Deep Reinforcement Learning for Efficient Navigation of the Moose Robot

Constança Fernandes
Department of Computer Science
University of Porto, Portugal
up202205398@edu.fc.up.pt

Isabel Sá
Department of Computer Science
University of Porto, Portugal
up202205388@edu.fc.up.pt

Pedro Silva
Department of Computer Science
University of Porto, Portugal
up202205150@edu.fc.up.pt

Abstract—**This paper presents a deep reinforcement learning (DRL) approach for efficient navigation of the Moose robot in complex environments with unknown rough terrain. We train an Advantage Actor-Critic (A2C) model to optimise the robot's trajectory while maintaining navigation performance. The proposed solution is validated in simulation using diverse terrain maps, and its effectiveness is compared with traditional planning algorithms such as Greedy and A*. Although the DRL agent was able to perform well in simpler settings, it consistently failed to learn an effective policy in more complex maps. The main challenges identified include reward sparsity—due to the low frequency of successful goal-reaching episodes—and premature episode terminations caused by the robot being detected as stuck. In contrast, traditional planners such as A* showed robust performance. These results suggest that while DRL can capture basic navigation behaviors, more intricate tasks require additional strategies such as curriculum learning or transfer learning from simpler environments.**

Index Terms—**Reinforcement Learning, Navigation, Rough Terrain, A2C, Moose Robot**

I. Introduction

Mobile robotic navigation in unstructured and rough terrains remains a formidable challenge due to the inherent trade-offs between robustness, efficiency, and adaptability. Real-world environments often present complex, uneven surfaces with varying slopes, obstacles, and uncertain conditions that severely limit the applicability of classical path planning methods. These traditional algorithms, while efficient in structured or well-mapped domains, struggle to maintain performance and safety when faced with irregular terrains or incomplete sensory information.

The Moose robot, a versatile mobile platform equipped with multimodal sensors including GPS, IMU, and LIDAR, offers an ideal testbed for advancing autonomous navigation research in such difficult scenarios. Its mobility capabilities enable exploration of novel control strategies that leverage rich perceptual inputs to negotiate rough terrain effectively.

Recent breakthroughs in deep reinforcement learning (DRL) have demonstrated promising results in enabling agents to learn complex navigation policies from raw sensory data, without explicit modeling of terrain or environmental dynamics. DRL techniques inherently balance exploration and exploitation, adapting to varying conditions by optimizing reward-driven behavior. This capability is particularly valuable for navigating uncertain and dynamic environments where precomputed paths may become invalid or unsafe.

This work aims to develop and rigorously evaluate a DRL-based navigation framework for the Moose robot that autonomously reaches target locations by optimally planning trajectories that maximize safety, energy efficiency, and traversal speed. By integrating terrain-aware perception, risk-sensitive planning, and continuous control through an Advantage Actor-Critic (A2C) algorithm, our approach seeks to overcome the limitations of purely geometric or heuristic path planners.

The contributions of this study are threefold: (i) the implementation of a synthetic terrain generation pipeline based on Perlin noise to simulate diverse and realistic topographies for training and evaluation; (ii) the integration and comparative analysis of classical (Greedy and A*) and learning-based navigation methods within a consistent simulation framework; (iii) a detailed examination of DRL policy robustness and generalization across varying terrain complexities and randomized navigation tasks.

Through extensive experiments in a high-fidelity simulation environment, this research provides new insights into the capabilities and limitations of DRL for autonomous rough-terrain navigation, paving the way for practical deployment of mobile robots in challenging outdoor scenarios

The remainder of this paper is organised as follows: Section II reviews related work in robotic navigation and deep reinforcement learning; Section III describes the methodology adopted, including the simulation environment and the A2C algorithm; Section IV presents the experiments and results; Section V discusses the findings; and finally, Section VI concludes the study and suggests directions for future work.

II. Related Work

Several approaches have been proposed for mobile robot navigation, including traditional path planners such as A* and greedy heuristics. These classical algorithms often rely on complete and accurate terrain information and do not explicitly optimise trajectory quality in uncertain or uneven environments. More recently, deep reinforcement learning (DRL) has shown great promise in addressing complex control tasks for navigation in rough terrain.

Notably, Zhang et al. [6] introduced a DRL-based framework for robot navigation over unknown rough terrain, demonstrating the potential of neural policy learning to adapt to complex and uncertain environments without explicit prior knowledge of the terrain. This approach enables autonomous agents to learn navigation policies that implicitly encode safety and efficiency considerations.

Subsequent works have incorporated terrain classification and slip estimation through deep learning methods to enhance robot perception and control [2]. Others have modeled terrain traversability and energy consumption of legged robots using inverse reinforcement learning, enabling more energy-efficient locomotion policies [1].

Recent advances also include the development of dynamic slip cost mapping techniques to improve autonomous navigation safety and efficiency [4]. Additionally, domain randomization techniques have been evaluated to improve the generalization capabilities of DRL locomotion policies trained in simulation for real-world applications [3].

Reliable planning approaches combining terrain awareness and DRL have also been proposed to improve navigation robustness in uneven outdoor environments [5].

Our approach integrates terrain-aware perception, risk-aware planning, and deep reinforcement learning into a unified framework to optimise navigation safety and efficiency in uneven environments, overcoming limitations of previous methods

III. Methodological Approach

In this section, we describe the experimental setup and the navigation strategies evaluated in our study. The Moose robot operates in a simulated uneven terrain environment, where each episode consists of a navigation task from a start point to a goal point. To ensure a fair and generalizable evaluation, the start and goal positions are randomly assigned at the beginning of every episode, for all tested methods—including Greedy, A*, and A2C-based reinforcement learning. This randomized configuration prevents overfitting to specific trajectories and promotes the development of robust navigation policies.

A. Terrain Generation Using Perlin Noise

To support the evaluation of navigation strategies in diverse topographical conditions, terrain surfaces are procedurally synthesised using Perlin noise — a gradient-based method known for generating smooth and naturalistic elevation profiles. Perlin noise is preferred over other noise models such as white noise or Simplex noise because it produces continuous, coherent gradients that mimic natural terrain features without abrupt transitions or overly repetitive patterns.

The simulation environment is built within Webots and ... incorporates a 200×200 cells grid that defines a three-dimensional surface. Terrain generation is based on a custom implementation of 2D Perlin noise, configured with a single octave and no bias correction. The resulting noise values are scaled to improve terrain characteristics, producing realistic elevation differences suitable for evaluating the robot's mobility and navigation performance.

To integrate the generated elevation data, the world model is modified programmatically, replacing the standard height map with the synthesised terrain. Importantly, the region immediately surrounding the robot's initial position is flattened during a post-processing step to avoid initial instability, such as unwanted tipping or sliding, at the start of each episode. The robot's initial coordinates are then adjusted to match this modified elevation, ensuring consistency between the terrain and the spawn location.

The procedural generation pipeline includes tools for analysing and visualising terrain characteristics. Statistical summaries — such as mean, standard deviation, and elevation range — are calculated to assess the spatial variability of each generated map. In addition, elevation surface heatmaps are produced using perceptually uniform colour gradients, allowing for quick visual inspection of terrain diversity.

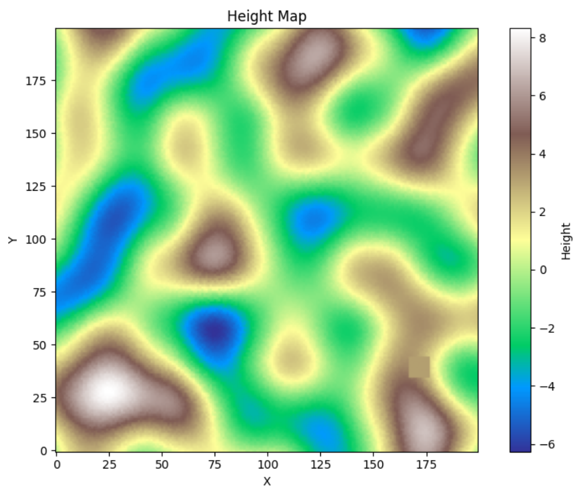

Fig. 1. Heatmap visualization of the generated terrain elevation using Perlin noise. The colour gradient reflects elevation levels, with warmer colours indicating higher elevations.

This synthetic terrain generation framework supports the randomisation of domains important for training, allowing learning and planning algorithms to generalise to a wide variety of environmental conditions. By adjusting noise parameters such as scale and frequency, it is possible to control the complexity and slope of the generated maps, facilitating systematic testing of the robustness of navigation policies.

B. Greedy Navigation Approach

As a baseline control strategy, a rule-based greedy navigation controller was implemented to guide the robot towards the target using only local sensor data.

At each simulation step, the robot estimates its current heading and position relative to the goal using data from its GPS and IMU sensors. The heading angle is extracted from the yaw component of the IMU, while the displacement vector to the goal is derived from GPS coordinates.

The controller prioritises actions that reduce the Euclidean distance to the target:

$$d_t = \sqrt{(x_g - x_t)^2 + (y_g - y_t)^2}$$

It then calculates the angular deviation $\Delta\theta$ between the robot's orientation and the vector to the goal. If $|\Delta\theta| > 0.01745$ radians (1 degree), the robot rotates in place to align with the goal direction. Note that this threshold is a hyperparameter and can be tuned to balance between precision and responsiveness. Once alignment is achieved, it proceeds forward with a constant velocity.

In addition to heading control, the robot monitors its pitch and roll values continuously. If these exceed a predefined inclination threshold (indicating unsafe slopes), the robot prioritises stability by entering a recovery manoeuvre: it induces rotation in the direction opposite to the tilt. This behaviour prevents traversal of steep regions and reduces the risk of falling or flipping, thereby improving navigation safety.

a) Recovery Behaviour: To detect stagnation or local minima, the controller compares successive GPS positions. If minimal displacement is observed over a defined window, the robot performs a rotational escape routine. This heuristic aims to dislodge the robot from potentially blocked or unstable terrain.

b) Termination Criteria: An episode terminates under the following conditions:

- The robot reaches within a specified proximity to the goal. This proximity threshold is also a configurable parameter and can be adjusted depending on task requirements.
- A timeout of 120 seconds is exceeded.

Despite its simplicity, this greedy strategy establishes a valuable performance baseline. It enables comparisons against planning-based and learning-based approaches under diverse terrain conditions.

C. Path Planning with A* Algorithm

Navigation commences with a global planning module based on the A* search algorithm, adapted to account for terrain irregularities. The operational environment is discretised into a two-dimensional grid, with each cell assigned a traversal cost influenced by local terrain features, such as slope and elevation.

Each cell considers an 8-connected neighborhood, allowing transitions in both axial and diagonal directions. Diagonal movements incur higher traversal costs, reflecting their increased Euclidean distance ($\sqrt{2}$) relative to axial steps. This preserves geometric consistency and penalises sharp turns on rugged terrain.

The pathfinding cost function is defined as:

$$f(n) = g(n) + h(n)$$

where $g(n)$ represents the cumulative cost from the origin to node n, computed as the Euclidean distance scaled by a terrain roughness factor derived from local elevation gradients.

To ensure physically plausible trajectories, impassable areas and steep inclines are assigned prohibitively high traversal costs. The slope at each cell is estimated via numerical gradients over a local elevation window, and terrain flatness is evaluated by measuring the terrain gradient derived from local elevation differences.

The heuristic term $h(n)$ estimates the remaining cost to the goal:

$$h(n) = \lambda \cdot \sqrt{(x_g - x_n)^2 + (y_g - y_n)^2}$$

where λ is a weighting factor (or tuning parameter) that modulates the influence of the Euclidean distance based on estimated terrain roughness. This formulation guides the planner towards geometrically direct yet traversable paths that respect the Moose's locomotion constraints.

This enhanced cost model enables the planner to favour routes that avoid steep or unstable regions, promoting both efficiency and safety in terrain-constrained environments.

D. Robot Control via A2C

To achieve adaptive and efficient locomotion over challenging terrains, a reinforcement learning-based controller was implemented using the Advantage Actor-Critic (A2C) algorithm. The control architecture integrates with a simulated robotic platform through a socket-based interface, which enables asynchronous communication between the learning agent and the physical simulation. At each timestep, the agent receives observations from the environment and returns continuous velocity commands for actuation.

a) Action Space: The robot operates within a continuous action space defined by two scalar values representing the normalised target velocities for the left and right wheel groups:

$$a = [v_{\text{left}}, v_{\text{right}}], \quad v \in [-1, 1]$$

The resulting actions are linearly scaled and applied to the corresponding wheel motors via velocity-based control, enabling smooth and interpretable locomotion.

b) Observation Space: At each control step, the agent receives a structured observation vector comprising proprioceptive and exteroceptive information:

- The robot's 3D position (x, y, z), retrieved from onboard GPS.
- Orientation angles (ϕ, θ, ψ) (roll, pitch, yaw), provided by the inertial measurement unit (IMU).
- A point cloud obtained from a 4-layer sparse 3D LIDAR, enabling obstacle detection and spatial perception.
- The absolute 3D coordinates of the goal $(x_{\text{goal}}, y_{\text{goal}}, z_{\text{goal}})$.

This multi-modal representation ensures that the agent maintains both spatial and situational awareness necessary for navigation over irregular and potentially unstable terrain.

c) Reward Function: The reward function is designed to balance goal-seeking behaviour with motion efficiency and operational safety. The scalar reward signal at each timestep R_t incorporates the following elements:

$$R_t = \begin{cases} r_{\text{goal}} & \text{if the goal is reached} \\ r_{\text{progress}} & \text{if the agent approaches the goal} \\ r_{\text{stagnation}} & \text{if the agent remains idle} \\ r_{\text{penalty}} & \text{if the agent becomes stuck or flips} \\ 0 & \text{otherwise} \end{cases}$$

where r_{goal} is a shaped terminal reward based on the distance travelled and time efficiency, r_{progress} is a continuous signal encouraging reduction in goal distance, and $r_{\text{stagnation}}$ is a negative reward applied when positional changes fall below a defined threshold over a prolonged interval.

E. Reward Structure and Termination Conditions

This conceptual reward framework is instantiated in practice as:

$$R_t = \begin{cases} 0.5 + \frac{d_{\text{start}}}{T_{\text{total}}} & \text{if } d_t \leq d_g \text{ (goal reached)} \\ \mu \cdot \frac{d_{\min} - d_t}{\Delta t} & \text{if } d_t < d_{\min} \text{ (progress is made)} \\ -0.5 & \text{if stuck too long, timeout, or robot fell} \\ 0 & \text{otherwise} \end{cases}$$

This reward design is shaped to encourage both navigation efficiency — by rewarding progress towards the goal and penalising delays — and safety, by discouraging behaviours that result in the robot becoming stuck or falling, which is critical in rough terrain environments.

where:

- d_t: current Euclidean distance to the goal.
- $d_{\min}$: minimum distance to the goal achieved so far.
- d_{start}: initial distance from start to goal.
- T_{total}: total elapsed time since the beginning of the episode.
- Δt: time interval between steps.
- μ: positive scaling factor.

Three termination conditions are monitored:

1) The robot reaches the goal ($d_t \leq d_g$).
2) The robot is stuck for more than a threshold number of steps.
3) The robot falls off the terrain or exceeds the episode time limit (600 seconds).

These criteria help to ensure efficient navigation and to penalize non-progress behavior. This reward structure follows the design proposed by Zhang et al. [6]. An additional termination condition is triggered when the robot falls outside the map boundaries, which can occur due to the absence of physical constraints (e.g., walls) in the simulated environment. The timeout threshold was set to 600 seconds to allow extended learning per episode; however, this limit was rarely reached during training, as the robot typically succeeded or failed earlier.

F. Training Protocol

Training employed a synchronous Advantage Actor-Critic (A2C) algorithm, where both the actor and critic were parameterised by a shared multilayer perceptron (MLP).The environment used for training and evaluation was the custom `MooseEnv`, providing continuous control of motor velocities and terrain observations. The `MooseEnv` environment provides continuous motor control and multi-modal observations, including proprioceptive data, terrain elevation, and a 4-layer LIDAR point cloud.

Key hyperparameters include:

- Learning rate: 1×10^{-4}
- Discount factor: $\gamma = 0.99$
- Entropy coefficient: 0.01 (to encourage exploration)
- Policy architecture: two hidden layers with ReLU activation functions

The agent was trained for 100,000 timesteps on each procedurally generated terrain map, cycling through a diverse set of environments designed to expose the agent to varying slopes, obstacles, and elevation patterns. This curriculum-style training strategy aimed to promote generalisation and terrain-aware behaviour. Over the full training regime, a total of 3 million timesteps were executed.

Training progress and performance metrics—such as cumulative reward, episode length, and goal success rate—were logged via TensorBoard. Periodic evaluations in deterministic mode were conducted to assess policy robustness.

A checkpointing mechanism was implemented to save the model periodically. If a pre-existing checkpoint was found, training resumed without resetting the timestep counter; otherwise, training was initiated from scratch.

During evaluation, the agent executed multiple episodes in deterministic mode, with progress feedback printed at regular intervals. Episodes terminated upon reaching the goal, falling off terrain, or exceeding the maximum allowed episode duration.

This training protocol enabled the development of a robust navigation policy capable of adapting to diverse terrain conditions and randomized goal positions with minimal supervision.

IV. Experimental Evaluation

This section presents the evaluation results of the navigation algorithms Greedy and A* over 100 runs (10 runs on each of 10 different terrain maps). The performance metrics considered include planned distance, actual distance traveled, completion time, average speed, and navigation outcome (success or failure).

A. Greedy Algorithm

Table I shows the aggregated statistics for the Greedy algorithm across the 100 runs.

The Greedy algorithm achieved a success rate of 80%, with an average completion time of 117.41 seconds. The average speed was 0.98 m/s. The small difference between planned and traveled distance indicates generally efficient path following, although some variability is present.

TABLE I
SUMMARY STATISTICS FOR THE GREEDY ALGORITHM (100 RUNS).

Metric	Mean	Std Dev	Min	Median	Max
Distance (m)	109.48	40.18	29.89	100.50	190.34
Distance Traveled (m)	110.44	40.10	22.09	103.63	190.55
Time (s)	117.41	43.64	20.03	105.15	211.72
Average Speed (m/s)	0.98	0.21	0.57	0.97	1.53
Result (Success=1)	0.80	0.40	0	1	1

B. A* Algorithm

Table II presents the summary statistics for the A* algorithm over the same set of runs.

TABLE II
SUMMARY STATISTICS FOR THE A* ALGORITHM (100 RUNS).

Metric	Mean	Std Dev	Min	Median	Max
Distance (m)	112.02	37.11	52.23	103.33	179.67
Distance Traveled (m)	113.67	37.56	52.23	103.34	183.99
Time (s)	109.03	39.89	33.32	94.00	214.97
Average Speed (m/s)	1.04	0.19	0.58	1.05	1.65
Result (Success=1)	0.85	0.36	0	1	1

The A* algorithm demonstrated a higher success rate of 85% and slightly better average speed (1.04 m/s) compared to Greedy. The average completion time was lower at 226.01 seconds, reflecting the efficiency of the path planning.

C. A2C Algorithm

Table III presents the performance of the A2C algorithm based on the collected data.

TABLE III
SUMMARY STATISTICS FOR THE A2C ALGORITHM (100 RUNS).

Metric	Mean	Std Dev	Min	Median	Max
Distance (m)	102.57	41.27	15.22	98.11	203.68
Distance Traveled (m)	79.96	70.91	0.03	82.63	289.81
Time (s)	63.51	69.55	13.16	48.16	678.76
Average Speed (m/s)	1.89	2.35	0.00	1.47	8.53
Result (Success=1)	0.00	0.00	False	0.00	False

Despite its high theoretical potential, the A2C algorithm failed to successfully complete any of the 100 navigation tasks (**0% success rate**). The average completion time was 63.51 seconds, which is lower than the heuristic methods, but this is likely due to premature termination or failure. While the average speed appears high (1.89 m/s), it is misleading, as most runs failed and the robot either did not move or made erratic movements. This is also reflected in the high standard deviation and extreme values observed in distance traveled and average speed.

D. Comparison and Discussion

The comparative analysis indicates that the A* algorithm outperforms the Greedy method in terms of success rate, speed, and completion time. Although the planned and traveled distances are similar, A* achieves better efficiency in path execution.

The A2C algorithm, however, failed all navigation attempts in this experiment. Although it reported high average speeds, the results suggest erratic behavior and lack of convergence in the policy. The large standard deviation in traveled distance and the near-zero values in many runs point to unstable or inconsistent navigation. This highlights the challenge of training DRL models in complex, variable terrains and reinforces the importance of extensive tuning, environment shaping, and policy regularization.

V. CONCLUSIONS AND FUTURE WORK

This study compared heuristic-based and learning-based navigation strategies for a mobile robot operating in irregular terrain environments. The A* algorithm consistently outperformed the Greedy approach, achieving higher success rates, faster completion times, and more efficient path execution. These results reaffirm the robustness and reliability of heuristic planning methods in structured and partially known environments.

On the other hand, the A2C-based deep reinforcement learning (DRL) agent failed to learn an effective navigation policy. Across 100 evaluation runs, the agent did not successfully complete any episode. We suspect that this failure stems from two main factors:

- **Sparse rewards:** Due to the large map size, the agent rarely reached the goal during training—in fact, this occurred only once throughout the entire training process. As a result, the reward signal was too infrequent to effectively guide policy learning.
- **Premature termination:** Most training episodes ended with the robot being classified as "stuck". Despite efforts to improve the stuck-detection mechanism, this issue persisted, causing a majority of episodes to terminate early and further reducing the opportunity for meaningful learning.

To address these limitations, future work will explore alternative strategies to support DRL learning, including:

- **Curriculum learning:** Starting training on smaller, simpler maps with denser rewards could help the agent learn basic navigation behaviors before scaling up to complex environments.
- **Transfer learning:** Leveraging policies pre-trained on simpler maps to initialize learning in the large-scale terrain could improve convergence and performance.
- **Reward shaping:** Incorporating intermediate rewards for progress toward the goal or for overcoming local elevation challenges may provide more effective feedback for learning.
- **Improved failure detection:** Refining the criteria used to determine when the robot is stuck can prevent premature episode termination and allow the agent to better explore the environment.

Despite the negative results for the DRL model in this iteration, the experiment offers valuable insight into the challenges of applying reinforcement learning in real-world-inspired robotic navigation. With improved design choices and training strategies, DRL still holds potential as a flexible and adaptive solution for autonomous navigation in irregular terrains.

References

[1] Yunhao Gan et al. Energy-based legged robots terrain traversability modeling via deep inverse reinforcement learning. *arXiv preprint arXiv:2207.03034*, 2022.

[2] Ramon Gonzalez and Karl Iagnemma. Deepterramechanics: Terrain classification and slip estimation for ground robots via deep learning. *arXiv preprint arXiv:1806.07379*, 2018.

[3] Miguel Silva and Rui Oliveira. Evaluating domain randomization in deep reinforcement learning locomotion tasks. *Mathematics*, 10(7):1123, 2022.

[4] H. Wang, Z. Liu, et al. Slipnet: Enhancing slip cost mapping for autonomous navigation. *arXiv preprint arXiv:2401.00001*, 2024.

[5] Kasun Weerakoon, Adarsh Jagan Sathyamoorthy, Utsav Patel, and Dinesh Manocha. Terp: Reliable planning in uneven outdoor environments using deep reinforcement learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9447–9453. IEEE, 2022.

[6] Kaichena Zhang, Farzad Niroui, Maurizio Ficocelli, and Goldie Nejat. Robot navigation of environments with unknown rough terrain using deep reinforcement learning. In *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–7, 2018.