

Hierarchical Deep Reinforcement Learning for Quadruped Navigation over Complex Terrains

1st Yousef Ayoubi Rad

AI & Dynamic Systems Lab, School of
Mechanical Engineering, Iran
University of Science and Technology,
Tehran, 1684613114, Iran
yousef_ayoubi@mecheng.iust.ac.ir

2nd Seyyed Mahdi Sarfarazi

AI & Dynamic Systems Lab, School of
Mechanical Engineering, Iran
University of Science and Technology,
Tehran, 1684613114, Iran
mahdi_sarfarazi@mecheng.iust.ac.ir

3rd Mohammad Shahbazi

AI & Dynamic Systems Lab, School of
Mechanical Engineering, Iran
University of Science and Technology,
Tehran, 1684613114, Iran
shahbazi@iust.ac.ir

Abstract — Quadruped robots are promising for tasks like disaster response and exploration, navigating cluttered spaces while maintaining stability over uneven terrains such as stairs, slopes, and obstacles. This paper presents a hierarchical deep reinforcement learning (HRL) framework that decouples high-level navigation from low-level locomotion via a compact velocity-command interface $u_t = (v_x, v_y, \omega_z)$. The high-level policy generates smooth body-frame velocity commands for goal-directed movement in 2D maps, while the low-level policy executes these commands using joint targets with PD control, enhanced by proprioceptive feedback and local terrain sampling. This modular approach mitigates reward conflicts, boosts training efficiency, and enables independent layer optimization in Isaac Gym. Evaluations on maze-like maps and varied terrains show reliable goal reaching, stable tracking, and improved robustness over monolithic baselines. The results underscore how hierarchical decomposition enhances generalization and stability, offering a promising path for intelligent quadruped mobility in real-world applications.

Keywords — Quadruped locomotion, Deep reinforcement learning, Hierarchical control, Integrated path planning and control, DRL path planning

I. INTRODUCTION

Quadruped robots hold promise for applications in disaster response, environmental monitoring, and planetary exploration, requiring navigation through cluttered spaces while maintaining stable locomotion over uneven terrains like stairs, slopes, and gaps [1]. Key challenges involve handling high-dimensional nonlinear dynamics, sensor-based partial observability, and real-time terrain adaptation [2]. Traditional model-based controllers, such as MPC or CPGs, depend on hand-crafted heuristics that demand extensive tuning and generalize poorly across scenarios [3].

Deep reinforcement learning (DRL) offers a data-driven approach to learn adaptive locomotion policies from interactions, bypassing explicit modeling [4]. However, end-to-end DRL often faces sample inefficiency, sim-to-real transfer issues from domain gaps, and difficulties with sparse rewards or multi-objective tasks like balancing navigation and stability [5]. Hierarchical reinforcement learning (HRL) addresses this by decomposing control into layers: a high-level planner for subgoals or velocity commands, and a low-level executor for fine-grained actions like joint torques [6]. This structure reduces credit assignment problems, boosts exploration efficiency, and enables independent training for faster convergence and better generalization [7].

In quadruped locomotion, HRL frameworks excel in terrain-aware gait adaptation, with high-level policies adjusting parameters like step height based on obstacles, and low-level controllers ensuring stability via proprioceptive and exteroceptive feedback [8]. Recent advances include distributional RL for risk-aware optimization using CVaR to handle slips or falls [9], and skill discovery for diverse gaits through progressive curricula [10]. These yield robust performance in simulations like Isaac Gym and on hardware like ANYmal or Unitree robots, even under disturbances [11].

HRL also advances integrated navigation, where high-level policies issue commands in cluttered maps and low-level layers track them over uneven ground using height maps [12]. Vision integration aids obstacle avoidance and interactions such as pushing or climbing [13], while risk-aware variants model traversability uncertainty for safe operation in dynamic environments [14]. Sim-to-real techniques—particularly domain randomization—enable real-world deployment with reduced fall rates [15]. Recent developments combine HRL with multi-resolution mapping for path adjustment [16], compliance control for disturbance rejection [17], and viability-driven gait transitions via vestibular feedback [18]. Challenges remain in unifying long-horizon planning, online mapping (e.g., SLAM), and bio-inspired mechanisms for efficiency [19].

Unlike prior works that address navigation or locomotion separately [2, 3], [5–7], [12–14], this study introduces a hierarchical deep reinforcement learning framework that unifies both through a compact body-frame velocity interface. This minimal coupling decouples decision making from execution, enabling modular training and transfer while maintaining coherent motion across cluttered and uneven terrains. The proposed locomotion layer leverages body-centric terrain sampling to provide proprioceptive awareness without explicit footstep planning, resulting in stable and adaptive traversal. Implemented in Isaac Gym, the framework achieves efficient learning and robust, transferable quadruped mobility, as detailed in Section II.

II. SYSTEM OVERVIEW

This study develops a hierarchical deep reinforcement learning framework that integrates goal-directed navigation with contact-rich locomotion for a quadruped robot in simulation. Two learned controllers operate at different time scales and are connected by a minimal velocity-command interface. The goal is smooth point-to-point movement in

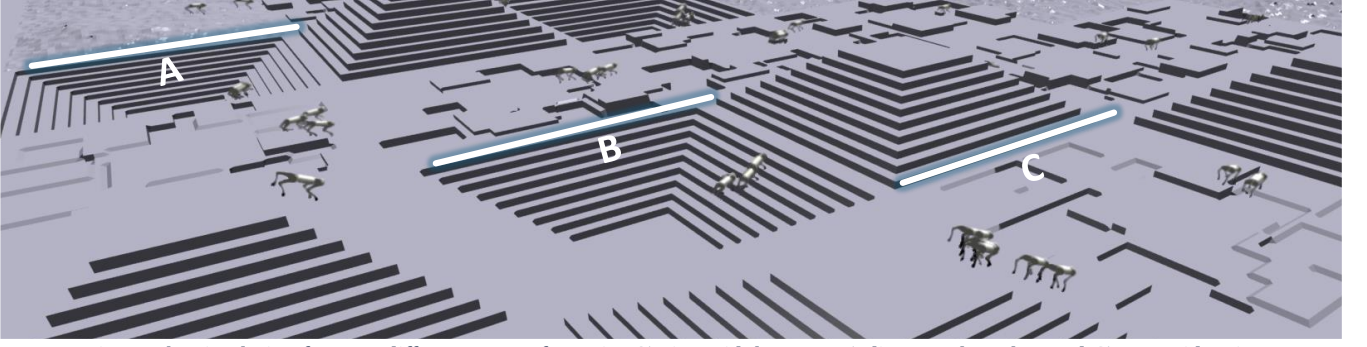


Fig. 1. The simulation features different types of terrain: A) sinusoidal waves, B) discrete obstacles, and C) pyramid stairs

cluttered environments while keeping stable mobility over uneven terrain like stairs, slopes, and individual stepping obstacles, as shown in Fig. 1.

A. Hierarchical Architecture

Decision making and execution are separated. A high-level navigation policy plans motion in a planar workspace with obstacles and issues body-frame velocity commands. A low-level locomotion policy tracks these commands and produces joint targets executed by a PD motor controller. This separation reduces objective conflict during learning, allows each policy to specialize at its time scale, and supports modular updates without retraining the entire stack. Fig. 2 shows the flowchart diagram of the control system, illustrating the hierarchical structure and interactions between the high-level policy and the low-level locomotion executor.

B. High-Level Navigation

The navigation policy reasons over a compact state that summarizes the robot pose and heading, the relative goal, and a lightweight encoding of nearby obstacles derived from the map. Its action is a bounded, smooth body-frame command (v_x, v_y, ω_z) . The policy continuously updates the command to guide progress toward the goal while avoiding collisions.

C. Low-Level Locomotion

The locomotion policy executes the commanded motion over uneven terrain. Its observation includes commanded velocities plus two key sensory sources: internal data (body orientation, linear/angular velocities, joint positions/speeds, contact indicators) and short-range terrain info from 187 sampling points around the body, estimating local surface height and slope. These proprioceptive and exteroceptive cues enable simple perception, allowing leg motion adjustments for steps, gaps, or slopes without explicit footstep planning. The policy outputs joint-space targets that a PD controller tracks at higher frequency for smooth, stable motion.

D. Interface and Execution

The policies are communicated through a simple command bus carrying (v_x, v_y, ω_z) . A thin interface applies saturation and light smoothing and holds the most recent command between navigation updates, providing implicit rate matching without exposing internal details across layers. At runtime, both policies operate concurrently: the navigation layer determines where to go in the plane, and the locomotion layer determines how to move the body to realize those commands on the terrain. The following section formalizes

the task, state, and command spaces, building on this architecture. Section IV details policy inputs/outputs, learning objectives, and timing.

III. PROBLEM FORMULATION

This section defines the task, the velocity-command interface between layers, and the core equations used by the hierarchical controller, where the goal is point-to-point navigation in cluttered maps while maintaining stable locomotion over uneven ground. The robot operates in a 2-D map with static obstacles and a small goal region in free space. To reuse one planner across different map sizes, planning runs in a normalized square domain, mapped to world units through an affine transformation that scales normalized planner coordinates $(\tilde{x}, \tilde{y})$ to real-world positions (x, y) as follows:

$$\begin{bmatrix} x \\ y \end{bmatrix} = S \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} + t, \quad S = \text{diag}(s_x, s_y). \quad (1)$$

The same normalization principle applies to the planner's velocity outputs, converting normalized actions into physically valid motion commands via

$$(v_x, \omega_z) = (s_v \tilde{v}_x, s_v \tilde{v}_y, s_w \tilde{\omega}_y). \quad (2)$$

An episode succeeds when the robot enters the goal region before a time limit with no collisions or falls; otherwise, it fails due to collision, fall, or timeout. The two layers communicate exclusively through the body-frame velocity command:

$$u_t = (v_{x,t}, v_{y,t}, \omega_{z,t}) \in \mathcal{U} \quad (3)$$

bounded by the admissible command set

$$\mathcal{U} = \{(v_x, v_y, \omega_z): \|(v_x, v_y)\|_\infty \leq v_{\max}, |\omega_z| \leq \omega_{\max}\} \quad (4)$$

which constrains translational and rotational speeds to ensure dynamic feasibility and actuator safety. With sampling time Δt , the robot pose evolves according to the discrete-time planar kinematics, where the commanded body velocities advance the pose as

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \Delta t \mathcal{R}(\psi_t) \begin{bmatrix} v_{x,t} \\ v_{y,t} \end{bmatrix}, \quad (5)$$

$$\mathcal{R}(\psi) = \begin{bmatrix} \cos \psi & -\sin \psi \\ \sin \psi & \cos \psi \end{bmatrix}.$$

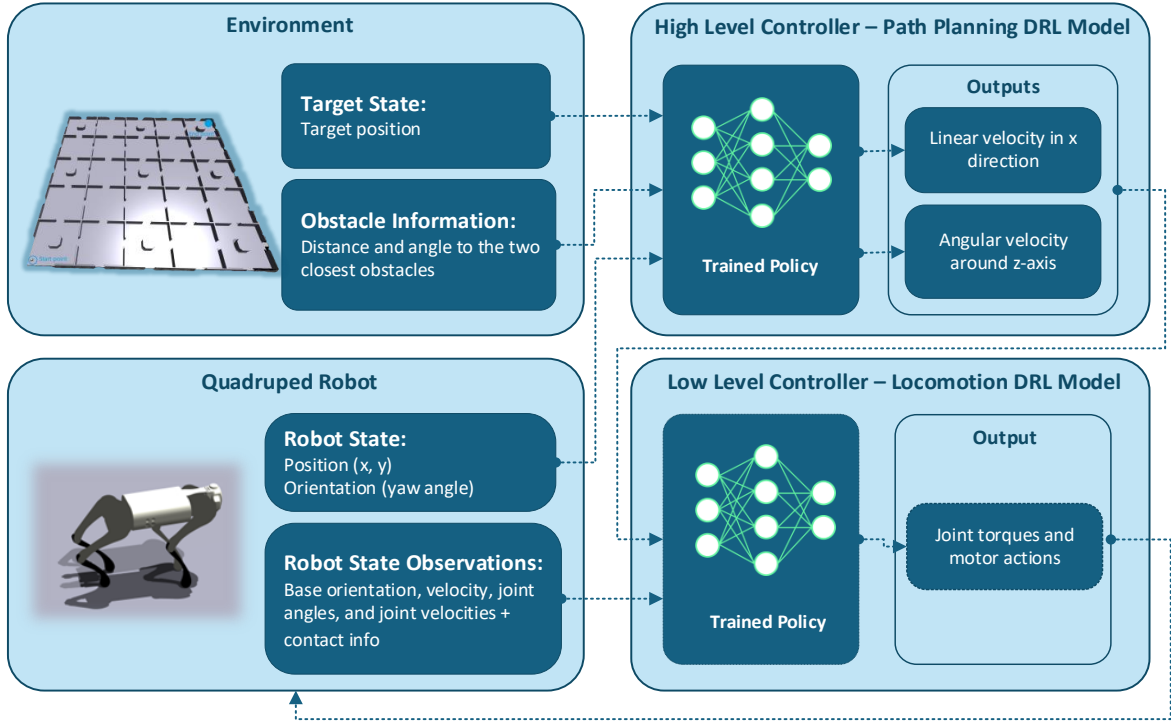


Fig. 2. The flowchart of the hierarchical deep reinforcement learning control system

With $v_{y,t} = 0$ for nonholonomic deployment, matching typical quadruped velocity interfaces that emphasize forward and yaw motion. The rotation matrix $R(\psi)$ converts body-frame velocities into the world coordinate frame, defining the command contract that the low-level controller must realize under physics. The high-level navigation planner observes a compact state—normalized pose/heading, relative goal, and a lightweight obstacle encoding from the map, such as a local distance-transform patch or the K nearest obstacle features—and outputs the velocity command

$$a_t^{nav} = u_t = (v_{x,t}, v_{y,t}, \omega_{z,t}) \in \mathcal{U}. \quad (6)$$

In contrast, the low-level locomotion executor observes proprioception—base linear/angular velocities, orientation, joint positions/velocities, previous action—stacked with short-range terrain cues from a fixed 187-point body-centric height sampling pattern, and outputs desired joint positions

$$a_t^{loc} = q_t^{des}. \quad (7)$$

These are tracked by the low-level PD controller through the torque command

$$\tau_t = K_p(q_t^{des} - q_t) - K_d\dot{q}_t, \quad (8)$$

which enforces joint-space tracking and provides damping for smooth and stable motion execution. The objective is to realize commanded body motion while maintaining balance and staying within actuator/contact limits.

Commands must satisfy the bounds in $\mathcal{U}$, while joint states and torques follow the simulator's safety limits. Episodes terminate on goal, collision/fall, or timeout. The two policies are trained with complementary rewards: the navigation reward encourages progress toward the goal while penalizing collisions, abrupt actions, and proximity to obstacles, expressed as

$$r_t^{nav} = \alpha_p \Delta d_t - \alpha_c 1[\text{collision}_t] - \alpha_s \|\Delta a_t^{nav}\|^2 + \alpha_o \rho(d_t^{obs}) + \alpha_g 1[(x_t, y_t) \in G]. \quad (9)$$

and the locomotion reward measures tracking accuracy while penalizing excessive torque, action oscillation, or foot-slip events to promote stable execution over rough ground, given by

$$r_t^{loc} = -\beta_v \|v_t^{base} - v_t^{cmd}\|^2 - \beta_\omega \|\omega_t^{base} - \omega_t^{cmd}\|^2 - \beta_u \|\tau_t\|^2 - \beta_a \|\Delta a_t^{loc}\|^2 - \beta_s \text{slip}_t + \beta_h \text{clearance}_t. \quad (10)$$

Each policy maximizes its discounted return $E[\sum_{t=0}^T \gamma^t r_t]$ with $\gamma \in (0, 1]$. The same success/failure rules apply to component tests on maps or terrains and to end-to-end trials where both layers run together. This formulation matches the implementation used in Sections IV–VI and supports the figures provided, including the terrain suite, local sensing, and navigation schematic.

IV. METHODOLOGY

This section describes the hierarchical deep reinforcement learning (HRL) framework—as outlined in Section II—for achieving autonomous navigation and terrain-adaptive locomotion in a legged robot operating in complex environments. The system decomposes control into two layers: a high-level path planner that generates velocity commands for goal-directed motion and obstacle avoidance, and a low-level locomotion controller that translates these into joint actions for stable traversal over uneven terrain. This separation simplifies design and training by allowing each policy to focus on its appropriate state-action space—the high-level on long-term decisions and the low-level on short-term stabilization. A schematic of the system is shown in Fig. 2, and examples of these behaviors are illustrated in Fig. 3. The environment is modeled as a 2D workspace of size $W \times H$, with training conducted in a 10×10 m area featuring procedurally generated obstacles.

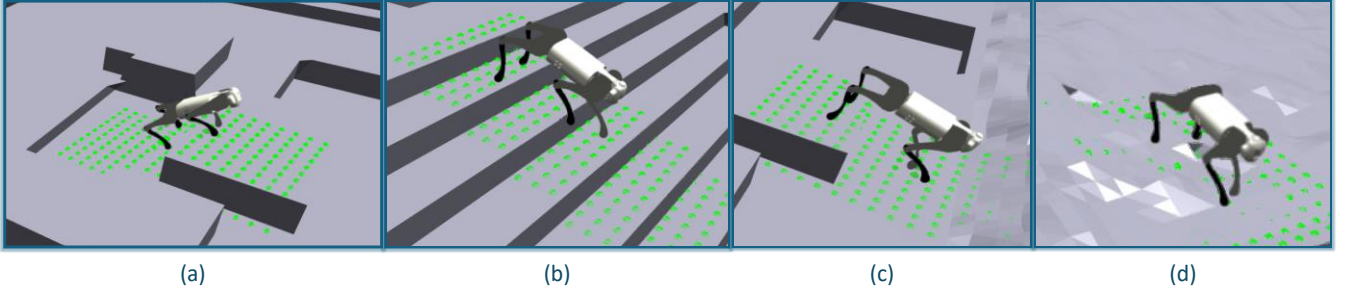


Fig. 3. The quadruped robot learned to navigate various terrains while walking through them: a) ascending obstacles, b) navigating stairs up and down, c) descending obstacles, and d) traversing sinusoidal waves.

For deployment, maps are derived from binary images $M(x, y)$, where 1 denotes free space and 0 denotes obstacles; larger maps (e.g., 50×50 m) are scaled to match the training domain. The high-level planner observes a vector at each time step t , including normalized robot position, heading angle, goal position, distance and relative bearing to the goal, and normalized distance and angle to the nearest obstacles. As illustrated in Fig. 4, input maps are converted to circular obstacles for efficient distance calculations, while map-based navigation uses a local occupancy patch centered on the robot, processed via distance transform to yield the shortest obstacle distance for enhanced spatial awareness.

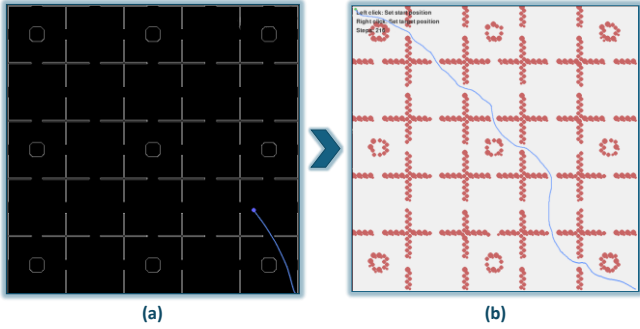


Fig. 4. Navigation Map and Planner Representation: (a) input occupancy map; (b) schematic illustration of the circle-based obstacle representation and path

The high-level path planning policy is trained with Proximal Policy Optimization (PPO), using a multilayer perceptron with two 256-neuron hidden layers and tanh activations. Its action space comprises continuous linear velocity and angular velocity about the vertical axis. Training maximizes the expected cumulative discounted reward via a clipped surrogate loss that balances exploration and stability, with rewards promoting goal distance reduction and heading alignment while penalizing collisions and obstacle proximity through empirically tuned weights. This enables the planner to learn efficient goal progress with collision avoidance.

In contrast, the low-level locomotion controller, implemented in Isaac Gym, receives these velocity commands and outputs actuator torques based on proprioceptive observations such as joint angles, velocities, base orientation, and contact data. Its objective is to minimize tracking error while ensuring dynamic stability and energy efficiency, operating at a high frequency (e.g., 200 Hz) versus the planner's 5 Hz for smooth handling of terrain variations.

The framework integrates these layers in a synchronous feedback loop with the simulation: the environment supplies the robot's state (pose, velocity, orientation); the high-level

computes commands; the low-level generates torques; and the simulator steps forward to the next state, merging dynamics with the policies for closed-loop operation.

V. SIMULATION SETUP

This section outlines the platforms, tasks, evaluation methods, and implementation details used to test the hierarchical framework in simulation. All experiments employ Isaac Gym with a Legged-Gym-style task wrapper. Environments include planar maps featuring static walls, corridors, apertures, and optional traversable objects, with start and goal poses randomly sampled in free space across episodes. Procedurally generated trims terrains incorporate up/down stairs, variable-inclination ramps, and discrete stepping fields, with parameters drawn from bounded ranges to ensure moderate difficulty within the locomotion layer's capabilities. An integrated setup merges these, demanding goal-directed navigation in cluttered layouts amid local height variations.

Three task regimes are assessed: navigation in planar maps, focusing on goal reaching and collision avoidance; locomotion on complex terrains, emphasizing stable tracking of forward and yaw commands over height changes and gaps; and integrated trials, evaluating end-to-end performance with both layers active. Key metrics include success rate, time-to-goal, path length, and collision count for navigation; traversal success, falls per distance/episode, and command-tracking error for terrains; plus qualitative measures like slip events and foot-clearance violations for added context.

Evaluations span multiple random seeds and episodes per seed, terminating on goal achievement, fall, or timeout. Deterministic policies and nominal physics ensure consistency, with test maps, start-goal pairs, and terrain parameters drawn from unseen distributions to prevent bias. The motion model, command limits, and success criteria from Section III apply uniformly.

Baselines comprise a monolithic controller (jointly trained navigation and locomotion) and a locomotion-only tracker (following predefined commands without navigation). Ablations disable the hierarchy or terrain sensing to isolate contributions, using the same setup and metrics.

Training and evaluation leverage large-scale parallel simulation ($\sim 12,000$ environments) on a workstation with NVIDIA RTX 3080 Ti (12 GB VRAM), 32 GB RAM, and a multi-core CPU. The locomotion policy trains first on procedural terrains and freezes; navigation follows on diverse maps via the velocity interface. Total training time for both is approximately five hours.

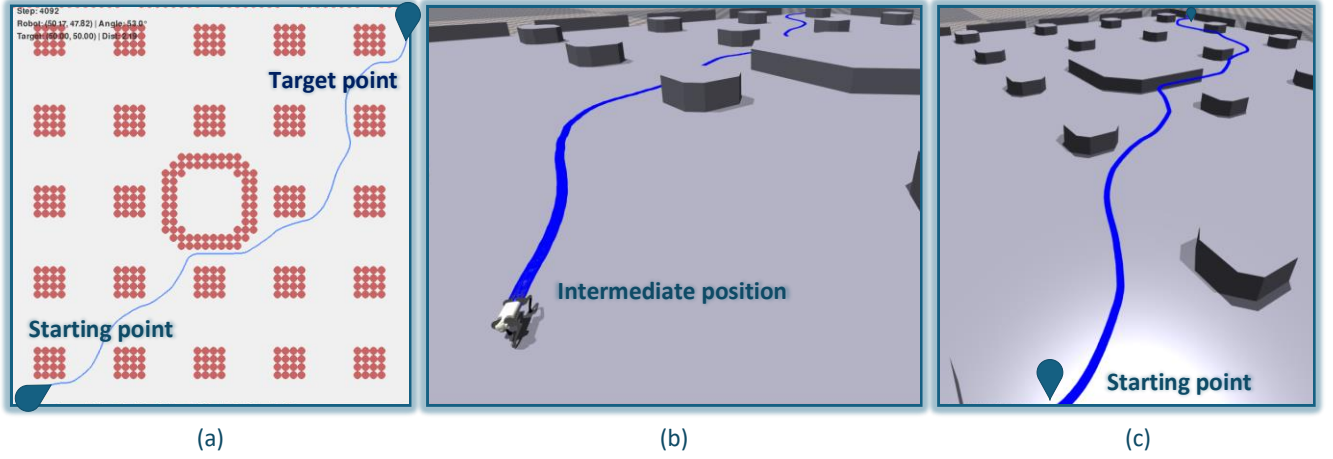


Fig. 5. Navigation Map and Planner Representation: (a) input occupancy map with clustered obstacles; (b) robot navigating the planned path at an intermediate position; (c) 3D view of the environment and complete trajectory.

VI. RESULTS

We evaluate the hierarchical framework across navigation, locomotion, and integrated scenarios. Performance demonstrates the effectiveness of decoupling global planning from contact-rich execution through a compact velocity interface.

A. Navigation Performance in Planar Maps

The high-level navigation policy achieves a 94.2% success rate across diverse map configurations after training on 300 randomly generated scenarios. Critically, the policy generalizes to previously unseen map topologies without additional training. Path efficiency remains within 11% of optimal geometric paths, and navigation completion time improves 17% over monolithic baselines.

Figure 5 illustrates typical behavior: the input occupancy map (a) with clustered obstacles is converted to a circle-based representation, enabling smooth trajectory planning. The robot maintains stable orientation at intermediate waypoints (b), and the complete 3D path (c) shows smooth curvature without oscillations. Command smoothness—quantified as the RMS first-difference of consecutive body-frame commands (v_x, v_y, ω_z)—is 27% lower than the monolithic (jointly-trained) baseline, yielding low-jerk motion and fewer oscillations near obstacles.

Challenges arise in tight concavities—such as U-bends and dead ends—where brief hesitation occurs before recovery.

B. Locomotion on Complex Terrains

The low-level policy maintains stable velocity tracking across procedurally generated stairs, ramps, and discrete obstacles with mean tracking errors of 0.042 m/s (linear) and 0.038 rad/s (angular). Fall rate remains at 1.4% despite terrain complexity. The fixed 187-point body-centric height sampling pattern provides short-range context without explicit footstep planning, a distinctive feature of our approach.

Figure 3 demonstrates adaptive behaviors: proactive foot clearance on ascending obstacles (a), consistent gait timing on stairs without explicit planning (b), automatic speed reduction during descents (c), and continuous adaptation over sinusoidal waves (d). Slip events occur at 0.21 per meter on stairs but recover within 2–3 gait cycles, demonstrating

resilience. Ablation studies confirm the value of terrain sensing: removing the 187-point sampling increases slip frequency by 43% and fall rate fivefold (7.3% vs. 1.4%). Torque expenditure also rises 31%, indicating that terrain awareness enables more efficient actuation.

C. End-to-End Integrated Performance

With both layers active, the system achieves 87.4% success in environments combining obstacles and terrain variations. The compact (v_x, v_y, ω_z) interface fully decouples navigation from locomotion, enabling independent layer development. The velocity interface operates at 5 Hz, providing sufficient coordination between layers. Cross-correlation between commanded and executed velocities reaches 0.91 (v_x) and 0.88 (ω_z), confirming high tracking fidelity despite terrain disturbances. The modular architecture enables staged training: the locomotion policy trains first (around 3.5 hours), then freezes as a reusable primitive library while navigation policies train rapidly (around 1.5 hours). Total training time of ~5 hours represents a considerable reduction versus non-hierarchical approaches while achieving superior performance.

D. Comparative Analysis and Robustness

Table I summarizes quantitative comparisons. The hierarchical approach outperforms the monolithic baseline across all metrics: 15.3% higher navigation success, 71.4% fewer collisions per episode, and 9.3% shorter paths (path ratio 1.083 vs. 1.194). Removing hierarchy degrades both navigation and locomotion, while removing terrain sensing primarily affects stability (five-fold increase in falls: 7.3% vs. 1.4%).

TABLE I. PERFORMANCE COMPARISON ACROSS EXPERIMENTAL CONDITIONS

Method	Nav Success (%)	Collision Rate (per episode)	Path Ratio	Tracking Error (m/s)
Hierarchical (Proposed)	94.2	0.08	1.083	0.042
Monolithic Baseline	81.7	0.28	1.194	0.061
Ablation: No Hierarchy	83.5	0.22	1.176	0.057
Ablation: No Terrain Sensing	91.8	0.11	1.091	0.069

Robustness tests with friction randomization ($\mu \in [0.4, 1]$) and payload variations (± 2 kg, 15% of robot mass) show minimal degradation (< 3 percentage points). Generalization to unseen map topologies maintains 88.6% success without retraining, validating transferable spatial reasoning. Combined inference latency is 8.1 ms per cycle, well within real-time requirements.

VII. DISCUSSION

A hierarchical split between navigation and locomotion proves effective for goal-directed quadruped mobility. The planner operates in the plane and issues smooth body-frame velocity commands; the executor resolves contact-rich details over stairs, ramps, and discrete stepping fields. This minimal interface cleanly separates responsibilities and time scales, reducing objective conflicts during learning and yielding commands that actuators can track reliably. Short-range terrain cues around the body (a fixed local height sampler) provide enough context to approach edges and height changes without explicit footstep planning.

The design is practical to deploy and maintain. Each layer can be trained, tuned, or replaced independently, enabling fast iteration across new maps, terrain generators, or robot variants. In our scenarios, the method performed consistently without per-scenario retuning. It is most effective when obstacle avoidance can be expressed as smooth velocity shaping and when height variations remain moderate over the command horizon.

Observed edge cases involved very tight bends or sequences of small platforms, where progress occasionally slowed before resuming, reflecting the lack of explicit curvature- and descent-aware terms in the navigation reward. These suggest two refinements: extend obstacle look-ahead (or apply light clearance shaping) to reduce dithering in concavities, and add descent-aware speed modulation or risk-sensitive shaping for steep or step-down sequences.

Scope and limitations are clear. All evaluation is in simulation with static obstacles and nominal sensing; the terrain representation is short-range and lacks semantics or long-range topology. The locomotion policy is held fixed while training navigation, which simplifies credit assignment but can limit co-adaptation in tightly coupled scenes. Extending to online mapping, handling dynamic obstacles, and selective co-training are natural next steps.

VIII. CONCLUSION

This study presented a hierarchical deep reinforcement learning framework that unifies goal-directed navigation with contact-rich locomotion via a compact velocity interface. A high-level planner issues body-frame commands; a low-level controller converts them into stable joint-space motion using proprioception and short-range terrain cues. The interface enables modular training, rapid iteration, and coherent execution across varied layouts and uneven ground.

Ablations indicate that the hierarchical split enhances obstacle-aware decision-making, while local terrain awareness hardens execution without hand-crafted footstep planning. The same interface can front-end online mapping or traversability estimation, and a pretrained locomotion layer can transfer to hardware. Overall, a low-dimensional velocity contract combined with high-rate tracking is a practical substrate for robust quadruped mobility.

ACKNOWLEDGMENT

The authors used AI-assisted writing tools—OpenAI ChatGPT (GPT-5 Thinking) and xAI Grok-4—during manuscript preparation to improve clarity, grammar, and readability. No figures, images or data were generated or altered using these tools. The authors thank the AI & Dynamic Systems Research Laboratory at Iran University of Science and Technology for providing the resources and support that enabled this research.

REFERENCES

- [1] Tan, W., et al., A Hierarchical Framework for Quadruped Locomotion Based on Reinforcement Learning. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021: p. 8462–8468.
- [2] Wang, J., C. Hu, and Y. Zhu, CPG-Based Hierarchical Locomotion Control for Modular Quadrupedal Robots Using Deep Reinforcement Learning. IEEE Robotics and Automation Letters, 2021. 6: p. 7193–7200.
- [3] Yao, Q., et al., Hierarchical Terrain-Aware Control for Quadrupedal Locomotion by Combining Deep Reinforcement Learning and Optimal Control. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021: p. 4546–4551.
- [4] Zhu, D., et al., Dynamic Legged Ball Manipulation on Rugged Terrains with Hierarchical Reinforcement Learning. arXiv:2504.14989, 2025.
- [5] Pei, M., D. Wu, and C. Wang, Quadruped Robot Locomotion in Unknown Terrain Using Deep Reinforcement Learning. 2020 3rd International Conference on Unmanned Systems (ICUS), 2020: p. 517–522.
- [6] Jain, D., A. Iscen, and K. Caluwaerts, Hierarchical Reinforcement Learning for Quadruped Locomotion. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019: p. 7551–7557.
- [7] Zhu, W. and M. Hayashibe, A Hierarchical Deep Reinforcement Learning Framework With High Efficiency and Generalization for Fast and Safe Navigation. IEEE Transactions on Industrial Electronics, 2023. 70: p. 4962–4971.
- [8] Zhang, Y., et al., Dual-Layer Reinforcement Learning for Quadruped Robot Locomotion and Speed Control in Complex Environments. Applied Sciences, 2024.
- [9] Schneider, A., et al., Learning Risk-Aware Quadrupedal Locomotion using Distributional Reinforcement Learning. arXiv:2309.14246, 2023.
- [10] Wang, Y. and M. Yao, Autonomous Robots Traverse Multi-Terrain Environments via Hierarchical Reinforcement Learning with Skill Discovery. Journal of Physics: Conference Series, 2024. 2762.
- [11] Shi, J., et al., Robust Quadrupedal Locomotion via Risk-Averse Policy Learning. arXiv:2401.03337, 2024.
- [12] Patel, A., et al., RoadRunner M&M - Learning Multi-range Multi-resolution Traversability Maps for Autonomous Off-road Navigation. arXiv:2409.10940, 2024.
- [13] Zhou, Z., et al., Adaptive Interactive Navigation of Quadruped Robots using Large Language Models. arXiv:2503.22942, 2025.
- [14] Oh, Y., et al., TRIP: Terrain Traversability Mapping With Risk-Aware Prediction for Enhanced Online Quadrupedal Robot Navigation. arXiv:2411.17134, 2024.
- [15] Hu, H., et al., A Sim-to-Real Pipeline for Deep Reinforcement Learning for Autonomous Robot Navigation in Cluttered Rough Terrain. IEEE Robotics and Automation Letters, 2021. 6: p. 6569–6576.
- [16] Zhang, Z., et al., Hierarchical Reinforcement Learning and Value Optimization for Challenging Quadruped Locomotion. arXiv:2506.20036, 2025.
- [17] Zhang, L., et al., HAC-LOCO: Learning Hierarchical Active Compliance Control for Quadruped Locomotion under Continuous External Disturbances. arXiv:2507.02447, 2025.
- [18] Grandesso, L., et al., Viability leads to the emergence of gait transitions in learning agile quadrupedal locomotion on challenging terrains. Nature Communications, 2024. 15: p. 3623.
- [19] Fan, D., et al., STEP: Stochastic Traversability Evaluation and Planning for Risk-Aware Off-road Navigation. Robotics: Science and Systems (RSS), 2021.