2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Toward Self-Righting and Recovery in the Wild: Challenges and Benchmarks

Rosario Scalise, Ege Caglar, Byron Boots, and Chad C. Kessens

***Abstract*— Self-recovery is a critical capability for robust, agile robots operating in the real world. Given truly challenging terrain, it is nearly inevitable that, at some point, the robot will fail and subsequently need to recover if it is to continue its task. One critical subset of recovery is standing back up after falling down (aka "self-righting"), an essential early milestone for babies learning to walk, and an existential capability for animals. While some robots can be designed with multiple orientations for mobility, most seeking to affect the world would significantly benefit from planners/policies that facilitate self-righting whenever possible.**

In this work, we present a series of challenges that outline why recovery in the wild is difficult. We then present a set of benchmark policies trained in simulation using deep reinforcement learning (RL) and the Student-Teacher approach. Finally, we evaluate the performance of these policies on a set of benchmark contexts in simulation, and provide baseline validation on a physical robot.

I. Introduction

Life is full of mishaps. We all inevitably stumble, slip, or fall. However, humans and animals alike can recover from their falls. In this work, we discuss the challenges hidden within the seemingly intuitive recovery behaviors we exhibit in everyday life, and how we can impart autonomous agents with these abilities.

Although the research community has pursued topics related to automated recovery both within robotics [1]–[3] and other fields like controls engineering [4], [5], the problem has yet to be solved in a general, scalable manner – especially when deployed in the wild. For example, existing solutions found on commercially available off-the-shelf quadrupeds such as the Boston Dynamics Spot [6], Ghost Robotics Vision 60 [7], or Unitree A1 [8] rely on open-loop movements that stop once the IMU designates the torso is upright [9] with no context-sensitivity. One reason for this is that general purpose mobile robots like quadrupeds and bipeds are only now starting to move out of laboratories (where recovery and reset for experiments is common-place) and into the field (where in-situ recovery is crucial, but highly challenging). As robots continue to gain more capabilities and therefore become more task-agnostic, we will need our self-recovery behaviors to track the diversity of contexts these new tasks bring our robots into.

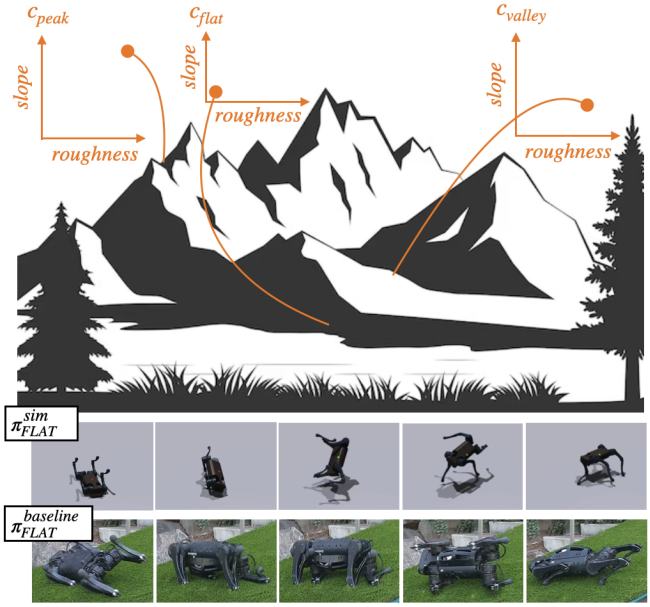

Fig. 1: *Top:* Generalized self-righting policies in the wild face a number of challenges induced by each unique context $c_{peak}, c_{flat}, c_{valley}$. *Bottom:* π_{FLAT} for the Unitree A1 robot executing a recovery maneuver in simulation and on real-world hardware.

Moreover, enabling new classes of tasks relies on having the confidence that we can "switch" between tasks at will. However, running controllers, policies, and planners that have not been tested at their interfaces (e.g. transitioning from surveying a flower to traversing a steep meadow) can result in failures. Sometimes, failing into a recovery controller might be sufficient, but other times these failures can be catastrophic, resulting in complete inversion of the robot. In these cases, the robot must self-right carefully enough that it can resume the next task without losing the progress it has already made. Preparing methods to "fill the gaps" when our controllers fail can give us the confidence to perform tasks that pose greater risk or need longer horizons to succeed, such as moving at faster speeds on more challenging terrains.

Finally, within the robot learning community, there are a number of reasons why recovery is key to unlocking the next era of lifelong learners. In the quest to deploy robots that collect data and continuously improve through reinforcement learning (RL), we would like our agents to recover even when exploring the world with suboptimal policies on real

This work was supported by the University of Washington (UW) and Oak Ridge Associated Universities (ORAU) under DEVCOM Army Research Laboratory (ARL) Cooperative Agreements W911NF-21-2-0291 and W911NF-16-2-0008 / W911NF-21-2-0038, respectively. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DEVCOM Army Research Laboratory or the U.S. Government.

The authors thank Matt Kaplan of DEVCOM ARL for thoughtful comments offered during early formulations of this work.

R Scalise (rosario@cs.uw.edu), E Caglar, & B Boots are with the Computer Science Department of the University of Washington, Seattle, WA, 98195 USA.

R Scalise is also an ORAU Fellow with ARL

C Kessens is with ARL, Aberdeen Proving Ground, MD 21005 USA.

hardware. Imparting "reset-access" to hardware [10] is one reason having robust self-righting policies is critical. In the same vein, when performing imitation learning (IL), a common problem is covariate shift, leading to cascading errors in control. One way to tackle this is through corrective feedback [11], and having robust recovery policies can provide a method to give this feedback automatically.

In this paper, we contribute a framework for investigating the problem of self-recovery in the wild that introduces:

- *Factors* that make this problem challenging,
- *Contexts* (a subset of which are shown) to test within,
- *Metrics* to quantify performance, and
- A baseline *benchmark policy set* to compare against.

We then present results of the baseline policy set in simulation and demonstrate a proof-of-concept real-world experiment on hardware.

II. Related Work

Motion planning for high degree-of-freedom (DoF) robots can be difficult due to high problem dimensionality. Most previous work has, therefore, constrained the scope of the problem to low DoFs on conventional, wheeled morphologies [2]. Even with clever sampling techniques, existing methods continue to struggle with high DoF systems in fully expressive 3D task spaces [12].

Existing methods for generalized self-righting [1], [12] assume righting maneuvers are performed quasi-statically. That is, each movement is performed with low inertia and minimal disruption to the environment. When introducing fast, dynamic movements, there are additional factors that affect whether the robot will remain in an upright state after arriving there, or if it will pass through the upright state on its way to a new, non-upright local minimum. Although most existing work on self-righting is quasi-static, there is some preliminary work on using dynamic movements to self-right a robot [2], though it is often highly specific to morphology [13]–[15].

Additionally, there have been attempts to minimize fall energy, such as work by [16], [17]. [18] and [19] attempt to unify many disparate fall policies under one global policy. There has also been work on recovering *while* falling [20], [21] as well as while slipping [22], [23].

Simulation frameworks can be leveraged to reduce the sampling complexity in obtaining complex dynamic behaviors. This includes using the simulator as part of a dynamics model, as well as providing a means of generating reward signals that lead to desired behaviors. For example, existing work [1], [2], [12], [24] conducted experimentation and algorithmic validation within custom physics simulators. There has been limited exploration of this problem utilizing tools from deep reinforcement learning [3], which only considers recovery on flat ground in a lab environment setting (which is also achievable using open loop control).

III. Challenges

Self-recovery in the wild is challenging due to the diversity of contexts a robot can find itself in, each of which possess their own risks and consequences of failure. We can also think of these challenges as defining tasks for which we desire robust policies. Consider the following conditions that influence self-righting:

Incline Angle: the angle of ground incline relative to the horizontal plane. Incline is particularly important when considering the yaw angle of the robot ψ relative to the ground plane, as different yaw angles can exhibit significant differences in stability.

Terrain Roughness: the degree of deviation in the amplitude of grid-sampled points on the ground from the nominal ground plane. It is particularly useful to think about this roughness factor at the scale of one robot body length. When terrain roughness begins to surpass the geometric size of the robot, it can become un-traversable.

Confined Spaces: the degree to which the geometric surroundings confine the robot's movements. We can formalize this as the ratio between robot body length relative to the distance between the two closest geometric walls it can fit between. Interaction with walls may significantly help or hinder self-righting attempts, and we should consider the inability to roll in a particular direction (e.g. for risk of falling off a landing or cliff).

Friction: the (static and/or dynamic) friction coefficient between the robot and the ground. This is determined by the interaction between the composition of the ground and the robot's materials, which may be different for different body parts contacting the ground. Complex ground may have its own differences in material (and thus friction) properties.

Terrain Deformability: the amount that the terrain changes its geometry in response to the forces generated by the robot. Two broad categories are reversible (including elastic) or irreversable (including plastic) deformation. Lack of repeatability and predictability from previous motions can frustrate real-time learning and worsen the severity of the situation. These factors are particularly challenging to model in simulation [25], though the community is starting to see more support in mainstream physics simulators.

Terrain Force Response: related to deformation, this is the exchange of energy that unfolds as the terrain changes in response to the forces generated by the robot. Terrain can include a mix of spring and damping forces affecting the efficiency of energy transmission, which can be important for controlling dynamic rolling conditions.

To begin addressing some of these factors, we propose a simulation challenge context: a grid composed of the cartesian product of $c_{\texttt{INCLINE}} \times c_{\texttt{ROUGH}}$ (shown in Fig. 2).

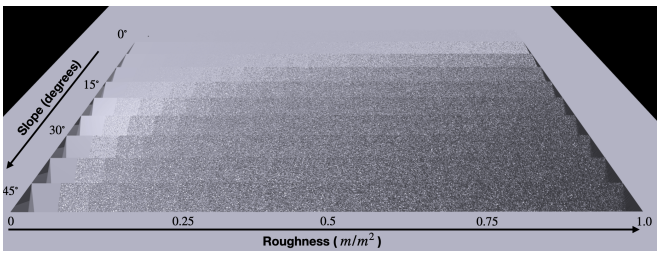

Fig. 2: Terrain grid over which we compute metrics.

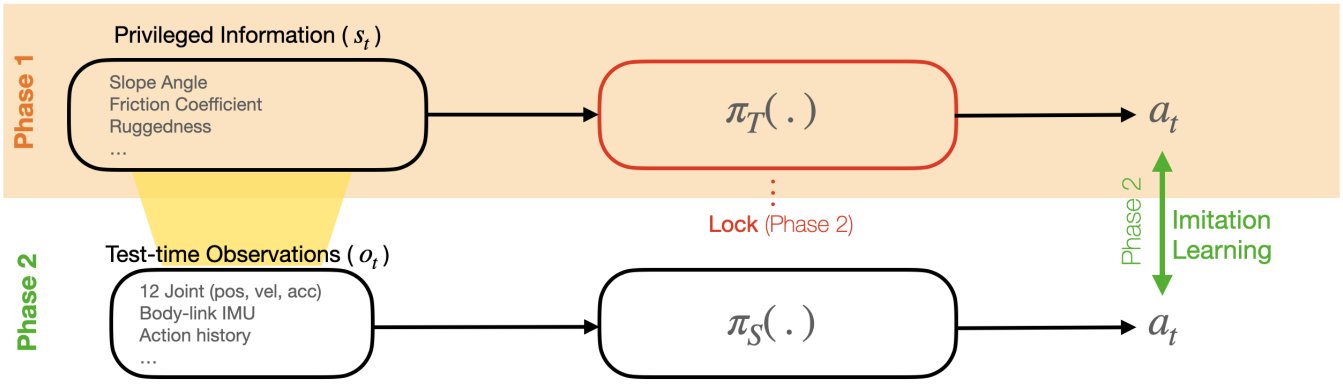

Fig. 3: The Student-Teacher paradigm used to train our policy described in Section V.

This is generated via a height map that is programmatically set to increase the slope percentage by 10% per row (on an interval of 0 to 100%). We then perturb the ground planes up and down by noise parameters that scale proportionally according to slope such that the bounds of the vertical roughness variance never exceed the body length of the robot.

Each tile in our map is represented as a set of points $\{(x_i, y_i, z_i)\}_1^N$ over the area A of a plane, roughly representing a plane in 3D. Since the quadruped is relatively small in scale, we assume that the natural terrain doesn't have significant higher-degree curvature in the area on which the robot will self-right. Hence, we believe adding noise to a first-order approximation (plane) of the terrain creates a suitable approximation for the contexts we would like to consider. After generation, to find the best-fit ground plane for each tile, we run a PCA transformation with 2 dimensions on these points to get $\{(\hat{x}_i, \hat{y}_i, \hat{z}_i)\}_1^N$, forming a smooth plane. Given the points are equally spaced apart, the area occupied by each point can be given by A/N. Then, the roughness metric for each tile is then calculated as the average distance of each point to its projection per unit area it occupies:

$$R = \frac{1}{N}\left(\sum_{i=1}^{N} ||(x_i, y_i, z_i) - (\hat{x}_i, \hat{y}_i, \hat{z}_i)||\right) \cdot \left(\frac{A}{N}\right)^{-1} \quad (1)$$

$$= \frac{1}{A}\sum_{i=1}^{N} ||(x_i, y_i, z_i) - (\hat{x}_i, \hat{y}_i, \hat{z}_i)|| \quad (2)$$

Note here that areas more densely populated by points result in higher roughness than areas where disturbances from the plane are more sparse.

IV. Metrics

While our ultimate goal is self-recovery, *how* recovery is achieved is also important. For this reason, we define a collection of metrics to quantify the characteristics of a given self-righting strategy:

Success Percentage: the proportion of possible initial states from which the robot can reach and remain in the set of (upright) goal states. As the initial states can be an unconstrained set (e.g., adding an arbitrary amount of initial linear and/or angular acceleration to the body link), we use a proxy for catastrophic failure. In our proxy, the robot is fully inverted and actively falling along the gravity vector at $1g$ from a height of 1 robot body length above the ground plane. Our goal set includes states that are within pitch and roll tolerances $\theta_{off} = \phi_{off} = 30^\circ$ from the gravity vector, as well as joint configurations q which result in all four feet being in contact with the ground and a body height $z_{body} \geq z_{standing}$, where $z_{standing}$ is defined as 80% of the fully-extended leg length.

It is important to acknowledge that the notion of success can vary depending on the initial distribution of states the downstream behavior controller expects. For example, a less robust locomotion controller might require starting in a much stricter subset of the state-space before it can be deployed nominally. To accommodate this, the set of acceptable goal states can be expanded or contracted depending on the requirements of the downstream controllers/policies. Note that in many cases, dynamic stability is sufficient to continue agile tasks, and achieving static stability is not necessary for a large range of successive behaviors.

Time-to-Recover (TTR): the time required from initiating the policy to successfully reaching the goal set region such that it is possible to remain in that set.

Distance-to-Recover (DTR): the translational distance (in the X-Y plane) that the robot traverses between initiating the policy and the final time step when the goal region is reached such that it is possible to remain in that set. We note that DTR could be productive if it gains ground towards a downstream task or helps traverse otherwise challenging terrain.

V. A Benchmark Self-Righting Policy

We now introduce benchmark policies trained in simulation using Deep Reinforcement Learning and the Student-Teacher approach outlined in [26], with care to model the Unitree A1 dynamics, terrain dynamics, and reward structure for the self-righting problem. For GPU-accelerated physics simulation, we utilize Nvidia Isaac Gym [27]. To train the teacher policy, we use a standard implementation of Proximal Policy Optimization (PPO) [28] configured with the hyper-parameters included in the Appendix. The teacher policy has

access to **privileged** information (i.e. beyond what sensors could provide) about the current context, including geometric features like the slope angle, and physical parameters such as friction. The reward structure used to encourage self-righting includes the terms found in Table I and can be expressed as:

List of Reward Terms and Coefficients		
Name	Formula	Weight
Orientation	$r_{\text{orient}} = \cos\theta_z$	2.5
Vertical Linear Velocity	$r_{\text{vel}} = \phi(\mathbf{v}_z)$	0.1
Torques	$r_{\text{tor}} = -\|\|\tau\|\|^2$	0.001
Base Height	$r_{\text{height}} = -\|\|z^* - z\|\|^2$	40
Collisions	$r_{\text{col}} = -n_c$	1
Action Rate	$r_{\text{act}} = -\|\|\mathbf{q}_j^* - \mathbf{q}_{j-1}^*\|\|^2$	0.01
Horizontal Position Drift	$r_{\text{drift}} = -\|\|\mathbf{x}_{\text{start}} - \mathbf{x}_{\text{end}}\|\|^2$	0.01
Joint Position Limits	$r_{\text{limit}} = -\max(0, \psi_{\text{high}} - \psi) + \max(0, \psi - \psi_{\text{low}})$	10
Feet Contact	$r_{\text{feet}} = \begin{cases} 1 & \text{all feet on ground} \\ 0 & \text{otherwise} \end{cases}$	8

TABLE I: List of reward terms and coefficients, where $\phi(x) = \exp\left(-\frac{\|\|x\|\|^2}{0.25}\right)$.

$$r_{total} = \sum_i r_i \cdot w_i \cdot dt \tag{3}$$

where dt is the time step duration and w_i is the weight. The reward weights were found by hyperparameter tuning on the lowest 10 roughness levels and lowest 3 slope levels to maximize the success rate.

Following convergence, we obtain a privileged policy which plays the role of the expert "Teacher", π_T. In the second phase of training, we deploy a frozen π_T and perform supervised learning to obtain a policy that maps current **unprivileged** observations (i.e. solely sensor based) to actions. After this process concludes, the "Student" policy π_S is deployable to the same observation space that the physical hardware will encounter in the real world.

Yielding a π_T that works well given different challenges (including varied dynamics or geometric contexts) is currently an open problem, and it is up to the training context designer to craft them in such a way that the learners will be exposed to a range of state-action trajectories. Although there are many possible design axes, we chose to uniformly randomize across dynamics parameters including friction, motor gains (stiffness and damping) and robot mass. In this work, we highlight the effect that changing geometric contexts can have and provide a benchmark policy $\pi_{\texttt{SELF-RIGHTING}}$ ($\pi_{\texttt{SR}}$) trained in three independent contexts, each controlled for geometric characteristics as described below:

Flat-ground Policy $\pi_{\texttt{FLAT}}$ ($\pi_{\texttt{SR}}$ trained only on $c_{\texttt{FLAT}}$): The first context specialization of our benchmark policy $\pi_{\texttt{SR}}$ emulates a laboratory environment much like [3]. We train using only flat ground geometry and do not expose the learner to any inclined or rough terrain.

Rough Policy $\pi_{\texttt{ROUGH}}$ ($\pi_{\texttt{SR}}$ trained only on $c_{\texttt{ROUGH}}$): In the second benchmark specialization, we train on a level ground plane, but introduce uneven roughness of varying amplitudes at a noise frequency scaled to match the length of half of the robot's body length.

Incline Policy $\pi_{\texttt{INCLINE}}$ ($\pi_{\texttt{SR}}$ trained only on $c_{\texttt{INCLINE}}$): For the third context specialization, we train only on a moderate incline of 20°.

Each training context is indicated by a light green box in the results figures (such as $c_{\texttt{FLAT}}$ shown in Fig. 4). Depending on the context, the policy changes behavior according to the following simplistic multiplexing scheme:

$$\pi_{\texttt{SR}} = \begin{cases} \pi_{\texttt{FLAT}} & \text{if } c_{\texttt{FLAT}} \\ \pi_{\texttt{ROUGH}} & \text{if } c_{\texttt{ROUGH}} \\ \pi_{\texttt{INCLINE}} & \text{if } c_{\texttt{INCLINE}} \end{cases}$$

In each of these cases, when the policy is deployed for testing either in simulation or on real hardware, it is deployed with test-time observations (i.e. the student version of the policy).

During training, we initialize the starting position randomly as $(x,y) \sim \mathscr{U}(0.1, 0.9)$ (where x and y are i.i.d. sampled) and a suitable base orientation. For the base orientation, we use the `scipy.spatial.transform` package to sample uniformly from $SO(3)$. If the rotation keeps the robot upright (meaning the normal vector to the robot's base has a positive $+z$ component) we apply a 180 degree rotation in the $+y$ axis to invert the robot. We also randomize the joint position configuration $q \in \mathbb{R}^{12}$, drawing from a uniform distribution $\sim \mathscr{U}(0, 2\pi)$ and using rejection sampling such that no initial joint positions result in self-collision or violate joint limits. To ensure maximum diversity in initial state, upon starting each episode, we drop the robot from 1 robot body length above the ground, allow physics to rollout a duration of $T_{wait} = 2$ s so the robot can settle, and then begin training.

VI. Experiments

A. Simulation Evaluation

We test each variation of the benchmark policy $\pi_{\texttt{SR}}$ on the Cartesian product of terrain contexts: $c_{\texttt{INCLINE}} \times c_{\texttt{ROUGH}}$ shown in Fig. 2. We report performance across 3 metrics: Success Percentage, TTR and DTR, shown for $\pi_{\texttt{FLAT}}$ in Figs. 4, 5, and 6, respectively. We see that $\pi_{\texttt{FLAT}}$ does quite well until terrain level 5 and roughness level 10. After this, performance starts to degrade on both axes. Fig. 11 shows that $\pi_{\texttt{ROUGH}}$, which introduces rough terrain context into the training, does much better on all rough terrain as expected, but similarly struggles to adapt at higher terrain slopes. Finally, Fig. 9 shows that $\pi_{\texttt{INCLINE}}$ exhibits good success on slopes gentler than the training set, but greater care in reward design might be required to generalize to even steeper slopes.

B. Physical Robot Evaluation

To test the performance of the benchmark behavior, we use the widely available Unitree A1 deployed on our experimental testbed, shown in Fig. 10. The results are given in Table II. We define a successful recovery as the robot starting from rest, running the self-righting controller, and landing on its four feet with no residual velocity. Additionally, we require

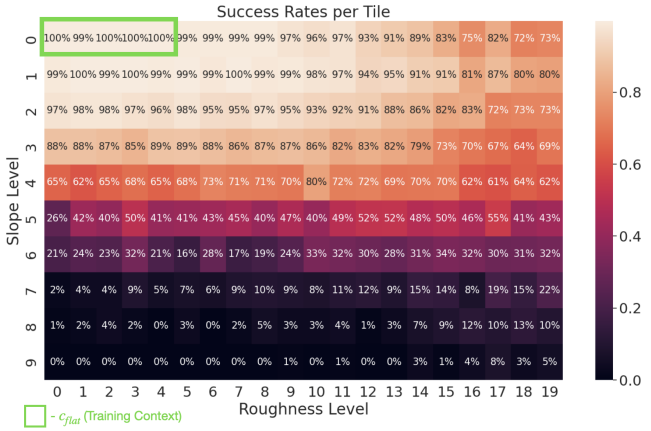

Fig. 4: π_{FLAT}: Success Rate (%) across slope angle and roughness grid

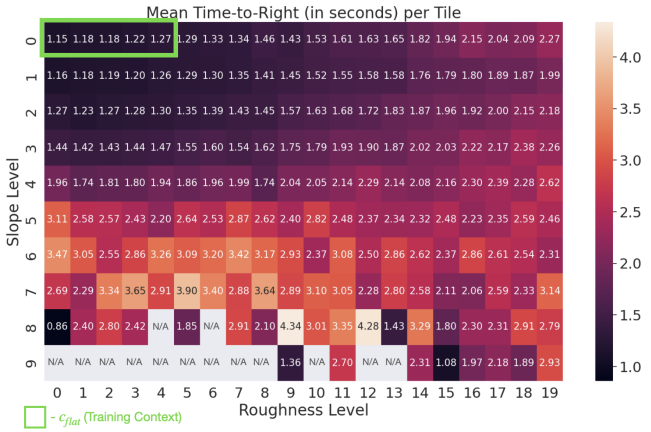

Fig. 5: π_{FLAT}: Mean Time-to-Right (s) across slope angle and roughness grid. N/A indicates no successful solutions.

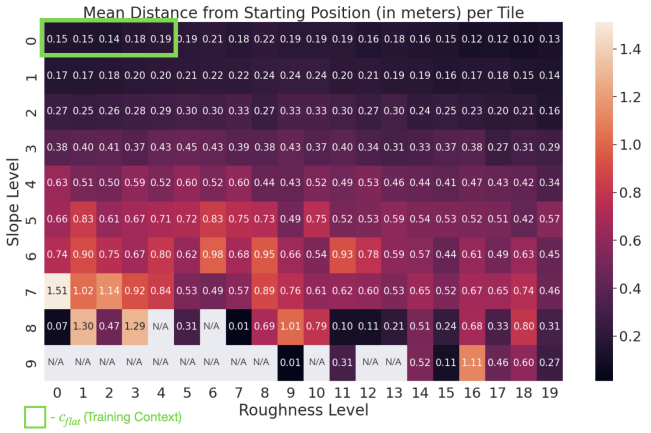

Fig. 6: π_{FLAT}: Mean Distance-to-Right (m) across slope angle and roughness grid. N/A indicates no successful solutions.

that the recovery occurs within less than one full roll-rotation and that the robot does not land on the flat ground below the incline.

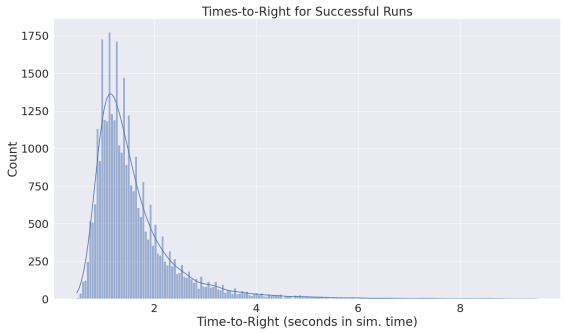

Fig. 7: π_{FLAT}: Mean Time-to-Right (s) distribution over successful runs. Each bin has a width of 0.1 seconds.

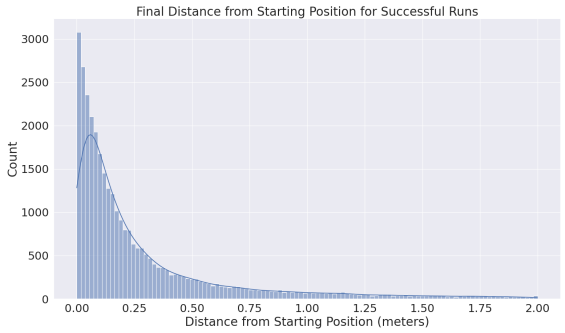

Fig. 8: π_{FLAT}: Mean Distance-to-Right (m) distribution over successful runs. Each bin has a width of 0.02 meters.

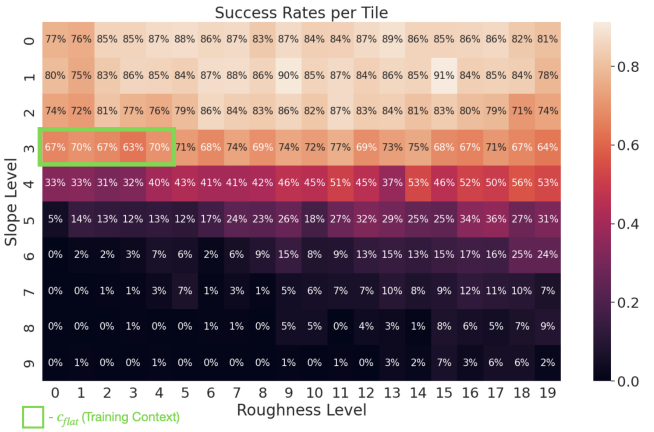

Fig. 9: π_{INCLINE}: success rate (%) across slope angle and roughness grid.

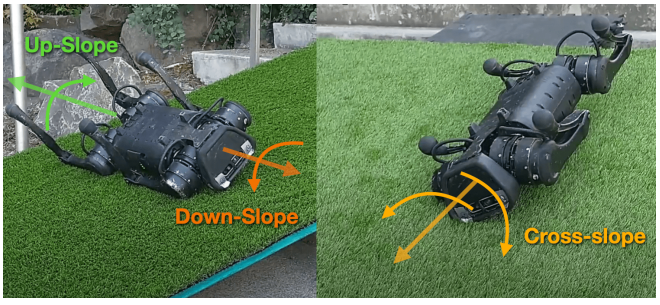

Fig. 10: The Unitree A1 on which we ran experiments and the 3 principle travel directions: Down-slope, Up-slope, and Cross-slope.

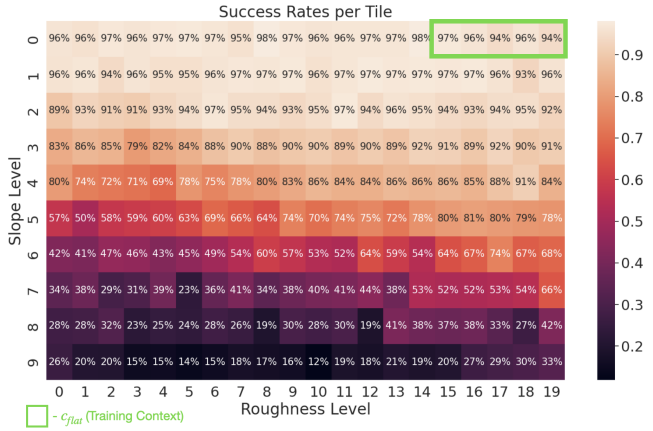

Fig. 11: π_{ROUGH}: success rate (%) across slope angle and roughness grid.

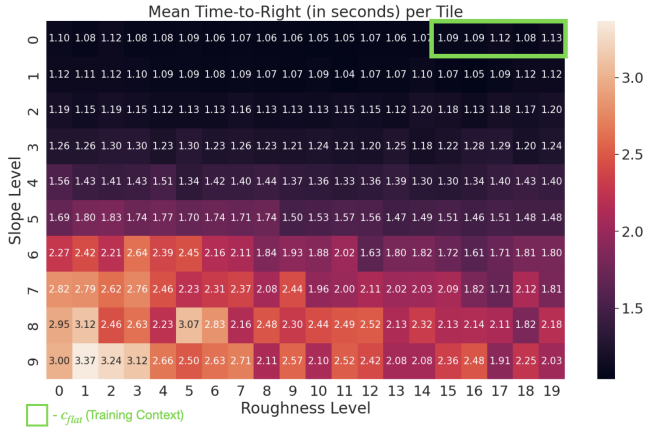

Fig. 12: π_{ROUGH}: Mean Time-to-Right (s) across slope angle and roughness grid.

Incline	Down-slope	Up-slope	Cross-slope
12°	✓	✓	✓
15°		✓	✓
18°			✓

TABLE II: Successful recovery results running the default self-righting controller baseline (✓) on the Unitree A1 from different initial yaw orientations shown in Fig. 10.

The baseline performance indicates that as the slope increases, the yaw angle that the robot has relative to the fall line of the slope greatly influences the likelihood of successful recovery. Note that the up-slope rolling behavior imparts a boost in performance on steeper slopes, preventing the robot from rolling past the upright state. Beyond this, cross-slope (e.g. rolling across the face of the slope) is the most likely to succeed. These results are shown in Table II. The environmental context of this testbed is estimated to be of roughness level 1 and lying between slope levels 2 and 3 (around 11° and 18° respectively).

VII. Conclusion

In this work, we introduce a broad formalization of the self-recovery problem, which includes: inherent *challenges*, *contexts* which are derived from those challenges, *metrics* that quantify how well recovery policies work, and finally an evaluation of the baseline behavior on real hardware.

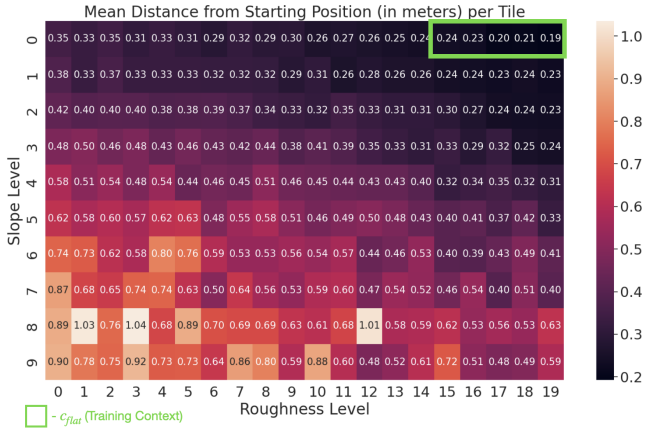

Fig. 13: π_{ROUGH}: Mean Distance-to-Right (m) across slope angle and roughness grid.

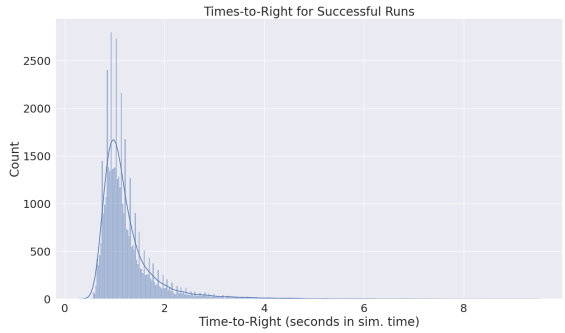

Fig. 14: π_{ROUGH}: Mean Time-to-Right (s) distribution over successful runs. Each bin has a width of 0.1 seconds.

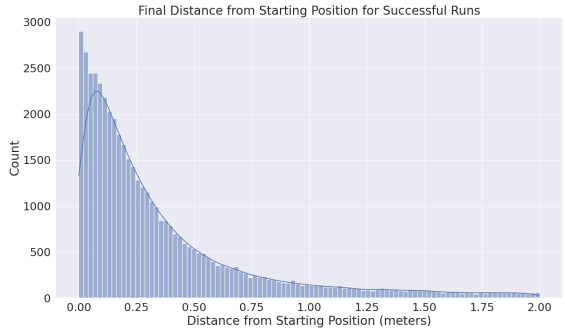

Fig. 15: π_{ROUGH}: Mean Distance-to-Right (m) distribution over successful runs. Each bin has a width of 0.02 meters.

The policy trained on high roughness and low slope contexts surprisingly outperform others even in their training context, suggesting a trade-off between ease of learning in a context (low slope) and generalizability of that context (high roughness). In future work, we hope to provide a more in-depth analysis of this phenomenon, which could be used in better model selection for contexts or better curriculum design for learning across all contexts.

The ultimate goal is to develop a $\pi_{\text{UNIVERSAL}}$ that will be able to recover in most contexts and can serve as a fail-safe policy for future researchers deploying policies on real hardware in the wild.

REFERENCES

[1] C. C. Kessens, D. C. Smith, and P. R. Osteen, "A framework for autonomous self-righting of a generic robot on sloped planar surfaces," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 4724–4729.

[2] C. C. Kessens and J. Dotterweich, "Ground-based self-righting using inertial appendage methods," in *Unmanned Systems Technology XIX*, vol. 10195. International Society for Optics and Photonics, 2017, p. 1019505.

[3] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," *arXiv preprint arXiv:1901.07517*, 2019.

[4] G. Zhang, H. Liu, Z. Qin, G. V. Moiseev, and J. Huo, "Research on self-recovery control algorithm of quadruped robot fall based on reinforcement learning," in *Actuators*, vol. 12, no. 3. MDPI, 2023, p. 110.

[5] A. G. Dobrikopf, L. Schulze, D. W. Bertol, and V. Barasuol, "Mpc-based reference governor control for self-righting of quadruped robots: Preliminary results," in *2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE)*. IEEE, 2022, pp. 85–90.

[6] Boston Dynamics. Spot - Boston Dynamics. [Online]. Available: https://bostondynamics.com/products/spot/

[7] Ghost Robotics. Vision 60 - Ghost Robotics. [Online]. Available: https://www.ghostrobotics.io/vision-60

[8] Unitree Robotics. Unitree A1 - Unitree Robotics. [Online]. Available: https://unitreerobotics.net/robotdog/unitree-a1/

[9] A. D. Perkins, M. Malchano, and S. Talebinejad, "Systems and methods for robotic self-right," Apr. 12 2016, US Patent 9,308,648.

[10] L. Smith, I. Kostrikov, and S. Levine, "A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning," *arXiv preprint arXiv:2208.07860*, 2022.

[11] J. Spencer, S. Choudhury, A. Venkatraman, B. Ziebart, and J. A. Bagnell, "Feedback in imitation learning: The three regimes of covariate shift," *arXiv preprint arXiv:2102.02872*, 2021.

[12] G. E. Mullins, C. Kessens, and S. K. Gupta, "An adaptive sampling approach for evaluating robot self-righting capabilities," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4233–4240, 2018.

[13] U. Saranli, A. Rizzi, and D. E. Koditschek, "Model-based dynamic self-righting maneuvers for a hexapedal robot," *International Journal of Robotics Research*, vol. 23, no. 9, pp. 903–918, September 2004.

[14] K. Byl, "Metastable legged-robot locomotion," Ph.D. dissertation, Massachusetts Institute of Technology, 2008.

[15] Q. Xuan and C. Li, "Coordinated appendages accumulate more energy to self-right on the ground," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6137–6144, 2020.

[16] Y. Ma, F. Farshidian, and M. Hutter, "Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators," in *40th IEEE Conference on Robotics and Automation (ICRA 2023)*, 2023.

[17] K. Chatzilygeroudis, V. Vassiliades, and J.-B. Mouret, "Reset-free trial-and-error learning for robot damage recovery," *Robotics and Autonomous Systems*, vol. 100, pp. 236–250, 2018.

[18] V. C. Kumar, S. Ha, and C. K. Liu, "Learning a unified control policy for safe falling," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3940–3947.

[19] S. Wang and K. Hauser, "Real-time stabilization of a falling humanoid robot using hand contact: An optimal control approach," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 454–460.

[20] J. Zico Kolter and A. Y. Ng, "The stanford littledog: A learning and rapid replanning approach to quadruped locomotion," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 150–174, 2011.

[21] T. Libby, A. M. Johnson, E. Chang-Siu, R. J. Full, and D. E. Koditschek, "Comparative design, scaling, and control of appendages for inertial reorientation," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1380–1398, 2016.

[22] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter, "Dynamic locomotion on slippery ground," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4170–4176, 2019.

[23] M. Khorram and S. A. A. Moosavian, "Push recovery of a quadruped robot on challenging terrains," *Robotica*, vol. 35, no. 8, pp. 1670–1689, 2017.

[24] C. C. Kessens, C. T. Lennon, and J. Collins, "A metric for self-rightability and understanding its relationship to simple morphologies," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 3699–3704.

[25] S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, and J. Hwangbo, "Learning quadrupedal locomotion on deformable terrain," *Science Robotics*, vol. 8, no. 74, p. eade2256, 2023.

[26] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on Robot Learning (CoRL)*, 2019.

[27] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox, "Gpu-accelerated robotic simulation for distributed reinforcement learning," in *Conference on Robot Learning*. PMLR, 2018, pp. 270–282.

[28] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.