

Learning Semantics-Aware Locomotion Skills from Human Demonstration

Yuxiang Yang¹, Xiangyun Meng¹, Wenhao Yu², Tingnan Zhang², Jie Tan², Byron Boots¹

¹University of Washington, ²Robotics at Google

{yuxiangy, xiangyun, bboots}@cs.washington.edu

{magicmelon, tingnan, jietan}@google.com

Abstract: The semantics of the environment, such as the terrain type and property, reveals important information for legged robots to adjust their behaviors. In this work, we present a framework that learns semantics-aware locomotion skills from perception for quadrupedal robots, such that the robot can traverse through complex offroad terrains with appropriate speeds and gaits using perception information. Due to the lack of high-fidelity outdoor simulation, our framework needs to be trained directly in the real world, which brings unique challenges in data efficiency and safety. To ensure sample efficiency, we pre-train the perception model with an off-road driving dataset. To avoid the risks of real-world policy exploration, we leverage human demonstration to train a speed policy that selects a desired forward speed from camera image. For maximum traversability, we pair the speed policy with a gait selector, which selects a robust locomotion gait for each forward speed. Using only 40 minutes of human demonstration data, our framework learns to adjust the speed and gait of the robot based on perceived terrain semantics, and enables the robot to walk over 6km without failure at close-to-optimal speed.¹

Keywords: Legged Locomotion, Semantic Perception, Imitation Learning, Hierarchical Control

1 Introduction

In order to operate in complex offroad environments, it is crucial for quadrupedal robots to adapt their motion based on the perception of the terrain ahead. As encountering new terrains, the robot needs to identify changes in key terrain properties, such as friction and deformation, and respond with the appropriate locomotion strategy to maintain a reasonable forward speed without incurring failures. In many cases, information about such terrain properties is more easily inferred from a terrain’s *semantics* class (e.g. grass, mud, asphalt, etc.), instead of its *geometric* shape (e.g. slope angle, smoothness). However, recent works in perceptive locomotion [1, 2, 3, 4, 5, 6, 7] mostly focus on the *geometric* aspect of the terrain, and do not make use of such *semantic* information.

In this work, we present a framework for quadrupedal robots to adapt locomotion behaviors based on perceived terrain semantics. The central challenge in learning such a semantic-aware locomotion controller is the high cost in data collection. On one hand, while simulation has become an effective data source for many robot learning tasks, it is still difficult to use simulation in offroad locomotion tasks, because modeling the complex contact dynamics accurately and rendering the environment photorealistically in such offroad terrains is not yet possible in simulation. On the other hand, data collection in the real world is time-consuming and requires significant human labor. Moreover, the robot needs to remain safe during the data collection process, as any robot failure can cause significant damage to the hardware and surrounding environment. Therefore, it is difficult to use standard reinforcement learning methods for this task.

Our framework addresses all concerns above, and learns semantics-aware locomotion skills directly in the real world. To reduce the amount of data required, we pre-train a semantic segmentation

¹Please see our video for more results: <https://youtu.be/V1XeZh1TSvA>

network on an off-road driving dataset, and extract a semantic embedding from the model for further fine-tuning. To avoid policy exploration in real-world environments, we collect speed and gait choices from human expert demonstrations, and train the policy using imitation learning [8]. To further improve robot’s traversability, we pair the speed policy with a gait selector, which selects a robust gait for each forward speed. With the pre-trained image embedding, the imitation learning setup, and the gait selector, our framework learns semantics-aware locomotion skills directly in the real world safely and efficiently.

We deploy our framework on an A1 quadrupedal robot from Unitree [9]. Using only 40 minutes of human demonstration data, our framework learns semantics-aware locomotion skills that can be directly deployed for offroad operation. The learned skill policy inspects the environment and selects an appropriate locomotion skill that is fast and failure-free, from slow and cautious stepping on heavy pebbles to fast and active running on flat asphalt. The learned framework generalizes well, and operates without failure on a number of trails not seen during training (over 6km in total). Moreover, our framework outperforms manufacturer’s default controller in terms of speed and safety. We further conduct ablation studies to justify the important design choices.

The technical contributions of this paper include:

1. We develop a hierarchical framework that adapts locomotion skills from terrain semantics.
2. We propose a safe and data-efficient method to train our framework directly in the real world, which only requires 40 minutes of human demonstration data.
3. We evaluate the trained framework on multiple trails spanning 6km with different terrain types, where the robot reached high speed while remaining failure-free.

2 Related Works

Terrain Understanding for Legged Robots Creating a perceptive locomotion controller is a critical step to enable legged robots to walk in outdoor, unstructured environments. Most importantly it allows robots to detect and react to terrain changes proactively before contact. Many prior works have focused on understanding terrain *geometry* from perceptive sensors [10, 11, 1, 5, 4]. However, such information can be insufficient for a robot to handle many challenging outdoor terrains as it does not reveal important terrain properties such as deformability or contact friction [6, 7, 12, 13]. To ameliorate this, recent works proposed to update the geometric understanding of a terrain with proprioceptive information [6, 7, 12]. However, these methods sacrifice proactivity, as the update cannot happen until *after* the robot has stepped on the terrain. Unlike these works, our framework infers terrain properties from its *semantics* [14, 15, 16, 17], so that the robot can detect terrain changes *before* contact, and select its locomotion strategies proactively.

Motion Controller Design for Perceptive Locomotion Another important question in perceptive locomotion is the design of a motion controller that can effectively make use of perceptive information. A common strategy is to create a low-level motion controller that plans precise foothold placements based on the perceived terrain [1, 2, 3, 4, 5]. While these methods have shown good results in highly uneven terrains, the high computational cost required for terrain understanding and rapid planning makes it infeasible for complex offroad environments. In this work, we devise a novel way to interface between perception and low-level motion controllers for legged robots, where the high-level perception model outputs the desired locomotion skills, including forward speed and robot gait, to a low-level motor controller. With our framework, the robot can select a safe and fast walking strategy for different terrains, which is crucial for offroad traversal.

Terrain Traversability Estimation Researchers have proposed a number of approaches to infer terrain traversability from geometric and semantic terrain properties, including manually designed [18], learned from self exploration [11], or learned from human demonstration [19]. While learning-based approaches provide more flexibility, they usually require large amounts of data, which is difficult to collect in the real world. As a result, most approaches rely on simulation as a source for training data [20]. For example, Frey et al. [11] learns a mapping from heightmap to traversability through self-exploration in simulation. Cai et al. [19] also learns a semantics-conditioned traversability map for off-road vehicles using a high-fidelity simulator. Unlike these approaches,

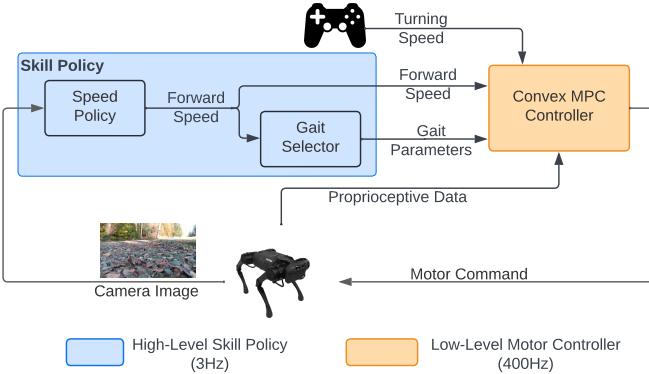


Figure 1: Our framework consists of a high-level skill policy and a low-level motor controller. The skill policy selects locomotion skills (gait and speed) based on camera images. The low-level controller computes motor commands for robot control.

our framework can be trained directly in the real world with 40 minutes of human demonstration data by leveraging a pre-trained semantic embedding.

3 Overview

Our hierarchical framework (Fig. 1) consists of a high-level skill policy and a low-level motor controller. At the high level, the skill policy receives the RGB image stream from onboard camera and determines the corresponding locomotion skill. Each skill consists of a desired forward speed and a corresponding locomotion gait, which are computed by the speed policy and gait selector, respectively. We train the speed policy using imitation learning from human demonstrations, and manually design the gait selector to find the appropriate gait for each forward speed. At the low level, a convex MPC controller [21] receives the skill command from the skill policy, and computes motor commands for robot control. In addition, the convex MPC controller can optionally receive a steering command from an external teleoperator, which specifies the desired turning rate.

4 Learning Speed Policies

In unstructured outdoor terrains, it is important for a robot to adjust its speed in response to terrain changes, so that it can traverse through different terrains efficiently and without failure. To achieve that, we design a speed policy, which computes the desired forward speed of the robot based on camera inputs. We train the speed policy using a two-staged procedure: First, we pre-train a semantic embedding from an offline dataset. After that, we collect human demonstrations and train the speed policy using imitation learning.

Pre-trained Semantic Embedding To reduce the amount of real-world data required to train the speed policy, we pre-train a semantic segmentation model and extract a semantic embedding for subsequent training. We implement the semantic segmentation model using FCHarDNet[22] and train it on an off-road driving dataset, RUGD [23]. We choose the FCHarDNet[22] model because of its compactness and real-time performance, and choose RUGD because of its similarity to images collected by the robot camera.

Although RUGD [23] provides pixel-wise semantic labels for every image, the raw semantic categories from the dataset can be insufficient in assessing terrain traversability. For example, RUGD labels both shallow and deep ground vegetation as "grass", on which the robot have notably different traversabilities. Instead, we extract a *semantic embedding* as the output before the final layer in FCHarDNet[22], which assigns a 48-dimensional embedding vector to each pixel in the input image (Fig. 2). We then compute a speed map by feeding each pixel's embedding through a fully-connected layer. Finally, the desired forward speed is obtained from averaging in a fixed region in

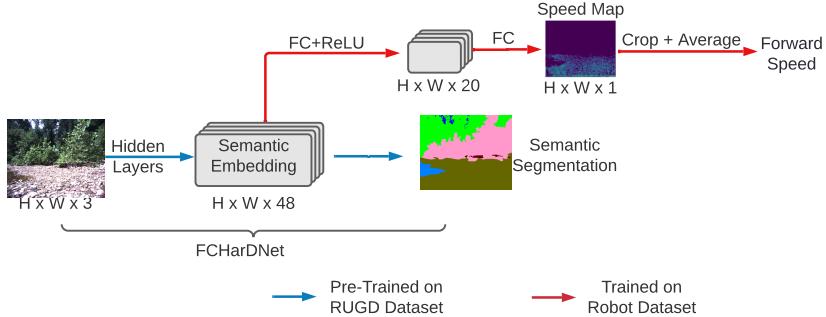


Figure 2: Architecture of our perception network. We extract a semantic embedding from a pre-trained semantic segmentation network, and use it to learn and predict forward speeds.

the speed map – a region at the bottom of the image, which roughly corresponds to a rectangular area 1m long, 0.3m wide in front of the robot.

Learning Speed Commands from Human Demonstration Even with the pre-trained image embedding, learning the speed policy directly in the real world using reinforcement learning is still challenging, due to omnipresent noise in real world environments and concerns for robot safety. Instead, we leverage human demonstration data and train the speed policy using imitation learning.

Collection of Demonstration Data We collect human demonstration data, where the human operator walks the robot on a variety of terrains, including asphalt, pebble, grass and dirt. During data collection, the operator gives speed command (between 0.5 m/s and 1.5m/s) to the robot using a joystick, while other components of the pipeline, such as the gait selector and motor controller, works accordingly (Fig. 1). Each time the camera captures a new image, we store the image and the corresponding speed command for policy training.

Policy Training We train the speed policy using behavior cloning [8], where we solve a supervised learning problem to minimize the difference between policy output and human command. To train the speed policy, we compute the mean-squared loss between the predicted forward speed (Fig. 2) and demonstrated speed, and update network parameters using back-propagation.

Policy Deployment At deployment time, the policy computes the desired forward speed from RGB camera images. We clip the policy output to be in [0.5, 1.5] m/s to match demonstration data.

5 Speed-based Gait Selector

In addition to speed, the *gait* of a legged robot, such as its foot swing height, can greatly affect its traversability, especially on uneven terrains. Moreover, change in robot speed usually leads to a change in the set of feasible gaits. For example, as the robot accelerates, it needs to increase its stepping frequency in order to catch up with the accelerated base [24, 25, 26]. To select a gait with high traversability for each speed, we design a heuristics-based *gait selector* that computes appropriate gait parameters based on desired forward speed.

Gait Parameterization In our design, each gait is parameterized by three parameters, stepping frequency, swing foot height, and base height. The **stepping frequency (SF)** determines the number of locomotion cycles each second. Similar to [25], we adopt a phase-based parameterization for gait cycles, where each leg alternates between swing and stance. In addition, we assume a trotting pattern for leg coordination, where diagonal legs move together and are 180° out-of-phase with the other diagonal. The trotting pattern is known for its stability and thereby is the default gait choice in most quadrupedal robots [21, 9]. The **swing foot height (SH)** determines the leg’s maximum ground clearance in each swing phase. While a higher swing height improves stability on uneven terrains by preventing unexpected contacts, a lower swing height is usually necessary for high-speed running. The **base height (BH)** specifies the height of robot’s center-of-mass. While a low base

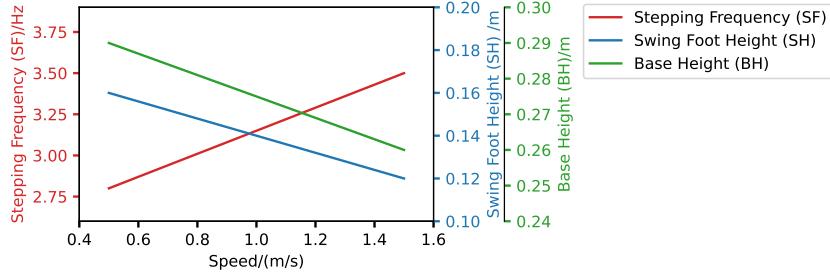


Figure 3: The gait selector selects gait parameters (SF, SH and BH) based on desired forward speed. For example, when the desired speed is 0.5m/s, the speed selector would choose a stepping frequency of 2.8Hz, a swing foot height of 0.16m, and a base height of 0.29m.

height gives better stability at high speeds, a higher base height can be beneficial when traversing through unknown obstacles.

Speed-Based Gait Selection We use empirical evidence to design the speed-based gait selector, which finds a gait with high traversability for each speed. More specifically, for the boundary speeds (0.5m/s and 1.5m/s), we first try different SFs with a nominal SH (0.12m) and BH (0.26m), and find the lowest SF that would still ensure base stability (2.8Hz and 3.5Hz). After that, we sweep over different values of SH and BH, to find the highest value of both that would allow the robot to walk robustly without falling. Lastly, we linearly interpolate the parameter values between the boundary speeds to find the gait for intermediate speeds. See Fig. 3 for details.

6 Low-level Convex MPC Controller

The low-level convex MPC controller computes and applies torques for each actuated degree of freedom, given the locomotion skills from the skill policy. Our low-level convex MPC controller is based on Di Carlo et al. [21] with two important modifications. Firstly, to walk robustly on slopes, we implemented a state estimator to estimate ground orientation, and adjust the robot pose to fit the inclination of the slope, similar to Gehring et al. [27]. Secondly, to reduce foot slipping, we designed an impedance controller for stance legs [28]. In addition to the motor torque command computed by MPC, the controller adds a small feedback torque to track the leg in its desired position. We found both techniques to significantly improve locomotion quality. Please refer to Appendix A for details.

7 Experiment and Result

To see whether our framework can learn to adapt locomotion skills based on terrain semantics, we deploy it to a quadruped robot and test it in a number of outdoor environments in the real world. We aim to answer the following questions in our experiments:

1. Can our framework operate without failure in complex offroad terrains for extended periods of time, and does it achieve the maximum safety speed for each terrain?
2. How does our framework generalize to unseen terrains?
3. How does our framework compare with existing baselines, and what is the contribution of each component to the overall framework?

7.1 Experiment Setup

We implement our framework on an A1 quadrupedal robot from Unitree [9]. We equip the robot with an Intel Realsense D435i camera to capture RGB images, and a GPS receiver to track its real-time location. We implement the entire control stack in the Robot Operating System (ROS) framework [29], and deploy it on a Mac Mini with M1 chip mounted on the robot. The convex MPC controller runs at 400Hz, and the speed policy and gait selector run at 3Hz.

To train the speed policy, we collected 7239 frames of data on a variety of terrains, which corresponds to 40 minutes of robot operation. The entire process, including robot setup, data collection

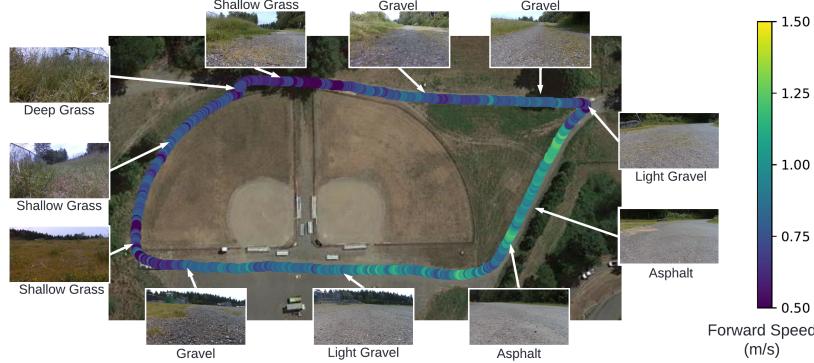
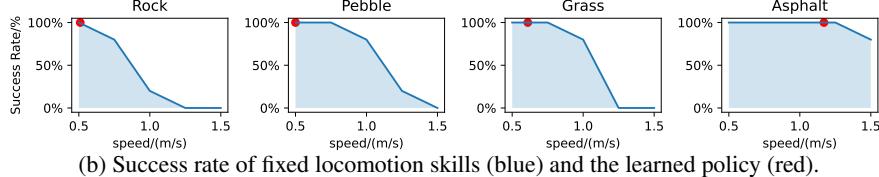


Figure 4: The test trail consists of multiple terrain types such as deep grass, shallow grass, gravel and asphalt. The learned skill policy adjusts the speed and gait based on terrain semantics, and walks faster on easier terrains.



(a) We test our robot on different terrain types.



(b) Success rate of fixed locomotion skills (blue) and the learned policy (red).

Figure 5: Our framework learns fast and failure-free locomotion skills. *Top:* We deployed our learned skill policy in 4 different terrains. *Bottom:* Our learned policy finds a failure-free locomotion skill with a close-to-optimal speed.

and battery swaps, took less than an hour. The speed policy is trained on a standard desktop computer with an Nvidia 2080Ti GPU, which took approximately 30 minutes to complete.

7.2 Fast and Failure-free Walking on Multiple Terrains

To evaluate the off-road traversability of our framework, we deploy our framework on an outdoor trail with multiple terrain types, including deep grass, shallow grass, gravel and asphalt (Fig. 4). Our controller switches between a wide range of skills as it traverses through the trail, from slow and careful stepping to fast and active walking, and completes the trail in 9.6 minutes, comparable to the performance of human demonstration (10 minutes).

To test the optimality of our framework, we deploy the learned skill policy on four different outdoor terrains, including rock, pebble, grass and pebble (Fig. 5a). We compare our semantics-aware skill policy with 5 fixed skills, where the speed linearly interpolates between 0.5m/s and 1.5m/s. For each skill, the corresponding gait is selected according to Fig. 3. For each terrain and skill combination, we repeat the experiment 5 times, and report the success rate, where a trial is considered successful if the robot does not fall over during the traversal (Fig. 5b). By comparing the success rate at different speeds, we obtained an approximation of the safe speed range for each terrain. We then test the optimality of our framework by comparing the average speed obtained by our learned skill policy on each terrain against these safe ranges.

The maximum safe speed varies significantly on different terrains. For example, while the robot can walk up to 1.25m/s without failure on asphalt, it can only walk up to 0.5m/s on rock, due to unexpected bumps and foot slips on the surface. Although not directly optimized for speed or robot safety, our learned policy finds a close-to-maximum speed in the failure-free region of each terrain

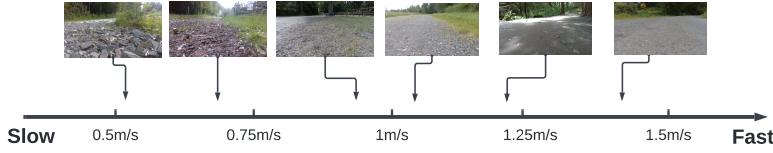


Figure 6: Desired speed computed by the skill policy. The policy prefers faster skills for rigid and flat terrains, and prefers slower skills for deformable or uneven terrains.

after learning from human demonstrations. We also noted that on pebble and grass, there is a slightly larger gap between the maximum safety speed and the speed selected by the skill policy. One reason for this is that the speed demonstrated by the human operator can be more conservative than the maximum safety speed.

7.3 Generalization to Unseen Terrains

To further test the generalizability of our framework, we deploy the robot on a number of outdoor trails not seen during training. The trails contain diverse terrain types, such as dirt, gravel, mud, grass and asphalt. The robot traverses through all of them without failure, and adjusts its locomotion skills based on terrain semantics. Please refer to Appendix. B.1 for details. To demonstrate the skill choices of our framework, we select a few key frames from the camera images and plot the corresponding speed in Fig. 6. In general, the skill policy selects a faster skill on rigid and flat terrains, and a slower speed on deformable or uneven terrains. At the time of writing, the robot has traversed through over 6km of outdoor trails without failure.

7.4 Comparison with Unitree’s Built-in Controllers

We compared our learned framework with a baseline, the Unitree’s built-in controller on the same trail (Fig. 4). We tested two modes of the built-in controller, a *normal* mode (Unitree-Normal) that walks up to 1m/s, and a *sports* (Unitree-Sport) mode that walks up to 1.5m/s. Both controllers do not include perception and assume a fixed gait at all times. Results can be found in Table 1. Although normal mode completed the entire trail without failure, it walks slower than our learned framework, especially on asphalts, due to limitations on the maximum speed. On the other hand, the sports mode controller failed to complete the course, and got stuck in deep grass twice due to insufficient swing foot clearance. Compared to the built-in controllers, our perception-enabled, terrain-adaptive framework completes the trail in a shorter time and without failures.

7.5 Ablation Studies

We further conduct ablation studies to justify important design choices.

Fixed Skill with No Adaptation For these baselines, we disable the perception module and operate the robot with a fixed locomotion skills. We test three skills, namely slow, medium and fast, operating at 0.5m/s, 1m/s, 1.5m/s, respectively, with the corresponding gait selected according to Fig. 3. Both the medium skill and the fast skill fail to complete the trail and incurred a large number of failures. While the slow skill completes the trail without failure, its traversal time is 50% longer compared to our learned framework.

Adapt Speed or Gait Only In our framework, we design a robot skill to be a combination of gait and forward speed. To justify this design, we design two policies, where the robot adapts the gait or the forward speed only. For the speed-only policy, we fix the gait to have a stepping frequency of 3.1Hz, a swing height of 0.14m, and a base height of 0.28m, which corresponds to a forward speed of 1.0m/s in Fig. 3, and allow the speed to be adapted in the same way as our framework. For the gait-only policy, we fix the base speed to be 0.8m/s, similar to the average speed attained by our learned policy, and adapt the gait in the same way as our framework. Both policies failed to complete the trail. For speed-only policy, we found the fixed gait to only work well when the base speed is close to 1m/s, and fails frequently at either higher or lower speeds. For the gait-only policy, the robot is mostly stable throughout the trail, but still fails twice on unexpected bumps.

Policy Type	Speed (m/s)	Gait Params (SF, SH, BH)	Traversal Time (min)	Number of Failures
Fixed-Slow	0.5	[2.8, 0.16, 0.29]	15	0
Fixed-Medium	1	[3.1, 0.14, 0.28]	∞	4
Fixed-Fast	1.5	[3.5, 0.12, 0.27]	∞	10+
Speed-Only	Adaptive	[3.1, 0.14, 0.28]	∞	9
Gait-Only	0.8	Adaptive	∞	2
Unitree-Normal	Tele-operated	N/A	11	0
Unitree-Sport	Tele-operated	N/A	∞	2
Fully-Adaptive (ours)	Adaptive	Adaptive	9.6	0

Table 1: Performance of different policies on the test trail. Compared to other policies, our framework completes the entire trail without failure in the shortest time.

Learning Without Human Demonstration Additionally, to demonstrate the importance of learning from human demonstration, we designed a baseline, in which we train the speed policy directly in the real world. For sample-efficiency concerns, we choose to optimize the speed policy using Bayesian Optimization. Please refer to Appendix. B.2 for the details. Despite its high sample efficiency, we find the training process unsafe and labor intensive, due to frequent attempts at unsafe locomotion actions and constant need for robot resets, and have to stop the experiment early. While additional efforts in safe and data-efficient real world learning might make direct real-world training possible, our framework provides a simple alternative approach, where the robot learns the speed policy using only 40 minutes of human demonstration.

8 Limitations and Future Work

In this work, we present a hierarchical framework to learn semantic-aware locomotion gaits from human demonstrations. Our framework learns to adapt locomotion skills for a variety of terrains using 40 minutes of human demonstration, and traverses over 6km of outdoor terrains without failure. One limitation of our framework is that its performance is limited by the quality of human demonstration. For example, while the same robot platform has achieved a speed over 2m/s [25, 9], the maximum speed learned by our framework is close to human walking speed (around 1.5m/s), as it is difficult for human to demonstrate at faster speeds. Another limitation of the current framework is that perception is only used to select locomotion skills but is not yet used in navigational tasks such as path planning. In future work, we plan to integrate navigation and path planning into our framework, so that the robot can operate fully autonomously in challenging off-road environments.

References

- [1] 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. In *5th Annual Conference on Robot Learning*, 2021.
- [2] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis. Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control. *arXiv preprint arXiv:2012.03094*, 2020.
- [3] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter. Robust rough-terrain locomotion with a quadrupedal robot. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5761–5768. IEEE, 2018.
- [4] 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)*, pages 2436–2442. IEEE, 2020.
- [5] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter. Perceptive locomotion in rough terrain—online foothold optimization. *IEEE Robotics and Automation Letters*, 5(4):5370–5376, 2020.
- [6] 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*, 7(62):eabk2822, 2022.
- [7] Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, and D. Pathak. Coupling vision and proprioception for navigation of legged robots. *CoRR*, abs/2112.02094, 2021. URL <https://arxiv.org/abs/2112.02094>.
- [8] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.
- [9] A1 Website. URL <https://www.unitree.com/products/a1/>.
- [10] P. Fankhauser, M. Bloesch, and M. Hutter. Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters*, 3(4):3019–3026, 2018.
- [11] J. Frey, D. Hoeller, S. Khattak, and M. Hutter. Locomotion policy guided traversability learning using volumetric representations of complex environments. *arXiv preprint arXiv:2203.15854*, 2022.
- [12] R. Yang, M. Zhang, N. Hansen, H. Xu, and X. Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=nhnJ3oo6AB>.
- [13] L. Wellhausen, A. Dosovitskiy, R. Ranftl, K. Walas, C. Cadena, and M. Hutter. Where should i walk? predicting terrain properties from images via self-supervised learning. *IEEE Robotics and Automation Letters*, 4(2):1509–1516, 2019.
- [14] J.-F. Lalonde, N. Vandapel, D. F. Huber, and M. Hebert. Natural terrain classification using three-dimensional lidar data for ground robot mobility. *Journal of field robotics*, 23(10):839–861, 2006.
- [15] Y. N. Khan, P. Komma, and A. Zell. High resolution visual terrain classification for outdoor robots. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1014–1021. IEEE, 2011.
- [16] F. Schilling, X. Chen, J. Folkesson, and P. Jensfelt. Geometric and visual terrain classification for autonomous mobile navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2678–2684. IEEE, 2017.
- [17] A. Shaban, X. Meng, J. Lee, B. Boots, and D. Fox. Semantic terrain classification for off-road autonomous driving. In *Conference on Robot Learning*, pages 619–629. PMLR, 2022.

- [18] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A.-A. Agha-Mohammadi. Step: Stochastic traversability evaluation and planning for risk-aware off-road navigation. In *Robotics: Science and Systems*, pages 1–21. RSS Foundation, 2021.
- [19] X. Cai, M. Everett, J. Fink, and J. P. How. Risk-aware off-road navigation via a learned speed distribution map. *arXiv preprint arXiv:2203.13429*, 2022.
- [20] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.
- [21] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1–9. IEEE, 2018.
- [22] P. Chao, C.-Y. Kao, Y.-S. Ruan, C.-H. Huang, and Y.-L. Lin. Hardnet: A low memory traffic network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3552–3561, 2019.
- [23] M. Wigness, S. Eum, J. G. Rogers, D. Han, and H. Kwon. A rugh dataset for autonomous navigation and visual perception in unstructured outdoor environments. In *International Conference on Intelligent Robots and Systems (IROS)*, 2019.
- [24] D. F. Hoyt and C. R. Taylor. Gait and the energetics of locomotion in horses. *Nature*, 292(5820):239–240, 1981.
- [25] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots. Fast and efficient locomotion via learned gait transitions. In *Conference on Robot Learning*, pages 773–783. PMLR, 2022.
- [26] Z. Fu, A. Kumar, J. Malik, and D. Pathak. Minimizing energy consumption leads to the emergence of gaits in legged robots. *arXiv preprint arXiv:2111.01674*, 2021.
- [27] C. Gehring, C. D. Bellicoso, S. Coros, M. Bloesch, P. Fankhauser, M. Hutter, and R. Siegwart. Dynamic trotting on slopes for quadrupedal robots. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5129–5135. IEEE, 2015.
- [28] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter. Dynamic locomotion on slippery ground. *IEEE Robotics and Automation Letters*, 4(4):4170–4176, 2019.
- [29] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.
- [30] A. Krause and C. Ong. Contextual gaussian process bandit optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL <https://proceedings.neurips.cc/paper/2011/file/f3f1b7fc5a8779a9e618e1f23a7b7860-Paper.pdf>.

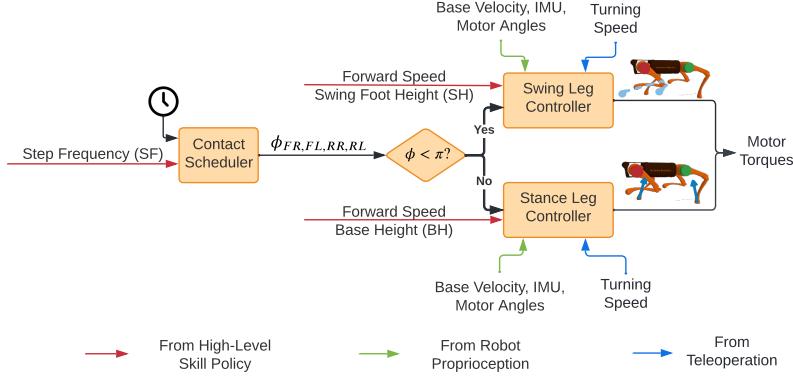


Figure 7: Overview of our low-level convex MPC controller. We use separate control strategies for swing and stance legs, and use a phase-based contact scheduler to determine the contact state of each leg.

A Details of the Convex MPC Controller

Our low-level convex MPC controller is similar to previous works [21, 25], which includes a contact scheduler to determine the contact state of each foot (*swing* or *stance*), and separate controllers for swing and stance legs. In addition, we implement a slope detector to react to changes in ground level, and an impedance controller to prevent excessive foot slipping.

A.1 Phase-based Contact Scheduler

The contact scheduler determines the contact state of each leg (swing or stance). Similar to previous work by Yang et al. [25], we adopt a phase-based representation for contact schedule. More specifically, we introduce four phase variables $\phi_{FR,FL,RR,RL} \in [0, 2\pi]$, one for each leg. The subscripts denote the identity of each leg (Front-Right, Front-Left, Rear-Right, Rear-Left). The phase of each leg denotes its progress in the current locomotion cycle. As a leg moves, its phase ϕ increases monotonically from 0 to 2π , and wraps back to 0 as the next locomotion cycle starts.

At each control step, we determine the leg phases using the following procedure. First, we progress the phase of the front-right leg based on the stepping frequency (SF), f , which is obtained from the gait selector (Section 5):

$$\phi_{FL} \leftarrow \phi_{FL} + 2\pi f \Delta t \quad (1)$$

where Δt is the control timestep (0.0025s). We then determine the remaining leg phases according to the trotting pattern, where diagonal legs move together, and 180° out-of-phase with the other diagonal:

$$\begin{aligned} \phi_{FL} &= \phi_{FR} + \pi \\ \phi_{RR} &= \phi_{FR} + \pi \\ \phi_{RL} &= \phi_{FR} \end{aligned} \quad (2)$$

Depending on the phase of each leg, its motor command is computed by either the swing controller ($\phi < \pi$) or the stance controller ($\phi \geq \pi$).

A.2 Swing and Stance Controller

We use separate control strategies for swing and stance legs, where the swing controller uses a PD control to track a desired swing foot trajectory, and the stance controller solves a model predictive control (MPC) problem to optimize for foot forces. Our controller design is based on the work by Di Carlo et al. [21] with modifications. We briefly summarize them here. Please refer to the original work [21] for further details.

Swing Controller The goal of the swing controller is for swing legs to track a desired swing trajectory. At each control step, the swing controller computes the swing trajectory by interpolating between three key positions, namely, the lift-off position $\mathbf{p}_{\text{lift-off}}$, the highest position in the air \mathbf{p}_{air} , and the landing position \mathbf{p}_{land} . $\mathbf{p}_{\text{lift-off}}$ is the recorded lift-off position at the beginning of the swing phase. \mathbf{p}_{air} is the foot’s highest position in the swing phase, where the height is determined by the swing height (SH) from the gait selector (Section 5). \mathbf{p}_{land} is the estimated landing position computed by the Raibert Heuristic. Once the desired swing trajectory is computed, the controller computes the leg’s desired position based on its progress in the current swing state. The controller then converts this desired position into joint angles using inverse kinematics, and uses a joint PD controller to track the desired position.

Stance Controller To find desired foot forces, the stance controller solves a MPC problem, where the objective is for the base to track a desired trajectory. The trajectory is generated by numerically integrating the desired forward speed and steering speed, where the forward speed is provided by the speed policy (Section 4) and the steering speed is provided by an external teleoperator. In addition, the height of the robot in this trajectory is determined by the base height (BH), which is provided by the gait selector (Section 5). To solve this MPC problem efficiently, we cast it as a quadratic program (QP), where the objective is a quadratic trajectory tracking loss, and the constraints include the linearized robot dynamics, actuator limits and approximated friction cones. Once the foot force is solved, we convert it into joint torque commands using jacobian transpose. Note that the final joint torque command is the sum of this torque computed by MPC and an additional torque from position feedback. Please see Appendix A.4 for details.

A.3 Slope Detection and Adaptation

To operate in complex offroad terrains, it is crucial for the robot to adapt its base pose and foot swing range based on terrain slope. Therefore, we implement a slope detector that estimates terrain orientation from foot contact information, and use it to adjust the robot’s pose on steep slopes.

Ground Orientation Estimation The orientation of the ground plane is estimated based on foot contact position and imu readings, similar to previous work by Gehring et al. [27]. To find the ground orientation, we first find the ground normal vector in the robot frame, $\mathbf{n}_{\text{robot}}$, from foot contact positions. More specifically, we keep track of the last contact position of each leg, $\mathbf{p}_{1,\dots,4}$, which is represented in *robot frame*. Assuming that the contact positions lie in the same ground plane, the relationship between the contact positions $\mathbf{p}_{1,\dots,4}$ and the ground normal vector $\mathbf{n}_{\text{robot}}$ can be represented by:

$$\mathbf{n}_{\text{robot}}^T \mathbf{p}_1 = \mathbf{n}_{\text{robot}}^T \mathbf{p}_2 = \mathbf{n}_{\text{robot}}^T \mathbf{p}_3 = \mathbf{n}_{\text{robot}}^T \mathbf{p}_4 \quad (3)$$

We find $\mathbf{n}_{\text{robot}}$ using the following least-squares formulation:

$$\min_{\mathbf{n}_{\text{robot}}} \left\| \begin{bmatrix} | & | & | & | \\ \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_3 & \mathbf{p}_4 \\ | & | & | & | \end{bmatrix}^T \mathbf{n}_{\text{robot}} - \mathbf{1} \right\|_2^2 \quad (4)$$

Finally, we convert the ground normal vector to the world frame based on the robot’s orientation, which is provided by the onboard IMU.

Slope Adaptation for Stance Legs To walk on steep slopes, the robot needs to maintain its body to be *ground-level*, instead of *water-level*, so that each leg has equal amount of swing space (Fig. 8). To achieve that, we express the base pose in the ground frame in the stance-leg MPC controller (Appendix A.2), and adapt the direction of gravity based on the estimated ground orientation.

Slope Adaptation for Swing Legs For swing legs, we adapt the *neutral swing position* to be aligned with the direction of gravity. This configuration allows the base to be controlled more easily even on slippery surfaces, as shown in previous work by Gehring et al. [27].

A.4 Impedance Controller to Counter Foot Slipping

Since the MPC-based stance leg controller assumes static foot contacts, additional slip-handling technique is usually required for the robot to walk on slippery surfaces. For example, to prevent foot

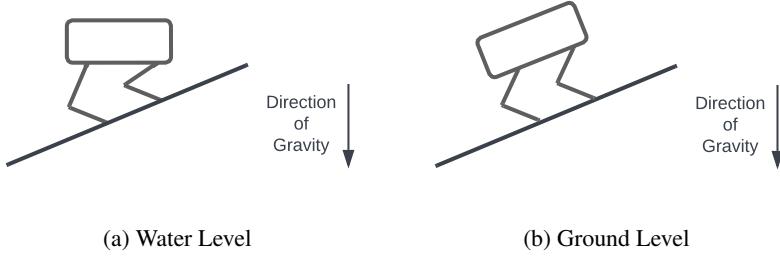


Figure 8: Comparison of stance leg strategies on an upward slope. **Left:** If the robot remains water-level, its front legs are overly retracted, and its rear legs are overly extended, which makes it hard to control both legs. **Right:** If the robot remains ground-level, both front and rear legs have equal amount of extension. We keep the robot ground-level and convert the direction of gravity to ground frame based on estimated ground orientation.



Figure 9: Examples of trails where we test our framework. Note that the trails are not seen during training.

slipping, Jenelten et al. [28] implemented a probabilistic slip detector and used impedance control with different gains for slip and non-slip legs. Similar to this work, we also implement an impedance controller to handle foot slips. However, our approach adopts a unified gain for both slip and non-slip legs, and does not require a slip detector.

In our impedance controller design, the desired torque for each actuated DoF is the sum of the torque computed by MPC and an additional position feedback:

$$\tau = \tau_{\text{MPC}} + k_p(q_{\text{ref}} - q) \quad (5)$$

where τ_{MPC} is the desired torque computed by the MPC solver (Appendix A.2), q is the joint position, and k_p is the position gain. We set the reference, q_{ref} to be the expected joint position as if the foot is in static contact with the ground. Intuitively, when the foot is not slipping ($q \approx q_{\text{ref}}$), the additional torque from the position feedback is small, and our controller falls back to the standard MPC controller. When the foot is slipping, the additional torque from position feedback will bring the leg to the non-slipping position. To determine q_{ref} , we first compute the desired position of the corresponding leg in Cartesian coordinates, p_{ref} . Assuming that the base is moving at the commanded velocity \bar{v} with no foot slip, the foot velocity in robot frame is just the negated base velocity, and the foot position can be computed by numerical integration:

$$p_{\text{ref}} \leftarrow p_{\text{ref}} - \bar{v}\Delta t \quad (6)$$

where Δt is the control timestep. We then convert from foot position p_{ref} to joint position q_{ref} using inverse kinematics.

B Experiment Details

B.1 Terrains for Generalization Test

We test the performance of our framework in a number of trails not seen during training. Please see Fig. 9 and Table 2 for some example trails. These trails include a number of terrain types that are not seen during training, such as mud, moss, mulch and dirt (Fig. 10). Our framework generalizes well to these terrains, and enables the robot to traverse through them quickly and safely.

	Trail 1	Trail 2	Trail 3	Trail 4
Trail Length (km)	0.45	0.41	0.2	0.51
Terrain Type	Dirt	Mixed	Asphalt	Mud
Average Speed (m/s)	0.59	0.74	0.94	0.59

Table 2: Summary of test trails. Our framework selects different locomotion skills (speed and gait) based on terrain type.



Figure 10: Our framework generalizes to unseen terrains, such as mud, moss, mulch and dirt.

B.2 Bayesian Optimization for Locomotion Skills

Problem Formulation We model the problem as a contextual continuous-action multi-armed bandit problem, where the robot decides a locomotion skill $a \in \mathcal{A}$ based on the observed context $c \in \mathcal{C}$ in order to maximize its reward $r : \mathcal{C} \times \mathcal{A} \rightarrow \mathbb{R}$. The context c is a low-dimension embedding for the terrain semantics. To compute c , we first average over the 48-dimensional semantic embedding (Section. 4) in a fixed region of the camera image, and reduce the embedding to 5 dimensions using PCA. The action a , is the desired forward speed. We use the same gait selector (Section. 5) to find the gait for the desired forward speed. To reduce noise in reward measurements, we execute the action a for 4 seconds *on the same terrain*, and compute the reward as follows:

$$r(c, a) = r_{\text{survival}} + \alpha_{\text{stability}} r_{\text{stability}} + \alpha_{\text{speed}} r_{\text{speed}} + \alpha_{\text{energy}} r_{\text{energy}} \quad (7)$$

The survival reward, $r_{\text{survival}} \in [0, 4]$ is the total amount of time that the robot walks without falling. The stability penalty, $r_{\text{stability}}$, is the sum of the square of base angular velocities. The speed reward, r_{speed} , is the desired forward speed, as specified by the action a . The energy penalty, r_{energy} , is the average power dissipation across all motors. We set the coefficients as $\alpha_{\text{stability}} = -10$, $\alpha_{\text{speed}} = 3$, $\alpha_{\text{energy}} = 0.001$, respectively.

Bayesian Optimization Setup We aim to solve the problem using the CGP-UCB algorithm [30], where the agent collects experiences and updates an estimated reward function. We model the reward function, $r(c, a)$, as a Gaussian Process, and learns it using experiences collected. At the beginning of each episode, the robot receives the context c , and finds the corresponding action that maximizes the following acquisition function:

$$a^* = \arg \max_{a \in \mathcal{A}} \mu(c, a) + \kappa \sigma(c, a) \quad (8)$$

where $\mu(c, a)$ and $\sigma(c, a)$ is the mean and standard deviation of the reward r estimated by the Gaussian Process, and $\kappa = 1.8$ is a parameter that trades-off between exploration and exploitation.

Real-world Results We first test the performance of our algorithm on a simplified task, where the robot walks on the same terrain at every episode, and receives a similar context c . We find that our algorithm learns quickly in this case, and converges to the optimal safe speed for each terrain in less than 20 trials. However, when the robot switches between different terrains during training, our algorithm becomes less stable, and keeps proposing highly dangerous actions to the robot, such as walking fast on difficult terrains. Moreover, our algorithm requires the robot to be manually reset after each episode, which is costly in the real world. Without additional effort in ensuring robot safety and automating the reset procedure, it can be difficult to deploy our algorithm to learn locomotion skills directly in complex outdoor terrains.