# Learning Robust Autonomous Navigation and Locomotion for Wheeled-Legged Robots

Joonho Lee[1*], Marko Bjelonic[2,†], Alexander Reske[2,†], Lorenz Wellhausen[2,†], Takahiro Miki[1], Marco Hutter[1]

[1]Robotic Systems Lab, ETH Zurich, Switzerland

[2]Swiss-Mile Robotics AG, Switzerland

[†]Substantial part of the work was carried out during their stay at 1

[*]Please address correspondence to jolee@ethz.ch

**This research addresses the challenges wheeled-legged robots face in achieving locomotion and autonomy in complex environments. We present a fully integrated system comprising locomotion control, mobility-aware navigation planning, and large-scale planning. We use model-free reinforcement learning (RL) techniques and privileged learning to develop a versatile controller capable of smoothly transitioning between walking and driving modes, jumping off tables, and overcoming other obstacles. The mobility-aware navigation tightly integrates with the locomotion controller using a hierarchical RL framework. This integration enables the robot to navigate complex environments with challenging terrain and static and dynamic obstacles. Autonomous deployments in the city of Zurich and Seville further validate our approach. The system successfully completes kilometer-scale missions, showing robustness and adaptability. Our research highlights the importance of integrated control systems for seamless navigation and locomotion in complex environments. The findings contribute to the feasibility of wheeled-legged robotic so-**

**lutions and hierarchical RL for large-scale navigation, with implications for last-mile delivery and beyond.**

# Summary

Hierarchical Reinforcement Learning Facilitates Robust and Efficient Autonomy in Real-World Deployments of Wheeled-Legged Robots

# Introduction

A significant portion of the population resides in urban areas, leading to a considerable challenge in supply-chain logistics, especially for last-mile deliveries. The increasing traffic and demand for faster delivery services put additional pressure on our roads. By shifting reliance from individual motorized transportation to smart and versatile robotic solutions, we can significantly improve the efficiency of urban delivery. Moreover, last-mile delivery routes are not limited to streets; they can be extended by utilizing indoor routes, providing an efficient alternative to human labor. To fulfill all these roles, robots must be fast and efficient on flat ground while being able to overcome obstacles like stairs. Traditional wheeled robots cannot surmount these obstacles effectively, and legged systems alone are inadequate in achieving the necessary velocity and efficiency. For instance, the ANYmal robot (*1*) can only operate for a maximum of 1 hour (*2, 3*) at half the speed of an average human walking (2.2 km/h on average (*4*)).

Wheeled-legged robots offer a comprehensive solution that addresses these requirements (*5–8*). Our research has been dedicated to the development of a wheeled-legged robot, as depicted in Figure 1, where actuated wheels are integrated with its legs (*6*). Unlike other logistics platforms, this design empowers the robot to operate effectively over long distances, enabling high-speed locomotion on moderate surfaces while maintaining agility on challenging terrains (*9, 10*).

Figure 1: Urban deployments. Our control system for the wheeled-legged robot has undergone extensive validation in various indoor and outdoor locations. The experiments took place in Zurich, Switzerland and in Seville, Spain. (Row 1) Locomotion challenges. (Row 2) Navigation challenges; dynamic and static obstacles, complex terrains, and narrow space. (Row 3) Locations in Zurich. (Row 4) Locations in Seville.

## Challenges

Despite the advancements mentioned above, the autonomous real-world deployment of such machines requires solutions for several open challenges:

### 1. Missing hybrid locomotion

So far, existing approaches for hybrid wheeled-legged locomotion (hybrid locomotion) build upon simple heuristics to decide when to step and when to drive (*10*) or rely on pre-defined gait sequences (*11, 12*). Most control strategies designed for legged robots incorporate hand-crafted gait patterns (*13, 14*) or motion primitives (*15, 16*) inspired by nature, but we cannot take observations from biological organisms for wheeled-legged robots. Determining an effective wheeled-legged gait for each situation is not straightforward, as speed and efficiency heavily depend on the direction of motion and chosen gait. For example, minimizing stepping can lead to a lower Cost of Transport (COT) (*10*) for wheeled-legged robots, but traditional methods for legged robots often do not consider gait switching, resulting in sub-optimal outcomes when applied to wheeled-legged robots. Some directly optimize for COT (*17, 18*) and demonstrated improved performance with gait adaptation, but the results are limited to indoor settings or moderate terrains with robots mostly moving forward. To generate more complex motions that combine driving and walking, trajectory optimization techniques have been utilized to directly optimize gait and discover complex behaviors such as terrain-aware gait and skidding (*9, 19*). However, these methods are computationally expensive and often rely on close-to-optimal initialization. Additionally, some of these approaches prioritize computational efficiency at the expense of model accuracy, such as by neglecting the dynamics of wheels, leading to sub-optimal performance on the real robot.

### 2. Mobility-unaware navigation

Urban environments consist of a significant portion of flat and open areas that require efficient high-speed traversal to cover large distances. At the same time, it is bristled with obstacles like stairs and uneven terrain. To achieve speed, efficiency, and obstacle negotiation capabilities, a navigation algorithm must consider the characteristics of hybrid locomotion. This understanding is crucial for issuing commands that optimize efficiency on flat terrain while maintaining agility when faced with obstacles. Many existing approaches (*20, 21*) define navigation costs, such as traversability (*21, 22*), independently of the robot's whole-body state. These methods often rely on sampling-based planning using estimated cost maps. Therefore, these approaches have limited capability to account for low-level locomotion behaviors, such as tracking error variations based on terrain, commanded velocity, or gait. Consequently, they may result in frequent turning and stepping actions that can decrease efficiency.

In addition to the previous points, the higher speed capabilities of wheeled-legged robots introduce the need for faster reaction times, which raises safety concerns and calls for the development of more responsive systems. State-of-the-art sampling-based planners designed for legged robots typically take several seconds to compute a path (*20*). However, when operating at speeds of multiple meters per second, relying on such planning methods would necessitate long foresight and could result in delayed decision-making. In dynamic environments or situations involving human presence, ensuring safety requires faster and more frequent decision-making capabilities than what traditional planning methods can provide.

### 3. Orchestration of too many sub-modules

Attaining autonomy in robotic systems poses a significant challenge that necessitates the seamless integration of various sub-modules. Team Cerberus undertook the development of an autonomy system for a classic legged robot during the DARPA Subterranean Challenge (*2*). The

endeavor involved careful engineering of individual components that play pivotal roles in the robot's autonomy. These components encompassed global navigation planning, state estimation, traversability estimation, local navigation planning, path following, and locomotion control. By carefully designing and integrating these individual modules, the legged robot achieved an autonomy system capable of tackling the challenges posed by the DARPA Subterranean Challenge.

In this context, our work focuses particularly on enhancing the coordination of local navigation and locomotion control, a critical subset of the autonomy system. In this aspect of autonomy, Team Cerberus incorporated multiple modules - traversability estimation, local navigation planning, path following, and locomotion control. The inter-module communication in current systems is largely guided by heuristic methods, which can limit the overall system's effectiveness and adaptability.

Smooth trajectories and robust operation often hinge on these engineered heuristics. This issue becomes evident in commonly observed behaviors, as reported in Tranzatto et al. (*2*), where robots frequently pause midway through a path to re-plan or exhibit zigzag motion while following a given path. Such discontinuous or oscillatory behavior can compromise efficiency and hinder the robot's ability to effectively respond to complex scenarios. In some cases, navigation failure may occur when computed navigation paths are not accurately followed (*20*). To address these challenges, a holistic approach is required to develop efficient and robust control systems that integrate multiple sub-modules.

## Contributions

In this work, we tackle the three key challenges faced by wheeled-legged robots when aiming for autonomy in complex and dynamic environments. We present a fully integrated system specifically designed for autonomous last-mile delivery in complex urban settings.

### 1. Hybrid locomotion control

We have developed a robust and versatile locomotion controller for wheeled-legged robots utilizing model-free Reinforcement Learning (RL) techniques and privileged learning (*15, 23, 24*). By leveraging model-free RL, which relies minimally on human intuition, we achieve a highly performant locomotion controller capable of making decisions regarding the gait and smoothly transitioning between walking and driving modes. To train our controllers, we utilize simulation environments and employ privileged learning. During the training, an agent utilizes additional information not available during the deployment phase to enhance the model's performance. In our approach, we employ the robot's motion (e.g., velocity and acceleration), terrain properties, and noiseless exteroceptive measurements as privileged information. During deployment, the final policy only relies on raw measurements from Inertial Measurement Unit (IMU), joint encoder readings, and noisy exteroceptive mapping, similar to the approach presented by Ji et al. (*25*), minimizing the use of heuristics for noise filtering and eliminating the need for state estimation for orientation and velocity estimates. This approach results in enhanced robustness when operating on challenging terrains and reduces the number of failure points in terms of locomotion control.

### 2. Mobility-aware navigation planning

The primary technical contribution of this work lies in the development of our learned high-level controller (HLC). This controller tightly integrates local navigation planning and path following with the hybrid locomotion controller described in the previous paragraph. This integration is achieved using a Hierarchical Reinforcement Learning (HRL) framework and our new method of training a navigation policy using pre-generated navigation graphs in simulation.

Training a HLC with the low-level controller (LLC) in the loop, HRL ensures the navigation planner fully aware of various characteristics of the locomotion controller, including tracking

behavior, gait phase, joint trajectories, latent embedding, and more. By leveraging HRL, we can effectively learn policies for solving tasks that can be decomposed into sub-tasks, making it an ideal approach for integrating the navigation planner, path follower, and locomotion controller. HRL ensures seamless coordination between these modules, leading to improved overall performance in complex real-world scenarios that involve challenging terrain, as well as static and dynamic obstacles.

We adopt the concept of "Navigation Graph" from computer games (*26, 27*) to provide solvable yet challenging navigation problems to the agent during training (see Fig. 2). The terrain is created using a procedural content generation algorithm called Wave Function Collapse (WFC) (*28*), and each terrain comes with a graph that defines feasible paths and safe areas. The training environment offers a diverse array of navigation challenges, such as detours, dynamic obstacles, rough terrains, and narrow passages. As detailed in the method section, we combine different obstacles in a controlled manner and reward RL agents for following the shortest path towards the goal point during training. This approach enhances performance in comparison to policies trained with randomly placed obstacles and goals.

### 3. System integration for large-scale navigation

The two-level hierarchy comprising the mobility-aware navigation planner (HLC) and hybrid locomotion controller (LLC) is integrated into a complete system for urban navigation. This two-level hierarchy replaces the conventional assembly of modules including traversability estimation, local navigation planner, path follower, and locomotion controller, along with the intermodule communication layers. The outcome is a streamlined and more robust navigation system, simplifying the complexities associated with traditional autonomy approaches.

We introduce city-scale global navigation and localization by leveraging pre-mapped digital twins. To create this digital twin for a new campus or city district, we use an off-the-shelf

reality capture device to generate georeferenced point cloud data. Based on this data, a global navigation graph is constructed, incorporating human input. Notably, the navigation graph only necessitates sparse waypoints at each intersection, as our hierarchical controller effectively handles obstacles encountered between these waypoints. During the robot's deployment, it localizes itself in relation to the reference point cloud using its LiDAR, IMU, and kinematics. With this setup, the robot can be provided with a single GPS goal, and it autonomously navigates toward the target location. This integration of city-scale global navigation, localization, navigation planning, and hybrid locomotion control enables efficient and autonomous navigation in urban environments.

Our system has undergone extensive validation through large-scale real-world deployments in urban environments, including Zurich (Switzerland) and Seville (Spain). In Fig. 1, we present challenging scenarios for both locomotion and navigation during our experiments, showing a diverse range of indoor and outdoor locations where our system was rigorously tested. These deployments serve as compelling demonstrations of the efficacy of wheeled-legged quadrupedal robots in achieving autonomy in complex and dynamic environments. During these deployments, our robot successfully completed kilometer-scale autonomous missions in urban settings, showcasing its ability to adapt its gait based on the situation and navigate smoothly and rapidly across various terrains and obstacles. These environments encompass wide-open outdoor spaces, challenging obstacles such as steps and stairs, transitions between indoor and outdoor environments, and multi-level indoor deployment. One noteworthy aspect is the high generalization capabilities of our robot, allowing it to effectively tackle dynamic and complex obstacles, even in cases where reference paths are obstructed.

Overall, our research underscores the significant potential of wheeled-legged robots for achieving robust autonomy in complex and dynamic environments. It emphasizes the importance of developing tightly integrated and robust control systems to enable fast and efficient

navigation. Furthermore, by showcasing our real-world use case, we provide a broader context for this research and assess the feasibility of wheeled-legged robotic solutions and HRL in large-scale navigation, with potential applications in last-mile delivery and beyond.

## System Overview

Figure 2 provides an overview of our system. As depicted in Fig. 2A, our robot is equipped with multiple payloads, including three LiDARs, an RGB stereo camera at the front, a delivery box, a 5G router, and a GPS antenna. They serve various purposes such as localization, terrain mapping, and human detection, contributing to the safety layer. We integrated an RGB camera with high-frequency object detection capabilities because lidar-based terrain mapping does not capture dynamic obstacles well. This allows for real-time tracking of people within a range of 20 meters. We add an offset to the elevation map using this information to create a buffer zone (Figure 4E). We provide more implementation details in supplementary section S2.

Our primary focus in this work is the system illustrated in Fig. 2B. Given a global navigation path, represented by a sequence of graph nodes, we extract two waypoints, denoted as 'WP1' and 'WP2'. Taking inspiration from the pure-pursuit tracking algorithm (*29*), we set two intermediate waypoints by interpolating between the robot's current position projected onto the path and the subsequent graph node, with a fixed look-ahead distance.

Our robot follows the intermediate waypoints using the low-level locomotion policy (LLC) which is commanded by the high-level navigation policy (HLC), both of which are neural networks trained through RL. HLC, fed with waypoints as input, generates velocity targets for LLC at 10 Hz, which aligns with the update rate of the onboard elevation mapping (*30*). LLC, in turn, generates joint position and wheel velocity commands at 50 Hz.

LLC is constructed by a Recurrent Neural Network (RNN)-based policy and builds upon the perceptive locomotion controller by Miki et al. (*16*). We applied modifications to the obser-

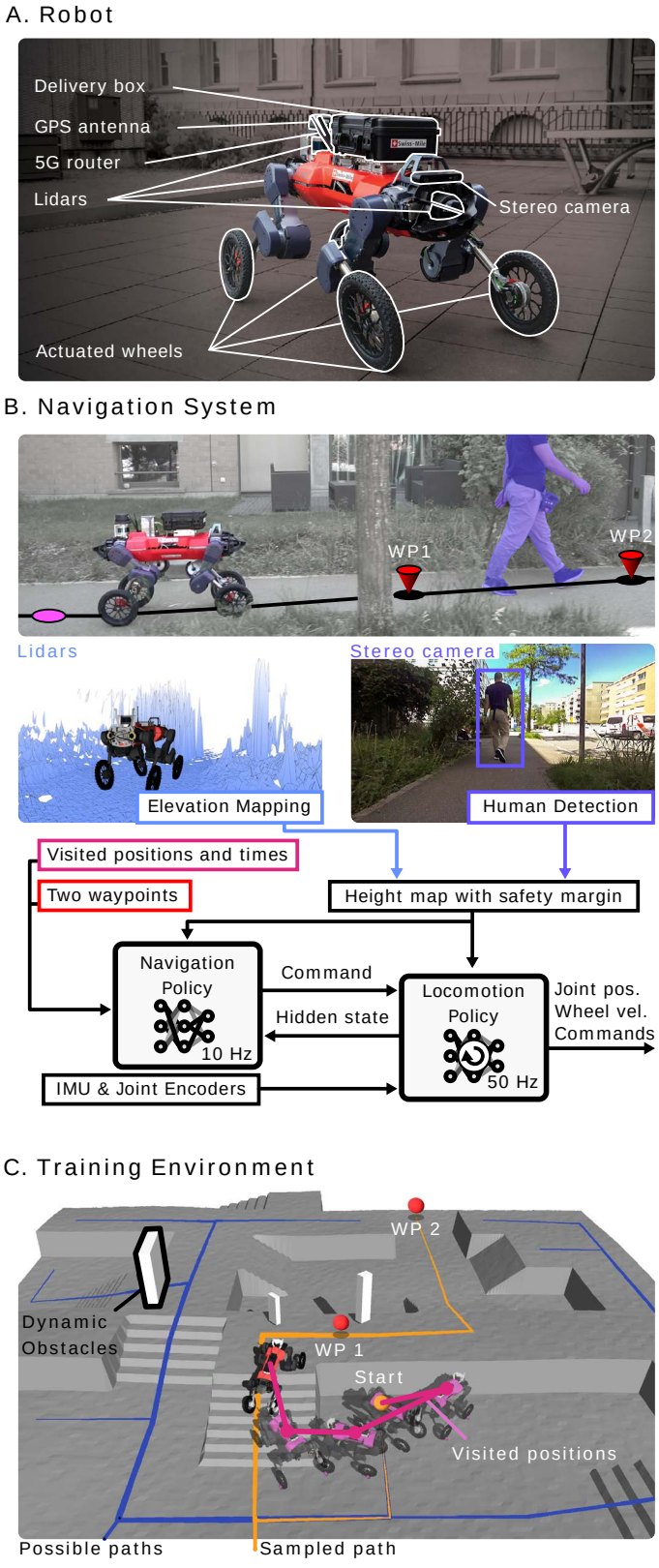


Figure 2: System overview. (A) Our wheeled-legged quadrupedal robot. (B) Overview of the navigation system. (C) Training environment. We leverage pre-generated obstacle-free paths to facilitate navigation learning.

vation and action space to improve robustness and remove engineered motion bias. Technical details are given in the method section.

The HLC is trained in the simulation environment depicted in Fig. 2C. In every episode, a new obstacle-free path is generated, and waypoints are defined along the path at random intervals. The HLC processes multiple input modalities including the hidden state of LLC, a terrain scan around the robot, and a sequence of previously visited positions with corresponding visitation times. Instead of using standard proprioceptive observations, HLC accesses the belief state of LLC. This latent state captures environmental information such as terrain properties and disturbances, as supported by (*15, 16*). Additionally, HLC processes 20 previously visited positions recorded at 50 cm intervals. They span the distance of up to 10 m, approximately the usual waypoint spacing. The history allows HLC to make informed decisions based on the robot's prior navigation experience.

# Results

We conducted autonomous navigation missions in different urban environments. These experiments took place in Zurich, Switzerland, and Seville, Spain. The capability of our system is summarized in Movie 1. Additionally, we show one full mission in supplementary video S1 to demonstrate the scale of each experiment.

## Kilometer-Scale Autonomous Deployments

Fig. 3 provides a summary of our mock-up delivery mission conducted in Glattpark, Zurich. Our robot covered a total distance of 8.3 km with minimal human intervention. The figure highlights the key challenges and outcomes of the mission.

We first show our workflow in Fig. 3A. To begin, we employed a handheld laser scanner to capture dense color point clouds of the experimental area. The scanning process took ap-

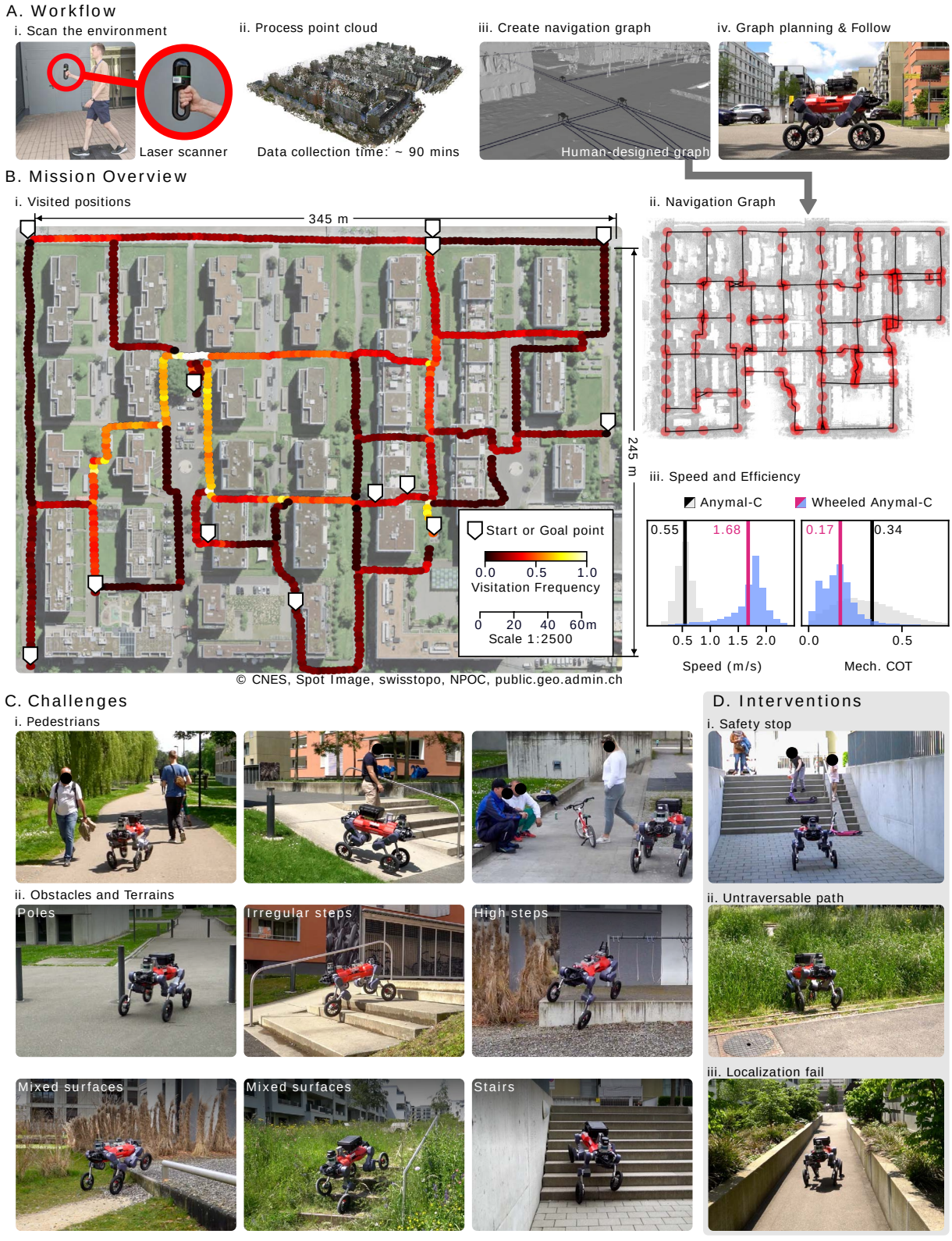


Figure 3: Large scale autonomous navigation experiment at Glattpark, Zurich. (A) Workflow: Offline preparation of the experiment. (B) Outcome of multiple missions. (C) Challenges faced by the robot. (D) Situations requiring interventions.

proximately 90 minutes to cover a 245 m x 345 m urban area. Subsequently, we georeferenced the point cloud and the data was converted into a mesh representation, facilitating the creation of a navigation graph and the placement of goal points by a human expert (see Fig. 3A-iii and Fig. 3B-ii). The purpose of the navigation graph is to provide topological guidance and to indicate social preferences, like avoiding landscaping and private property.

Selected goal points are sent to the robot via mobile network and the reference path is computed onboard using a shortest-path algorithm (*31*). The resulting path is converted into robot-relative coordinates for our navigation policy using LiDAR-localization in the pre-scanned point cloud map. Note that the point cloud is purely for localization and is not otherwise used for navigation (*32*). We have found this localization method to be more robust among high-rise buildings than a GPS-based approach.

Fig. 3B-i illustrates the paths traversed by our robot during multiple long-distance experiments, each lasting more than 30 minutes. Throughout these experiments, we manually selected 13 distant goal points to maximize coverage of the experimental area. This setup required the robot to navigate diverse obstacles in order to reach each goal point successfully. Major challenges are shown in Fig. 3C. Our learned hierarchical controller played a crucial role in enabling the robot to overcome these. By changing gait and modulating speed adaptively, the navigation controller achieved robust and responsive navigation.

Fig. 3B-iii presents histograms of the speed and mechanical COT while the robot was in motion. We define mechanical COT as

$$COT_{mech} = \sum_{\text{all joints}} [\tau\dot{\theta}]^{+}/(mg|v_{xy}^{b}|) \quad ,$$

where $\tau$ denotes joint torque, $\dot{\theta}$ is joint speed, $mg$ is the total weight, and $|v_{xy}^{b}|$ is the horizontal speed of the robot's base. This quantity represents positive mechanical power exerted by the actuator per unit weight and unit locomotion speed (*6, 15*).

Our robot achieved an average speed of 1.68 m/s with a mechanical COT of 0.16. In comparison, we provide data on the average speed and COT of one ANYmal robot which primarily traversed flat and urban terrains during the DARPA Subterranean challenge (*2*)[1]. Our robot demonstrated three times the speed with a 53% lower COT. Note that we only compared the output mechanical power. Other significant factors contributing to energy loss, such as heat loss and mechanical loss from the actuators' transmission, also come into play during constant walking motions.

The improvement is mainly attributed to the driving mode, which evenly distributes weight across all four legs, keeping leg joints relatively static. Constant stepping leads to concentrated loads on fewer legs, requiring higher joint torques and speeds. During driving, joint actuators contribute almost zero mechanical COT ($\approx 0.01$). Compared to a typical ANYmal robot during locomotion, our robot's wheels exert about 1.2 times the total mechanical power, while achieving 3.4 times higher locomotion speed on average. Upon evaluating the average $\sum \tau^2$ solely for leg joints, our robot exhibits a 16 % lower value, despite being both heavier (≈12 kg) and faster. This quantity is directly related to the heat loss (*33, 34*).

Fig. 3C shows the major challenges encountered by our robot, including pedestrians, various obstacles, and non-flat terrains. Our robot demonstrated the capability to navigate around walking pedestrians in various situations, even on slopes or stairs, as depicted in supplementary video S1. Additionally, our robot exhibited the ability to handle thin obstacles, such as the pole shown in the first image of Fig. 3C-ii, as well as various discrete terrains like steps and stairs.

HLC and LLC are trained to minimize $COT_{mech}$ and $\sum \tau^2$. As a result, the robot mostly drives on flat terrain. However, when encountering uneven surfaces, the robot dynamically switches to a stepping gait. Importantly, this gait switching is learned without any handcrafted heuristics like Central Pattern Generator (CPG)s or predefined gait sequences. Furthermore,

[1] ANYmal 4 in this video: https://youtu.be/fCHOU-fw2c0?list=PLE-BQwvVGf8HJYuIH85–ul9DEVDvmt9B

our controller demonstrates robustness in handling various surfaces, including grass, sand, or gravel, which can be attributed to the privileged training of the LLC (*15*).

We intervened during the mission in three circumstances, presented in Fig. 3D. Firstly, there were instances where children were in the robot's path. Although our navigation module would have most likely safely navigated around children, as it did for adults, we prioritized safety and stopped the robot proactively.

Secondly, we encountered situations where the waypoints were located within untraversable regions. Tall grass had grown on a trail in the time between creating the navigation graph and performing experiments. Consequently, it presented an obstacle in the local height map used for navigation. The robot safely stopped in front of the tall grass and we manually triggered global re-planning to go around the obstruction.

Lastly, we encountered challenges with localization in geometrically degenerate environments, such as long corridors. This meant that the reference path became invalid and provided infeasible, potentially hazardous waypoints. Our robot's controller was still able to operate safely by relying on onboard local terrain mapping, but was unable to reach the goal point until localization was recovered.

## Local Navigation

In Fig. 4, we present example scenarios that best show the local navigation capability of our system. The sequence of these scenarios can be viewed in supplementary video S2.

In the first case (Fig. 4A), we show the exploratory behavior when the robot encounters a blocked path. The robot goes reverse or move along the wall, searching for an opening until it finds the stairs leading to the final waypoint. The robot's explicit position memory allows it to reason about its previously visited positions and navigate through complex paths.

Fig. 4B shows our robot's ability to navigate narrow corridors. It successfully maneuvers

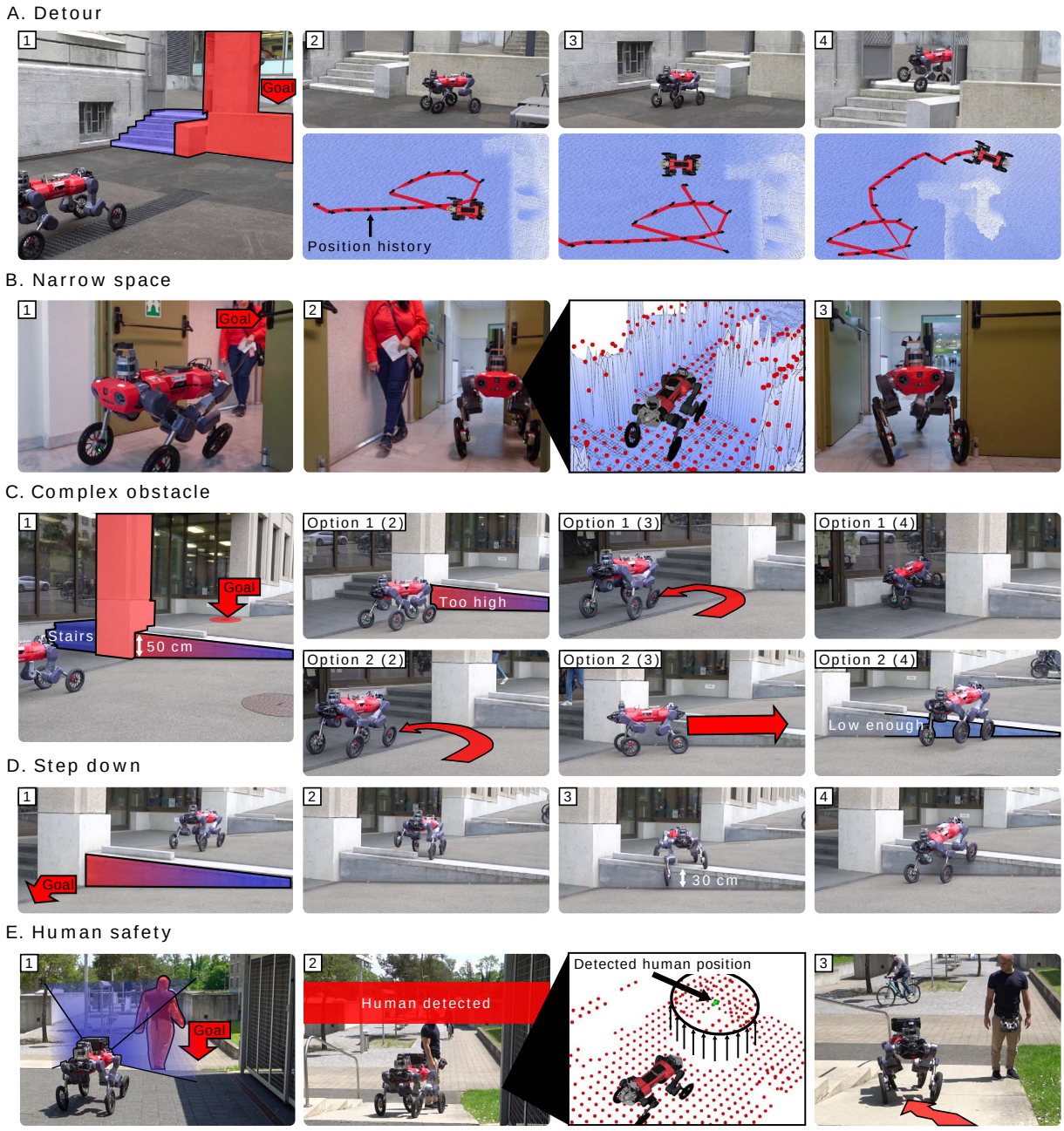


Figure 4: Obstacle negotiation. (A) Navigating around blocked routes by actively exploring the area and finding alternative paths. (B) Safely traversing a narrow space. (C) Our robot exhibits two different ways to traverse the complex obstacle. (C, D) Our robot shows an asymmetric understanding of traversability, being able to traverse higher steps when going down. (E) Ensuring safety around humans by incorporating additional human detection and overriding height scan values.

through two doors with a human standing in-between, where the gap is as wide as the robot's width. The robot navigates through the narrow space without any collisions even though human safety was not enabled in this particular case. This example shows our navigation controller's precision and ability to adjust its trajectory in real-time, making it well-suited for environments with limited space and tight passageways.

We conducted a test with a complex obstacle depicted in Fig. 4C-1. The obstacle consisted of a small staircase on one side and a step with variable height ranging from 0 to 50 cm on the other side. When a waypoint was provided above the step, our robot showcased two different approaches to traverse it. Initially, when the route was obstructed, the robot drove backward and started exploring. During this phase, it could either find the stairs to climb up or continue exploring for a feasible step height. In the second case, by driving along the step and assessing the terrain, the robot determined a viable height of approximately 20 cm. This example shows the effectiveness of our hierarchical controller, seamlessly adapting its gait based on the terrain, and exhibiting versatility in navigating complex paths.

We observed that HLC has an asymmetric understanding of traversability when going up and down the step (Fig. 4D). Specifically, the robot was able to traverse higher steps when descending, indicating that it has a more advanced understanding of the terrain compared to cost-map approaches for traversability estimation (*21, 30*). These methods use symmetric traversability maps that are independent of motion direction, whereas our approach makes decisions based on the current terrain, the robot's state, and the characteristic of the low-level locomotion control.

In Fig. 4E, we visualize our strategy for augmenting the static, local elevation map with dynamic obstacles. We incorporated camera-based human detection to add a height offset in a radius of 50 cm around individuals. As the robot encounters a person moving along its path, HLC, trained to handle dynamic obstacles of various sizes, ensures a constant distance from the person, allowing the robot to successfully overtake them.

## Hybrid Locomotion

We evaluated our LLC over various real-world terrains to observe emerging gaits and assess its robustness. We provide highlights of our locomotion experiments in supplementary video S3. LLC adapts gaits depending on the command velocity and terrain. We tested the policy on various real-world terrains, as illustrated in Fig. 5. Our previous Model Predictive Control (MPC)-based controller (*10*) lacks robustness and cannot operate in the environments depicted in Fig. 5. Additionally, our controller reached the peak speed of 5.0 m/s on flat terrain. The hardware limit allows for a maximum speed of 6.3 m/s, which is determined by the maximum joint speed of 45 rad/s multiplied by the wheel radius of 0.14 m.

Fig. 5AB presents distinct behaviors that are influenced by the terrain. When traversing a large discrete obstacle (High step), the robot displays an asymmetric gait combining creeping (*35*) and driving. Conversely, on bumpy terrain where the height deviations are comparable to the wheel's radius (Uneven ground), the robot leverages the wheels to move forward. The policy adjusts the reach of each leg to maintain the main body's stability and keep the wheels in contact with the terrain, acting as an active suspension. When climbing stairs or steep hills, the robot trots like a normal point-foot quadruped (*16*) (Stair and Steep uphill). Depending on the terrain conditions, such as slope or friction, the gait frequency may vary. Additionally, the policy adjusts the main body's height based on the situation. For instance, when descending a slope, the policy lowers the body height to enhance stability and prevent tipping over.

In Fig.5C, we present two scenarios involving high discrete obstacles. In Fig. 5C-i, we commanded our LLC to drive down a table approximately 60 cm high. As the front legs descend, the robot stretches down its front legs and crouches the hind legs to maintain a level main body. Once the front legs touch down, the front wheesl roll forward to rebalance. In Fig. 5C-ii, we show our robot traversing a block of approximately 40 cm high. In the middle of the block (second figure), none of the wheels are in contact with the ground. Then the robot crawls forward


Figure 5: Different behaviors on various terrains. (A) Terrain types and motion sequences. The robot is moving from left to right following target velocity commands given by the joystick (up to 2 m/s). (B) Corresponding wheel contact sequences. (c) Two challenging discrete obstacles. (D) The maximum step height and slope that the LLC can overcome with a given command velocity.

with its knees until one of the wheels regains contact. This example shows the advantage of using model-free RL (*36*).

Quantitative evaluation of the locomotion performance is presented in Fig.5D. In the first plot, we present the maximum traversable height of the step depending on the command speed. Our robot can traverse higher steps when descending compared to ascending. This observation aligns with the results shown in Fig.4CD, where our HLC avoids high steps and instead chooses to detour to avoid knee collision and ensure safety. In the second case, we tested LLC on a slope with fixed friction coefficient of 0.7 in simulation.

The robot was commanded with a fixed linear velocity to ascend the slope, and success was determined by its ability to climb up for 2 meters. We observed that the stepping behavior, as depicted in Fig. 5A, emerged only on steeper slopes with command speeds over 0.5 m/s. With the stepping gait, the robot was able to climb steeper slopes. This analysis demonstrates the complex characteristics of our LLC in terms of gait patterns and traversability. Conventional navigation planning and path-following approaches would struggle to identify and adapt to such complexities.

## Comparison to a Conventional Navigation Approach

We compared our approach with the conventional sampling-based navigation planner by Wellhausen et al. (*4*). This local navigation planner, used by the Cerberus team in the Subterranean Challenge (*2*), is designed for the autonomous deployment of legged robots. For both methods, we used the same LLC.

We conducted experiments in a point-goal navigation setup, as depicted in Fig.4A. The area was scanned with a laser scanner to create a simulation environment, shown in Fig.6A, with fixed start and goal points.

Fig. 6B illustrates the field of view of our HLC and the baseline. To accommodate the

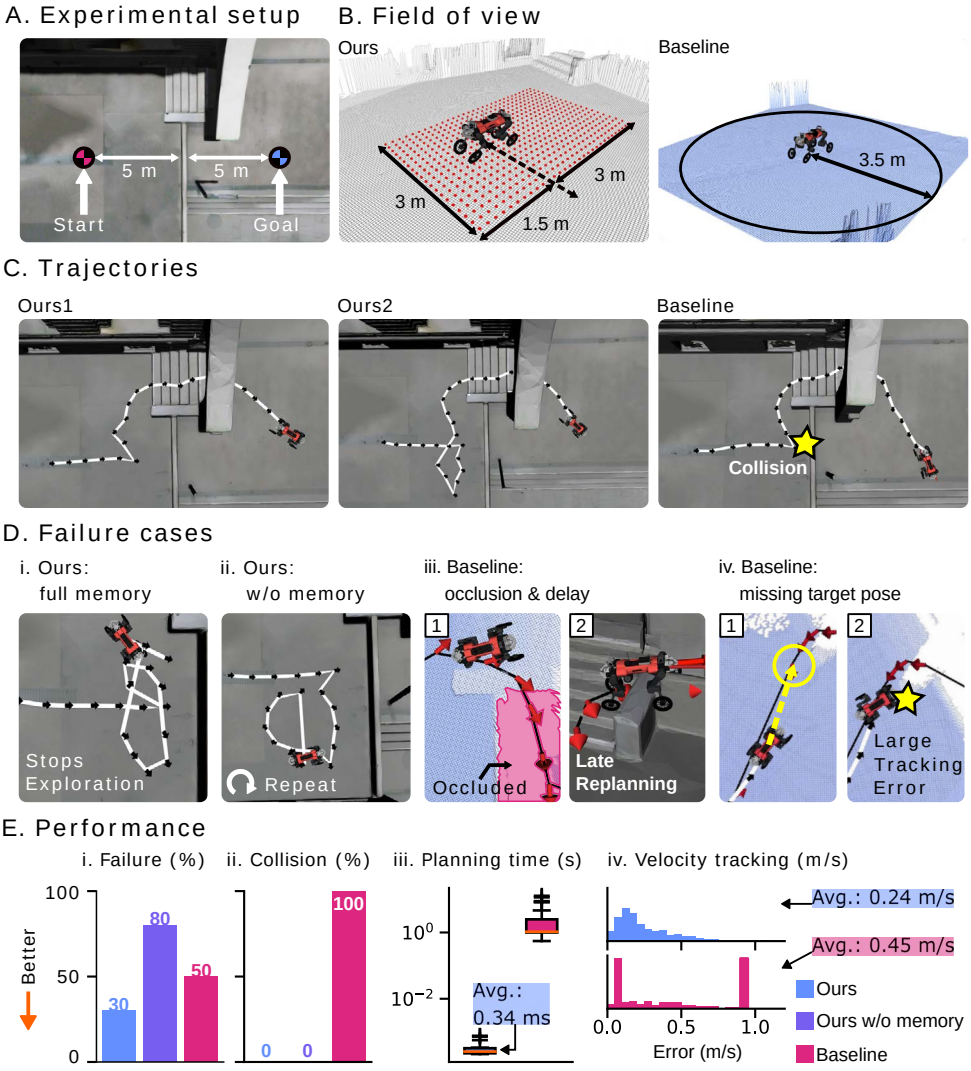


Figure 6: Comparison to a conventional approach in a point-goal navigation setup. (A) Experimental setup. (B) Field of view comparison between different methods. (C) Trajectories of the two methods. Our method displays two distinct trajectories depending on the initial exploration direction. (D) Failure cases. (E) Quantitative evaluation of performance. The experiments were repeated 10 times per method.

limitations during physical deployment, we limit the range of the map to 3.5 meters in both $x$ and $y$ directions. This decision is particularly important when the robot is moving at high speeds, reaching up to 2 m/s. Using a larger map size significantly slows down the elevation map update and results in high delay and empty spots in the map.

Fig. 6C presents the trajectories of both approaches that successfully reached the goal. Our approach demonstrates a tendency to explore the environment and discover an opening at the staircase, as discussed in the previous section. In contrast, the baseline consistently collides due to the occlusion and delay issue, which will be further explained below.

Fig.6D presents failure cases. Our approach sometimes gets stuck during exploration when the position memory buffer is full. We also trained our approach without memory, resulting in repetitive behaviors and difficulty escaping local minima (Fig.6D-ii).

The baseline method faced two challenges: occlusion handling and tracking error of the locomotion controller. While several heuristics can help mitigate the occlusion problem, the baseline method's ability to handle changing situations is limited due to the delay in re-planning. Concerning the second issue, most existing methods assume perfect tracking; however, the actual locomotion controller experiences delays and tracking errors. Fig. 6D-iv illustrates this issue, where distant pose targets lead to high-velocity commands that prevent the robot from stopping at the next waypoint, resulting in collisions. This problem becomes more pronounced when dealing with fast-moving robots or when deploying on rough terrain.

In the quantitative analysis shown in Fig. 6E, Our approach with full memory showed the lowest failure rate. Our method without memory exhibited the highest failure rate. Notably, only our approach achieved a collision-free path, demonstrating the accurate steering capability of our navigation policy while respecting the capabilities of the locomotion policy.

The comparison of the failure rate highlights the advantage of exploratory behavior in partially observable scenarios. Unlike the sampling-based baseline, which is limited to exploring

within the provided map, our method enables the robot to dynamically explore new areas, resulting in a higher success rate. Furthermore, the results obtained from 'Ours w/o memory' emphasize the importance of the memory mechanism in facilitating effective exploration in static environments.

Another significant benefit of our approach lies in its computational efficiency (Fig. 6E-iv). From updating observations to inferring the neural network, our high-level controller takes only 0.34 ms on average. In contrast, depending on the complexity of the environment, the baseline sometimes requires more than 1 s to update the navigation plan on a desktop machine (AMD Ryzen 9 3950X, GeForce RTX 2080).

As mentioned previously, the baseline's high failure rate can also be attributed to imperfect tracking of the low-level policy. Fig. 6E-iv presents a histogram illustrating the tracking error of trajectories from the baseline, which shows an approximately 88% higher on average compared to our approach. Notably, the baseline exhibits a single high peak in the histogram, which occurs when there are discrete changes in the command velocity to the LLC or when the robot is commanded too close to obstacles and LLC refuses to follow the command. In contrast, our high-level controller, trained in conjunction with the low-level controller, demonstrates well-distributed tracking error statistics with consistently low tracking error.

# Discussion

The presented wheeled-legged robot system demonstrates significant advancements in achieving autonomy and robustness in complex urban environments. The integration of mobility-aware navigation planning and hybrid locomotion contributes to the system's ability to navigate challenging terrain and obstacles while ensuring efficient and fast navigation.

Our experiments validated the effectiveness of the proposed system in real-world scenarios. Our wheeled-legged robot successfully completed kilometer-scale autonomous missions in ur-

ban environments. It successfully navigated through various obstacles such as stairs, irregular steps, natural terrain, and pedestrians.

Our results demonstrate several notable advantages over conventional navigation planning approaches. Firstly, our hierarchical controller actively explores areas beyond its current perception. Unlike traditional sampling-based approaches, our method enables the robot to dynamically explore new areas, improving the success rate. The integration of memory allows the robot to reason about previously visited positions, enhancing its decision-making capabilities in complex environments.

Another major advantage of our approach is its responsiveness. The controller dynamically reacts to unperceived obstacles and effectively navigates through urban environments with pedestrians, continuously adapting to changing situations. The incorporation of real-time data and fast computation enables the robot to leverage up-to-date information, enhancing its ability to navigate challenging terrains and avoid obstacles.

Moreover, the presented hybrid locomotion controller exhibits robustness and versatility in traversing various rough terrain. The adaptive gaits observed in our experiments, such as the asymmetric gait for large discrete obstacles, wheel-based locomotion for bumpy terrain, and trotting gait for stairs and steep hills, demonstrate the controller's capability to efficiently traverse diverse terrains.

However, there are still important aspects to consider for future improvements. One such aspect is the incorporation of semantic information into our system. Currently, our system primarily relies on geometric information for navigation, with minimal utilization of semantic information (to adjust the height map for human safety). More advanced scene understanding, such as pavement detection or visual traversability estimation (*37*), will allow the robot to make more informed decisions during navigation. This is exemplified by the work of Sorokin et al. (*38*), where they suggest enhancing a robot's ability to visually differentiate terrains, leading

to safer urban navigation.

Another important requirement is fast perception with a wide field of view. Our HLC relies on a limited field of view of up to 3 meters to the front of the robot. This is inherently limited by using elevation mapping (*30*). Our system's perception capabilities, although effective for the demonstrated scenarios, may present limitations for faster missions or in environments with high uncertainty. Our robot hardware is capable of locomotion up to 6.2 m/s, but we couldn't demonstrate the maximum speed during autonomous deployment due to the delayed and limited mapping. Removing terrain elevation mapping and relying on the fast raw sensory stream would be a promising direction for future improvement.

In conclusion, the presented wheeled-legged robot system demonstrates the potential for achieving robust autonomy in complex and dynamic urban environments using data driven approaches. While challenges remain, such as improving perception capabilities or reducing human labor in map creation, our research paves the way for future advancements in the field of wheeled-legged robots and autonomous urban applications.

Overall, our research contributes to the growing body of knowledge on wheeled-legged robots and autonomous navigation in urban environments. The presented system's robustness, adaptability, and efficiency hold great promise for transforming last-mile delivery and addressing the challenges of urban mobility.

# Materials and Methods

Our main objective, as depicted in Figure 2B, is to develop a robust control system that enables the robot to navigate along a predefined global path consisting of a sequence of waypoints spaced approximately 2 m to 20 m apart. The global path can be generated using a graph planner (*39*) or defined manually. It is important to note that while the global planning aspect is essential for the overall navigation process, it is outside the scope of this work.

Due to space constraints, a comprehensive validation of our method is presented in the supplementary section S4 and S5.

## Overview of the Approach

Inspired by the existing literature (*40, 41*), in which hierarchical decomposition of complex tasks enables faster learning and higher performance, we employ HRL to extend our previous learning-based velocity tracking controller for quadrupedal robots (*16*) to waypoint tracking navigation. In this section, we present an overview of our method, starting with the definition of the hierarchical structure.

### Defining Hierarchy

To tackle the waypoint tracking navigation problem, we adopt the two-level HRL framework inspired by (*42*). Existing literature offers different hierarchical structures, which can be categorized into three distinct approaches:

1. End-to-End: This approach involves a single policy that learns both locomotion and navigation behavior in an end-to-end manner.

2. Latent goal: Some existing works suggest using learned latent sub-goals for HRL (*43, 44*) for simplicity and flexibility. There is no need to explicitly define intermediate goals, and the task assignment within the hierarchy is learned.

3. Explicit goal: The low-level policy handles the locomotion, such as (*16*), and the high-level policy (or a planner) solves navigation by commanding sub-goals. The high-level policy can also define gait parameters (e.g., trot, walk, driving). Locomotion controllers with predefined gait sequences (*9, 13, 45*) require an additional gait command.

We explicitly define sub-goals for practical reasons (option 3). Although the first two approaches may offer simpler implementations, explicitly separating the controllers provides benefits. Firstly, it allows us to develop low-level controllers independently, enhancing the interpretability of the behavior. Secondly, we leverage widely used intermediate representations in legged robotics, such as gait and target velocity. This design choice enables us to reuse the trained low-level policy across various high-level applications, promoting versatility and adaptability in our system.

While our final high-level policy only outputs the velocity command, we also explored commanding gait patterns similarly to Tsounis et al. (*46*). The experiment is described in supplementary section S5.

**Training Procedure**

Our training procedure involves training two policies: a low-level policy and a high-level policy. Here is an overview of the training procedure:

1. Low-level Policy Teacher Training: We begin by training the teacher policy for the low-level locomotion policy. The teacher policy is trained to follow random velocity targets (and optionally gait parameters) on rough terrains using a Proximal Policy Optimization (PPO) algorithm (*47*). In this step, privileged information, including the robot's motion, terrain properties, and noiseless exteroceptive measurements, is utilized to enhance the locomotion performance and convergence of the policy.

2. Low-level Policy Student Training: We then train the student low-level policy, which is deployable on the robot. The student policy observes a sequence of noisy and biased IMU measurements, joint states, and noisy height scans as input, instead of directly observing the privileged information. By imitating the teacher policy and utilizing an RNN-based

architecture (*16*), the student policy learns to extract meaningful features from the temporal data and performs robust locomotion.

3. High-level Policy Training: During high-level policy training, we use the trained student low-level policy as a fixed component and focus on training the high-level policy. The high-level policy is trained using a PPO algorithm. The training data is generated in our custom-built simulation environment that provides feasible paths and dense reward to the learning agent. This approach is further explained in the next section.

4. Optional Fine-tuning: An optional phase of alternating training can be conducted for both policies to enhance their coordination and potentially improve motion smoothness. However, our experiments showed only marginal enhancements from this step. Therefore, we decided to skip the alternating training phase for our final policies.

## Graph-guided Navigation Learning

Navigation graphs, commonly employed in computer games for autonomously navigating AI characters in synthetic environments (*26, 27*), play a crucial role in our navigation learning approach. Inspired by game development, we utilize pre-generated navigation graphs to define initial states, assign feasible paths, and design the reward function during the training of our high-level policy.

### World Generation

The automatic terrain generation method, illustrated in Fig. 7, establishes connectivity between different areas of the terrain (tiles), resulting in a navigation graph that defines viable paths. For example, tiles with stairs in $x$-direction are exclusively connected to floor tiles along the $x$-axis.

To generate diverse and realistic terrain layouts, we utilize the WFC algorithm. The WFC algorithm automatically composes different terrain features, including stairs, floors, and other

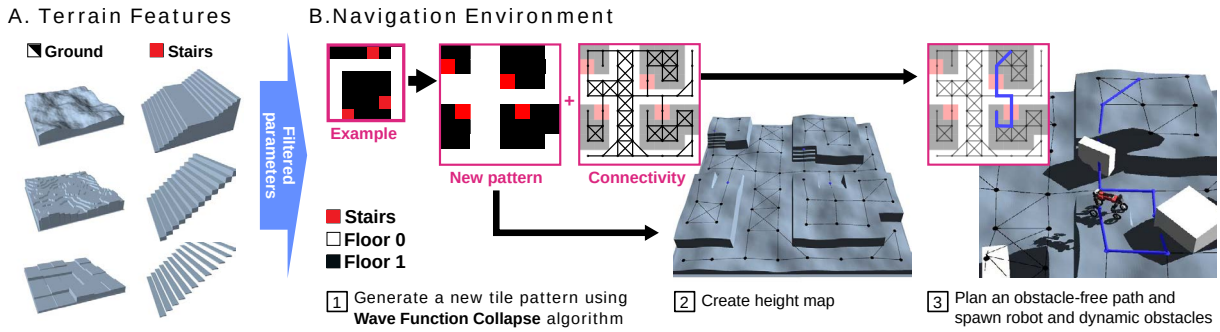


Figure 7: Procedural generation of the navigation world. (A) Filtered parameterized terrain during low-level policy training. (B-1) Generating new tile maps and connectivity graph using the Wave Function Collapse algorithm. (B-2) Created height map terrain with filtered floor features and stair parameters. (B-3) Randomly generated navigation path between two nodes provides waypoints during training. Dynamic obstacles (white boxes) are added randomly.

obstacles. By simultaneously creating connectivity between the tiles, we obtain a navigation graph.

The WFC algorithm divides an input tile map (referred to as ”Example” in Fig. 7B) into smaller chunks and rearranges them to create new N by N patterns. This procedural generation approach enables us to generate a wide variety of navigation worlds with different styles of corridors, rooms, and obstacles.

In our approach, we define three types of tiles: Stair, Floor 0, and Floor 1. We provide their relationship to the WFC algorithm along with example images. The WFC algorithm calculates the probability of each tile type and determines the connectivity to neighboring tile types. By randomly generating tile maps based on these probabilities, we compose the existing terrain features, resulting in varied and realistic training environments. The parameters for the parameterized floor and stairs are selected during the low-level policy training using the terrain filtering algorithm by Lee et al. (*15*) (see supplementary section S6).

### Using Navigation Graphs for RL

We generate random feasible paths for training using the pre-generated navigation graphs (connectivity graph). We employ Dijkstra's algorithm (*31*) to find a path between two randomly selected nodes within the graph. Along the graph edge, we sample waypoints by interpolating between the robot's current position projected onto the path and the subsequent graph node, with a fixed look-ahead distance. The distance is sampled uniformly from [5.0, 20.0] $\mathrm{m}$ every episode. At the end of each path, we include the last node twice as two waypoints. This approach ensures that the agent has clear instructions on the desired trajectory and endpoint.

During the initial training phase, a positive reward is given when the agents successfully navigate along the planned path on the graph. The reward gradually diminishes, and we let the policy train with a sparse reward at the end. The reward function is defined as follows.

$$r_{h,dense} \coloneqq \begin{cases} 1.0 & |e_{wp^1}| < 0.75 \\ clip(v \cdot \widehat{e_{wp^1}}, 0.0, v_{thres})/v_{thres} & \text{otherwise} \end{cases} \tag{1}$$

where $e_{wp^1} = p_{robot} - wp^1$ and $v_{thres} = 0.5$. $p_{robot}$ and $wp^1$ denote the positions of the robot and the nearest waypoint, respectively.

This reward mechanism encourages the robot to follow a planned navigation graph, minimizing the geodesic distance to the final goal. The path entails detours, rather than simply moving towards a waypoint. This approach challenges agents with paths that incorporate tight gaps and sharp turns, thereby pushing their capabilities.

### Dynamic Obstacles

In addition to the static structure generated by the WFC algorithm, we introduce dynamic obstacles during the training. They are randomly placed within the environment and move towards the robot.

The dynamic obstacles are represented by white boxes in Fig. 7B-iii, and their number,

positions, and velocities are randomly generated at the start and during each episode. These obstacles move towards the robot at speeds ranging from 0.1 m/s and 0.5 m/s.

## High-level Policy Details

This section is focused on detailing the Markov Decision Process (MDP) governing $\pi_{hi}$, encompassing observations and actions. Detailed information about the reward function can be found in supplementary section S3.

### Observation

The observation space of $\pi_{hi}$ is defined with three different modalities:

1. Exteroceptive observation: The exteroceptive observation follows the definition by Miki et al. (*16*). We sample height values around the robot from the 2.5D elevation map (*30*). Due to limited memory and computational resources onboard, the robot's field of view is 3 meters to the front and 1.5 meters in other directions. We prioritize shifting the scan pattern towards the front due to the farther perception range afforded by the forward-facing RGB camera. In addition, the exteroceptive observation includes two previous scans taken at 0.1 s and 0.2 s before to account for the dynamic environments.

2. History of position and time information : To aid $\pi_{hi}$ in exploration, we introduce an additional position buffer. We save the visited positions in the world frame at regular intervals of 0.5 meters, along with the corresponding visitation time. The time information includes how many time steps the robot stayed in each position. The most recent 20 positions and their respective time information are provided to the high-level policy in the robot frame.

3. Waypoint and previous actions: In addition to the aforementioned perception information, $\pi_{hi}$ takes the two waypoints, which are extracted from the navigation graph, as input.

Additionally, the policy is given the positions of the two previously observed waypoints and the three previous outputs of $\pi_{hi}$. The previous actions assist the policy in making a smoother command trajectory.

**Exploration Bonus**

During training, we encourage exploration of $\pi_{hi}$ using the explicit position buffer. This is achieved through an exploration bonus incorporated into the reward function. The exploration bonus, denoted as $r_{exp}$, is calculated as the sum of costs $C(s_t, wp_t^1, p_{buf}^i)$ over the positions in the buffer $P_{buf}$.

The cost function $C(p_{robot}, wp^1, p_{buf}^i)$ is defined as

$$C(p_{robot}, wp^1, p_{buf}^i) \coloneqq \begin{cases} 0.0 & |p_{robot} - wp^1| < 0.75 \\ -n_{buf}^i & |p_{robot} - p_{buf}^i| < 1.0 \end{cases} . \tag{2}$$

Here, $p_{robot}$ represents the position of the robot, $wp^1$ denotes the first waypoint, $p_{buf}^i$ represents the $i$-th position saved in the position buffer, and $n_{buf}^i$ corresponds to the number of visits for the $i$-th position in the position buffer.

In essence, if the robot is not close to the first waypoint and is near a position saved in the position buffer, the agent incurs a penalty proportional to the number of timesteps it remains in that position. This penalty encourages the agent to explore new areas and prioritize progress towards the first waypoint.

By incorporating this exploration bonus into the reward function, we promote the exploration behavior of $\pi_{hi}$ during training, enhancing the system's ability to explore and overcome local minima.

**Bounded Action Space**

Instead of the commonly used Gaussian action distribution, we adopt Beta Distribution to represent a bounded action space for the $\pi_{hi}$, as introduced by Chou et al. (*48*). This offers several

benefits. Firstly, it allows us to define hard limits on the outputs, enhancing safety and interpretability. Additionally, working with a bounded action space makes it easier to regularize the motion and control the behavior of the agent.

Specifically, we define the bounds of HLC's commands as follows: $v_x \in [-1.0, 2.0]$ m/s, $v_y \in [-0.75, 0.75]$ m/s, and $\omega_z \in [-1.25, 1.25]$ rad/s. The shift in the $v_x$ range encourages the policy to consistently face forward during locomotion, aligning with the orientation of the RGB camera mounted on the robot. We provide additional details in the supplementary section S8.

**Network Architecture**

For $\pi_{hi}$, we employ a combination of architectures tailored for specific input types. For position history, we utilize 1D Convolutional Neural Network (CNN) layers followed by max pooling, similar to PointNet (*49*), enabling permutation-invariant processing of spatial information. The height scan around the robot is processed using a 3-layer 2D CNN layers followed by an Multi Layer Perceptron (MLP) layer. Other inputs and the output layer are handled by plain MLP layers commonly used for non-spatial data processing and policy generation. For the beta distribution parameters, we use the Sigmoid function at the output layer.

## Low-level Policy Details

In this section, we provide details for low-level teacher and student training. The MDP for the low-level teacher policy inherits from Miki et al. (*16*), with modification to observation and action spaces. The reward function is defined in detail in supplementary section S3, and the details on the privileged training is given in supplementary section S7.

The low-level policy is trained to achieve velocity tracking on rough terrains generated randomly. These terrains, designed following the approach of Miki et al. (*16*), are illustrated in Fig.7A. Each terrain is generated by two to three parameters. During training, we apply the

parameter filtering algorithm by Lee et al. (*15*). Please refer to supplementary section S6 for details. The low-level policy is commanded by linear velocity in the x and y direction, as well as yaw rate. Linear x velocity is uniformly sampled from [-2.5, 2.5] m/s, y velocity from [-1.2, 1.2] m/s, and yaw rate from [-1.5, 1.5] rad/s. In each episode, a new command is sampled, with a 0.005 probability of random resampling.

**Observation**

1. Exteroceptive Observation: For the low-level policy, we sample points around the robot's wheels from a circular pattern, the same as Miki et al. (*16*).

2. Proprioceptive Observation: The proprioceptive observation is utilized exclusively by the student policy. It consists of measurements obtained from body IMU and joint encoders. These measurements provide information about the robot's body acceleration, angular velocities, joint angles, and joint velocities. The student policy processes the sequence of proprioceptive measurements to derive the state of the robot, e.g., pose, velocity, and acceleration. Unlike previous works that relied on estimated pose and twist by a model-based state estimator (*16, 33, 50*), we directly use the IMU measurement consisting of linear acceleration and angular velocity as input to the recurrent student policy.

3. Privileged Observation: A privileged observation is employed during the training of the low-level teacher policy. It encompasses various essential components, including noise-less joint states, foot contact state, terrain normal at each foot, foot contact force, robot velocity, and gravity vector in the robot's base frame (*16*).

4. Velocity Command: 3-dimensional vector consisting of target base horizontal velocity (2) and target base yaw rate (1).

### Action

The low-level policy's action is a 16-dimensional vector consisting of joint position commands (12 joints) and wheel velocity commands (4). The joint position and velocity commands are given to the PD controller of each actuator.For a more detailed explanation of the simulation of the actuators, we refer the readers to supplementary section S2.

In contrast to our prior work (*16*), we discard the use of the CPG in the action space to remove any engineered bias in the motion. For a detailed comparative study of various action spaces, refer to supplementary section S5.

### Network Architecture

The low-level teacher policy is implemented as a plain MLP network, while the low-level student policy adopts a Gated Recurrent Unit (GRU) architecture by Miki et al. (*16*).

# Supplementary materials



All supplementary videos: `https://bit.ly/3qPWWgx`

# References


1. M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, others, Anymal-a highly mobile and dynamic quadrupedal robot, *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 38–44 (IEEE, 2016).

2. M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, others, Cerberus in the darpa subterranean challenge, *Science Robotics* p. eabp9742 (2022).

3. F. Bjelonic, J. Lee, P. Arm, D. Sako, D. Tateo, J. Peters, M. Hutter, Learning-based design and control for quadrupedal robots with parallel-elastic actuators, *IEEE Robotics and Automation Letters* 1611–1618 (2023).

4. L. Wellhausen, M. Hutter, Rough terrain navigation for legged robots using reachability planning and template learning, *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6914–6921 (IEEE, 2021).

5. N. Kashiri, L. Baccelliere, L. Muratore, A. Laurenzi, Z. Ren, E. M. Hoffman, M. Kamedula, G. F. Rigano, J. Malzahn, S. Cordasco, P. Guria, A. Margan, N. G. Tsagarakis, Centauro: A hybrid locomotion and high power resilient manipulation platform, *IEEE Robotics and Automation Letters* 1595–1602 (2019).

6. M. Bjelonic, C. D. Bellicoso, Y. de Viragh, D. Sako, F. D. Tresoldi, F. Jenelten, M. Hutter, Keep rollin'—whole-body motion control and planning for wheeled quadrupedal robots, *IEEE Robotics and Automation Letters* 2116–2123 (2019).

7. V. Klemm, A. Morra, C. Salzmann, F. Tschopp, K. Bodie, L. Gulich, N. Küng, D. Mannhart, C. Pfister, M. Vierneisel, others, Ascento: A two-wheeled jumping robot, *ICRA Int. Conf. on Robotics and Automation (ICRA)*, 7515–7521 (2019).

8. W. Reid, B. Emanuel, B. Chamberlain-Simon, S. Karumanchi, G. Meirion-Griffith, Mobility mode evaluation of a wheel-on-limb rover on glacial ice analogous to europa terrain, *IEEE Aerospace Conference*, 1–9 (2020).

9. M. Bjelonic, R. Grandia, M. Geilinger, O. Harley, V. S. Medeiros, V. Pajovic, E. Jelavic, S. Coros, M. Hutter, Offline motion libraries and online mpc for advanced mobility skills, *The International Journal of Robotics Research* 903–924 (2022).

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

11. M. Geilinger, R. Poranne, R. Desai, B. Thomaszewski, S. Coros, Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels, *ACM Transactions on Graphics (TOG)* p. 160 (2018).

12. M. Hosseini, D. Rodriguez, S. Behnke, State estimation for hybrid locomotion of driving-stepping quadrupeds, *2022 Sixth IEEE International Conference on Robotic Computing (IRC)*, 103–110 (2022).

13. C. D. Bellicoso, F. Jenelten, C. Gehring, M. Hutter, Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots, *IEEE Robotics and Automation Letters* 2261–2268 (2018).

14. F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, M. Hutter, Dynamic locomotion on slippery ground, *IEEE Robotics and Automation Letters* 4170–4176 (2019).

15. J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter, Learning quadrupedal locomotion over challenging terrain, *Science robotics* **5** (2020).

16. T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter, Learning robust perceptive locomotion for quadrupedal robots in the wild, *Science Robotics* p. eabk2822 (2022).

17. W. Xi, Y. Yesilevskiy, C. D. Remy, Selecting gaits for economical locomotion of legged robots, *The International Journal of Robotics Research* 1140–1154 (2016).

18. Y. Yang, T. Zhang, E. Coumans, J. Tan, B. Boots, Fast and efficient locomotion via learned gait transitions, *Conference on Robot Learning*, 773–783 (PMLR, 2022).

19. G. Bellegarda, K. Byl, Trajectory optimization for a wheel-legged system for dynamic maneuvers that allow for wheel slip, *2019 IEEE 58th Conference on Decision and Control (CDC)*, 7776–7781 (IEEE, 2019).

20. L. Wellhausen, M. Hutter, Artplanner: Robust legged robot navigation in the field, *Field Robotics*, 413–434 (2023).

21. J. Frey, D. Hoeller, S. Khattak, M. Hutter, Locomotion policy guided traversability learning using volumetric representations of complex environments, *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5722–5729 (IEEE, 2022).

22. R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, A. Giusti, Learning ground traversability from simulations, *IEEE Robotics and Automation letters* 1695–1702 (2018).

23. V. Vapnik, R. Izmailov, Learning using privileged information: Similarity control and knowledge transfer, *Journal of Machine Learning Research* 2023–2049 (2015).

24. D. Chen, B. Zhou, V. Koltun, P. Krähenbühl, Learning by cheating, *Conference on Robot Learning*, 66–75 (PMLR, 2020).

25. G. Ji, J. Mun, H. Kim, J. Hwangbo, Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion, *IEEE Robotics and Automation Letters* (2022).

26. CryEngine, Ai and navigation system - cryengine 5 documentation, `https://docs.cryengine.com/pages/viewpage.action?pageId=26869983`. [Online; accessed August-2022].

27. Unreal Engine, Navigation system - unreal engine 5 documentation, `https://docs.unrealengine.com/5.0/en-US/navigation-system-in-unreal-engine/`. [Online; accessed August-2022].

28. M. Gumin, Wave function collapse algorithm, `https://github.com/mxgmn/` (2016).

29. J. M. Snider, others, Automatic steering methods for autonomous automobile path tracking, *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08* (2009).

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

31. E. DIJKSTRA, A note on two problems in connexion with graphs, *Numerische Mathematik* 269–271 (1959).

32. E. Jelavic, J. Nubert, M. Hutter, Open3d slam: Point cloud based mapping and localization for education, *Robotic Perception and Mapping: Emerging Techniques, ICRA 2022 Workshop*, p. 24 (ETH Zurich, Robotic Systems Lab, 2022).

33. J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, M. Hutter, Learning agile and dynamic motor skills for legged robots, *Science Robotics* p. eaau5872 (2019).

34. S. Seok, A. Wang, M. Y. Chuah, D. J. Hyun, J. Lee, D. M. Otten, J. H. Lang, S. Kim, Design principles for energy-efficient legged locomotion and implementation on the mit cheetah robot, *Ieee/asme transactions on mechatronics* 1117–1129 (2014).

35. R. B. McGhee, A. A. Frank, On the stability properties of quadruped creeping gaits, *Mathematical Biosciences* 331–351 (1968).

36. J. Lee, J. Hwangbo, M. Hutter, Robust recovery controller for a quadrupedal robot using deep reinforcement learning, *arXiv preprint arXiv:1901.07517* (2019).

37. L. Wellhausen, R. Ranftl, M. Hutter, Safe robot navigation via multi-modal anomaly detection, *IEEE Robotics and Automation Letters* 1326–1333 (2020).

38. M. Sorokin, J. Tan, C. K. Liu, S. Ha, Learning to navigate sidewalks in outdoor environments, *IEEE Robotics and Automation Letters* 3906–3913 (2022).

39. M. Kulkarni, M. Dharmadhikari, M. Tranzatto, S. Zimmermann, V. Reijgwart, P. De Petris, H. Nguyen, N. Khedekar, C. Papachristos, L. Ott, others, Autonomous teamed exploration of subterranean environments using legged and aerial robots, *2022 International Conference on Robotics and Automation (ICRA)*, 3306–3313 (IEEE, 2022).

40. O. Nachum, H. Tang, X. Lu, S. Gu, H. Lee, S. Levine, Why does hierarchy (sometimes) work so well in reinforcement learning?, *arXiv preprint arXiv:1909.10618* (2019).

41. D. Jain, K. Caluwaerts, A. Iscen, From pixels to legs: Hierarchical learning of quadruped locomotion, *Proceedings of the 2020 Conference on Robot Learning*, J. Kober, F. Ramos, C. Tomlin, eds., 91–102 (PMLR, 2021).

42. O. Nachum, S. S. Gu, H. Lee, S. Levine, Data-efficient hierarchical reinforcement learning, *Advances in neural information processing systems* **31** (2018).

43. A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, K. Kavukcuoglu, Feudal networks for hierarchical reinforcement learning, *International Conference on Machine Learning*, 3540–3549 (PMLR, 2017).

44. D. Jain, A. Iscen, K. Caluwaerts, Hierarchical reinforcement learning for quadruped locomotion, *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7551–7557 (IEEE, 2019).

45. S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, I. Havoutis, Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control, *IEEE Transactions on Robotics* (2022).

46. V. Tsounis, M. Alge, J. Lee, F. Farshidian, M. Hutter, Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning, *IEEE Robotics and Automation Letters* 3699–3706 (2020).

47. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347* (2017).

48. P.-W. Chou, D. Maturana, S. Scherer, Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution, *International conference on machine learning*, 834–843 (PMLR, 2017).

49. C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660 (2017).

50. N. Rudin, D. Hoeller, P. Reist, M. Hutter, Learning to walk in minutes using massively parallel deep reinforcement learning, *Conference on Robot Learning*, 91–100 (PMLR, 2022).

51. A. A. Hagberg, D. A. Schult, P. J. Swart, Exploring network structure, dynamics, and function using networkx, *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, J. Millman, eds., 11 – 15 (Pasadena, CA USA, 2008).

52. M. Mueller, A. Dosovitskiy, B. Ghanem, V. Koltun, Driving policy transfer via modularity and abstraction, *Proceedings of The 2nd Conference on Robot Learning*, A. Billard, A. Dragan, J. Peters, J. Morimoto, eds., 1–15 (PMLR, 2018).

53. A. Agarwal, A. Kumar, J. Malik, D. Pathak, Legged locomotion in challenging terrains using egocentric vision, *Conference on Robot Learning*, 403–415 (PMLR, 2023).

54. G. Kahn, P. Abbeel, S. Levine, Badgr: An autonomous self-supervised learning-based navigation system, *IEEE Robotics and Automation Letters* 1312–1319 (2021).

55. G. Kahn, P. Abbeel, S. Levine, Land: Learning to navigate from disengagements, *IEEE Robotics and Automation Letters* 1872–1879 (2021).

56. Y. Kim, C. Kim, J. Hwangbo, Learning forward dynamics model and informed trajectory sampler for safe quadruped navigation, *Conference on Robotics-Science and Systems (RSS)* (Robotics: Science and Systems Foundation, 2022).

57. J. Truong, D. Yarats, T. Li, F. Meier, S. Chernova, D. Batra, A. Rai, Learning navigation skills for legged robots with learned robot embeddings, *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 484–491 (IEEE, 2021).

58. D. Hoeller, L. Wellhausen, F. Farshidian, M. Hutter, Learning a state representation and navigation in cluttered and dynamic environments, *IEEE Robotics and Automation Letters* 5081–5088 (2021).

59. M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, J. Nieto, Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations, *IEEE Robotics and Automation Letters* 4423–4430 (2018).

60. T. Manderson, J. C. G. Higuera, S. Wapnick, J. Tremblay, F. Shkurti, D. Meger, G. Dudek, Vision-based goal-conditioned policies for underwater navigation in the presence of obstacles, *Conference on Robotics-Science and Systems (RSS)* (Robotics: Science and Systems Foundation, 2020).

61. H. Wang, S. Chen, S. Sun, Diffusion model-augmented behavioral cloning, *CoRR* **abs/2302.13335** (2023).

62. N. Savinov, A. Dosovitskiy, V. Koltun, Semi-parametric topological memory for navigation, *International Conference on Learning Representations* (2018).

63. Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, D. Pathak, Coupling vision and proprioception for navigation of legged robots, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 17273–17283 (2022).

64. E. Wijmans, A. Kadian, A. S. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, D. Batra, Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames, *International Conference on Learning Representations* (2019).

65. J. Choi, K. Park, M. Kim, S. Seok, Deep reinforcement learning of navigation in a complex and crowded environment with a limited field of view, *2019 International Conference on Robotics and Automation (ICRA)*, 5993–6000 (IEEE, 2019).

66. E. Wijmans, M. Savva, I. Essa, S. Lee, A. S. Morcos, D. Batra, Emergence of maps in the memories of blind navigation agents, *International Conference on Learning Representations* (2023).

67. K. Zhu, T. Zhang, Deep reinforcement learning based mobile robot navigation: A review, *Tsinghua Science and Technology* 674–691 (2021).

68. H. Surmann, C. Jestel, R. Marchel, F. Musberg, H. Elhadj, M. Ardani, Deep reinforcement learning for real autonomous mobile robot navigation in indoor environments, *arXiv preprint arXiv:2005.13857* (2020).

69. R. Yang, M. Zhang, N. Hansen, H. Xu, X. Wang, Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers, *Deep RL Workshop NeurIPS 2021* (2021).

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

71. P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, others, On evaluation of embodied navigation agents, *CoRR* **abs/1807.06757** (2018).

72. H. Kolvenbach, D. Bellicoso, F. Jenelten, L. Wellhausen, M. Hutter, Efficient gait selection for quadrupedal robots on the moon and mars, *14th International Symposium on Artificial*

*Intelligence, Robotics and Automation in Space (i-SAIRAS 2018)* (ESA Conference Bureau, 2018).

73. P. N. Ward, A. Smofsky, A. J. Bose, Improving exploration in soft-actor-critic with normalizing flows policies, *arXiv preprint arXiv:1906.02771* (2019).

74. W. Zhou, S. Bajracharya, D. Held, Plas: Latent action space for offline reinforcement learning, *Conference on Robot Learning*, 1719–1735 (PMLR, 2021).

75. A. Allshire, R. Martín-Martín, C. Lin, S. Manuel, S. Savarese, A. Garg, Laser: Learning a latent action space for efficient reinforcement learning, *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 6650–6656 (IEEE, 2021).

76. D. P. Kingma, M. Welling, Auto-encoding variational bayes, *2nd International Conference on Learning Representations* (2014).

77. L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using real nvp, *International Conference on Learning Representations* (2016).

78. J. C. Brant, K. O. Stanley, Minimal criterion coevolution: a new approach to open-ended search, *Proceedings of the Genetic and Evolutionary Computation Conference*, 67–74 (2017).

79. O. E. L. Team, A. Stooke, A. Mahajan, C. Barros, C. Deck, J. Bauer, J. Sygnowski, M. Trebacz, M. Jaderberg, M. Mathieu, N. McAleese, N. Bradley-Schmieg, N. Wong, N. Porcel, R. Raileanu, S. Hughes-Fitt, V. Dalibard, W. M. Czarnecki, Open-ended learning leads to generally capable agents, *CoRR* **abs/2107.12808** (2021).

80. A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, D. Scaramuzza, Learning high-speed flight in the wild, *Science Robotics* p. eabg5810 (2021).

81. S. Ross, G. Gordon, D. Bagnell, A reduction of imitation learning and structured prediction to no-regret online learning, *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 627–635 (JMLR Workshop and Conference Proceedings, 2011).

# Acknowledgments

**Funding**: This work was supported by the Mobility Initiative grant funded through the ETH Zurich Foundation, European Union's Horizon 2020 research and innovation programme under grant agreement No 101070405, European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme grant agreement No 852044, Swiss National Science Foundation through the National Centre of Competence in Digital Fabrication (NCCR dfab), European Union's Horizon Europe Framework Programme under grant agreement No 101070596 and Apple Inc. **Author contribution**: J.L. conceived the main idea for the approach and was responsible for the implementation and training of the controllers. The high-level policy was trained by J.L. while M.B. developed the simulation environment for the low-level policy and also trained it. The navigation system and safety layer were collaboratively devised and implemented by J.L., M.B., A.R., and L.W. Real-world experiments were planned and executed with contributions from A.R. and L.W. Initial setup of the low-level controller was facilitated by T.M. All authors played a significant role in system integration and experimentation. **Other contributors**: We appreciate Turcan Tuna for helping us integrating Open3D SLAM, Julian Keller for the API integration, and Giorgio Valsecchi for the hardware support. **Conflict of interest**: The authors declare that they have no competing interests. **Data and materials availability**: All (other) data needed to evaluate the conclusions in the paper are present in the paper or the Supplementary Materials.

# Supplementary Materials

## S1. Nomenclature

| | |
|---|---|
| $\hat{(\cdot)}$ | normalized vector |
| $(\cdot)_{des}$ | desired quantity |
| $v$ | linear velocity of the robot in world frame |
| $v^B$ | linear velocity of the robot in $B$ frame |
| $p_A$ | position of $A$ in world frame |
| $wp^1$ | first waypoint position |
| $wp^2$ | second waypoint position |
| $P_{buf}$ | the position buffer |
| $p^i_{buf}$ | the $i$-th position saved in the position buffer |
| $n^i_{buf}$ | The number of visits for the $i$-th position saved in the position buffer |
| $\omega$ | angular velocity |
| $\tau$ | joint torque |
| $q$ | joint position |
| $\psi$ | yaw angle |
| $r_f$ | linear position of a foot |
| $e_g$ | gravity vector |
| $I_{c,body}$ | index set of body contacts |
| $I_{c,wheel}$ | index set of wheel contacts |
| $\|\cdot\|$ | cardinality of a set or $l_1$ norm |
| $\|\|\cdot\|\|$ | $l_2$ norm |

## S2. Implementation Details

In this section, we add some implementation details that are necessary to enable kilometer-scale autonomous deployments and a successful sim-to-real transfer.

### Localization

Using Open3D SLAM (*32*), we localize based on the data of the Velodyne VLP-16 LiDAR mounted on top of the robot in a known point cloud map generated with the Leica BLK2GO reality capture device. Notably, Open3D SLAM only uses well-known algorithms, such as Iterative Closest Point (ICP), in their core form while using Open3D as a powerful backend for 3D data processing. For this work, we adapted Open3D SLAM to use odometry from IMU and joint encoder data as prior for scan-to-map matching. We expect this high-frequency odometry source to contribute to the systems robustness because scan-to-scan matching might fail at higher speeds or in degenerate environments.

### Global Planning

For global navigation, we also rely on well-established methods and libraries. Leveraging the reality capture data, we manually design a sparse navigation graph offline as shown in Fig. 3. During deployment we compute the shortest path from the robot's current position to the goal position with Dijkstra using the networkX (*51*) Python library.

### Waypoint Selection

The global path obtained from the navigation graph can contain nodes that are tens of meters apart while the high-level policy was trained with waypoints that are at most 10 m apart. To resolve this issue, we introduce a simple waypoint selection method called "anchor pursuit", inspired by the "pure pursuit" path following algorithm. If the next path nodes is less than 3 m from the current robot position, we select it as a waypoint. In case it is farther, we instead project

the robot position onto the path and select a waypoint with a 3 m lookahead distance. This ensures that the nodes or so-called anchor points are always approached and the robot doesn't take undesired shortcuts around these core waypoints of the global path. Also note that this is different from interpolating the global path at fixed distances and sequentially approaching these sub-waypoints since with "anchor pursuit" the sub-waypoints are moving forward with the robot. This allows for greater freedom in circumnavigating obstacles, since interpolated waypoints do not need to be followed exactly. In future work, one could add exteroceptive information, e.g. semantically segmented images (*52*), to generate more sophisticated refined paths for the hierarchical controller or even incorporate this capability into the high-level policy itself.

**Local Terrain Mapping**

To obtain the extereoceptive observations on real hardware, we use a GPU-accelerated geometric terrain mapping approach (*30*), which provides a local elevation map around the robot based on the data of the two Robosense RS-Bpearl dome LiDARs mounted at the front and rear of the robot. The points for the foot scan for the locomotion policy and the base scan for the navigation policy are extracted from the elevation map. Removing the local terrain mapping and directly providing the raw extereoceptive data, e.g. via depth images (*53*) or point clouds, is subject to future work and can help to reduce processing delays.

**Human Detection**

For human detection, we use the Stereolabs ZED 2i stereo camera. Their SDK provides a "Spatial Object Detection" feature, which detects humans and provides an estimate of the human's position in the camera frame. As shown in Fig. 4B, this position is used to augment the foot and base scans with a safety margin around the human.

### Modeling Actuators

For successful sim-to-real transfer, it is crucial to simulate the dynamics of the joint actuators. The non-linear characteristics of robotic actuators such as joint friction, delay, and backlash are very difficult to model with simple analytic models. Instead of modeling the whole actuator, we used neural network models to simulate the complex dynamics efficiently.

The joint actuators are fully simulated by an actuator network (*33*), which is trained with accurate torque measurements from Series Elastic Actuator (SEA)s. Wheel actuators are pseudo-direct drives and we do not have access to accurate torque measurements. Instead, we learn the mapping from velocity command and a history of past velocity readings to the motor current, i.e., $I_t = f(\dot{\phi}_{target}|\dot{\phi}_{t-1}, \dot{\phi}_{t-2}, \cdots)$, using a neural network. Then, the torque is computed by $\tau_t = K_\tau * GR * I_t$ where $K_\tau$ is the torque constant and $GR$ is the gear ratio. Additionally, we simulated joint friction such that $\tau_t = K_\tau * GR * I_t + \tau_{friction}$. The friction is modeled by two terms: Coulomb friction $\tau_{friction,C} = -C_!\dot{\phi}$ and the stick friction $\tau_{friction,S} = -C_2 sgn(\dot{\phi})$ with constants $C_1$ and $C_2$. $C_1$ and $C_2$ are randomized and included in the privileged observation.

## S3. Reward Functions

In this section, we provide a detailed explanation on the reward function for each agent. We categorize the reward functions into three groups:

- High-level reward ($r_h$): high-level objectives such as waypoint or goal position tracking.
- Low-level reward ($r_l$): achieving the commands given by the $\pi_{hi}$.
- Regularization reward ($r_r$): constraint-related objectives and regularization terms.

The main objective of the $\pi_{hi}$ is defined by $r_h$, and the $\pi_{lo}$ focuses on the low-level control (e.g., pose control, balancing, and locomotion control) by maximizing the discounted sum of

$r_l$. $r_r$ defines additional sub-objectives such as joint velocity penalty, torque minimization, or action smoothness. Such regularization objectives are often introduced for robotic applications to avoid damaging hardware or to facilitate sim-to-real transfer. In RL, it is done by adding $r_r$ to the reward functions (*15, 25, 50*).

The low-level policy is trained using $r_l + r_r$, and the high-level policy is trained using $r_h + w_l \cdot (r_l + r_r)$ with a constant scale $w_l$. We chose the value of $w_l$ such that the expected sum of $r_h$ and $r_l + r_r$ are at a similar magnitude. This makes $r_h$ generate smooth and low-effort trajectories that respect the capability of the low-level policy.

**High-level Rewards**

The high-level reward ($r_h$) is defined as a linear combination of functions below.

- *Goal reaching.* The basic sparse reward for reaching the first waypoint ($wp^1$):

$$r_{h,goal} \coloneqq \begin{cases} 1.0 & |p_{robot} - wp^1| < 0.75 \\ 0.0 & \text{otherwise} \end{cases}. \tag{3}$$

- *Dense tracking reward* Explained in the main text:

$$r_{h,dense} \coloneqq \begin{cases} 1.0 & |e_{wp^1}| < 0.75 \\ clip(v \cdot \widehat{e_{wp^1}}, 0.0, v_{thres})/v_{thres} & \text{otherwise} \end{cases} \tag{4}$$

  where $e_{wp^1} = p_{robot} - wp^1$ and $v_{thres} = 0.5$.

- *Exploration bonus.* Explained in the main text:

$$r_{h,exp} \coloneqq \sum_{P_{buf}} C(s_t, wp_t^1, p_{buf}^i), \tag{5}$$

  where

$$C(p_{robot}, wp^1, p_{buf}^i) \coloneqq \begin{cases} 0.0 & |p_{robot} - wp^1| < 0.75 \\ -n_{buf}^i & |p_{robot} - p_{buf}^i| < 1.0 \end{cases}. \tag{6}$$

- *Near-goal stability.* This reward motivates the robot to stay still near the goal point. This reward is a part of regularization reward, but this is only active during the high-level policy training:

$$r_{h,stability} \coloneqq \begin{cases} \exp\left(-2.0||v||^2\right) & |p_{robot} - wp^1| < 0.75 \\ 0.0 & \text{otherwise} \end{cases}. \tag{7}$$

**Low-level Rewards**

The low-level reward ($r_l$) is modified from the reward terms by Miki et al. (*16*). $r_l$ is defined with the linear combination of the following terms:

- *Linear velocity.* This term encourages the policy to follow a desired horizontal velocity (velocity in $xy$ plane) command:

$$r_{lv} \coloneqq \begin{cases} 2.0\exp(-2.0 \cdot ||v_{xy}^{body}||^2), & \text{if } |v_{des}| < 0.05 \\ \exp(-2.0||v_{xy}^{body} - v_{des}||^2) + v_{des} \cdot v_{xy}^{body}, & \text{otherwise} \end{cases}, \tag{8}$$

  where $v_{des} \in \mathbb{R}^2$ is the desired horizontal velocity.

- *Angular velocity.* This term encourages the policy to follow a desired yaw velocity command:

$$r_{av} \coloneqq \exp(-2.0(\omega_z^{body} - \omega_{des})^2), \tag{9}$$

- *Body motion.* This term penalizes the body velocity in directions not part of the command:

$$r_{bm} \coloneqq -1.25(v_z^{body})^2 - 0.4|\omega_x^{body}| - 0.4|\omega_y^B|. \tag{10}$$

- *Body orientation.* This reward penalizes the angle between the $z$-axis of the world and the $z$-axis of the robot's body:

$$r_{ori} = \arccos(R_b(3,3))^2, \tag{11}$$

where $R_b(3,3)$ is the last element of the rotation matrix representation of the body orientation.

- *Body height.* This reward motivates the policy to keep the height of the robot's base above the ground ($h_{base}$) around 0.55 m with the tolerance of 0.05 m:

$$r_h = max(0.0, |h_{base} - 0.55| - 0.05). \tag{12}$$

**Regularization Rewards**

- *Torque penalty.* We penalize the joint torques to prevent damaging joint actuators during deployment and to reduce energy consumption ($\tau \propto$ electric current):

$$r_\tau \coloneqq -\sum_{i \in joints} ||\tau_i||^2. \tag{13}$$

- *Joint motion.* This term penalizes joint velocity and acceleration to avoid vibrations:

$$r_s = -c_k \sum_{i=1}^{12} (\dot{q_i}^2 + 0.01\ddot{q_i}^2), \tag{14}$$

where $\dot{q_i}$ and $\ddot{q_i}$ are the joint velocity and acceleration, respectively.

- *Target smoothness.* The magnitude of the first and second order finite difference derivatives of the target joint positions are penalized such that the generated joint trajectories become smoother:

$$r_s = -c_k \sum_{i=1}^{12} ((q_{i,t,des} - q_{i,t-1,des})^2 + (q_{i,t,des} - 2q_{i,t-1,des} + q_{i,t-2,des})^2), \tag{15}$$

where $q_{i,t,des}$ is the joint target position of joint $i$ at time step $t$.

- *Joint position constraint.* This term introduces a soft constraint in the joint space. To avoid the knee joint flipping in the opposite direction, we give a penalty for exceeding a

threshold:

$$r_{jc,i} = \begin{cases} -(q_i - q_{i,th})^2, & \text{if } q_i > q_{i,th} \\ 0.0 & \text{otherwise} \end{cases}, \tag{16}$$

$$r_{jc} = \sum_{i=1}^{12} r_{jc,i}, \tag{17}$$

where $q_{i,th}$ is a threshold value for the $i$th joint. We only set thresholds for the knee joint.

- *Bad contact penalty.* Contacts with the environment are penalized except for the wheels:

$$r_{bc} \coloneqq -|I_{c,body} \backslash I_{c,wheel}|. \tag{18}$$

- *Survival bonus.* Not terminating is densely rewarded:

$$r_{h,surv} \coloneqq 1.0 \quad \text{while not terminated.} \tag{19}$$

## S4. Comparison to Related Works and Validation of Our Method

In this section, we present a survey of relevant research concerning the development of navigation policies for mobile robots using RL. Focused on the works based on model-free RL, we extract essential design choices and conduct a comparative analysis with our approach.

It's noteworthy that alternative methodologies also exist, including those based on offline RL or model-based RL (*54–56*). For instance, Kahn et al. (*54, 55*) showed outdoor navigation using an offline-trained dynamics model combined with a sampling-based planner. However, we maintain our focus on the model-free approach and local navigation setup (up to 20 m distance to goal) because we aim to develop a highly responsive control policy with a high control rate and minimal onboard planning as described in the introduction.

### Key Design Choices in Existing Literature

Upon reviewing the existing literature, we have identified several critical factors that contribute to enhancing the navigation performance of a learned agent. These key design choices observed

in prior research include:

1. Modularity and Abstraction: Navigation problems are often addressed by decomposing them into sub-problems and then tackling each using specialized sub-modules. This modular approach, exemplified by hierarchical control systems, streamlines the developmental process as evidenced in existing studies (*41, 44, 57*). In the context of HRL, separating low-level locomotion and high-level navigation on different time scales benefits exploration and performance (*40, 44*), with higher-level agents operating at a slower frequency. The temporal abstraction by design enhances exploration and improves final performance in some cases. Furthermore, integrating pre-trained perception modules has shown to enhance the navigation proficiency in complex settings, as demonstrated in the investigations carried out by Müller et al. (*52*) and Hoeller et al. (*58*).

   Our approach aligns with this paradigm. HLC operates at a slower frequency and utilizes the pre-trained RNN encoder from the LLC as a state representation. Our experiment below shows the importance of the former in navigation performance, while the latter plays a crucial role in robust robot deployment, especially on challenging terrains (depicted in supplementary video S4.).

2. Utilization of Expert Demonstration: Many existing works make use of expert demonstration, often sourced from either executing a sampling-based planner (*59*) or human demonstrations (*60, 61*). Imitation learning is a widely adopted approach for autonomous driving and navigation domains, offering accelerated learning and enhanced performance.

   In our case, while we could generate expert demonstrations using a sampling-based planner (*20*), we refrained from this approach due to its significant computational overhead. Instead, we opted to leverage pre-generated paths from our simulation environment and trained a high-level policy to adhere to them via path sampling and dense reward.

3. Memory: Memory, whether explicit or implicit, plays a significant role in point-goal navigation. Literature frequently incorporates explicit memory mechanisms (*62, 63*) or employs RNN architectures to address this need (*58, 64, 65*). Extensive analysis by Wijmans et al. (*66*) highlights the contribution of memory in successfully accomplishing navigation tasks.

   Similarly, we incorporate memory, albeit in a simplified manner. Tailoring our approach to real-world requirements, we designed our formulation to employ simple models (e.g., MLP or shallow CNN) and interpretable states. Rather than depending on generic, navigation-agnostic RNN structures, we integrate explicit information about visited positions and times.

4. Dense Reward: Dense reward-shaping is a prevalent strategy due to the inherent difficulty in training sparse reward formulations. Notably, dense reward functions are frequently employed to incentivize policy progression or to penalize collision occurrences (*41, 67, 68*). Some works define the dense reward functions based on the geodesic distance, which is the shortest obstacle-free path to the goal (*57, 59, 64*).

   In our approach, we leverage dense rewards only during the initial training phase. This is because the dense rewards based on the shortest obstacle-free path do not account for dynamic obstacles or intricate environmental dynamics such as varying friction and disturbances.

While it's not mentioned above, similarly to locomotion research, the sim-to-real approach is widely adopted. Using large amounts of synthetic data improves training efficiency and robustness, owing to the large amount of data collected from diverse simulated scenarios.

We have integrated the above principles from the literature into our approach. Our HRL formulation follows the task decomposition appearing in the literature, and our simulation en-

vironment is designed based on the insights from points 2 to 4.

**Validation of Our Approach**

We proceed to conduct a comparative analysis between our approach and the baselines defined below. The first two baselines aim to validate the effectiveness of our graph-guided navigation learning approach, whereas the subsequent baselines are derived from existing literature. These subsequent baselines are meant to isolate and identify the distinct contributions of each component within our approach.

1. Without path sampling: This baseline evaluates the advantage of using pre-generated, obstacle-free paths to generate waypoints during training. While it is trained in an environment identical to ours, the goals are uniformly distributed across the terrain without accounting for obstacles or ensuring a feasible path.

2. Without WFC Features: This baseline evaluates the importance of providing various navigation challenges during the training. Instead of using terrain features generated by WFC, this baseline is trained over rough terrains with randomly placed obstacles.

3. Memoryless: This baseline does not incorporate position history, aligning it with the reactive policies presented by Jain et al. (*41*) and Pfeiffer et al. (*59*). It is trained in the same environment as ours.

4. No temporal abstraction: This baseline evaluates the importance of temporal abstraction for the navigation task. We train a high-level policy with the same control frequency as the low-level locomotion policy (50 Hz).

5. End-to-end: This baseline assesses the advantages of the hierarchical decomposition of the task. It is a single policy directly outputting joint control commands, trained to pursue

| Policies | SPL (Success Rate) by Path Length | |
|---|---|---|
| | 5 to 10 m | 10 to 20 m |
| **Ours** | **0.897 (0.901)** | **0.689 (0.763)** |
| Baseline 1. Without path sampling | 0.858 (0.840) | 0.497 (0.559) |
| Baseline 2. Without WFC features | 0.865 (0.871) | 0.302 (0.305) |
| Baseline 3. Memoryless | 0.873 (0.897) | 0.526 (0.573) |
| Baseline 4. No temporal abstraction | 0.798 (0.823) | 0.370 (0.397) |
| Baseline 5. End-to-end | 0.304 (0.318) | 0.045 (0.046) |

Table S1: Performance comparison between different navigation policies. Values in the parenthesis indicate success rates. Each value represents an average taken from 1000 randomly generated terrains and paths.

waypoints. The policy is trained with $r_h + r_l$, without the velocity tracking rewards. This formulation is in line with the work of Yang et al. (*69*) and Rudin et al. (*70*).

We conduct evaluations based on the methodology proposed by Anderson et al. (*71*), employing the Success weighted by Path Length (SPL). The SPL is given by:

$$\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{\max(p_i, l_i)}$$

where $l_i$ corresponds to the shortest path distance between the starting position and the goal in episode $i$, $p_i$ represents the actual path length traversed by the robot, and $S_i$ is a binary indicator denoting the success of episode $i$. In our experiment, we compute $l_i$ as the shortest distance on the navigation graph.

We evaluated the SPL and success rate for different path lengths ($l_i$), as shown in Table S1. This was done using 1,000 randomly generated terrains, with randomly sampled paths ranging from 5 to 20 m. A waypoint is given at the path's endpoint. An episode is considered successful when the robot approaches the waypoint within 50 cm in 60 s.

We begin by validating our graph-guided navigation learning approach with baselines 1 and 2. Both baselines show degraded performance for the distant goals. When a policy is trained to track arbitrary goals (baseline 1), we observed that the policy becomes overly conservative

with distant goals. This is due to the high occurrence of infeasible goals, leading to increased failures during training. Without WFC-generated terrain features during training (baseline 2), the randomly generated environment fails to provide a policy with diverse challenges like tight spaces and transitioning terrains. Consequently, baseline 2 exhibits limited performance when navigating complex environments. Results from these baselines underscore the importance of a well-structured training environment that offers quality training data.

We then evaluate each component of our navigation policy. As given in Table S1, our approach shows higher SPL and success rates compared to the ablated baselines. When the memory is removed, the baseline 3 showed approximately 16 % lower SPL for the distant goals (beyond 10 m). The baseline 3 still shows more than 50 % success thanks to its explorative behavior, depicted in Fig. 6D-ii. The limited impact on performance can be attributed to our focus on local navigation. However, in scenarios that demand extensive memory capabilities, as demonstrated in (*64*), this component becomes more important. Baselines 4 and 5 results show the importance of the hierarchical decomposition of the problem. Baseline 4, lacking temporal abstraction, and baseline 5, omitting problem sub-division (modularity), both encounter increased difficulty in solving complex navigation tasks with distant goals. Regarding baseline 5, the need to address rough-terrain locomotion and point-goal navigation within a single MDP introduces challenges in terms of reward shaping. To achieve better results, extra engineering efforts would be necessary.

In conclusion, we have validated our approach across different training environments and MDP formulations from existing literature. The analysis shows that each individual component is important in enhancing navigation performance. Our approach effectively incorporates key concepts from existing literature. In particular, our terrain generation and learning strategy significantly contribute to the final performance.

## S5. Comparison of Different Architectures

In this section, we provide our experiment with different policies. We explored three different approaches to learning gait-switching behavior:

- No Explicit Gait Design (Ours1): High-level policy commands velocity target and low-level policy outputs joint position target and wheel velocity target directly. No CPG is used. No stylistic rewards such as the air-time reward by (*25, 50*) or foot clearance reward by (*15*).

- Hierarchical Gait Selection (Ours2): High-level policy outputs desired foot contact states per foot (either 1 or 0) in addition to the velocity command. Low-level policy learns gait following and velocity tracking. The low-level policy is first trained with known gait patterns, e.g., trot, pace, and static walk, and then trained alternatingly with the high-level policy.

  For the low-level policy, we introduced an additional gait-tracking reward, defined as

  $$r_{gait} \coloneqq 0.1 \cdot \sum_{i \in 0,1,2,3} \mathbb{1}(fc(i) = fc(i)_g), \tag{20}$$

  where $fc_(i)$ denotes the desired contact state of the $i$-th foot and $fc_(i)_g$ is the target contact state given by the high-level policy.

- Baseline (*15, 16*): This is an end-to-end approach, with single policy. Trained with $r_h + r_r$. We used the same action space as Miki et al. (*16*). The gait frequency and duty factor are fixed to 1.0 Hz and 0.5, respectively, and the initial phases are randomized.

We conduct an experiment where the robot is commanded toward a waypoint behind a $1\,\mathrm{m} \times 1\,\mathrm{m}$ obstacle. The experimental setup and motion sequences are shown in Fig.S1.

Hierarchical controllers only step when faced with the obstacle (Fig.S1C and Fig.S1D). Ours1 and Ours2 show similar behaviors with different gait frequencies and timing. On the

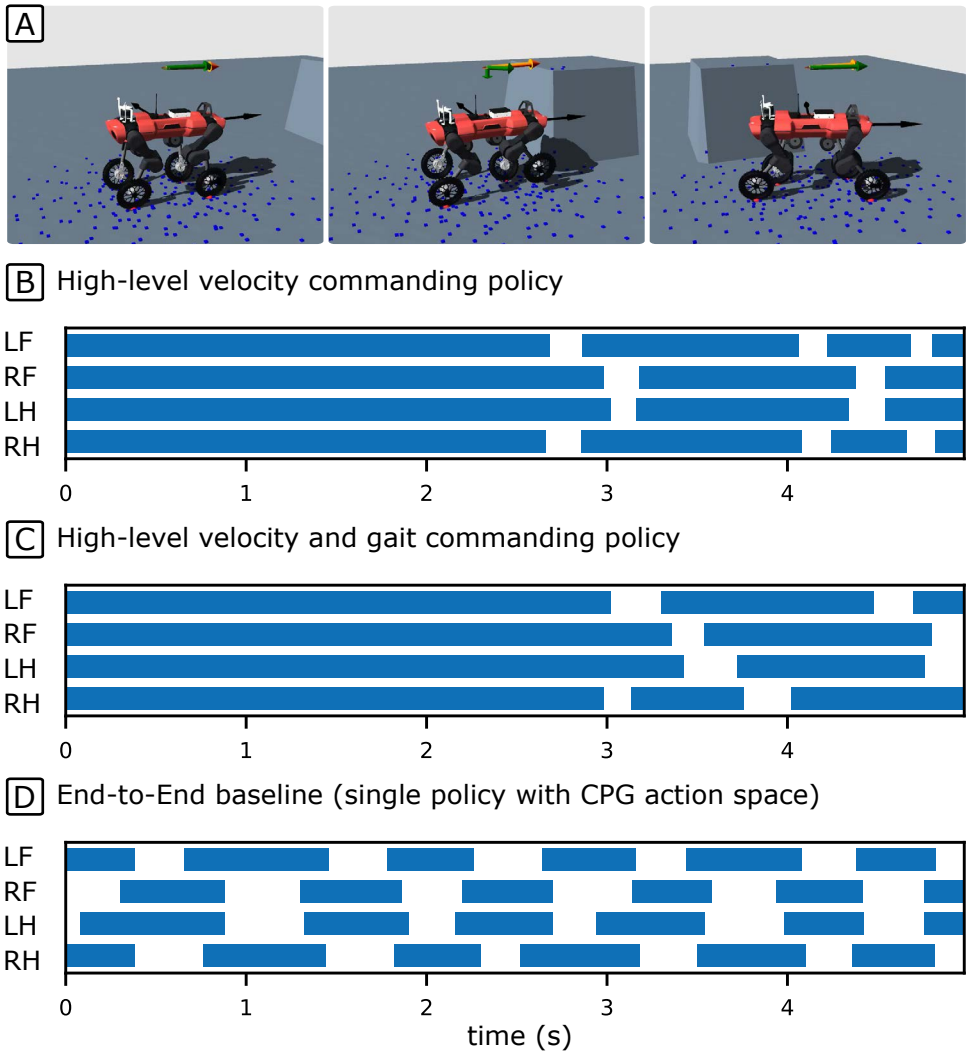


Figure S1: Different controllers during collision avoidance. On flat terrain with an obstacle, the waypoint is given behind the obstacle. (A) Motion sequence of our controller. (B-D) Foot contact sequences of different approaches. The robot faces the obstacle at around 2.0 s.

other hand, the baseline shows regular stepping and can adapt gait frequency, but keeps stepping even when it is not necessary. This is mainly due to the fixed CPG that limits the exploration of different walking patterns. In (*15*) and (*16*), the gait frequency is manually set to 0 when no stepping is desired.

Our final controller follows approach 1 for simplicity, but both resulted in similar performance and gait-switching behavior.

The second approach follows the traditional separation of locomotion and gait planning (*9, 13, 45*). The modular design simplifies the locomotion control problem with a fixed gait and allows for individual gait analysis (*72*). To learn gait patterns, we use a learned action space that maps the output of the high-level policy, which is defined as Beta distribution, to a distribution of gait parameters (*73–75*). The generative model is trained with known gait parameters (*6, 13*).

**Learned Action Space for Gait-generating High-level Policy**

For the gait commanding high-level policy, we had to implement a special action space. Exploring the space of gait parameters with the commonly used Gaussian distribution can be inefficient because not all the real-valued vectors can represent feasible gaits, and the feasible parameters can be sparsely distributed. To improve exploration and accelerate learning, we use a learned gait generator as the action space of the high-level policy.

Existing works have proposed using generative models such as Variational Autoencoder (VAE)s (*74, 75*) or a normalizing flow (*73*) to transform the action distribution into a different, possibly multi-modal, distribution. Wenxuan et al. (*74*) and Allshire et al. (*75*) proposed to pre-train generative models with existing motion data for higher sample efficiency.

Similarly, we construct a learned latent action space with a RealNVP model (*73*) that generates gait patterns from a Beta distribution. We chose RealNVP instead of VAE (*76*) because the RealNVP can be updated during the RL update by policy gradient thanks to its invertibil-

ity (*73, 77*).

We construct a stochastic policy $\pi(a|s)$ by two neural network modules in series. Firstly, an MLP outputs parameters for the Beta distribution that serves as a base distribution. Then follows an invertible normalizing flow layer to get $a = f_\psi(z)$, where $z \sim \mathcal{N}(\mu_\theta(s), \sigma_\theta(s))$. $f_\psi$ denotes a RealNVP. We can directly use the RealNVP policy instead of Gaussian policies within RL algorithms since it is possible to compute the log-likelihood of the action by

$$\log\big(\pi(a|s)\big) = \log\big(p_z(f_\psi^{-1}(a))\big) + \log\left(\left|\det\left(\frac{\partial f_\psi^{-1}(a)}{\partial a^T}\right)\right|\right). \tag{21}$$

The RealNVP layers are pre-trained to generate gait parameters from a uniform distribution. It is trained by minimizing the log-likelihood:

$$\mathbb{E}_x\left\{-\log\big(p_z(f_\psi^{-1}(x))\big) - \log\big(\,\big|\det\big(\partial f_\psi^{-1}(x)/\partial x^T\big)\big|\big)\right\}, \tag{22}$$

where $x$ is sampled uniformly from known gait parameters.

## S6. Filtering Terrain Parameters

To ensure that the low-level policy training focuses on traversable terrain, we employ the adaptive terrain curriculum method introduced by Lee et al. (*15*). This selection process specifically targets the low-level policy training phase.

Using a genetic algorithm based on the Minimal Criterion (MC) (*78*), we avoid terrain parameters that are either too difficult or too easy for the agents. The fitness function, denoted as $f(c_\mathcal{T}, \pi)$, is defined as follows:

$$f(c_\mathcal{T}, \pi) = \begin{cases} \mathbb{E}\{\nu(s_t \mid c_\mathcal{T})\} & \text{if} \quad t_l < \mathbb{E}\{\nu(s_t \mid c_\mathcal{T})\} < t_h \\ 0.0 & \text{otherwise} \end{cases}. \tag{23}$$

Here, $c_\mathcal{T}$ denotes the terrain parameter being evaluated, and $\pi$ represents the policy being trained. The expected value $\mathbb{E}\nu(s_t \mid c_\mathcal{T})$ is computed over the trajectories generated by the policy during each iteration, where $\nu(s_t \mid c_\mathcal{T})$ is a score function reflecting the successful

traversal of a sampled terrain at state $s_t$. In our case, $\nu(s_t)$ is set to 1.0 if the velocity tracking error is less than 20% of the command speed.

The threshold parameters $t_l$ and $t_h$ define the MC, ensuring that terrain parameters with a success rate between $t_l$ and $t_h$ are selected. In other words, terrain parameters that fall within this success rate range are considered feasible for training.

These selected terrain parameters are then reused to generate tile maps for the subsequent high-level policy training, ensuring that the high-level policy is trained on feasible terrain environments suitable for navigation.

The concept of dynamic task generation and open-ended learning, demonstrated by the Open-Ended Learning Team at DeepMind (*79*), further supports the effectiveness of this approach. The automatic generation of new solvable problems enhances the agent's generalization capabilities.

## S7. Privileged Training

We follow the privileged learning method proposed by Lee et al. (*15*) for robust Sim-to-Real transfer. The policy trained by RL serve as "teacher policy". It uses the is the ground-truth state $s_t$ from simulation which includes privileged information $x_t$. $x_t$ includes ground friction coefficient or ground reaction forces, which are not directly observable in the real world.

A recurrent "student policy" network is trained in a supervised fashion without $x_t$. The student policy imitates the teacher and learns to construct an internal representation of the world from a sequence of the noisy real-world observations. A policy trained in this way has proven to be more adaptive and robust in real-world settings with high disturbances and noisy observations (*15, 16, 80*).

We employ the DAgger (*81*) algorithm for imitation learning. We collect trajectories using the low-level student policy and label target actions using the teacher policy. The loss function

is defined as

$$\mathcal{L} \coloneqq \mathbb{E}_{(s_t,o_t)\sim\mathcal{D}} \left\{ (\pi^{\text{teacher}}(s_t) - \pi^{\text{student}}(o_t, h_t))^2 \right\}, \tag{24}$$

where $o_t$ denotes the observation and $h_t$ denotes the hidden state of the student policy. $o_t$ is a noisy version of $s_t \setminus x_t$.

## S8. Bounded Action Space for HLC

We employ the Beta Distribution for the action space of HLC (*48*). The beta distribution is a continuous probability distribution defined on the interval [0, 1], characterized by two shape parameters, $\alpha$ (alpha) and $\beta$ (beta). These parameters determine the shape and probabilities associated with different outcomes.

The probability density function (PDF) of the beta distribution is given by the formula $f(x; \alpha, \beta) = x^{\alpha-1}(1-x)^{\beta-1}/B(\alpha, \beta)$, where $x$ represents the random variable representing the probability, and $B(\alpha, \beta)$ is the beta function. The expected value (mean) of the beta distribution is given by $E[X] = \alpha/(\alpha + \beta)$, and the shape parameters $\alpha$ and $\beta$ determine the variance of the distribution. Higher values of $\alpha + \beta$ lead to lower variance.

We modify the parameterization of the beta distribution to define the bounded action space for our high-level policy. Instead of directly outputting the $\alpha$ and $\beta$ parameters from $\pi_{hi}$, we design our policy to directly output the mean and the sum of $\alpha$ and $\beta$. More specifically, let $a_1$ and $a_2$ be the outputs of the high-level policy $\pi_{hi}(s_t)$, we have $\alpha = a_1 \cdot a_2$ and $\beta = a_2 - (a_1 \cdot a_2)$. This design choice eliminates the need for additional computing of the mean after inference.

| Parameter | Value |
|---|---|
| discount factor | 0.99 |
| KL-d target | 0.01 |
| clip range | 0.2 |
| entropy coefficient | 0.001 |
| max. episode length (s) | 10.0 |
| dt (s) | 0.02 |
| batch size | 500000 |
| num. minibatches | 20 |
| num. epochs | 4 |
| learning rate | adaptive* |

Table S2: Hyperparameters for LLC teacher policy training. ($*$) Follows the implementation by Rudin et al. (*50*).

| Parameter | Value |
|---|---|
| discount factor | 0.991 |
| KL-d target | 0.01 |
| clip range | 0.2 |
| entropy coefficient | 0.001 |
| max. episode length (s) | 15.0 |
| dt (s) | 0.1 |
| batch size | 150000 |
| num. minibatches | 10 |
| num. epochs | 5 |
| learning rate | adaptive |

Table S3: Hyperparameters for HLC training.