Name	Definition	Noise
Base Linear Velocity	${}_{\mathcal{B}}\mathbf{v}$	$\mathcal{U}(-0.1, 0.1)$
Robot Base Angular Velocity	${}_{\mathcal{B}}\boldsymbol{\Omega}$	$\mathcal{U}(-0.2, 0.2)$
Base Projected Gravity	${}_{\mathcal{B}}\mathbf{g}$	$\mathcal{U}(-0.05, 0.05)$
Robot Joint Positions	$\mathbf{q}_j$	$\mathcal{U}(-0.01, 0.01)$
Joint Velocities	$\dot{\mathbf{q}}_j$	$\mathcal{U}(-1.5, 1.5)$
Foot Position Command	$\mathbf{p}^*_{foot}$	-
Previous Actions	$\mathbf{q}^*_j$	-
Height Scan	$[h_{1,1}, h_{1,2}, ..., h_{l,k}]$	See Fig. 2-I
Contact Switch	$b_{switch} \in \{0, 1\}$	-

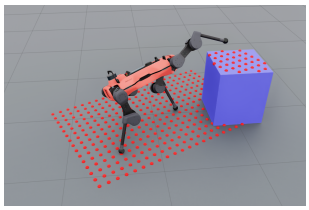

Fig. 3. We use a height scan of size $2.4\,\mathrm{m} \times 1.6\,\mathrm{m}$ with a resolution of $0.1\,\mathrm{m}$. The grid is shifted to the front by $0.2\,\mathrm{m}$ to cover the reach of the pedipulating foot.

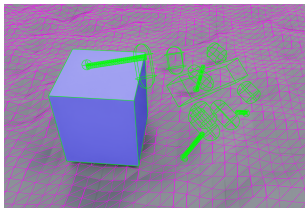

Fig. 4. We use a detailed collision model for the robot and train on rough ground to encourage the policy to take higher steps. The obstacles are simple cuboids.

tasks [1], [24], which keeps the dimension of the observation vector low. A low-dimensional observation vector reduces training times and complexity for the relatively complex task of learning pedipulation and obstacle avoidance end-to-end.

D. Commands

Commands define the goal of the task and need to be provided by the human operator or a high-level planner. Following previous work on pedipulation [5], we only command the foot position and allow the rest of the whole-body control to emerge during training. The foot position commands are sampled uniformly from the command space. In stage one, we apply a command space curriculum. In the subsequent stages, we provide separate command spaces for each obstacle scenario (Fig. 2).

While whole-body obstacle avoidance is desirable, we additionally need the option to allow the pedipulating foot to make contact with the environment to solve manipulation tasks, such as pushing doors open or moving objects. To this end, we provide an additional boolean command b_{switch}. During training, this switch determines whether penalties on the pedipulating foot's contact forces are applied (Table III). The policy then learns to avoid obstacles when the switch is off and to allow contacts when it is on. Note that the remaining bodies of the robot always have contacts penalized and should thus always avoid obstacles.

TABLE III

REWARD TERMS SUMMARY. $\mathbf{F}$ DENOTES THE SUM OF EXTERNAL FORCES ON A BODY. SUPERSCRIPT 1: ONLY APPLIED IN STAGES 2 AND 3. SUPERSCRIPT 2: ONLY APPLIED IF THE CONTACT SWITCH b_{switch} IS ZERO. SUBSCRIPT $\mathcal{B}$: VALUE EXPRESSED IN THE ROBOT BASE FRAME.

Reward Term Name	Definition	Weight
Command Tracking	$exp\Big(-\frac{\left\|\mathbf{p}^*_{foot}-\mathbf{p}_{foot}\right\|}{0.8}\Big)$	14.0
Termination	if $\|\mathbf{F}_{base}\| > 1.0$	-500.0
Base Linear Velocity z	${}_{\mathcal{B}}\dot{z}^2$	-2.0
Base Angular Velocity xy	$\left\|{}_{\mathcal{B}}\boldsymbol{\omega}^2_{xy}\right\|$	-0.05
Torques	$\boldsymbol{\tau}^T\boldsymbol{\tau}$	$-2.0\mathrm{e}{-5}$
Joint Velocities	$\dot{\mathbf{q}}^T\dot{\mathbf{q}}$	-0.04
Joint Accelerations	$\ddot{\mathbf{q}}^T\ddot{\mathbf{q}}$	$-2.5\mathrm{e}{-7}$
Action Rate	$\dot{\mathbf{q}}^{*T}\dot{\mathbf{q}}^*$	-0.02
Contact Events	$\sum_{i\in\{robot\_bodies\}}(\|\mathbf{F}_i\| > 0.1)$	-2.0
Contact Events for Obstacle Avoidance		
Pedipulating Foot[1,2]	$\left\|\mathbf{F}_{RF\_foot}\right\| > 0.001$	-80.0
Remaining Feet[1]	$\sum_{i\in\{remaining\_feet\}}(\|\mathbf{F}_i\| > 0.001)$	-20.0
Hips and Knees[1]	$\sum_{i\in\{knees\}\cup\{hips\}}\|\mathbf{F}_i\| > 0.001$	-40.0
Contact Forces for Obstacle Avoidance		
Pedipulating Foot[1,2]	$\left\|\mathbf{F}_{RF\_foot}\right\|$	-40.0
Remaining Feet[1]	$\sum_{i\in\{remaining\_feet\}}\|\mathbf{F}_i\|$	-0.2
Hips and Knees[1]	$\sum_{i\in\{hips\}\cup\{knees\}}\|\mathbf{F}_i\|$	-0.2

E. Rewards

In the first training stage (Fig. 2), we use a dense foot tracking reward, penalties for undesired contacts and excessive base movement, and regularization rewards following [5]. Specifically, we regularize the joint positions $\mathbf{q}$, their first and second derivatives, torques $\boldsymbol{\tau}$, and the rate of change of the actions $\dot{\mathbf{q}}^*$. We terminate the episode when the external force on the base exceeds a threshold, which indicates that the robot fell over. Smaller forces on the base, which can occur due to self-collisions, are only penalized. When introducing the obstacles in the second training stage, we add a contact reward structure with individually tuned weights for different parts of the robot's body (Table III, Contact Events/Forces for Obstacle Avoidance). To provide a smoother transition of the reward structure, we apply a curriculum on these rewards: They linearly increase from zero to their final weight over 5000 steps. We penalize both contact force magnitudes and contact events to prevent two exploitation strategies:

1) Without penalizing contact events, the policy could learn to push an obstacle with minimal force, which would be a limited penalty in return for an exponential command tracking reward.
2) Without penalizing the force magnitudes, on the other hand, the policy could push the obstacle out of the way with a short but high-intensity push.

We apply this idea to all collision bodies of the robot to avoid exploitation strategies like pushing obstacles away

with the hips, for example, which receive lower penalties than the pedipulating foot. Note that we choose a minimum force threshold for the contact events to avoid false positives due to numerical inaccuracies. We only penalize the non-pedipulating feet for contacts with obstacles but not the ground, as those forces should clearly be allowed. Finally, we give the contact rewards for the pedipulating foot based on the contact switch command b_{switch} (Sec. III-D).

F. Sim-to-Real Transfer

On hardware, we sample the height scan from an elevation map created by the proprietary ANYmal software based on the depth cameras on the four sides of the robot (Fig. 2-I). We tuned the elevation mapping parameters to be conservative about occlusions and not assume thin-walled objects where the top of the object cannot be seen (e.g. Fig. 10).

This map, however, is subject to noise and artifacts, which are not present in the idealized simulation. To enable the sim-to-real transfer, we increase the randomization of the height scan observations in the third training stage. With likelihood 0.05, we set points to random values sampled from $\mathcal{U}(0.0m, 1.3m)$, emulating the artifacts and edge noise in the elevation map. With likelihood 0.3, we set points to zero, emulating the effects of occlusions and blind spots. We also add drift and random noise.

In addition, over all three stages, we add the following randomizations. Following our previous work [5], we choose the ground to be a height field (see Fig. 4) uniformly sampled from $\mathcal{U}(-0.05m, 0.05m)$, which encourages the policy to take higher steps. To make the policy robust to unseen disturbances, we apply external forces to the robot's base and the pedipulating foot. We also randomize friction parameters and the robot's mass.

IV. Results and Analysis

We first evaluate obstacle-avoiding performance in simulation for ease of repeatability and visualization. Secondly, we demonstrate that these skills transfer to hardware by performing experiments on the ANYmal quadruped [25].

A. Simulation Testing

1) Reaching Around a Corner: We demonstrate the benefits of our perceptive pedipulation policy by comparing it to our previous blind policy [5] in the simple scenario of reaching around a corner (see Fig. 5). Due to the dense tracking reward (see Sec. III-E), the blind policy (A) moves the pedipulating foot to the foot position command on a direct path. As an obstacle is obstructing this path, the foot collides with the corner of the obstacle, which leads to the robot falling over. In contrast, our perceptive policy (B) is aware of the obstacle and navigates around it, taking a longer path to the foot position command, while successfully avoiding collisions.

2) Free Space Tracking: We investigate the effect of the contact switch on the free space tracking accuracy, i.e., when there are no obstacles in the robot's vicinity. We randomly sample 1024 foot position commands from a $2\,\mathrm{m}$ x $2\,\mathrm{m}$ x $1.3\,\mathrm{m}$ command space and average the tracking errors. We get an average tracking error of 0.057m with the switch set to zero. With the switch set to one, we get 0.047m. This experiment indicates that disabling the contact switch does degrade the tracking accuracy, albeit only slightly.

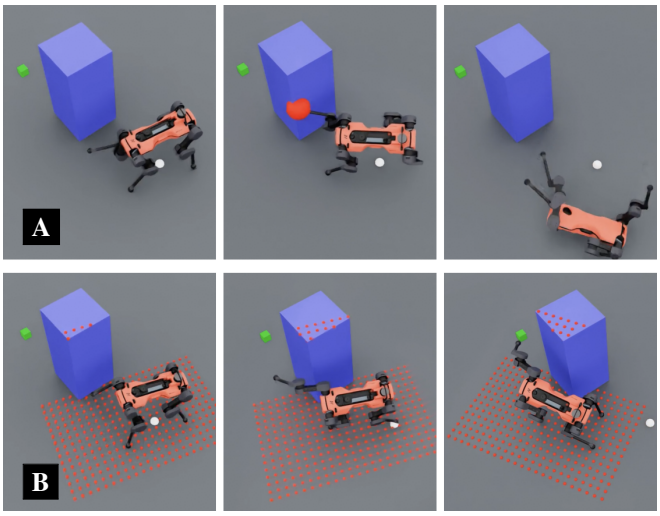

Fig. 5. The blind pedipulation policy [5] tries to reach the foot position command (green). The obstacle in the path of the foot leads to a collision (red) and eventually to a critical failure of the policy (A). Our perceptive pedipulation policy can reach around a corner while avoiding collisions (B).

3) Single Obstacle: To test local obstacle avoidance, we first consider a single obstacle (Fig. 6-A to Fig. 6-D). We move the foot position command from one side of the obstacle to the other at a constant height of $0.4\,\mathrm{m}$. As opposed to training, here, we provide a dense trajectory, as this is representative of a teleoperation scenario. When the entire command trajectory is out of collision, the robot's foot tracks it as expected. When the command moves through an obstacle, the foot takes a longer path around the obstacle, which results in a larger tracking error but avoids collisions (Fig. 6-B).

When the obstacle's height is lowered enough, the foot will take a path above the obstacle instead of around it, if this path results in a lower tracking error (Fig. 6-C). This result demonstrates that the policy does indeed consider the height of the obstacle and does not just avoid any regions in the xy-plane that have nonzero height. Finally, if the trajectory is moved too far into an obstacle, the policy will keep the foot on one side, behaving conservatively while continuing to avoid collisions (Fig. 6-D). This behavior likely occurs due to the dense reward structure. Going around the obstacle would reduce the reward until the robot can move toward the target again.

Our policy successfully avoided self-collisions and collisions with the environment in all four tests. In each test, the command moved back and forth twice. This example also shows that training on relatively wide obstacles generalizes to thinner obstacles.

4) Multiple Obstacles: The final simulation experiment tests the collision-avoidance behavior near multiple obstacles

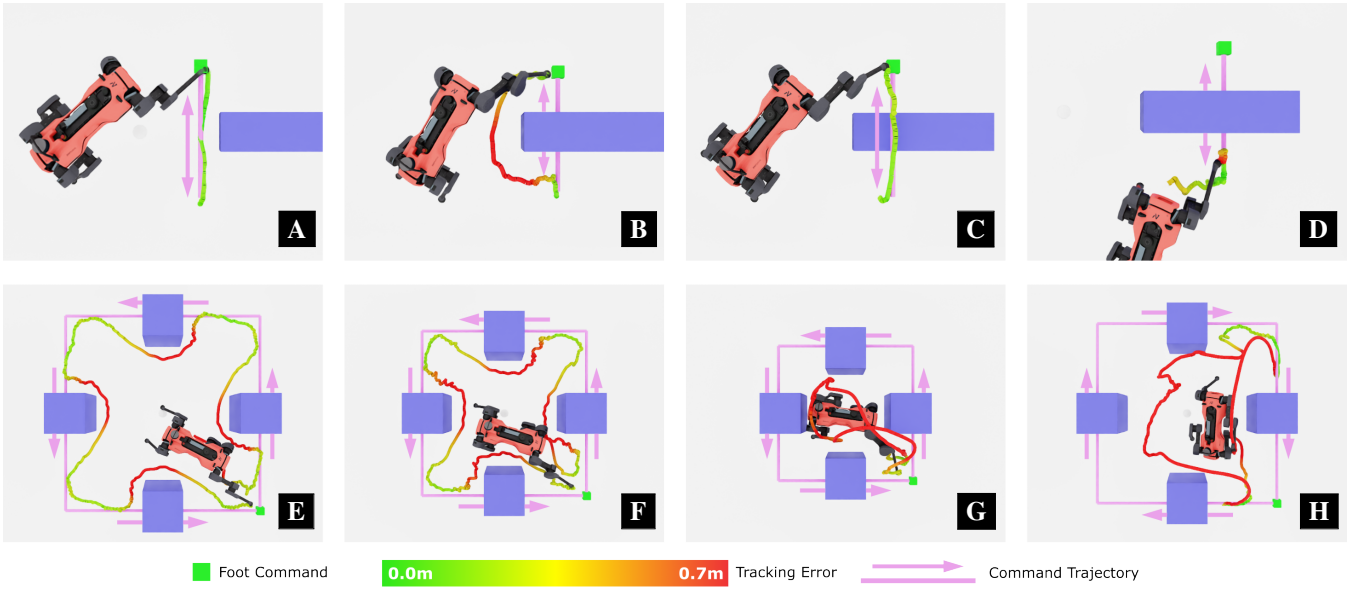

Fig. 6. We evaluate obstacle avoidance in simulation, providing dense foot position command trajectories for single and multiple obstacle scenarios. In the top row, we use a single obstacle and provide commands to move the foot from one side to another. In the bottom row, the robot needs to follow a square ring pattern around four obstacles. While effectively avoiding obstacles in many scenarios, the policy sometimes performs adversely. For instance, getting stuck due to a limited perceptive field of view (D) or insufficient free space to rotate the base freely (G).

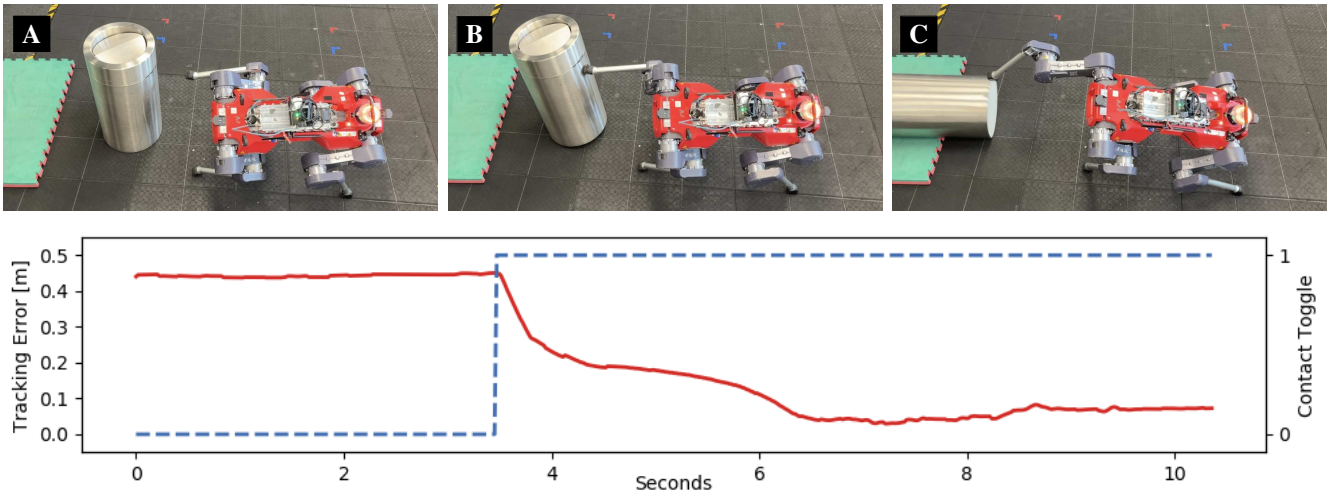

Fig. 7. With a constant foot position command inside the obstacle and the contact switch at 0 (A), the robot avoids the obstacle with a large tracking error (red). Once we set the contact switch to 1, the foot moves to the commanded position (B), tipping over the obstacle in the process (C). This switching helps enable interactions with the environment when desired. The final tracking error achieved is $0.073\,\mathrm{m}$.

(Fig. 6-E to Fig. 6-H). When the obstacles are far apart (Fig. 6-E), the policy uses its pedipulation capabilities and tripod gait to track the command trajectory, avoiding obstacles when it encounters them. Moving the obstacles closer together (Fig. 6-F), the policy first displays a similar behavior as before until we reach a limit at a gap size of $1.3\,\mathrm{m}$, where rotating the base would likely lead to collisions (Fig. 6-G). At this point, the policy stops turning the base and tracks the foot position commands behind it by reaching behind its back. In this situation, some low-force collisions occur with the back side of the robot base.

Changing the trajectory of the foot position command from counterclockwise to clockwise significantly degrades tracking performance (Fig. 6-H). Here, the robot tends to reach behind its back even when there is enough space to turn the base. From this pose, the policy struggles to rotate the base towards the target and return to the default tracking behavior. One possible reason for this behavior is that the policy only tracks points but not complete trajectories. Moving the foot to the back is the easiest way to reach the point behind the robot, as seen in the bottom right of Fig. 6-H. However, this position is unfavorable for tracking the rest of the trajectory. Observing trajectories instead of individual points could mitigate this issue.

B. Hardware Experiments

1) Contact Switch: First, we demonstrate the functionality of the contact switch with a simple example (Fig. 7). When we set the switch to obstacle avoidance and move the foot

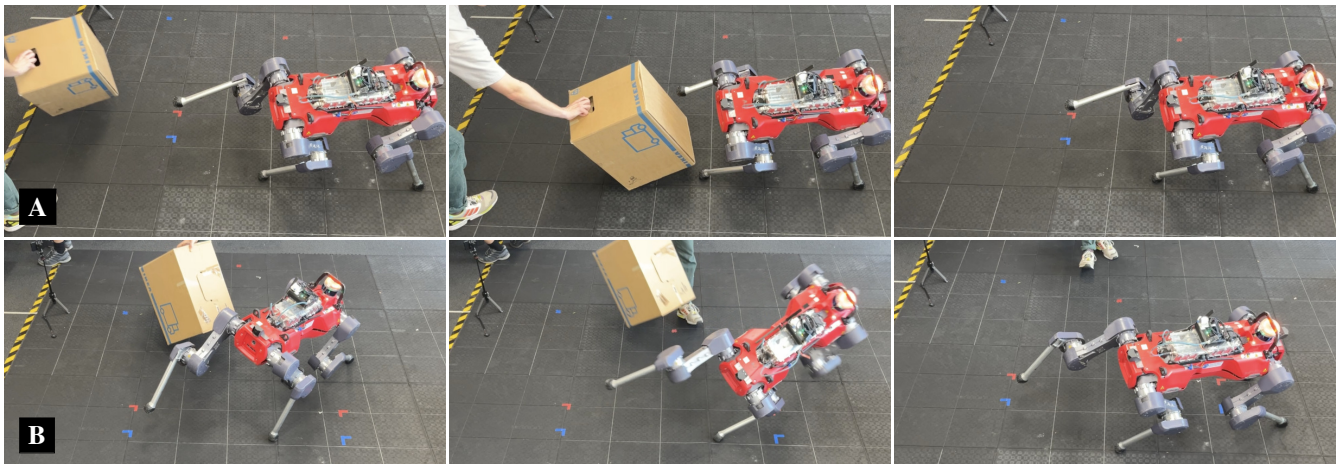

Fig. 8. The learned policy is capable of avoiding dynamic obstacles, although it is not explicitly trained for them in simulation. With the obstacle approaching from the front, the robot retracts its pedipulating foot to avoid collisions (A). When the obstacle comes from the side, the robot repositions the base and resumes tracking the foot position command (B).

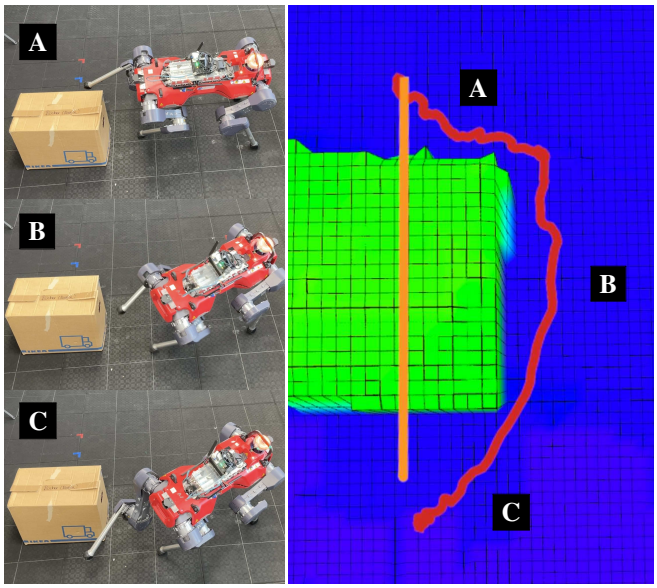

Fig. 9. The foot position command (orange) is moved back and forth along a straight line through the cardboard box via teleoperation. The actual foot trajectory (red) goes around the box.

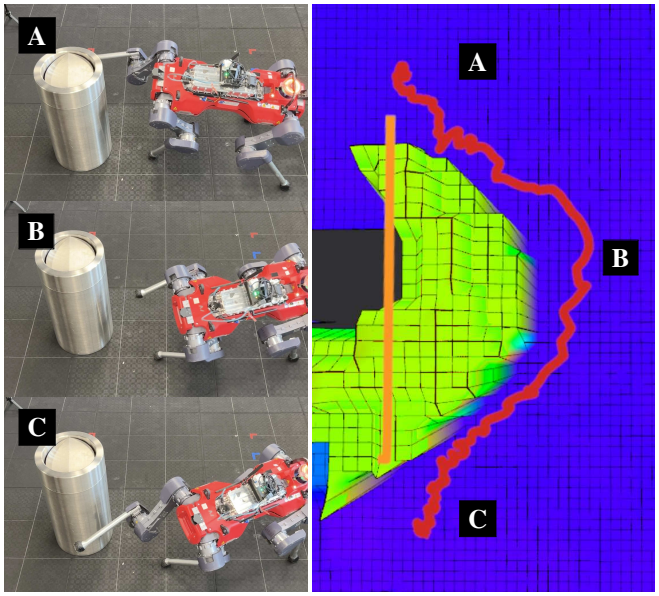

Fig. 10. The foot position command (orange) is moved back and forth along a straight line through the metal bin via teleoperation. The actual foot trajectory (red) goes around the bin, which has a geometry not seen during training. We handle holes in the elevation map conservatively by replacing them with large height values.

position command inside the obstacle, the policy successfully avoids it. Once we toggle the switch to allow contacts with the pedipulating foot, the robot pushes the obstacle out of the way to reach the commanded position.

2) Single Obstacle: The obstacle-avoiding behaviors demonstrated in simulation transfer effectively to real hardware. The policy can track foot positions through obstacles like boxes (Fig. 9) while avoiding collisions. The robot repositions its base using a tripod gait when the foot position command is out of reach or obstructed. We successfully repeated this experiment five times.

Notably, we perform the same tests on a round obstacle (Fig. 10), which is not part of the training scenarios. The policy's success in avoiding this new geometry highlights its generalization capabilities. We believe this works because the policy does not just recognize and avoid the rough shapes of obstacles. Instead, the agent learns to avoid the individual columns of space encoded in the height scan if they are occupied. We infer that the policy considers each cell in the grid individually, enabling it to precisely avoid obstacles regardless of their shape. Note that this excludes very small obstacles, which we consider perception artifacts.

3) Dynamic Obstacles: Even though the obstacles do not move actively in simulation, the policy displays a dynamic obstacle-avoiding behavior. It successfully moves the pedipulating leg away from an incoming obstacle and returns to the commanded foot position once it is removed (Fig. 8-A). The elevation map update rate limits the reaction speed of

the controller. The robot avoids obstacles coming in from the side by repositioning its base (Fig. 8-B). This behavior, however, only works for obstacles on the robot's right side.

V. Conclusion

This paper addressed the challenge of local collision avoidance for legged robots performing pedipulation. We developed a deep reinforcement learning policy incorporating a 2.5D height scan for perceptive information, enabling obstacle-avoiding whole-body control while tracking foot position commands.

Through simulation and hardware experiments, we demonstrated that our policy generalizes from a minimal set of obstacle scenarios during training to previously unseen obstacle geometries. Even though only trained on non-moving obstacles, the policy displays dynamic obstacle-avoiding behavior. The discretized nature of our perception representation allows the policy to generalize to unseen scenarios. However, it limits the set of observable obstacle geometries.

Human operators or high-level planners could easily and safely use our controller for pedipulation tasks near obstacles, as it tackles both pedipulation and obstacle avoidance end-to-end. Our contact switch additionally provides the option to switch between whole-body obstacle avoidance - including the foot - and using the foot to interact with the environment. We see this work as a first step towards robust mobile pedipulation in obstacle-rich and unknown environments.

VI. Limitations and Future Work

While our can perform obstacle-avoiding foot command tracking repeatedly, there are some corner cases that the controller does not handle well. These include obstacles directly under the base or confined spaces that are not seen during training. The choice of perception representation is limited to structures that can be represented in 2.5D and that are larger than the step size of the height scan grid.

Moving from a 2.5D elevation map to a 3D representation is desirable to overcome these limitations.

VII. Acknowledgements

We thank René Zurbrügg for assisting with the simulation implementation of the ray-cast sensor. Additionally, we thank Andrei Cramariuc for his feedback.

References

[1] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[2] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Proceedings of the 5th Conference on Robot Learning*, vol. 164 of *Proceedings of Machine Learning Research*, pp. 91–100, 2022.

[3] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: Learning a unified policy for manipulation and locomotion," in *Conference on Robot Learning (CoRL)*, 2022.

[4] M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg, "Articulated object interaction in unknown scenes with whole-body mobile manipulation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2022.

[5] P. Arm, M. Mittal, H. Kolvenbach, and M. Hutter, "Pedipulate: Enabling manipulation skills using a quadruped robot's leg," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[6] X. Cheng, A. Kumar, and D. Pathak, "Legs as manipulator: Pushing quadrupedal agility beyond locomotion," in *Conference on Robot Learning (CoRL)*, 2023.

[7] Z. He, K. Lei, Y. Ze, K. Sreenath, Z. Li, and H. Xu, "Learning visual quadrupedal loco-manipulation from demonstrations," *arXiv preprint arXiv:2403.20328*, 2024.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[9] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, pp. 4569–4574, 2011.

[10] T. Miki, J. Lee, L. Wellhausen, and M. Hutter, "Learning to walk in confined spaces using 3d representation," in *2024 International Conference on Robotics and Automation (ICRA)*, IEEE, 2024.

[11] L. Chen, Z. Jiang, L. Cheng, A. C. Knoll, and M. Zhou, "Deep reinforcement learning based trajectory planning under uncertain constraints," *Frontiers in Neurorobotics*, vol. 16, 2022.

[12] J. Chiu, J.-P. Sleiman, M. Mittal, F. Farshidian, and M. Hutter, "A collision-free mpc for whole-body dynamic locomotion and manipulation," *2022 International Conference on Robotics and Automation (ICRA)*, pp. 4686–4693, 2022.

[13] F. Shi, T. Homberger, J. Lee, T. Miki, M. Zhao, F. Farshidian, K. Okada, M. Inaba, and M. Hutter, "Circus anymal: A quadruped learning dexterous manipulation with its limbs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, p. 2316–2323, 2021.

[14] S. G. Jeon, M. Jung, S. Choi, B. Kim, and J. Hwangbo, "Learning whole-body manipulation for quadrupedal robot," *IEEE Robotics and Automation Letters*, vol. 9, pp. 699–706, 2023.

[15] C. Lin, X. Liu, Y. Yang, Y. Niu, W. Yu, T. Zhang, J. Tan, B. Boots, and D. Zhao, "Locoman: Advancing versatile quadrupedal dexterity with lightweight loco-manipulators," *arXiv preprint arXiv:2403.18197*, 2024.

[16] M. Bjelonic, R. Grandia, O. Harley, C. Galliard, S. Zimmermann, and M. Hutter, "Whole-body mpc and online gait sequence generation for wheeled-legged robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8388–8395, 2021.

[17] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini, "Mpc-based controller with terrain insight for dynamic legged locomotion," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2436–2442, 2020.

[18] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2497–2503, 2022.

[19] L. Petrović, "Motion planning in high-dimensional spaces," *arXiv preprint arXiv:1806.07457*, 2018.

[20] K. Almazrouei, I. Kamel, and T. Rabie, "Dynamic obstacle avoidance and path planning through reinforcement learning," *Applied Sciences*, vol. 13, no. 14, 2023.

[21] D. Honerkamp, T. Welschehold, and A. Valada, "N^2m^2: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments," *IEEE Transactions on Robotics*, 2023.

[22] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, p. 3740–3747, June 2023.

[23] J. Frey, D. Hoeller, S. Khattak, and M. Hutter, "Locomotion policy guided traversability learning using volumetric representations of complex environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2022.

[24] T. Miki, L. Wellhausen, R. Grandia, F. Jenelten, T. Homberger, and M. Hutter, "Elevation mapping for locomotion and navigation using gpu," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2273–2280, IEEE, 2022.

[25] ANYbotics, "Anymal-d datasheet." `https://www.anybotics.com/anymal-technical-specifications.pdf`. Accessed: 2024-07-01.