Towards Safer Navigation: Reward Shaping with Prior Topographic Knowledge in Habitat Simulator Point Goal Navigation

Jiajie Zhang, Linkai Zu, Jintian Hu, Tianyang Zhao
ShanghaiTech University
393 Huaxia Middle Road, Pudong, China
{zhangjj2023, zhlk2024, hujt2024, zhaoty2024}@shanghaitech.edu.cn

Abstract

*Navigation is essential for intelligent agents in diverse environments. Traditional modular approaches are robust but lack the adaptability of deep learning methods. Reinforcement learning (RL) agents show promise but often struggle with safety and collision avoidance due to limited use of environmental maps. This paper addresses these challenges by integrating topographic knowledge into the RL framework through a novel reward shaping technique that uses distance information from a top-down map to maintain safe distances from obstacles, enhancing path safety without incorporating map data into agent observations. We introduce a new metric, **path_safety**, to evaluate navigation safety. Experiments using Habitat Simulator's PointGoal Navigation task demonstrate that RGBD agents trained with reward shaping achieve significantly higher path_safety scores compared to those without it. These improvements persist even without active reward shaping, indicating that agents have internalized safe navigation strategies. In contrast, RGB-only agents do not show similar enhancements, highlighting the importance of depth information. This work improves the reliability and safety of RL-based navigation agents, advancing their applicability to real-world dynamic and complex environments.*

1. Introduction

Navigation stands as a cornerstone capability for intelligent agents, underpinning their ability to interact effectively within diverse environments. For humans, navigating from point A to point B is often intuitive, especially when the spatial relationship is clearly defined (e.g., "B is 20 meters north of A"). However, replicating this proficiency in robotic systems poses significant challenges. Effective navigation for robots entails the seamless integration of multiple sub-tasks, including perceiving three-dimensional structures, estimating ego-motion, planning optimal routes, and managing various forms of uncertainty[4]. The ability to navigate safely and efficiently is crucial for the deployment of intelligent agents in real-world applications, ranging from autonomous vehicles to service robots in dynamic environments.

Traditionally, robotic navigation has relied on modular pipelines that decompose the navigation problem into distinct sub-components such as mapping, localization, planning, and control [1, 2, 4]. Each module is meticulously hand-engineered to address specific aspects of the navigation task, and these modules are subsequently integrated by human designers to form comprehensive navigation systems. This classical approach has been extensively studied and successfully implemented in numerous robotic platforms over the past decades. Despite their widespread use, these traditional methods often fall short in terms of robustness and accuracy, particularly when confronted with the complexities and unpredictabilities inherent in real-world scenarios.

The advent of deep learning has introduced a promising alternative to classical navigation methodologies. Inspired by its transformative impact across various domains of artificial intelligence, deep learning-based approaches offer the potential to learn navigation strategies directly from data, thereby minimizing the need for extensive hand-engineering[14, 21, 22, 16, 24]. These data-driven methods can leverage underlying regularities within the data to discover sophisticated strategies that may surpass manually crafted rules. Notably, end-to-end training of deep convolutional networks using reinforcement learning has demonstrated significant advancements in navigation tasks within simulated environments[21, 16, 24]. Such approaches enable the development of agents that can autonomously learn to navigate without explicit decomposition into sub-tasks, achieving state-of-the-art performance in various benchmarks.

While learning-based navigation agents have demonstrated promising advancements in virtual environments such as AI Habitat[33], MIMOS[26], and Stanford Large-

Scale 3D Indoor Spaces (S3DIS)[12], transitioning these agents to real-world applications remains fraught with challenges. Virtual training environments offer significant advantages, including reduced training costs and exceptionally high simulation speeds—platforms like Meta's Habitat Simulator can process up to 10,000 frames per second, effectively shifting the primary bottleneck to network optimization[33]. Despite these benefits, a substantial performance gap persists between simulated environments and real-world settings. In practical applications, traditional modular navigation stacks are preferred due to their established robustness and reliability. In contrast, learning-based agents frequently encounter issues such as collision-prone global path planning and difficulties in managing unforeseen obstacles during both training and deployment phases. These shortcomings not only undermine the reliability of learning-based agents but also pose significant risks and potential costs when deployed in real-world environments without adequate safety measures.

A critical difference between reinforcement learning (RL)-based navigation agents and classical navigation systems lies in their utilization of environmental maps. While traditional methods rely on detailed maps to inform their planning and control strategies, RL-based agents typically operate without explicit map information, relying instead on learned representations from sensory inputs. This discrepancy contributes to the aforementioned safety and reliability issues, as the absence of map awareness can lead to inadequate collision avoidance and suboptimal path planning.

In this paper, we address these challenges by integrating prior topographic knowledge into the reinforcement learning framework to enhance the safety of navigation agents. Specifically, we propose a reward shaping technique that incorporates map-based information, thereby enabling the agent to maintain safe distances from obstacles and plan safer paths. Instead of directly feeding grid map images into the agent's policy network, our method calculates the distance between the agent and the nearest obstacles at each step and integrates this information into the reward function. This approach incentivizes the agent to prefer actions that result in greater clearance from obstacles, thereby reducing the likelihood of collisions. Additionally, we introduce a novel metric, termed "path safety," which quantifies the average distance between the agent's trajectory and environmental obstacles. This metric provides a more nuanced assessment of navigation safety, complementing traditional performance indicators and offering deeper insights into the agent's ability to navigate safely.

Through the incorporation of prior topographic knowledge and the introduction of path safety metrics, our work represents a significant step towards bridging the gap between simulated training environments and real-world deployment of learning-based navigation agents. By enhancing the safety and reliability of RL-based agents, we move closer to achieving autonomous systems capable of robustly navigating complex, real-world environments.

2. Related Work

2.1. Traditional Navigation Stack

Navigation has been a fundamental area of research in robotics for decades, encompassing various sub-tasks such as mapping, localization, path planning, and motion control [8, 4]. Classical navigation pipelines typically integrate these components to enable autonomous movement in complex environments. Simultaneous Localization and Mapping (SLAM) is a pivotal technique that combines mapping and localization, with most approaches utilizing metric maps [13, 5]. Alternatively, topological mapping methods have also been explored to represent environments [10]. Path planning within these frameworks often relies on established algorithms that ensure efficient and collision-free routes, as extensively reviewed by LaValle [6][28]. These traditional methods have been successfully implemented across diverse robotic platforms, including wheeled [7], legged [15], and aerial robots [9], demonstrating their versatility and robustness in real-world applications.

2.2. End-to-End Learning for Navigation

The advent of deep learning has introduced end-to-end approaches to navigation, primarily through the application of deep reinforcement learning (deep RL) [14, 17, 24, 25]. These methods leverage neural network architectures to learn navigation policies directly from sensory inputs, bypassing the need for explicit modular decomposition. Techniques such as asynchronous training and curiosity-driven exploration have been employed to enhance the learning efficiency and exploratory capabilities of agents [17, 25]. Despite their potential, many deep RL-based navigation models utilize generic architectures without integrating domain-specific knowledge from classical navigation research. For instance, the UNREAL framework employs a CNN-LSTM architecture trained with auxiliary tasks to improve performance [16]. Additionally, hybrid approaches have been developed where learning-based modules, such as deep networks for pose estimation, are incorporated into broader navigation systems [19, 11, 23]. Advanced methods like the Value Iteration Network (VIN) and the end-to-end mapper-planner CMP represent state-of-the-art efforts to blend traditional planning with learning-based techniques [21, 18]. However, these approaches often demand extensive training data and computational resources, limiting their scalability and adaptability to novel environments.

2.3. Simulation Environments for Navigation Research

Simulation platforms play a crucial role in the development and evaluation of navigation algorithms, offering controlled and scalable environments for experimentation. Several simulators have been developed, each with unique strengths and limitations. Gazebo has been a staple in the ROS community due to its extensive community support and a wide array of prebuilt assets, although it falls short in terms of photorealism and simulation speed [3]. Unity, combined with the MLAgents Toolkit, provides a versatile game engine-based environment suitable for embodied agents, yet it lacks native photorealistic rendering and compatibility with large-scale 3D datasets like Replica and Matterport3D [28, 34, 20]. Isaac Sim offers high configurability and photorealistic rendering but is not openly accessible [32], while Sapien provides realistic physics-rich environments with limited task support [35].

In this context, the AI Habitat Simulator emerges as a preferred choice for our study. Habitat Sim v2 is renowned for its high-speed simulation and photorealistic rendering capabilities, supporting datasets such as Replica and Matterport3D [36]. The modular design of the Habitat framework facilitates seamless integration with various 3D scene datasets, making it an ideal platform for benchmarking and comparison [33, 20, 30]. Specifically, we utilize the Matterport3D dataset within the AI Habitat Simulator, as it is the only publicly available dataset with reported navigation results in Habitat, allowing us to benchmark our approach against existing v1 agent performances [33].

2.4. Comparison of Classical and Learning-based Navigation Methods

Recent studies have benchmarked classical and learned navigation methods within complex 3D environments, highlighting the strengths and limitations of each approach [31, 33]. While classical methods offer reliability and interpretability through well-established algorithms, learned agents demonstrate adaptability and potential for handling diverse scenarios. However, learned agents often require vast amounts of training data and exhibit limited generalization to unseen environments [26, 29]. Our work builds on this comparative analysis by introducing reward shaping techniques that incorporate prior topographic knowledge, aiming to enhance the safety and efficiency of navigation in realistic settings.

3. Method

3.1. Framework Overview

We build upon a standard reinforcement learning (RL) framework for point-goal navigation implemented within the Habitat Simulator environment. The agent is trained using the Proximal Policy Optimization (PPO) algorithm, and we start from a conventional baseline reward function. At each timestep t, the baseline reward r_t is defined as:

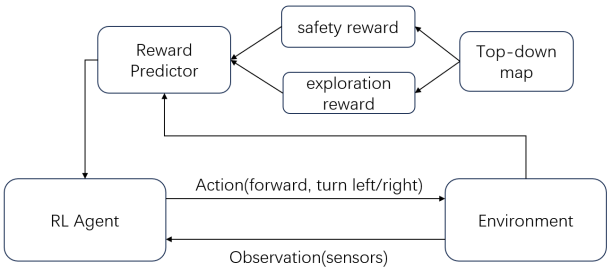

Figure 1. reward shaping: add safety aware reward and exploration reward into basic reward

$$r_t = \begin{cases} s + d_{t-1} - d_t + \lambda & \text{if goal is reached} \\ d_{t-1} - d_t + \lambda & \text{otherwise} \end{cases} \quad (1)$$

where d_t is the agent's distance to the goal at time t, s is a success bonus, and λ is a small positive constant that encourages movement. This baseline reward focuses primarily on guiding the agent toward the goal without explicitly considering navigational safety.

In this study, we introduce a novel approach to enhance safety by integrating prior topographic knowledge through reward shaping. Unlike methods that directly incorporate static top-down maps into the observation space—thereby deviating from the RL principle of learning through environment-driven feedback—we leverage this knowledge exclusively within the reward function. This ensures the agent learns to navigate safely while preserving its ability to generalize to unseen environments. The state space $\mathcal{S}$ remains consistent with standard PointGoal-Nav tasks, and the action space $\mathcal{A}$ consists of discrete navigation commands.

3.2. Reward Shaping with Prior Topographic Knowledge

To promote safer navigation paths, we augment the baseline reward r_t with two additional shaping terms: an exploration reward $R_{\text{exploration}}$ and a safety-aware reward R_{safety}. These terms are layered on top of the original reward, giving the total reward:

$$R_{\text{total}}(s_t) = r_t + \alpha R_{\text{exploration}}(s_t) + \beta R_{\text{safety}}(s_t) \quad (2)$$

Here, r_t is the original baseline reward, and $R_{\text{exploration}}$, R_{safety} are additive shaping signals. The scaling factors α and β balance the contribution of these additional rewards.

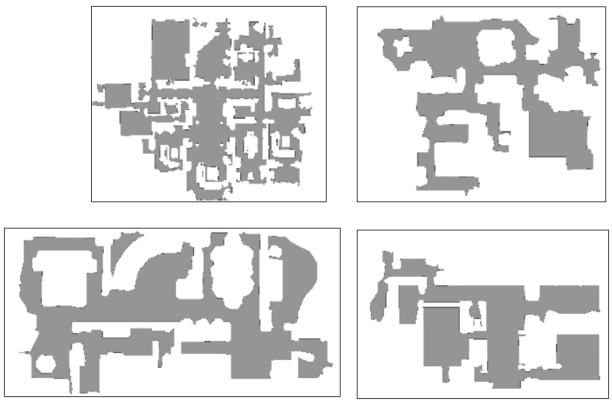

Figure 2. examples of the top-down blank maps that we used for computing the safety reward during the training process

By integrating top-down map information into the reward function rather than the agent's observation space, we adhere to the foundational RL paradigm. The agent still learns from its interactions, but now benefits from a reward signal that indirectly encodes environmental constraints and desirable navigation patterns.

3.2.1. Dual Reward Mechanism

Our reward shaping strategy employs a dual reward mechanism: one component encourages exploration, while the other promotes safe navigation.

Exploration Reward ($R_{\text{exploration}}$) The exploration reward $R_{\text{exploration}}$ is designed to counterbalance the possibility of overly cautious behavior that might arise from the safety penalties. It encourages the agent to explore new regions and avoid stagnation:

$$R_{\text{exploration}} = \lambda_1 \Delta d_{\text{goal}} + \lambda_2 \mathbb{I}(s_t \in \mathcal{V}) - \lambda_3 \mathbb{I}(s_t \in \mathcal{H}) \quad (3)$$

Here, Δd_{goal} measures progress toward the goal, $\mathcal{V}$ represents valid navigable positions, and $\mathcal{H}$ denotes historically visited states. The indicator function $\mathbb{I}(\cdot)$ ensures that rewards are context-sensitive. Coefficients λ_1, λ_2, and λ_3 control the relative importance of progress, validity, and historical uniqueness.

Safety Reward (R_{safety}) The safety reward R_{safety} encourages the agent to maintain a safe distance from obstacles. Using a distance transform of the environment's top-down map, we assign tiered rewards based on the agent's proximity to obstacles. Let $d(s_t)$ be the distance from the agent to the nearest obstacle at state s_t:

$$R_{\text{safety}}(d) = \begin{cases} -0.2 & \text{if } d < 0.5 \text{ m} \\ 0.1 & \text{if } 0.5 \text{ m} \leq d < 2.0 \text{ m} \\ 0.0 & \text{if } d \geq 2.0 \text{ m} \end{cases} \quad (4)$$

This tiered structure provides a strong disincentive for getting too close to obstacles, modest encouragement for maintaining a reasonably safe margin, and neutrality when the agent is sufficiently distant.

3.2.2. Obstacle Avoidance via Distance Transforms

To efficiently determine the agent's proximity to obstacles, we employ a distance transform-based algorithm. We precompute the distance to the nearest obstacle for every navigable point on the top-down map. This allows for $O(1)$ queries during training and ensures timely and accurate feedback. The pixel-to-meters conversion aligns the distance thresholds with real-world scales, making the safety reward more interpretable and effective.

3.3. Exploration Efficiency Optimization

To further refine the agent's behavior, we incorporate multiple strategies to enhance exploration efficiency without sacrificing safety. Effective movement rewards ensure that the agent is incentivized to make meaningful progress toward the goal, preventing it from abandoning promising routes due to high avoidance penalties. Concurrently, invalid area penalties impose negative rewards when the agent attempts to enter unsafe or non-viable regions, thereby discouraging such paths and maintaining strict adherence to safety constraints. Additionally, a historical position tracking mechanism is implemented to avoid repetitive exploration of the same areas, promoting movement towards unexplored and potentially more promising regions. This comprehensive, multi-layered approach effectively balances the necessity for thorough exploration with stringent safety requirements, resulting in more reliable and efficient navigation.

3.4. Integration with PPO Framework

Our reward shaping mechanism integrates seamlessly with the PPO algorithm by computing the total reward R_{total} at each timestep, which combines the baseline reward r_t with the shaping signals. This enhanced reward is utilized for policy and value function updates. We maintain independent reward statistics for the baseline and shaping rewards, allowing separate assessment of exploration and safety incentives. Additionally, the shaping rewards are dynamically accumulated, enabling the reward function to adapt across different training phases and continuously guide the agent's behavior. A dedicated real-time monitoring system tracks key metrics, including the newly introduced `path_safety` measure, which quantifies the minimum distance between the agent and obstacles over entire

trajectories, providing immediate feedback on navigational safety. Furthermore, the reward scales α and β are adjustable during training to prioritize exploration or safety as needed, offering the flexibility necessary to achieve the desired balance in various environments or training phases. By integrating these components into PPO, we preserve its stability and optimization efficiency while substantially enhancing the agent's navigation safety and exploration capabilities.

In summary, our method maintains the foundational RL structure while augmenting it with carefully designed reward shaping terms grounded in prior topographic knowledge. This design encourages safe and efficient navigation, yielding policies that not only reach their goals but also do so with significantly enhanced safety margins.

4. Experiments

In this section, we describe the experimental setup used to evaluate our proposed method for enhancing navigation safety through reward shaping with prior topographic knowledge in the Habitat Simulator's Point Goal Navigation (PointGoalNav) task.

4.1. Task Definition

We adopt the PointGoal navigation task as outlined by Anderson et al. [27], which serves as our primary experimental framework. In this task, the agent is randomly positioned and oriented within an environment and must navigate to specified target coordinates relative to its initial position. Notably, the agent does not have access to a ground-truth map and must rely solely on its sensory inputs for navigation. Our approach aligns with the PointGoal Navigation task definition by utilizing the true map information exclusively for reward shaping, rather than incorporating it directly into the agent's observations or outputs.

4.2. Agent Configuration

4.2.1. Embodiment and Action Space

The agent is modeled as a cylindrical entity with a diameter of 0.2 meters and a height of 1.5 meters. The action space comprises four discrete actions: `turn_left`, `turn_right`, `move_forward`, and `stop`. These actions correspond to precise movements, where turning actions result in a rotation of 10 degrees, and the `move_forward` action propels the agent forward by 0.25 meters. The `stop` action is used by the agent to indicate the completion of its navigation task upon reaching the goal.

4.2.2. Sensory Inputs

The agent is equipped with a single RGB color vision sensor positioned 1.5 meters above the base center and oriented forward. This sensor captures color images at a resolution of 256×256 pixels with a 90-degree field of view. Additionally, an idealized depth sensor shares the same position and orientation as the RGB sensor, matching its field of view and resolution. Agents utilizing only the RGB sensor are referred to as `RGB` agents, while those incorporating both RGB and depth information are designated as `RGBD` agents. Furthermore, all agents have access to an idealized GPS and compass, providing precise location coordinates and orientation relative to the goal.

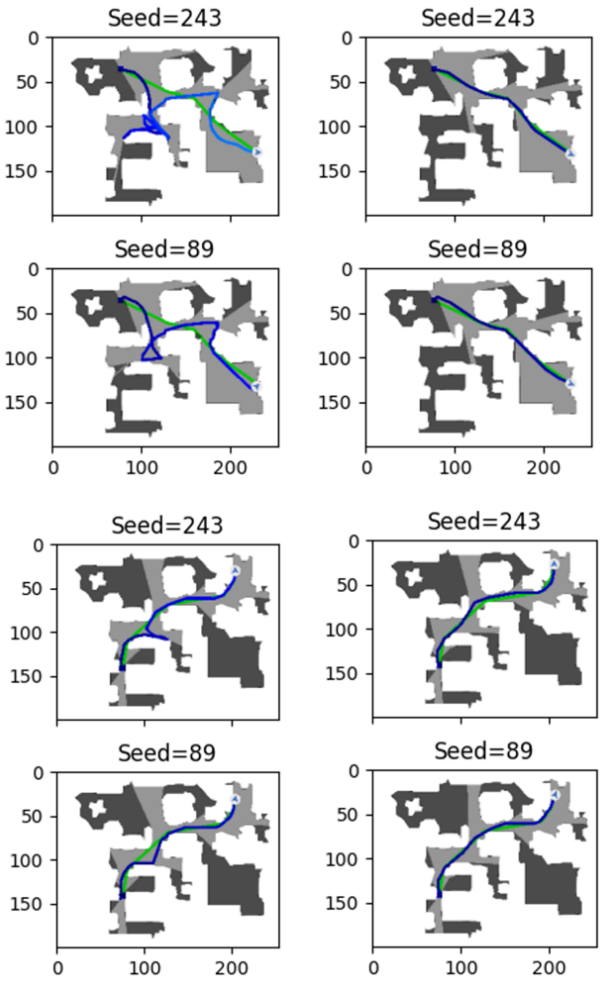

Figure 3. Comparison of the paths taken by the RGBD agent with and without reward shaping during testing. The left figure shows the path without reward shaping, where the agent comes close to obstacles and even collides with them. The right figure illustrates the path taken by the agent when trained with a safety reward, maintaining a safe distance from the walls.

4.3. Experimental Setup

4.3.1. Episode Specification

Each navigation episode begins with the agent placed at a randomly selected navigable position and orientation within

the environment. The target location is also randomly chosen, ensuring that a feasible navigable path exists from the starting position to the goal. During an episode, the agent is permitted to execute up to 500 actions. This action limit is substantially higher than the number required by an optimal agent to reach any goal, allowing ample opportunity for the agent to explore and navigate effectively. After each action, the agent receives sensory feedback from its active sensors to inform subsequent decisions.

4.3.2. Training Procedure

The training process is distributed across four simulator worker threads, each handling an equal subset of the training scenes. Each thread generates blocks of 500 training episodes per scene, with the order of these blocks shuffled to ensure diverse training experiences. Training proceeds by iterating through shuffled blocks indefinitely until a cumulative total of 5 million agent steps is reached across all worker threads. This training duration is consistent with previous studies [33]. The training time required to reach 5 million steps is approximately 10 GPU-hours for `RGBD` agents and 9.5 GPU-hours for `RGB` agents.

4.4. Evaluation Metrics

4.4.1. Success Criteria

A navigation episode is deemed successful if the agent issues a `stop` action within 0.2 meters of the target coordinates, as measured by the geodesic distance along the shortest navigable path from the agent's final position to the goal. If the agent fails to satisfy this condition within 500 actions, the episode is considered unsuccessful.

4.4.2. Success Weighted by Path Length (SPL)

We employ the Success weighted by Path Length (SPL) metric [27] to evaluate the agent's performance. For a given episode, let l denote the geodesic distance of the shortest path from the starting position to the goal, and p represent the distance traversed by the agent. The SPL is defined as:

$$\text{SPL} = S \cdot \frac{l}{\max(p, l)}$$

where S is a binary indicator of success (1 if the episode is successful, 0 otherwise). This metric accounts for both the success rate and the efficiency of the agent's navigation path, providing a balanced measure of performance.

4.4.3. Path Safety Metric

To quantitatively evaluate the safety of the agent's navigation trajectory, we introduce a path safety metric that leverages the distance transform of a top-down occupancy map. Let $\tau = \{p_0, p_1, \ldots, p_T\}$ denote the agent's path, where each $p_t \in \mathbb{R}^3$ is a 3D world coordinate (x_t, y_t, z_t). The top-down map M is a 2D grid, where $M(u, v)$ indicates whether the cell is navigable ($M(u, v) = 0$) or occupied by an obstacle ($M(u, v) = 1$). We compute a distance transform $D(u, v)$ such that $D(u, v)$ is the Euclidean distance from cell (u, v) to the nearest obstacle.

Each 3D position p_t on the path is projected onto the 2D map coordinates (u_t, v_t) using a known mapping function:

$$(u_t, v_t) = f(x_t, z_t; M)$$

where $f(\cdot)$ translates world coordinates (x_t, z_t) into map coordinates (u_t, v_t).

The path safety metric, denoted as $\texttt{path\_safety}(\tau)$, is defined as the average distance from every projected path position to the nearest obstacle:

$$\texttt{path\_safety}(\tau) = \frac{1}{T+1} \sum_{t=0}^{T} D(u_t, v_t). \quad (5)$$

If any path position falls outside the valid map range or if no valid positions are found, the path safety defaults to zero. This metric thus provides a meaningful measure of how safely the agent navigates, as higher values indicate that the agent consistently maintains a greater average distance from obstacles along its trajectory.

Agent	Success Rate	SPL
gibson_rgbd	0.230	0.262
gibson_rgbd_shaped	0.236	0.279
mp3d_rgbd	0.343	0.341
mp3d_rgbd_shaped	0.353	0.339

Table 1. Average Success Rate and SPL for different Agents (averaged over multiple seeds).

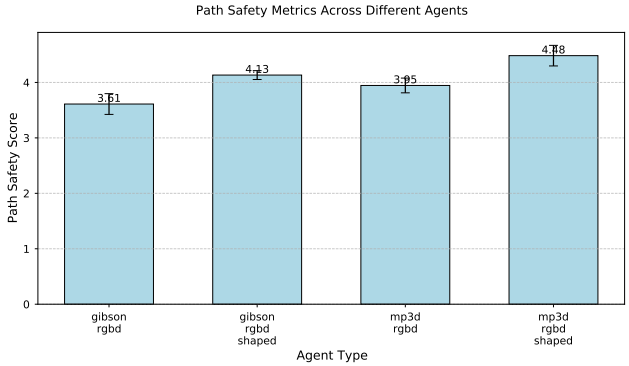

Figure 4. Comparison of path safety scores between RGBD agents with and without reward shaping. The figure shows that, with our reward shaping, the paths taken by the RGBD agents maintain a larger margin from obstacles. We trained the Gibson agent and MP3D agent, then tested them on the MP3D dataset. The results demonstrate that both RGBD agents with reward shaping perform better in an unseen environment

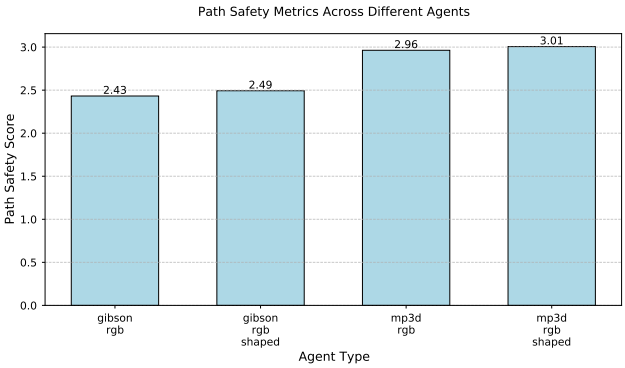

Figure 5. Comparison of path safety scores between RGB agents with and without reward shaping. A limitation we discovered is that reward shaping did not significantly impact the path safety performance of RGB-based agents during testing, particularly when the agents encountered unseen environments.

5. Discussion

The experimental results indicate that reward shaping using prior topographic knowledge significantly enhances the safety of agents' navigation paths. Agents trained with reward shaping consistently maintain greater distances from obstacles, as evidenced by higher `path_safety` scores in Figure 4. This improvement is critical for real-world robotic applications where collision avoidance is paramount. Notably, during evaluation without active reward shaping, agents still navigate safely, maintaining high `path_safety` scores. This demonstrates that agents have learned to associate safe distances with high-value actions through the interplay of state (depth sensor data), actions (proximity adjustments), and shaped rewards. The alignment of the Proximal Policy Optimization (PPO) algorithm with safety objectives ensures that agents prioritize safe paths, reinforced by depth observations that enable effective strategy execution. However, this benefit is primarily observed in `RGBD` agents, as `RGB` agents do not exhibit significant improvements in path safety with reward shaping, especially in unseen environments (Figure 5). This limitation highlights the essential role of depth information in effective reward shaping and suggests that RGB-only agents may require additional mechanisms to achieve similar safety enhancements.

References

[1] Andrew J Davison and David W Murray. "Mobile robot localisation using active vision". In: *Computer Vision—ECCV'98: 5th European Conference on Computer Vision Freiburg, Germany, June 2–6, 1998 Proceedings, Volume II 5*. Springer. 1998, pp. 809–825.

[2] Guilherme N DeSouza and Avinash C Kak. "Vision for mobile robot navigation: A survey". In: *IEEE transactions on pattern analysis and machine intelligence* 24.2 (2002), pp. 237–267.

[3] Nathan Koenig and Andrew Howard. "Design and use paradigms for gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*. Vol. 3. Ieee. 2004, pp. 2149–2154.

[4] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. "Probabilistic robotics. 2005". In: *Massachusetts Institute of Technology, USA* (2005).

[5] Hugh Durrant-Whyte and Tim Bailey. "Simultaneous localization and mapping: part I". In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110.

[6] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[7] Sebastian Thrun et al. "Stanley: The robot that won the DARPA Grand Challenge". In: *Journal of field Robotics* 23.9 (2006), pp. 661–692.

[8] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. "Visual navigation for mobile robots: A survey". In: *Journal of intelligent and robotic systems* 53 (2008), pp. 263–296.

[9] Michael Blösch et al. "Vision based MAV navigation in unknown and unstructured environments". In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 21–28.

[10] Emilio Garcia-Fidalgo and Alberto Ortiz. "Vision-based topological mapping and localization methods: A survey". In: *Robotics and Autonomous Systems* 64 (2015), pp. 1–20.

[11] Alex Kendall, Matthew Grimes, and Roberto Cipolla. "Posenet: A convolutional network for real-time 6-dof camera relocalization". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 2938–2946.

[12] Iro Armeni et al. "3d semantic parsing of large-scale indoor spaces". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 1534–1543.

[13] Cesar Cadena et al. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". In: *IEEE Transactions on robotics* 32.6 (2016), pp. 1309–1332.

[14] Alexey Dosovitskiy and Vladlen Koltun. "Learning to act by predicting the future". In: *arXiv preprint arXiv:1611.01779* (2016).

[15] Marco Hutter et al. “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.

[16] Max Jaderberg et al. “Reinforcement learning with unsupervised auxiliary tasks”. In: *arXiv preprint arXiv:1611.05397* (2016).

[17] Volodymyr Mnih. “Asynchronous Methods for Deep Reinforcement Learning”. In: *arXiv preprint arXiv:1602.01783* (2016).

[18] Aviv Tamar et al. “Value iteration networks”. In: *Advances in neural information processing systems* 29 (2016).

[19] Eric Brachmann et al. “Dsac-differentiable ransac for camera localization”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 6684–6692.

[20] Angel Chang et al. “Matterport3d: Learning from rgb-d data in indoor environments”. In: *arXiv preprint arXiv:1709.06158* (2017).

[21] Saurabh Gupta et al. “Cognitive mapping and planning for visual navigation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2616–2625.

[22] Saurabh Gupta et al. “Unifying map and landmark based representations for visual navigation”. In: *arXiv preprint arXiv:1712.08125* (2017).

[23] Iaroslav Melekhov et al. “Relative camera pose estimation using convolutional neural networks”. In: *Advanced Concepts for Intelligent Vision Systems: 18th International Conference, ACIVS 2017, Antwerp, Belgium, September 18-21, 2017, Proceedings 18*. Springer. 2017, pp. 675–687.

[24] Emilio Parisotto and Ruslan Salakhutdinov. “Neural map: Structured memory for deep reinforcement learning”. In: *arXiv preprint arXiv:1702.08360* (2017).

[25] Deepak Pathak et al. “Curiosity-driven exploration by self-supervised prediction”. In: *International conference on machine learning*. PMLR. 2017, pp. 2778–2787.

[26] Manolis Savva et al. “MINOS: Multimodal indoor simulator for navigation in complex environments”. In: *arXiv preprint arXiv:1712.03931* (2017).

[27] Peter Anderson et al. “On evaluation of embodied navigation agents”. In: *arXiv preprint arXiv:1807.06757* (2018).

[28] Arthur Juliani. “Unity: A general platform for intelligent agents”. In: *arXiv preprint arXiv:1809.02627* (2018).

[29] Yi Wu et al. “Building generalizable agents with a realistic and rich 3d environment”. In: *arXiv preprint arXiv:1801.02209* (2018).

[30] Fei Xia et al. “Gibson env: Real-world perception for embodied agents”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9068–9079.

[31] Dmytro Mishkin, Alexey Dosovitskiy, and Vladlen Koltun. “Benchmarking classic and learned navigation in complex 3d environments”. In: *arXiv preprint arXiv:1901.10915* (2019).

[32] Filipe Figueredo Monteiro et al. “Simulating real robots in virtual environments using NVIDIA’s Isaac SDK”. In: *Simpósio de Realidade Virtual e Aumentada (SVR)*. SBC. 2019, pp. 47–48.

[33] Manolis Savva et al. “Habitat: A platform for embodied ai research”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9339–9347.

[34] Julian Straub et al. “The Replica dataset: A digital replica of indoor spaces”. In: *arXiv preprint arXiv:1906.05797* (2019).

[35] Fanbo Xiang et al. “Sapien: A simulated part-based interactive environment”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11097–11107.

[36] Andrew Szot et al. “Habitat 2.0: Training home assistants to rearrange their habitat”. In: *Advances in neural information processing systems* 34 (2021), pp. 251–266.