AUTONOMOUS VEHICLES

Learning robust autonomous navigation and locomotion for wheeled-legged robots

Joonho Lee*†, Marko Bjelonic‡, Alexander Reske‡, Lorenz Wellhausen‡, Takahiro Miki, Marco Hutter

Autonomous wheeled-legged robots have the potential to transform logistics systems, improving operational efficiency and adaptability in urban environments. Navigating urban environments, however, poses unique challenges for robots, necessitating innovative solutions for locomotion and navigation. These challenges include the need for adaptive locomotion across varied terrains and the ability to navigate efficiently around complex dynamic obstacles. This work introduces a fully integrated system comprising adaptive locomotion control, mobility-aware local navigation planning, and large-scale path planning within the city. Using model-free reinforcement learning (RL) techniques and privileged learning, we developed a versatile locomotion controller. This controller achieves efficient and robust locomotion over various rough terrains, facilitated by smooth transitions between walking and driving modes. It is tightly integrated with a learned navigation controller through a hierarchical RL framework, enabling effective navigation through challenging terrain and various obstacles at high speed. Our controllers are integrated into a large-scale urban navigation system and validated by autonomous, kilometer-scale navigation missions conducted in Zurich, Switzerland, and Seville, Spain. These missions demonstrate the system's robustness and adaptability, underscoring the importance of integrated control systems in achieving seamless navigation in complex environments. Our findings support the feasibility of wheeled-legged robots and hierarchical RL for autonomous navigation, with implications for last-mile delivery and beyond.

INTRODUCTION

A substantial portion of people reside in urban areas, leading to a considerable challenge in supply-chain logistics, especially for last-mile deliveries. Increasing traffic and demands for faster delivery services put additional pressure on roads. Moreover, last-mile delivery routes are not limited to streets but can also include indoor routes. By shifting reliance from individual motorized transportation to smart and versatile robotic solutions, we can substantially improve the efficiency of urban delivery and provide an efficient alternative to human labor. To fulfill all of 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/hour on average (*4*)].

Wheeled-legged robots offer a comprehensive solution that addresses these requirements (*5*–*8*). Our research focuses on developing a wheeled-legged robot, as depicted in Fig. 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*). However, to fully leverage such machines in autonomous real-world applications, it is essential to address several challenges, including solving hybrid wheeled-legged locomotion (hybrid locomotion), achieving smooth and efficient navigation, and implementing a complete system that seamlessly integrates locomotion and navigation modules into an autonomous application.

Robotic Systems Lab, ETH Zurich, Zurich, Switzerland.
*Corresponding author. Email: jolee@ethz.ch
†Present address: Neuromeka, Seoul, Korea.
‡Present address: Swiss-Mile Robotics AG, Zurich, Switzerland.

First, hybrid locomotion remains a challenging area in legged robotics. Existing approaches for hybrid locomotion were built on simple heuristics to decide when to step and when to drive (*10*) or rely on predefined gait sequences (*11*, *12*). Most control strategies designed for legged robots incorporated handcrafted 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 because 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 suboptimal outcomes when applied to wheeled-legged robots. Some directly optimized 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 used 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. In addition, some of these approaches prioritize computational efficiency at the expense of model accuracy, such as by neglecting the dynamics of wheels, leading to suboptimal performance on the real robot.

Second, traditional navigation planning methods often overlook the unique characteristics of highly dynamic robots, leading to suboptimal navigation plans. Urban environments are mostly covered with flat and open areas that require efficient high-speed traversal to cover large distances. At the same time, they are bristled with obstacles like stairs and uneven terrain. To achieve speed, efficiency, and obstacle-negotiation capabilities, a navigation algorithm must consider the characteristics of dynamic 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*) are based on explicit navigation costs, such as traversability (*21*, *22*), without considering the robot's whole-body states. They focus on generating kinematic navigation plans by sampling-based planning on these estimated cost maps. As a result, these approaches often cannot account for various dynamic characteristics of the robot, such as tracking error variations depending 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 shorter reaction times, which raises safety concerns and calls for more responsive control 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 collisions in dynamic environments. 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.

Last, attaining autonomy in robotic systems poses a substantial engineering challenge, requiring seamless integration of various submodules. Traditionally, these submodules are developed in isolation with a focus on each component's functionality. Their coordination

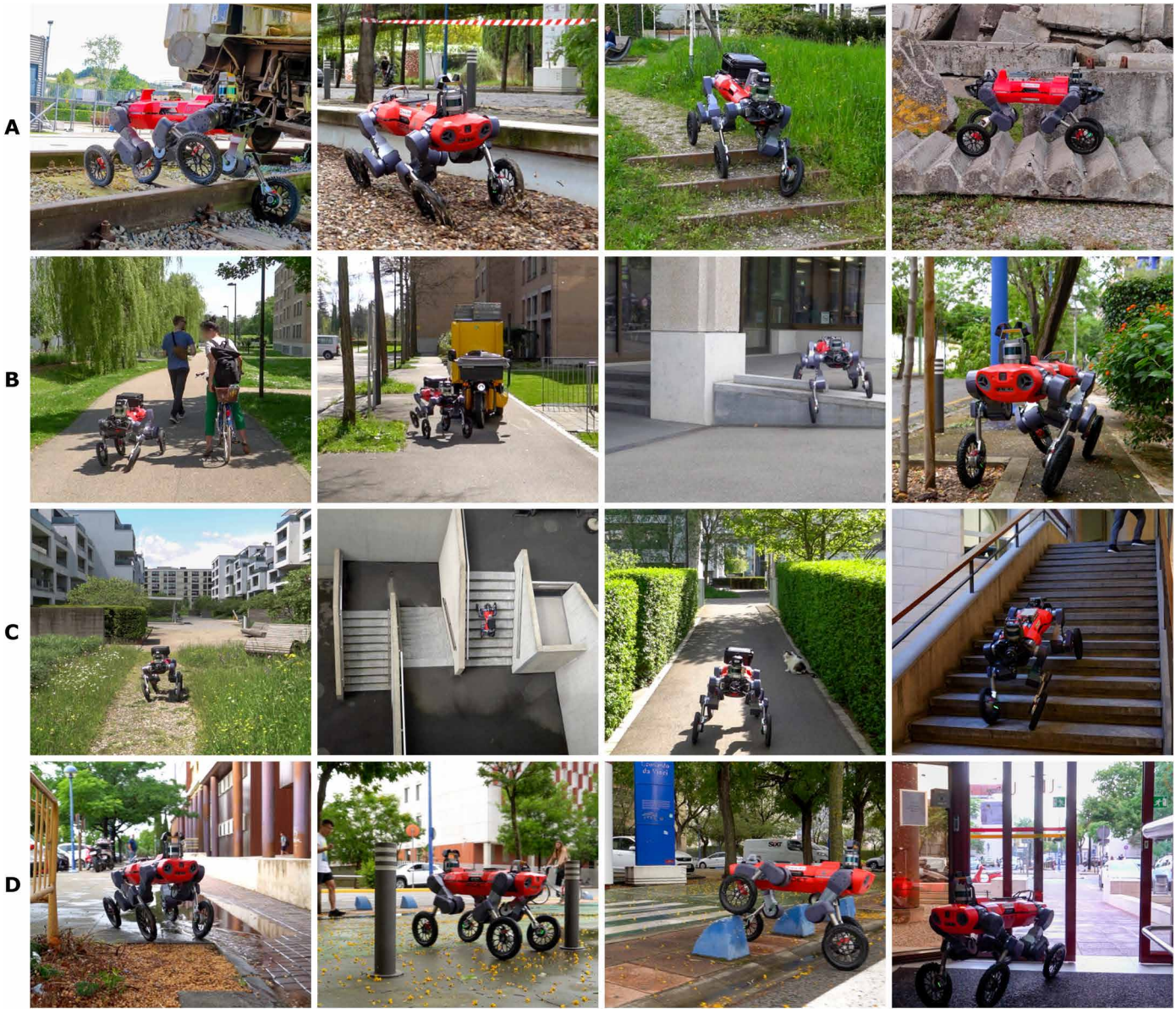

Fig. 1. Deployments in urban environments. Our control system for the wheeled-legged robot has undergone extensive validation in various indoor and outdoor locations. The experiments took place in Zurich, Switzerland, and Seville, Spain. (**A**) Locomotion challenges. (**B**) Navigation challenges: dynamic and static obstacles, complex terrains, and narrow space. (**C**) Locations in Zurich. (**D**) Locations in Seville.

relies heavily on heuristic methods for intermodule communications, and these engineered heuristics often limit smooth and robust operation. Team Cerberus, for example, undertook the development of an autonomy system for a classic legged robot during the DARPA Subterranean Challenge (*2*). The project uncovered substantial insights into operational challenges faced in real-world robotic applications. Issues observed during the challenge included robots frequently pausing midway through a path to replan or exhibiting zigzag motion while attempting to adhere to a predetermined route. Such discontinuous or oscillatory behavior can compromise efficiency and hinder the robot's ability to respond to complex dynamic scenarios. In some cases, navigation failure may occur when computed navigation paths are not accurately followed (*20*).

In this work, we developed a large-scale autonomous navigation system for wheeled-legged robots that enabled seamless coordination between navigation and locomotion controls. Our approach integrated hybrid locomotion control, which was developed using model-free reinforcement learning (RL) and privileged learning (*15*, *23*, *24*), with a navigation controller optimized through hierarchical RL (HRL). Both locomotion and navigation controllers were trained using simulated data. The controllers were integrated into a global navigation system using digital-twin models, high-resolution laser scans of real locations, for experiment planning and onboard localization. Extensive testing in urban areas of Zurich, Switzerland, and Seville, Spain demonstrated the system's autonomy in navigating complex environments, completing kilometer-scale missions across various terrains and obstacles. Our learned controllers enabled adaptive gait selection, efficient terrain negotiation, and responsive navigation that avoided static and dynamic obstacles safely. Our practical evaluations underscored the potential of wheeled-legged robots for achieving efficient and robust autonomy in real-world applications. Additional comparative studies validated the advantage of our tightly integrated navigation controller over traditional systems.

RESULTS

Movie 1 summarizes our main results.

System overview

We first present a detailed overview of each component comprising our autonomous navigation system. Figure 2 provides an overview of our system.

Movie 1. Summary of main contributions.

Robot

The wheeled-legged robot used in this work is depicted in Fig. 2A. The robot carried multiple payloads, including three light detection and ranging (LIDAR) sensors, a stereo camera at the front, a delivery box, a 5G router, and a global positioning system (GPS) antenna. They served various purposes, such as localization, terrain mapping, and human detection, contributing to the safety layer. We integrated a stereo camera with high-frequency object-detection capabilities because point cloud–based terrain mapping does not capture dynamic obstacles well. This allowed for real-time tracking of people within a range of 20 m. We created a buffer zone in the elevation map by adding an offset around the detected human positions, as detailed in the "Local navigation" section later. We provide more technical details in the Supplementary Materials.

Navigation system

Our navigation system is illustrated in Fig. 2B. Given a global navigation path, represented by a sequence of graph nodes, we extracted two waypoints, denoted as WP1 and WP2. Taking inspiration from the pure-pursuit tracking algorithm (*25*), 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 followed the intermediate waypoints using the low-level controller (LLC), which is commanded by the high-level controller (HLC), both of which are neural networks trained through RL. HLC, fed with the waypoints as input, generated velocity targets for LLC at 10 Hz, which aligns with the update rate of the onboard elevation mapping (*26*). LLC, in turn, generated joint position and wheel velocity commands at 50 Hz.

The primary technical contribution of this work lies in the development of our HLC. This controller addresses local navigation planning and path-following control together, which traditionally necessitated separate modules.

Locomotion controller (LLC)

We have developed a robust and versatile locomotion controller for wheeled-legged robots by leveraging model-free RL. Our LLC is driven by a recurrent neural network (RNN)–based policy and builds on the perceptive locomotion controller by Miki *et al.* (*16*). We applied modifications to the observation and action space to improve robustness and removed the engineered motion primitives [central pattern generator (CPG) in (*16*)]. Technical details are given in Materials and Methods.

With minimal dependence on human intuition, we achieved a locomotion controller capable of making decisions regarding the gait and transitioning between walking and driving modes. The locomotion controller is trained in simulation environments through privileged learning (*15*, *23*, *24*). During the training, an agent uses additional information that is only available during the training phase to enhance the model's performance. We used the robot's motion information, including velocity and acceleration, terrain properties, and noiseless exteroceptive measurements, as privileged information. During deployment, the final policy only relied on raw measurements from the inertial measurement unit (IMU), joint encoders, and onboard terrain elevation mapping. Similarly to the approach presented by Ji *et al.* (*27*), we used the raw IMU and encoder measurements instead of using a conventional state estimator. This reduces the use of heuristics for noise filtering and eliminates the need for accurate state estimation for orientation and velocity estimates. This approach resulted in enhanced robustness when

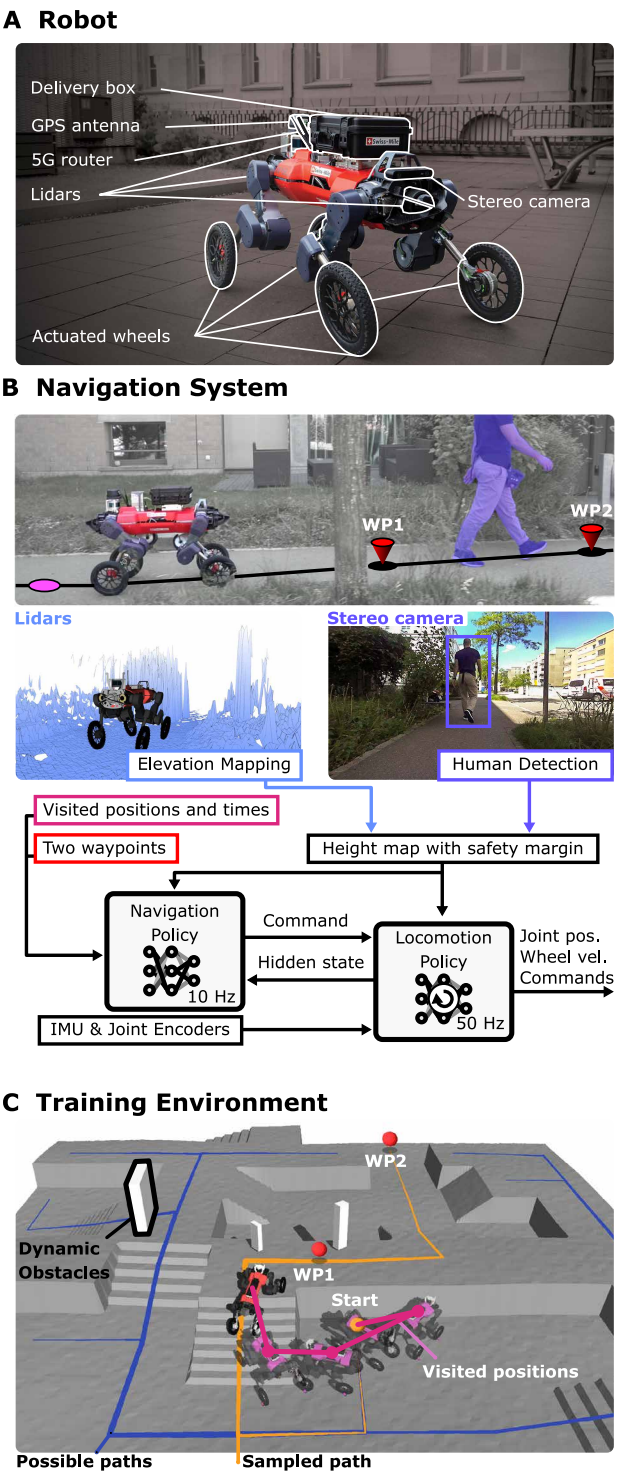

Fig. 2. System overview. (**A**) Our wheeled-legged quadrupedal robot is equipped with various payloads for onboard terrain mapping, obstacle detection, and localization. (**B**) Overview of the navigation system. The system is driven by two neural network policies operating at different levels. The high-level navigation policy observes two waypoints (WP1 and WP2) and generates target velocity commands for the locomotion policy. The low-level locomotion policy then controls joint actuators and follows the velocity commands. (**C**) Our training environment was designed to dynamically generate new navigation paths for each episode, optimizing the learning process. By leveraging pregenerated obstacle-free paths, we enhanced the navigation capabilities of our system.

operating on challenging terrains, resulting in fewer failure points in terms of locomotion control.

Mobility-aware navigation controller (HLC)

The HLC replaces the traditional navigation setup comprising path planning, path following, and intermodule communication layers (*2*). Instead of explicitly planning future poses and computing reference velocities, our HLC directly computes the velocity targets at a high frequency.

The HLC processes multiple input modalities, including the hidden state of LLC policy, terrain height values around the robot, and a sequence of previously visited positions, with corresponding visitation times. Instead of using standard proprioceptive observations, the HLC accesses the belief state of the LLC. This latent state captures environmental information such as terrain properties and disturbances, as supported by (*15*, *16*). In addition, the HLC processes 20 previously visited positions recorded at 50-cm intervals. They span a distance of up to 10 m, approximately the usual waypoint spacing. The history allows the HLC to make informed decisions based on the robot's prior navigation experience.

Our HLC was trained in the simulation environment depicted in Fig. 2C. In every episode, new obstacle-free paths were generated, and two waypoints were sampled along the path at random intervals.

Training environment

We have adopted the concept of "navigation graph" from computer games (*28*, *29*) to provide solvable yet challenging navigation problems to the agent during training (see Fig. 2). The simulation environment was crafted using a procedural content generation algorithm called wave function collapse (WFC) (*30*), and a graph outlining feasible paths and safe areas was constructed with the terrain. The training environment offered diverse navigation challenges, including detours, dynamic obstacles, rough terrains, and narrow passages. As detailed in Materials and Methods, we combined different obstacles in a controlled manner and rewarded RL agents for following the shortest path toward the goal point during training. This approach yielded improved performance compared with policies trained with randomly placed obstacles and goals.

Kilometer-scale autonomous deployments

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. In addition, we show one full mission in movie S1 to demonstrate the scale of each experiment.

Figure 3 summarizes our mock-up delivery mission conducted in Glattpark, Zurich. Our robot covered a total distance of 8.3 km with minimal human intervention.

We first show our workflow in Fig. 3A. To begin, we used a handheld laser scanner to capture dense color point clouds of the experimental area. The scanning process took approximately 90 min to cover a 245 m–by–345 m urban area. Subsequently, we georeferenced the point cloud, and the data were converted into a mesh representation, facilitating the creation of a navigation graph and the placement of goal points by a human expert (see Fig. 3, A-iii and B-ii). The purpose of the navigation graph is to provide topological guidance and to indicate social preferences, like avoiding landscaping and private property.

During the robot's deployment, it localizes itself with respect to the prescanned reference point cloud using its LIDAR, IMU, and joint encoder readings. With this setup, the robot can be provided

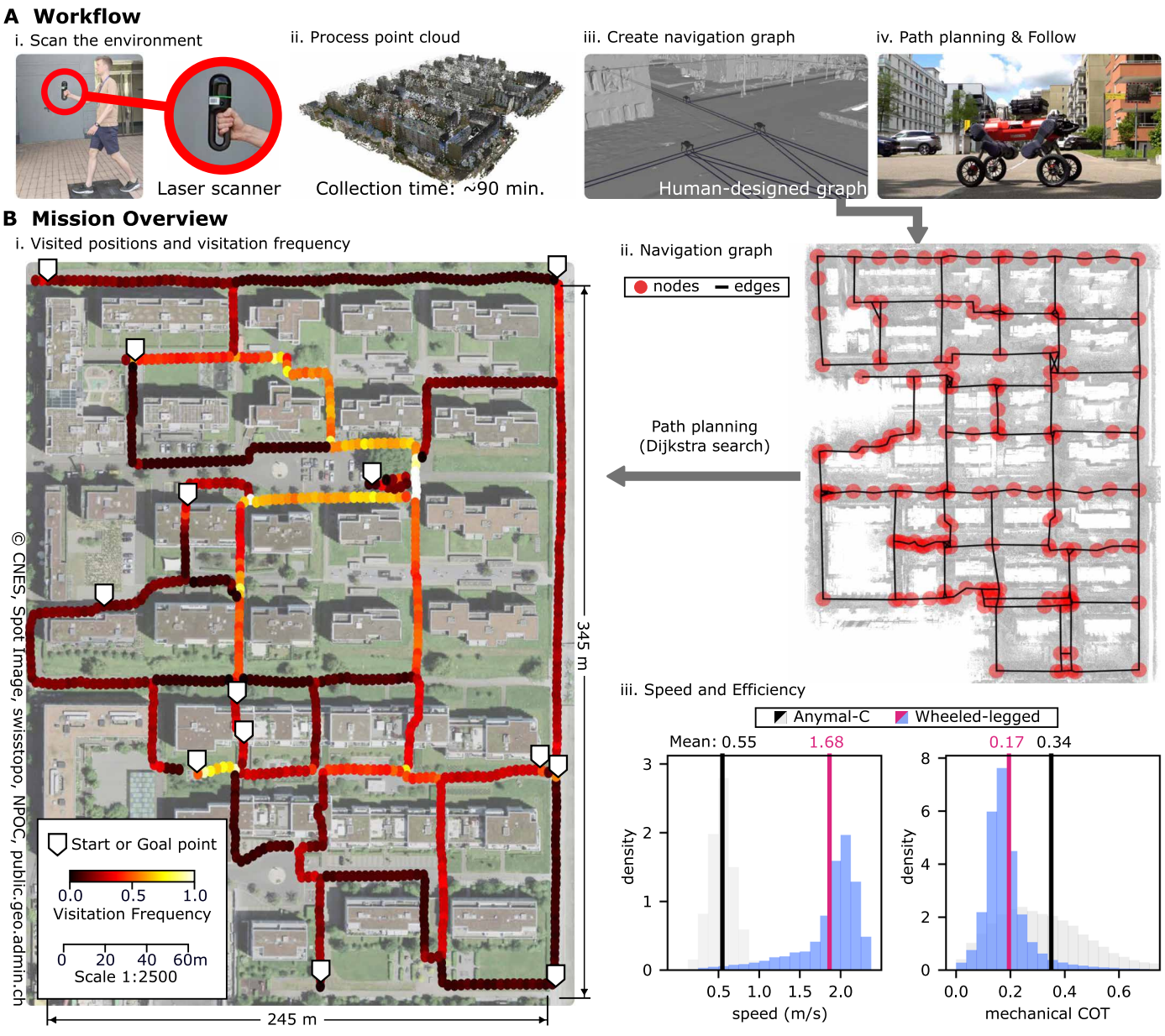

Fig. 3. Large-scale autonomous navigation experiment at Glattpark, Zurich. (**A**) Our city navigation workflow began with offline preparation, involving scanning the test area using a handheld laser scanner and constructing a navigation graph. (**B**) The robot autonomously navigated the urban environment to reach 13 predetermined goal points, selected in an arbitrary order. (i and ii) Path planning within the city was facilitated by the pregenerated navigation graph. (iii) Moving speed and COT_{mech} compared with a normal legged robot (ANYmal-C).

with a single GPS goal, and it autonomously navigates toward the target location. Selected goal points are sent to the robot via a mobile network, and the reference path is computed onboard using the shortest path algorithm (*31*). The resulting path is converted into robot-relative coordinates for our navigation policy using LIDAR localization in the prescanned 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.

Figure 3B-i illustrates the paths traversed by our robot during multiple long-distance experiments, each lasting more than 30 min. 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 to reach each goal point successfully.

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

$$COT_{\text{mech}} = \sum_{\text{all joints}} [\tau\dot{\theta}]^{+} / (mg|v_{xy}^{b}|) \quad (1)$$

where τ denotes the joint torque, $\dot{\theta}$ is the 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 COT_{mech} of 0.16. For comparison, we provide data on the average speed and COT of one ANYmal robot that primarily traversed flat and urban terrains during the DARPA Subterranean Challenge (*2*). The data were from ANYmal 4 in (*33*). Our robot demonstrated three times the speed with a 53% lower COT. Note that we only compared the output mechanical power. Other major factors contributing to energy loss, such as heat loss and mechanical loss from the actuators' transmission, also come into play during constant walking motions, thus potentially reducing overall efficiency.

The improvement is mainly attributed to the driving mode, which evenly distributes weight across all four legs, keeping leg joints relatively static. Constant stepping led to concentrated loads on fewer legs, requiring higher joint torques and speeds. During driving, joint actuators contributed almost zero COT_{mech} (≈0.01). Compared with a typical ANYmal robot during locomotion, our robot's wheels exerted approximately 1.2 times the total mechanical power while achieving an average locomotion speed 3.4 times higher. Upon evaluating the average $\sum\tau^2$ solely for leg joints, our robot exhibited a 16% lower value, despite being both heavier (≈12 kg) and faster. This quantity is directly related to the heat loss (*34*, *35*).

Figure 4 shows the major challenges encountered by our robot, including pedestrians, various obstacles, and nonflat terrains. Our robot demonstrated the capability to navigate around pedestrians in various situations, even on slopes or stairs, as depicted in Movie 1 and movie S1. In addition, our robot could avoid thin obstacles, such as the pole shown in the first image of Fig. 4A-ii, as well as various discrete terrains like steps and stairs.

Because the HLC and LLC are trained to minimize COT_{mech} and $\sum\tau^2$, the robot mostly drives on flat terrain. However, when encountering uneven surfaces, the robot switched to a stepping gait. This gait switching was learned without handcrafted heuristics like CPG or predefined gait sequences. Furthermore, our controller demonstrated 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. 4B. First, 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.

Second, we encountered situations where the waypoints were located within untraversable regions. For instance, tall grass had grown on a trail between the creation of the navigation graph and the robot deployment, obstructing the path. 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 replanning to go around the obstruction.

Last, 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 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. 5, we present example scenarios that best show the local navigation capability of our system. The sequence of these scenarios can be viewed in movie S2.

In the first case (Fig. 5A), we show the exploratory behavior when the robot encounters a blocked path. The robot reversed and moved along the wall, searching for an opening until it found the stairs leading to the final waypoint. The robot's explicit position memory enabled it to reason about its previously visited positions and navigate through the complex obstacles.

Figure 5B shows our robot's ability to navigate narrow corridors. It safely maneuvered through two doors with a human standing between them, where the gap was as wide as the robot's width. The robot navigated through the narrow space without collision even though human detection was not enabled in this deployment. This example showcases the precision and real-time trajectory adjustment of our navigation controller, making it suitable for environments with limited space and tight passageways.

We conducted a test with a complex obstacle depicted in Fig. 5C-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.

Initially, when faced with the obstructed route, the robot drove backward and began exploring. During this phase, it could either find the stairs to climb up or continue exploring to find a lower step height. In the second case, after driving along the step, the robot found a viable height of approximately 20 cm. This example shows the effectiveness of our hierarchical controller seamlessly adapting its gait on the basis of the terrain and exhibiting versatility in navigating complex paths.

We observed that the HLC has an asymmetric understanding of traversability when going up and down the step (Fig. 5D). Specifically, the robot was able to traverse higher steps when descending, indicating that it has a more advanced understanding of the terrain compared with cost-map approaches for traversability estimation (*21*, *26*). Traditional methods often use symmetric traversability maps that are independent of motion direction, whereas our approach makes decisions on the basis of the current terrain, the robot's state, and the characteristics of the LLC.

In Fig. 5E, we visualize our strategy for augmenting the static, local elevation map with dynamic obstacles. We used camera-based human detection to introduce a height offset within a radius of 50 cm around individuals. As the robot encountered a person moving along its path, the HLC, trained to handle dynamic obstacles of various sizes, maintained a constant distance from the person, enabling the robot to safely overtake.

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 movie S3. The LLC adapts gaits depending on the command velocity and terrain. We tested the policy on various real-world terrains, as illustrated in Fig. 6. Our previous model predictive control (MPC)–based controller (*10*) lacks robustness and cannot operate in the environments depicted in Fig. 6. In addition, 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.

Figure 6A presents distinct behaviors depending on the terrain. When traversing a large discrete obstacle (Fig. 6-i), the robot displayed an asymmetric gait combining creeping (*36*) and driving. When climbing stairs or steep hills, the robot trotted like a normal point-foot quadruped (*16*) (Fig. 6A, ii and iii). Conversely, the robot drove over the bumpy terrain where the height deviations were comparable to the wheel's radius (Fig. 6A-iv). The policy adjusted the reach of each leg to keep the main body stable and kept the wheels in contact with the terrain, acting as an active suspension. The gait pattern varied depending on the terrain conditions, such as slope or friction. In addition, the policy adjusted the main body's height on the basis of the situation. For instance, when descending a slope, the policy lowered the body height to enhance stability and prevent tipping over (Fig. 6A-v).

In Fig. 6B, we present two scenarios involving high discrete obstacles. In Fig. 6B-i, we commanded our LLC to drive down a table approximately 60 cm high. As the front legs descended, the robot stretched down its front legs and crouched the hind legs to maintain a level main body. Once the front legs made contact with the ground, the front wheels rolled forward to regain balance. In Fig. 6B-ii, we show our robot traversing a block approximately 40 cm high. In the middle of the block (ii-2), all the wheels were in the air. Then, the robot crawled forward with its knees until one of the wheels regained contact. This example shows the advantage of using model-free RL (*37*).

Quantitative evaluation of the locomotion performance is presented in Fig. 6C. In Fig. 6C-i, we present the maximum traversable height of the step depending on the command speed. Our robot could traverse higher steps when descending compared with ascending. This observation aligns with the results shown in Fig. 5 (C and D), where our HLC avoided high steps to avoid knee collisions and ensure safety. In Fig. 6C-ii, we tested the LLC on slopes with a 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 m. We observed that the stepping behavior, as depicted in Fig. 6A-iii, emerged only on steep slopes with command speeds more than 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 model-based planning and path-following approaches would struggle to identify and adapt to such complexities.

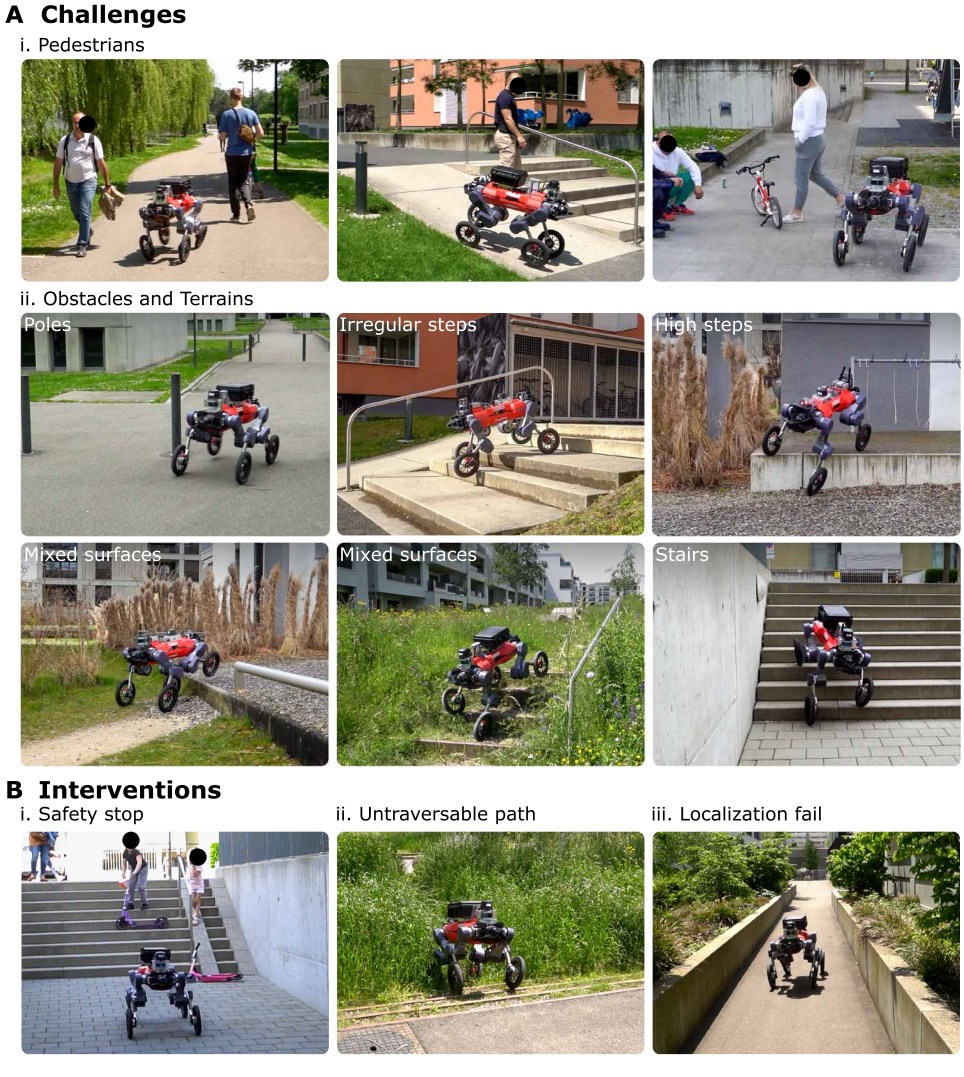

Fig. 4. Challenges in the populated urban environment. (**A**) The urban environment presents various obstacles. Some have to be avoided, such as pedestrians or poles, and others can be traversed, such as stairs or steps. The robot makes intelligent decisions on the basis of its onboard perception. (**B**) We had to intervene and stop the mission in these three cases.

Comparison with a conventional navigation approach

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

We conducted experiments in a point-goal navigation setup, as depicted in Fig. 5A. The area was scanned with a laser scanner to create a simulation environment, shown in Fig. 7A, with fixed start and goal points.

Figure 7B illustrates the field of view of our HLC and the baseline. To accommodate the limitations during physical deployment, we limited the range of the map to 3.5 m in both x and y directions. This decision was particularly important when the robot moved at high speeds, reaching up to 2 m/s. Using a larger map slowed down the elevation mapping update and resulted in high delay and unupdated regions in the map.

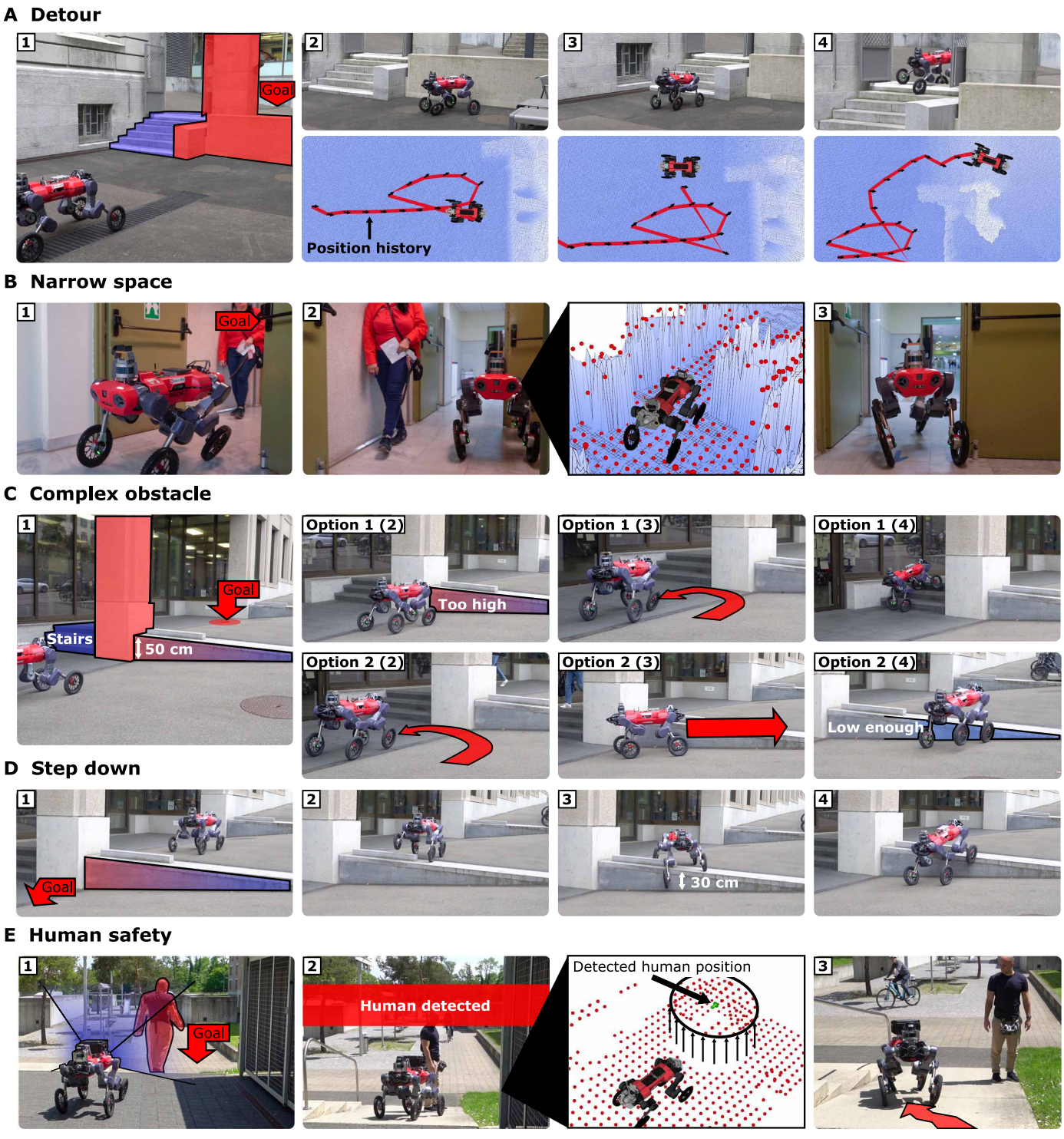

Fig. 5. Obstacle negotiation. (**A**) Our robot navigates around blocked routes by actively exploring the area and finding alternative paths. (**B**) Safe traversal of a narrow space. (**C**) Our robot exhibits two different ways to traverse the complex obstacle. (C and **D**) Our robot shows an asymmetric understanding of traversability, being able to traverse higher steps when going down. (**E**) We ensure safety around humans by incorporating additional human detection and overriding height scan values.

Figure 7C presents the trajectories of both approaches that successfully reached the goal. Our approach explored the environment until it found the staircase. The baseline could also solve the problem when occluded regions were assumed traversable. However, the baseline consistently collided because of the delay issue, which is explained below.

Figure 7D presents failure cases. Our approach sometimes got stuck after exploring a wide open area (Fig. 7D-i). The position

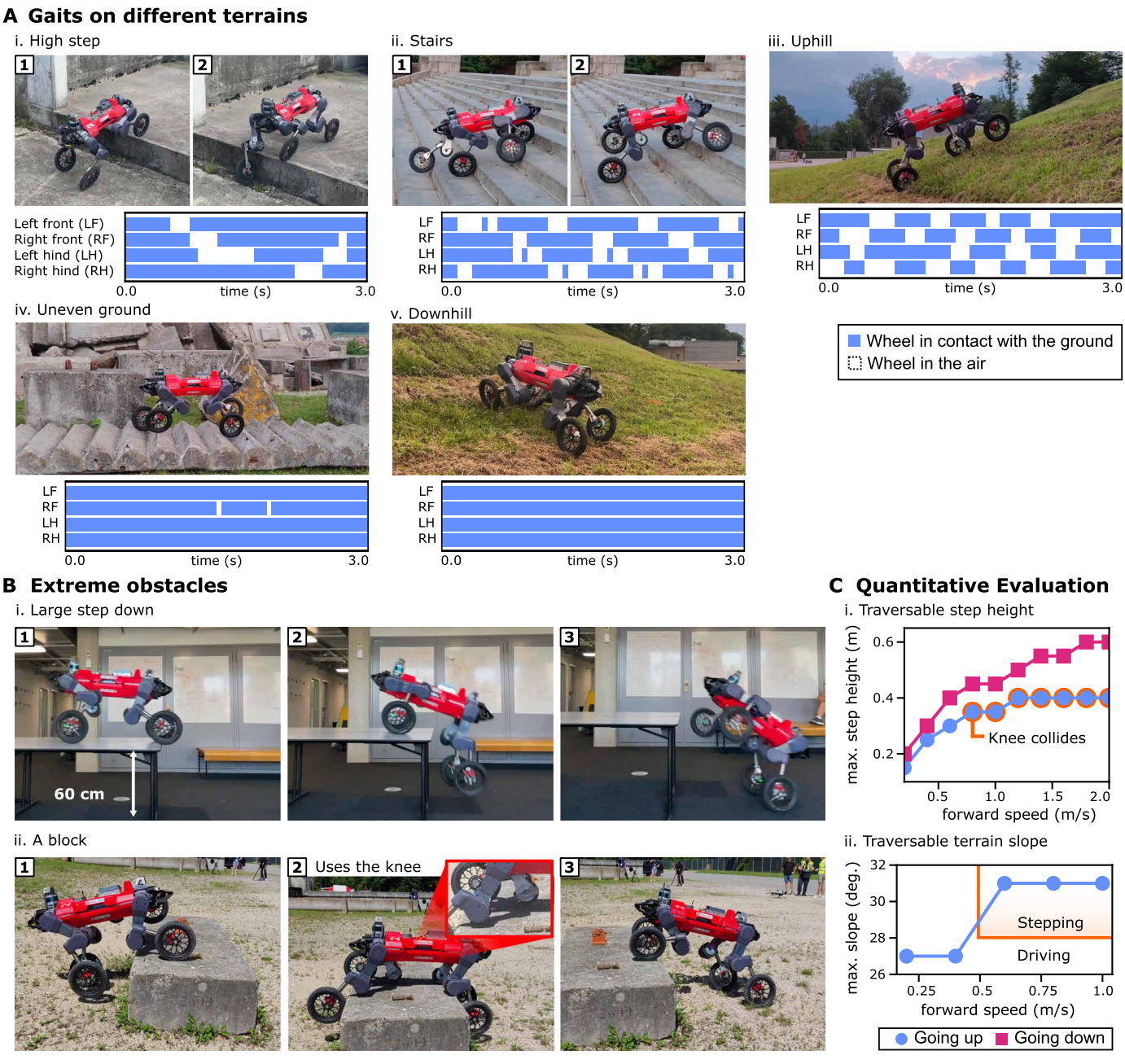

Fig. 6. Locomotion behaviors in different situations. (**A**) Gaits on different terrains. The robot is moving from left to right following target velocity commands given by the joystick (up to 2 m/s). The plots show the wheel contact sequences for each terrain. (**B**) Motion sequences over two extreme obstacles. (i) The robot underwent a full flight phase during the drop while maintaining stability. (ii) When traversing high obstacles, the robot sometimes leverages other body parts such as its knees. (**C**) (i) The maximum step height and (ii) the maximum terrain slope traversed by our locomotion controller with a given command velocity, when ascending and descending.

memory became full, and the agent did not explore further. In addition, we trained our approach without memory to validate the importance of positional memory. The memoryless policy exhibited repetitive behaviors and struggled to escape local minima (Fig. 7D-ii).

The baseline method faced two challenges: occlusion handling and tracking error of the locomotion controller. Although several heuristics could help mitigate the occlusion problem, the baseline method's ability to handle changing situations was limited because of the delay in replanning. Concerning the second issue, most existing methods assume perfect tracking; however, the actual locomotion controller experienced delays and tracking errors. Figure 7D-iv illustrates this issue, where distant pose targets led to high-velocity commands and overshoot. The robot could not accurately track the next waypoint and collided. This problem becomes more pronounced when dealing with fast-moving robots on rough terrain.

In the quantitative analysis shown in Fig. 7E-i, our approach with full memory showed the lowest failure rate. Our method without

memory exhibited the highest failure rate. 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 enabled the robot to dynamically explore new areas, resulting in a higher success rate. Furthermore, the results obtained from "Ours without memory" emphasize the importance of the memory mechanism in facilitating effective exploration in static environments. Only our approach achieved collision-free trajectories (Fig. 7E-ii). This is attributed to the accurate steering capability of our HLC, which respects the capabilities of the locomotion policy.

Another benefit of our approach lies in its computational efficiency (Fig. 7E-iii). From updating observations to inferring the neural network, our HLC took 0.34 ms on average. In contrast, depending on the complexity of the environment, the baseline sometimes requires more than a second to update the navigation plan on a desktop machine (AMD Ryzen 9 3950X, GeForce RTX 2080).

The baseline's high failure rate could also be attributed to the imperfect path following. In Fig. 7E-iv, a histogram illustrates the tracking error distributions of both approaches. The average tracking errors for our approach and the baseline were 0.24 and 0.45 m/s, respectively. The baseline exhibited a peak at high tracking error in the histogram, which occurs when there are discrete changes in the command velocity or when the robot is commanded too close to obstacles, causing the LLC to refuse to follow the command. In contrast, our HLC, trained in conjunction with the LLC, demonstrated evenly distributed tracking error statistics with consistently low tracking errors.

DISCUSSION

The presented wheeled-legged robot system demonstrates substantial 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 completed kilometer-scale autonomous missions in urban environments with minimal human intervention. It navigated through various obstacles such as stairs, irregular steps, natural terrain, and pedestrians.

Our results demonstrate several notable advantages over conventional navigation planning approaches. First, 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

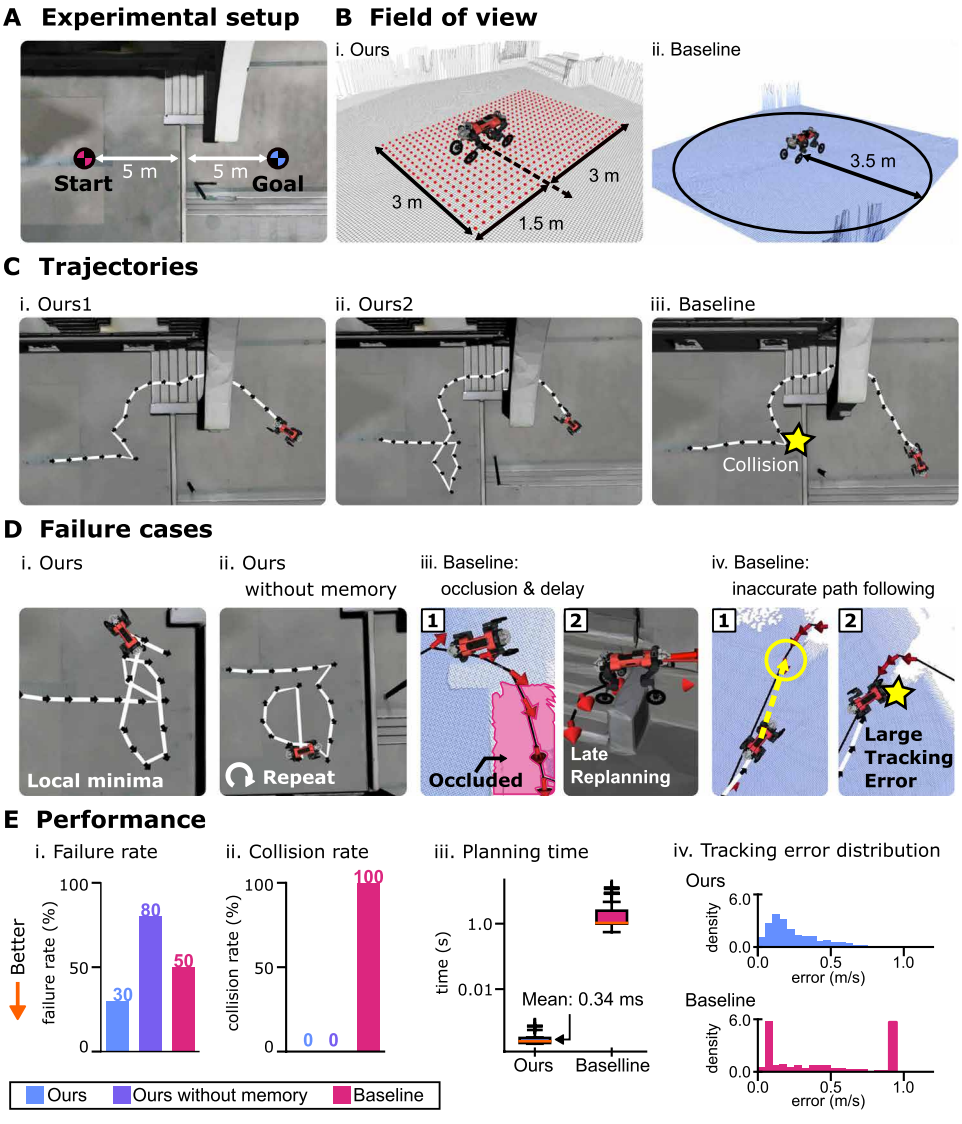

Fig. 7. Comparison with a conventional approach in a point-goal navigation setup. (**A**) Experimental setup. A goal point is given across the complex obstacle comprising stairs and a wall. The robot is initialized with uniformly sampled yaw angles between $-\pi/4$ and $\pi/4$, facing the goal point. (**B**) Field of views of our approach and the baseline. (**C**) Trajectories of the two methods. Our method displays two distinct trajectories depending on the initial exploration direction. (**D**) Failure cases. (i) Our controller got stuck when the exploration path became longer than its memory capacity. (ii) Without memory, our approach often fell into local minima. (iii) Baseline suffered from overconfidence in occlusion and delays in replanning. (iv) The high pose tracking error of the path-following controller of the baseline led to frequent collisions. (**E**) Quantitative evaluation of performance. The experiments were repeated 10 times per method. (i and ii) Failure and collision rates of the 10 trials. (iii) Planning time comparison. The error bar denotes one SD. Plus signs indicate outliers. (iv) Distributions of tracking errors during the experiments. For each method, histograms were generated using two trajectories.

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 use of semantic information (to adjust the height map for human safety). More advanced scene understanding, such as pavement detection or visual traversability estimation (*38*), will allow the robot to make more informed decisions during navigation. This is exemplified by the work of Sorokin *et al.* (*39*), who suggested 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 m to the front of the robot. This is inherently limited by using elevation mapping (*26*). 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 was capable of locomotion up to 6.2 m/s, but we could not demonstrate the maximum speed during autonomous deployment because of 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. Although challenges remain, such as improving perception capabilities and 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 Fig. 2B, was 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 to 20 m apart. The global path can be generated using a graph planner (*40*) or defined manually. Although the global planning aspect is essential for the overall navigation process, it is outside the scope of this work. Because of space constraints, a comprehensive validation of our method is presented in the Supplementary Materials.

Overview of the approach

Inspired by the existing literature (*41*, *42*), in which the hierarchical decomposition of complex tasks enables faster learning and higher performance, we used HRL to extend our previous learning-based velocity tracking controller (*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 adopted the two-level HRL framework by Nachum *et al.* (*43*). Various hierarchical structures have been explored in the literature.

Initially, we considered an end-to-end strategy as done by Rudin *et al.* (*44*). This method trains a unified policy to simultaneously manage locomotion and navigation tasks, without any hierarchical structure.

An alternative explored in the literature involves a two-level hierarchy, where a high-level policy directs the low-level policy by issuing latent subgoals at a lower frequency. Some works used learned latent subgoals for HRL (*45*, *46*), which offer simplicity and flexibility. There is no need to explicitly define intermediate goals, and the task assignment within the hierarchy is learned.

Our approach instead adopted an explicitly defined subgoal within a two-level hierarchy. In our setup, the low-level policy focused on locomotion tasks, and the high-level policy focused on navigation by commanding target base velocities to the low-level policy. We opted for explicitly defining subgoals for practical reasons.

Although the first two approaches could have provided simpler implementations, our decision to explicitly separate the control tasks enabled the independent development of the controllers. This separation not only simplified collaborative development efforts, allowing teams to work simultaneously on distinct system aspects, but also aligned with common practices in legged robotics. Consequently, this approach facilitated the reuse of pretrained low-level policies across a range of high-level applications, enhancing the system's versatility and adaptability.

Despite our high-level policy primarily outputting base velocity commands, we also explored commanding gait patterns similarly to Tsounis *et al.* (*47*). The experiment is described in the Supplementary Materials.

Training procedure

We trained a low-level policy and a high-level policy sequentially. The low-level policy training involved two stages: teacher policy training followed by student policy training. Then, the high-level policy was trained using the trained low-level student policy.

We began by training the teacher policy for the low-level locomotion policy. The teacher policy was trained to follow random velocity targets (and optionally gait parameters) on rough terrains using a PPO algorithm (*48*). In this step, privileged information, including the robot's motion, terrain properties, and noiseless exteroceptive measurements, was used to enhance the locomotion performance and convergence of the policy.

Subsequently, the deployable student policy was trained. Unlike the teacher policy, the student policy receives a sequence of noisy and biased IMU measurements, joint states, and noisy height scans

as input instead of directly accessing privileged information. Through imitation learning from the teacher policy and leveraging an RNN encoder (*16*), the student policy was trained to extract features from the temporal data necessary for robust locomotion.

The trained student low-level policy was then regarded as a fixed component, and a high-level navigation policy was trained using a PPO algorithm. The training data were collected in our custom-built simulation environment. This approach is further explained in the next section.

In addition to the previous three stages, an optional phase of alternating training could be conducted for both policies to enhance their coordination and potentially improve motion smoothness. However, our experiments showed only marginal enhancements from this, and therefore, we did not conduct any further fine-tuning.

Graph-guided navigation learning

Navigation graphs, commonly used in computer games for autonomously navigating characters in synthetic environments (*28*, *29*), played a crucial role in our navigation learning approach. Inspired by game development, we used pregenerated navigation graphs to define initial states, assign feasible paths, and design the reward function during the training of our high-level policy.

World generation

Our automatic terrain generation method, illustrated in Fig. 8, establishes connectivity between different areas of the terrain (tiles), resulting in a navigation graph across the training environment. For example, tiles with stairs in the x direction are exclusively connected to floor tiles along the x axis.

To generate diverse and realistic terrain layouts, we used the WFC algorithm. This algorithm automatically combines various terrain features, such as stairs, floors, and other obstacles. The output of the WFC algorithm provided both the composed terrain and the connectivity information between the tiles.

The WFC algorithm divides an input tile map (referred to as "Example" in Fig. 8B) into smaller chunks and rearranges them to create new N-by-N patterns. This procedural generation approach enabled us to generate a wide variety of navigation worlds with different combinations of corridors, rooms, and obstacles.

We defined three types of tiles: stair, floor 0, and floor 1. We provided 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 composed the existing tiles, resulting in varied and realistic training environments. The parameters for the parameterized floor and stairs were selected during the low-level policy training using the terrain filtering algorithm by Lee *et al.* (*15*). See the Supplementary Materials for details.

Using navigation graphs for RL

We used Dijkstra's algorithm (*31*) to find a path between two randomly selected nodes within the graph. Along the graph edge, we sampled two 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 was sampled uniformly from 5.0 to 20.0 m in every episode. At the end of each path, we included the last node twice as two waypoints. This approach ensured that the agent had clear instructions on the desired trajectory and endpoint.

During the initial training phase, a positive reward was given when the agent moved along the planned path on the graph. The reward gradually diminished, and we let the policy train with a sparse reward at the end. The reward function was defined as follows

$$r_{h,dense}\begin{cases}1.0 & |e_{wp^1}|<0.75\\ clip(v\cdot\widehat{e_{wp^1}},0.0,v_{thres})/v_{thres} & \text{otherwise}\end{cases} \quad (2)$$

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

This reward mechanism encouraged the agent to follow the shortest distance on the navigation graph, minimizing the geodesic distance to the final goal. The path entailed detours rather than simply moving straight toward a waypoint. This approach challenged 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 introduced dynamic obstacles during the training. The dynamic obstacles were randomly placed within the environment and moved toward the robot.

The dynamic obstacles are shown by white boxes in Fig. 8B-iii. Their number, positions, and velocities were randomly generated for each episode. These obstacles moved toward the robot at speeds ranging from 0.1 to 0.5 m/s.

High-level policy details

This section is focused on detailing the Markov decision process (MDP) governing the high-level policy (π_{hi}), encompassing observations and actions. Detailed information about the reward function can be found in Supplementary Materials and Methods.

Observation

The observation space of π_{hi} contains four different modalities. First, π_{hi} observed exteroceptive measurements from terrain mapping for obstacle avoidance. The exteroceptive observation followed the definition by Miki *et al.* (*16*). We sampled height values around the robot from the robot-centric elevation map (*26*). Because of limited memory and computational resources onboard, the robot's field of view was restricted to 3 m to the front and 1.5 m in other directions. We prioritized shifting the scan pattern toward the front because of the farther perception range afforded by the forward-facing camera. In addition, the exteroceptive observation included two previous scans taken at 0.1 and 0.2 s before to account for the dynamic environments.

Second, π_{hi} observed the hidden states of the locomotion controller instead of estimated robot states such as gravity vector or twist. Using the hidden state of the RNN locomotion policy improved the robustness of our system. This is explained in detail in the "Low-level policy details" section.

To facilitate exploration, we used an additional position buffer. We recorded the visited positions in the world frame at regular intervals of 0.5 m, along with the corresponding visitation time. The time information included how many time steps the robot stayed in each position. The most recent 20 positions and their respective time information were provided to the policy in the robot frame.

Last, two waypoints were observed. π_{hi} observed a short history of two previously given waypoints and three previous outputs of π_{hi}.

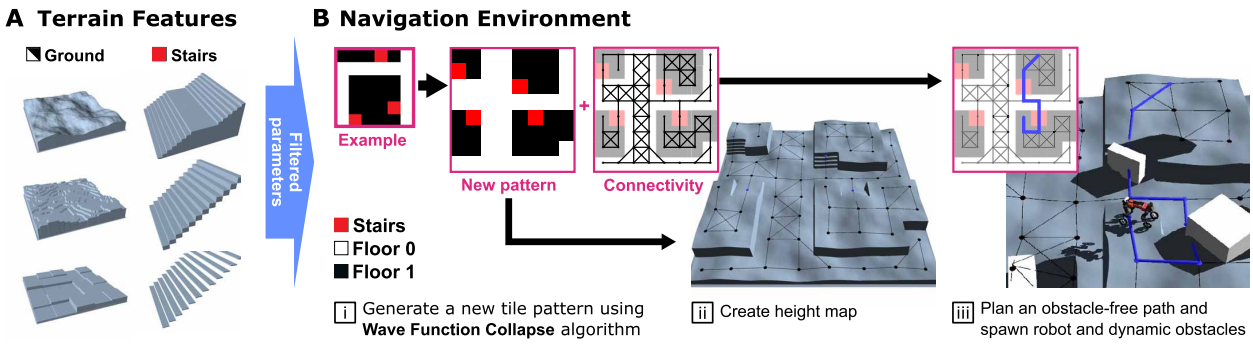

Fig. 8. Procedural generation of the navigation world. (**A**) Filtered parameterized terrain during low-level policy training. (**B**) (i) Generating new tile maps and connectivity graphs using the WFC algorithm. (ii) Created height map terrain with filtered floor features and stair parameters. (iii) Randomly generated navigation path between two nodes provides waypoints during training. Dynamic obstacles (white boxes) are added randomly.

This history of waypoints and actions assisted the policy in making a smoother trajectory.

Exploration bonus

During training, we encouraged the exploration using the explicit position buffer. This was achieved through an exploration bonus added to the reward function. The exploration bonus, denoted as r_{exp}, was calculated as the sum of costs $C(s_t, \mathrm{wp}_t^1, p_{buf}^i)$ over the positions in the buffer P_{buf}.

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

$$C(p_{robot}, wp^1, p_{buf}^i) := \begin{cases} 0.0 & |p_{robot} - wp^1| < 0.75 \\ -n_{buf}^i & |p_{robot} - p_{buf}^i| < 1.0 \end{cases} \quad (3)$$

Here, p_{robot} represents the position of the robot, wp^1 denotes the first waypoint, p_{buf}^i represents the ith position saved in the position buffer, and n_{buf}^i corresponds to the number of visits for the ith 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, then the agent incurs a penalty proportional to the number of time steps it stayed in that position. This penalty encouraged the agent to explore new areas and prioritize progress toward the first waypoint.

Bounded action space

Instead of the commonly used Gaussian action distribution, we used beta distribution to represent a bounded action space for π_{hi}, as introduced by Chou *et al.* (*49*). This offered several benefits. First, it allowed us to define hard limits on the outputs, enhancing safety and interpretability. In addition, working with a bounded action space made it easier to regularize the motion and control the behavior of the agent.

Specifically, we defined the bounds of the 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 encouraged the policy to consistently face forward during locomotion, aligning with the orientation of the camera mounted on the robot. We provide additional details in the Supplementary Materials.

Network architecture

We used a combination of architectures tailored for specific input types. For position history, we used one-dimensional convolutional neural network (CNN) layers followed by max pooling, similar to PointNet (*50*), enabling permutation-invariant processing of spatial information. The height scan around the robot was processed using three-layer two-dimensional CNN layers followed by a multilayer perceptron (MLP) layer. Other inputs and outputs were processed by plain MLP layers commonly used for nonspatial data. For the beta distribution parameters, we used the sigmoid function at the output layer.

Low-level policy details

The MDP for the low-level teacher policy was inherited from Miki *et al.* (*16*) with modification to observation and action spaces. The reward function and the details on the privileged training are provided in the Supplementary Materials.

The low-level policy was trained to achieve velocity tracking on random rough terrains. These terrains, designed by Miki *et al.* (*16*), are illustrated in Fig. 8A. Each terrain type was generated by two or three parameters. During training, we applied the parameter filtering algorithm by Lee *et al.* (*15*).

The low-level policy is commanded by linear velocity in the x and y directions, as well as the yaw rate. Linear x velocity is uniformly sampled from −2.5 to 2.5 m/s, y velocity from −1.2 to 1.2 m/s, and yaw rate from −1.5 to 1.5 rad/s. In each episode, a new command is sampled, with a 0.005 probability of random resampling.

Observation

The observation includes three types of information: a sequence of both exteroceptive and proprioceptive measurements, alongside the velocity command. For the exteroceptive perception, we sampled height values around the robot's wheels from a circular pattern, the same as Miki *et al.* (*16*).

The proprioception included measurements obtained from body IMU and joint encoders. These measurements conveyed information about the robot's body acceleration, angular velocities, joint angles, and joint velocities.

As previously discussed, instead of relying on estimated pose and twist by a model-based state estimator as done in several existing works (*16*, *34*, *51*), we directly used IMU measurements consisting of linear acceleration and angular velocity. This shift was motivated by the observation that conventional state estimators often result in high errors in case of wheel slippage or discrete height changes. In movie S4, we show a failure case of a locomotion controller due to a state estimation error. The command was provided as a three-dimensional vector, including the target base horizontal velocity and target base yaw rate.

Privileged observation

Privileged observation was only used for teacher policy training. It included noiseless 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*).

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 were given to the PD controller of each actuator. For a more detailed explanation of the simulation of the actuators, we refer the readers to the Supplementary Materials.

In contrast to our prior work (*16*), we discarded the use of the CPG in the action space to remove any engineered bias in the motion. A detailed comparative study of various action spaces is provided in the Supplementary Materials.

Network architecture

The low-level teacher policy was implemented as a plain three-layer MLP, and the low-level student policy was based on the gated recurrent unit architecture by Miki *et al.* (*16*).

Statistical analysis

Statistical analyses were performed using Python. For all the results, we computed the mean and SD over the full trajectory using the Numpy library. The box plot in Fig. 7 was generated using the Matplotlib library. We collected data at 400 Hz using either onboard state estimator or ground truth from simulation. A low-pass filter with a cutoff frequency of 5 Hz was applied to the state measurements to reduce high-frequency noise. The heatmap in Fig. 3 was generated by counting the visitation every 1 m based on the point cloud localization. For the COT comparison in kilometer-scale autonomous deployments, we only considered the data points where the linear speed is higher than 0.2 m/s. For the tracking error histogram in Fig. 7, we used data points with a command speed higher than 0.5 m/s.

Supplementary Materials

This PDF file includes:

Supplementary Materials and Methods
Fig. S1
Tables S1 to S3
References (*52*–*82*)

Other Supplementary Material for this manuscript includes the following:

Movies S1 to S4

REFERENCES AND NOTES

1. M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A Melzer, M. Hoepflinger, Anymal-a highly mobile and dynamic quadrupedal robot, in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2016), pp. 38–44.
2. M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, K. Alexis, Cerberus in the DARPA subterranean challenge. *Sci. Robot.* **7**, 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 Robot. Autom. Lett.* **8**, 1611–1618 (2023).
4. L. Wellhausen, M. Hutter, Rough terrain navigation for legged robots using reachability planning and template learning, in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2021), pp. 6914–6921.
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 Robot. Autom. Lett.* **4**, 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 Robot. Autom. Lett.* **4**, 2116–2123 (2019).
7. V. Klemm, A. Morra, C. Salzmann, F. Tschopp, K. Bodie, L. Gulich, N. Küng, D. Mannhart, C. Pfister, M. Vierneisel, F. Weber, R. Deuber, R. Siegwart, Ascento: A two-wheeled jumping robot, in *2019 International Conference on Robotics and Automation* (IEEE, 2019), pp. 7515–7521.
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, in *IEEE Aerospace Conference* (IEEE, 2020), pp. 1–9.
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. *int. J. Robot. Res.* **41**, 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, in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2021), pp. 8388–8395.
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 Transac. Graph.* **37**, 1–12 (2018).
12. M. Hosseini, D. Rodriguez, S. Behnke, State estimation for hybrid locomotion of driving-stepping quadrupeds, in *2022 Sixth IEEE International Conference on Robotic Computing (IRC)* (IEEE, 2022), pp. 103–110.
13. C. D. Bellicoso, F. Jenelten, C. Gehring, M. Hutter, Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots. *IEEE Robot. Autom. Lett.* **3**, 2261–2268 (2018).
14. F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, M. Hutter, Dynamic locomotion on slippery ground. *IEEE Robot. Autom. Lett.* **4**, 4170–4176 (2019).
15. J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter, Learning quadrupedal locomotion over challenging terrain. *Sci. Robot.* **5**, eabc5986 (2020).
16. T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter, Learning robust perceptive locomotion for quadrupedal robots in the wild. *Sci. Robot.* **7**, eabk2822 (2022).
17. W. Xi, Y. Yesilevskiy, C. D. Remy, Selecting gaits for economical locomotion of legged robots. *int. J. Robot. Res.* **35**, 1140–1154 (2016).
18. Y. Yang, T. Zhang, E. Coumans, J. Tan, B. Boots, Fast and efficient locomotion via learned gait transitions, in *Conference on Robot Learning* (MLResearchPress, 2022), pp. 773–783.
19. G. Bellegarda, K. Byl, Trajectory optimization for a wheel-legged system for dynamic maneuvers that allow for wheel slip, in *2019 IEEE 58th Conference on Decision and Control (CDC)* (IEEE, 2019), pp. 7776–7781.
20. L. Wellhausen, M. Hutter, Artplanner: Robust legged robot navigation in the field. arXiv:2303.01420 (2023).
21. J. Frey, D. Hoeller, S. Khattak, M. Hutter, Locomotion policy guided traversability learning using volumetric representations of complex environments, in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2022), pp. 5722–5729.
22. R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, A. Giusti, Learning ground traversability from simulations. *IEEE Robot. Autom. Lett.* **3**, 1695–1702 (2018).
23. V. Vapnik, R. Izmailov, Learning using privileged information: Similarity control and knowledge transfer. *J. Mach. Learn. Technol.* **16**, 2023–2049 (2015).
24. D. Chen, B. Zhou, V. Koltun, P. Krähenbühl, Learning by cheating, in *Conference on Robot Learning* (MLResearchPress, 2020), pp. 66–75.
25. J. M. Snider, "Automatic steering methods for autonomous automobile path tracking" (Tech. Rep. CMU-RITR-09-08, Robotics Institute, Pittsburgh, PA, 2009).
26. T. Miki, L. Wellhausen, R. Grandia, F. Jenelten, T. Homberger, M. Hutter, Elevation mapping for locomotion and navigation using GPU, in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2022), pp. 2273–2280.
27. 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 Robot. Autom. Lett.* **7**, 4630–4637 (2022).
28. CryEngine, Ai and navigation system - CryEngine 5 documentation; https://docs.cryengine.com/pages/viewpage.action?pageId=26869983.
29. Unreal Engine, Navigation system - Unreal Engine 5 documentation; https://docs.unrealengine.com/5.0/en-US/navigation-system-in-unreal-engine/.
30. M. Gumin, Wave function collapse algorithm; https://github.com/mxgmn/ (2016).
31. E. W. Dijkstra, A note on two problems in connexion with graphs. *Numer. Math* **1**, 269–271 (1959).
32. E. Jelavic, J. Nubert, M. Hutter, Open3d slam: Point cloud based mapping and localization for education, in *Robotic Perception and Mapping: Emerging Techniques, ICRA 2022 Workshop* (IEEE, 2022) p. 24.

33. Robotic Systems Lab, Team Cerberus wins the DARPA subterranean challenge; https://youtu.be/fCHOU-fw2c0?si=LjksAckgpSwfMqTC (2023).
34. J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, M. Hutter, Learning agile and dynamic motor skills for legged robots. *Sci Robot.* **4**, eaau5872 (2019).
35. 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 Trans. Mechatron.* **20**, 1117–1129 (2015).
36. R. B. McGhee, A. A. Frank, On the stability properties of quadruped creeping gaits. *Math. Biosci.* **3**, 331–351 (1968).
37. J. Lee, J. Hwangbo, M. Hutter, Robust recovery controller for a quadrupedal robot using deep reinforcement learning. arXiv:1901.07517 (2019).
38. L. Wellhausen, R. Ranftl, M. Hutter, Safe robot navigation via multi-modal anomaly detection. *IEEE Robot. Autom. Lett.* **5**, 1326–1333 (2020).
39. M. Sorokin, J. Tan, C. K. Liu, S. Ha, Learning to navigate sidewalks in outdoor environments. *IEEE Robot. Autom. Lett.* **7**, 3906–3913 (2022).
40. M. Kulkarni, M. Dharmadhikari, M. Tranzatto, S. Zimmermann, V. Reijgwart, P. De Petris, H. Nguyen, N. Khedekar, C. Papachristos, L. Ott, R. Siegwart, M. Hutter, K. Alexis, Autonomous teamed exploration of subterranean environments using legged and aerial robots, in *2022 International Conference on Robotics and Automation (ICRA)* (IEEE, 2022), pp. 3306–3313.
41. O. Nachum, H. Tang, X. Lu, S. Gu, H. Lee, S. Levine, Why does hierarchy (sometimes) work so well in reinforcement learning? arXiv:1909.10618 (2019).
42. D. Jain, K. Caluwaerts, A. Iscen, From pixels to legs: Hierarchical learning of quadruped locomotion, in *Proceedings of the 2020 Conference on Robot Learning*, J. Kober, F. Ramos, C. Tomlin, Eds. (MLResearchPress, 2021), pp. 91–102.
43. O. Nachum, S. S. Gu, H. Lee, S. Levine, Data-efficient hierarchical reinforcement learning, in *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett, Eds. (NeurIPS, 2018), pp. 3307–3317.
44. N. Rudin, D. Hoeller, M. Bjelonic, M. Hutter, Advanced skills by learning locomotion and local navigation end-to-end, in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2022), pp. 2497–2503.
45. A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, K. Kavukcuoglu, Feudal networks for hierarchical reinforcement learning, in *International Conference on Machine Learning* (MLResearchPress, 2017), pp. 3540–3549.
46. D. Jain, A. Iscen, K. Caluwaerts, Hierarchical reinforcement learning for quadruped locomotion, in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019), pp. 7551–7557.
47. V. Tsounis, M. Alge, J. Lee, F. Farshidian, M. Hutter, Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robot. Autom. Lett.* **5**, 3699–3706 (2020).
48. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms. arXiv:1707.06347 (2017).
49. P.-W. Chou, D. Maturana, S. Scherer, Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution, in *International Conference on Machine Learning* (MLResearchPress, 2017), pp. 834–843.
50. C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2017), pp. 652–660.
51. N. Rudin, D. Hoeller, P. Reist, M. Hutter, Learning to walk in minutes using massively parallel deep reinforcement learning, in *Conference on Robot Learning* (MLResearchPress, 2022), pp. 91–100.
52. A. A. Hagberg, D. A. Schult, P. J. Swart, Exploring network structure, dynamics, and function using networkx, in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, J. Millman, Eds. (SciPy, 2008), pp. 11–15.
53. M. Mueller, A. Dosovitskiy, B. Ghanem, V. Koltun, Driving policy transfer via modularity and abstraction, in *Proceedings of the 2nd Conference on Robot Learning*, A. Billard, A. Dragan, J. Peters, J. Morimoto, Eds. (PMLResearchPress, 2018), pp. 1–15.
54. A. Agarwal, A. Kumar, J. Malik, D. Pathak, Legged locomotion in challenging terrains using egocentric vision, in *Conference on Robot Learning* (MLResearchPress, 2023), pp. 403–415.
55. G. Kahn, P. Abbeel, S. Levine, Badgr: An autonomous self-supervised learning-based navigation system. *IEEE Robot. Autom. Lett.* **6**, 1312–1319 (2021).
56. G. Kahn, P. Abbeel, S. Levine, Land: Learning to navigate from disengagements. *IEEE Robot. Autom. Lett.* **6**, 1872–1879 (2021).
57. Y. Kim, C. Kim, J. Hwangbo, Learning forward dynamics model and informed trajectory sampler for safe quadruped navigation, in *Conference on Robotics-Science and Systems (RSS)* (Robotics: Science and Systems Foundation, 2022).
58. J. Truong, D. Yarats, T. Li, F. Meier, S. Chernova, D. Batra, A. Rai, Learning navigation skills for legged robots with learned robot embeddings, in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2021), pp. 484–491.
59. D. Hoeller, L. Wellhausen, F. Farshidian, M. Hutter, Learning a state representation and navigation in cluttered and dynamic environments. *IEEE Robot. Autom. Lett.* **6**, 5081–5088 (2021).
60. 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 Robot. Autom. Lett.* **3**, 4423–4430 (2018).
61. 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, in *Conference on Robotics-Science and Systems (RSS)* (Robotics: Science and Systems Foundation, 2020).
62. H. Wang, S. Chen, S. Sun, Diffusion model-augmented behavioral cloning. arxiv:/2302.13335 (2023).
63. N. Savinov, A. Dosovitskiy, V. Koltun, Semi-parametric topological memory for navigation, paper presented at The Sixth International Conference on Learning Representations, Vancouver, Canada, 30 April to 3 May 2018 (ICLR, 2018).
64. Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, D. Pathak, Coupling vision and proprioception for navigation of legged robots, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2022), pp. 17273–17283.
65. 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, paper presented at The Seventh International Conference on Learning Representations, 6 to 9 May 2019, New Orleans, USA (ICLR, 2019).
66. 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, in *2019 International Conference on Robotics and Automation (ICRA),* (IEEE, 2019), pp. 5993–6000.
67. E. Wijmans, M. Savva, I. Essa, S. Lee, A. S. Morcos, D. Batra, Emergence of maps in the memories of blind navigation agents. *AI Matters* **9**, 8–14 (2023).
68. K. Zhu, T. Zhang, Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Sci. Technol.* **26**, 674–691 (2021).
69. 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:2005.13857 (2020).
70. R. Yang, M. Zhang, N. Hansen, H. Xu, X. Wang, Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers, in *Deep RL Workshop NeurIPS 2021* (NeurIPS, 2021).
71. P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, A. Zamir, On evaluation of embodied navigation agents. arxiv:/1807.06757 (2018).
72. S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, I. Havoutis, Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control. *IEEE Transac. Robot.* **38**, 2908–2927 (2022).
73. H. Kolvenbach, D. Bellicoso, F. Jenelten, L. Wellhausen, M. Hutter, Efficient gait selection for quadrupedal robots on the Moon and Mars, in *14th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2018)* (ESA Conference Bureau, 2018).
74. P. N. Ward, A. Smofsky, A. J. Bose, Improving exploration in soft-actor-critic with normalizing flows policies. arXiv:1906.02771 (2019).
75. W. Zhou, S. Bajracharya, D. Held, Plas: Latent action space for offline reinforcement learning, in *Conference on Robot Learning* (MLResearchPress, 2021), pp. 1719–1735.
76. A. Allshire, R. Martin-Martin, C. Lin, S. Manuel, S. Savarese, A. Garg, Laser: Learning a latent action space for efficient reinforcement learning, in *2021 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2021), pp. 6650–6656.
77. D. P. Kingma, M. Welling, Auto-encoding variational bayes, paper presented at 2nd International Conference on Learning Representations, Banff, Canada, 14 to 16 April 2014 (ICLR, 2014).
78. L. Dinh, J. Sohl-Dickstein, S. Bengio, Density estimation using real nvp, paper presented at The Fourth International Conference on Learning Representations, 6 to 9 May 2016, Miami, USA (ICLR, 2016).
79. J. C. Brant, K. O. Stanley, Minimal criterion coevolution: A new approach to open-ended search, in *Proceedings of the Genetic and Evolutionary Computation Conference* (ACM, 2017), pp. 67–74.
80. Open-Ended Learning 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. arxiv:/2107.12808 (2021).
81. A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, D. Scaramuzza, Learning high-speed flight in the wild. *Sci. Robot.* **6**, eabg5810 (2021).
82. S. Ross, G. Gordon, D. Bagnell, A reduction of imitation learning and structured prediction to no-regret online learning, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (JMLR Workshop and Conference Proceedings, 2011), pp. 627–635.

Acknowledgments: We appreciate T. Tuna for helping us integrate Open3D SLAM, J. Keller for the API integration, and G. Valsecchi for the hardware support. A substantial part of the work

was carried out during the stay of M.B., A.R., and L.W. at the Robotic Systems Lab, ETH Zurich. **Funding:** This work was supported by the Mobility Initiative grant funded through the ETH Zurich Foundation, European Union's Horizon 2020 research and innovation program under grant agreement numbers 101070405 and 101016970, 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 numbers 852044 and 101070596, and Apple Inc. **Author contributions:** J.L. conceived the main idea for the approach and was responsible for the implementation and training of the controllers. J. L. trained the high-level policy, whereas 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 contributed to system integration and experimentation. **Conflict of interest:** The authors declare that they have no competing interests. **Data and materials availability:** All data needed to support the conclusions of this manuscript are included in the main text or the Supplementary Materials. The datasets and codes to generate Figs. 3, 6, and 7 are available at DOI: 10.5061/dryad.gxd2547tg.

Submitted 1 June 2023
Accepted 27 March 2024
Published 24 April 2024
10.1126/scirobotics.adi9641

Science Robotics

Learning robust autonomous navigation and locomotion for wheeled-legged robots

Joonho Lee, Marko Bjelonic, Alexander Reske, Lorenz Wellhausen, Takahiro Miki, and Marco Hutter

Sci. Robot. **9** (89), eadi9641. DOI: 10.1126/scirobotics.adi9641

Editor's summary

Last-mile deliveries in urban areas lead to increasing traffic problems and supply chain issues. By converting to robotic deliveries, routes can move from the street to alternate pathways, but traditional legged robots lack the necessary efficiency and battery life. By augmenting a legged robot with wheels, Lee *et al*. used hybrid wheeled-legged locomotion to improve the cost of transport while rolling on flat surfaces yet still allowing for legged navigation over obstacles like stairs. Using various reinforcement learning techniques, the robot was trained to smoothly transition between walking and driving modes and to navigate challenging terrain and dynamic obstacles including pedestrians. The robot was tested for large-scale navigation through an autonomous trek of more than 10 kilometers in Zurich, Switzerland, and Seville, Spain. The hybrid wheeled-legged platform successfully addressed many challenges associated with robotic urban mobility. —Melisa Yashinski

Science Robotics (ISSN 2470-9476) is published by the American Association for the Advancement of Science. 1200 New York Avenue NW, Washington, DC 20005. The title *Science Robotics* is a registered trademark of AAAS.