Reinforcement learning driven behavioral entropy based frontier exploration

Changjun Li

Delft University of Technology

TU Delft

Reinforcement learning driven behavioral entropy based frontier exploration

by

Changjun Li(5602351)

Thesis committee:

Dr. rer. nat. M. (Marija) Popovic (Responsible supervisor)
Dr.ir. E. van Kampen
Dr. I.I. (Ingeborg) de Pater

TU Delft

Contents

Abstract

Autonomous robotic exploration must balance fast map growth with robustness under uncertainty. Among frontier-based methods, information-theoretic variants (e.g., Shannon/Rényi) are fast and reliable. This thesis integrates Behavioral Entropy (BE)—which models human-like risk perception—into a planner-in-the-loop deep reinforcement learning(DQN) framework that learns the BE risk parameter α online.

The policy selects α from a discrete action set, uses occlusion-aware visibility to compute BE-based information gain, and delegates motion to a classical A* planner; reward shaping couples coverage/entropy reduction with motion cost. Action-usage analysis reveals an adaptive risk schedule—aggressive early, conservative mid-episode, selectively aggressive late—enabling faster cleanup of residual area. Overall, a reinforcement-learning method with risk-attitude tuning yields a robust, planner-compatible explorer.

Keywords: Reinforcement Learning; Information Theory; Entropy-based Exploration; Behavioral entropy; Deep Q-Learning

1 Introduction

With the rapid growth of autonomous robotic deployments in disaster response, urban mapping, and planetary exploration, there is a need for frameworks that can make real-time decisions for exploration both quickly and robustly. In post-disaster search and rescue, robots must navigate unknown, cluttered environments, continuously update their maps, and avoid missing trapped victims; in planetary exploration, communication delays demand a high degree of onboard autonomy and adaptability.

Currently, frontier-based exploration with occupancy grid mapping [1] as spatial representation is a widely used solution in robotics exploration and model unknown environments. It have been proven effective in static, structured settings but often falter when faced with sensor noise, pose uncertainty, and moving obstacles. Traditional frontier exploration assumes that a robot has perfect knowledge of its own state and relies on map data for decision-making. In reality, these assumptions do not always hold. Uncertainties arise from factors such as pose errors and mapping inaccuracies [2]. To address these limitations, information-theoretic approaches have been introduced to optimize frontier selection to reduce uncertainty. Common methods include (BGS) entropy (Boltzmann–Gibbs–Shannon) and Rényi entropy driven informative exploration algorithms, both of which measure uncertainty in an unknown environment [3–5].

To improve the information theory exploration, an entropy measurement called *behavioral entropy* has been proposed by Suresh and Nieto-Granda [6] and integrated with exploration, which extends traditional entropy measures by incorporating Prelec's probability weighting function [7] to model human-like risk perception. This formulation provides a more nuanced representation of uncertainty, potentially leading to better decision-making in robotic exploration. However, behavioral entropy remains challenging to apply effectively: parameter tuning is still largely heuristic, relying on expert intuition and noise level[6]. α is the only free parameter that can be tuned in Behavioral entropy calculation: $\alpha < 1$ overweights small probabilities (cautious, local), while $\alpha > 1$ underweights small probabilities and overweights large ones (bolder, long-range).

In parallel, recent advancements in reinforcement learning (RL) have demonstrated that robots can learn exploration strategies autonomously through interaction with their environment [8]. RL-based frontier exploration has extensively utilized Shannon and Rényi entropy to quantify uncertainty and optimize decision-making. For instance, Li *et al.* [9] proposed a deep RL framework that integrates entropy-based rewards with classical navigation, improving exploration efficiency by decomposing the process into decision, mapping, and planning modules. Luizza *et al.* [10] surveyed RL-driven exploration, identifying challenges such as exploration–exploitation trade-offs, reward shaping, and slow convergence. Leong *et al.* [11] combined frontier-based exploration with graph-SLAM and RL to enhance map accuracy and frontier selection in visually complex scenarios. While effective, these methods still rely on fixed entropy models and often struggle when environments change dynamically.

Inspired by the behavioral entropy and RL driven information theory based frontier exploration mentioned above, the focus in this paper is whether adapting the behavioral entropy risk parameter over time can outperform fixed-α heuristics and strong classical baselines under the same mapping and planning setup. We therefore train a planner-in-the-loop DQN whose discrete action is the BE shape parameter α, compute behavioral entropy-based information gain and lead the frontier selection. The central hypothesis is that *joint* learning of frontier selection base on entropy-parameter adaptation improves exploration efficiency compared with fixed-parameter baselines.

Contributions

- A planner-in-the-loop RL formulation for frontier exploration that embeds BE-based information

gain and treats (α) (a parameter to tune exploration attitude) as learnable decision variables rather than static hyperparameters [6, 7]

- A evaluation across synthetic and realistic layouts, including ablations against greedy frontier selection, Shannon/Rényi utilities, and fixed-parameter BE, as well as analysis of computational trade-offs in candidate evaluation [5, 12–14].
- An empirical characterization of how α evolves with exploration. Usage analysis over step bins and state summaries (coverage/entropy) reveals a stage-wise pattern: larger α early (bolder expansion), smaller α in the middle (local clean-up), and selective larger-α spikes late for residual pockets.

Scope. The focus is on ground robots with 2-D occupancy grids and standard global–local planning (A* + reactive local control). [15, 16].

2 Related Work

This thesis studies autonomous exploration for mapping: a mobile robot must travel and build a 2-D map of an unknown environment quickly and reliably under sensing and motion constraints. Typical application domains include indoor mapping (domestic/office robots), time-critical search-and-rescue, and resource-limited field missions such as underground or planetary exploration—settings where coverage speed, robustness under uncertainty, and path efficiency matter. At a high level, the goal is to maximize knowledge gained per unit effort (time, distance, or energy). The overall exploration loop is summarized in Fig. 2.2. A preview of the problem is shown in figure 2.1.

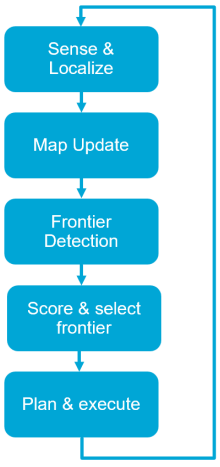


Figure 2.2: Classic pipeline of a frontier-based exploration system. .

This chapter reviews exploration strategies most relevant to this thesis from three angles: (i) classical and heuristic *frontier exploration*, (ii) *information-theoretic* frontier utilities that quantify uncertainty, and (iii) *reinforcement learning (RL)* approaches that learn exploration behavior from interaction. Foundations such as occupancy grids, SLAM (Simultaneous Localization and Mapping) back-ends, and low-level frontier extraction are considered preliminary and are discussed in detail in Chapter 3.

2.1. Classical and Robotics Frontier Exploration

Frontier-based exploration is a classical strategy for autonomous mapping. We maintain a probabilistic occupancy grid where each cell has an occupancy probability $p \in [0, 1]$ (free ≈ 0, occupied ≈ 1, unknown = unobserved). A *frontier* is the boundary between *free* and *unknown* cells [17] (details in Chapter 3). By repeatedly navigating to frontier regions, the robot incrementally pushes outward the known area and expands coverage. Selecting the *nearest frontier* is a strong baseline due to its simplicity and reliability in unknown environments [17]. Many studies report that this greedy choice often yields reasonably short tours, sometimes comparable to more elaborate lookahead planners; for instance, Holz observed that "although exploring closest frontiers is a rather simple and naive strategy, the resulting paths are reasonably short and do not rank behind those acquired using more sophisticated strategies" [5].

In practice, frontier pipelines are widely used on ground robots for *indoor mapping and inspection* via ROS packages such as `explore_lite` [18]; multi-robot variants parallelize coverage by sharing frontier cues while keeping decisions decentralized [19]. The *Distance Advantage* heuristic explicitly scores a frontier by its distance to the robot and its isolation from other frontiers, which shortens tours and mitigates greedy "ping–pong" effects [20]. Global-tour formulations convert candidate frontiers/viewpoints into a TSP-like ordering to reduce total path length; for aerial exploration, *FUEL* exemplifies a hierarchical pipeline that builds a near-optimal tour and then refines locally for dynamic feasibility [14]. These geometry-driven methods are lightweight and planner-friendly, but they rely on hand-crafted scores and can struggle under uncertainties, bad sensing quality, or topology changes significantly.

2.2. Information-Theoretic Frontier Exploration

Information–theoretic methods use the probabilities already stored in the *occupancy grid* to score each candidate frontier by the *expected reduction of map uncer-*

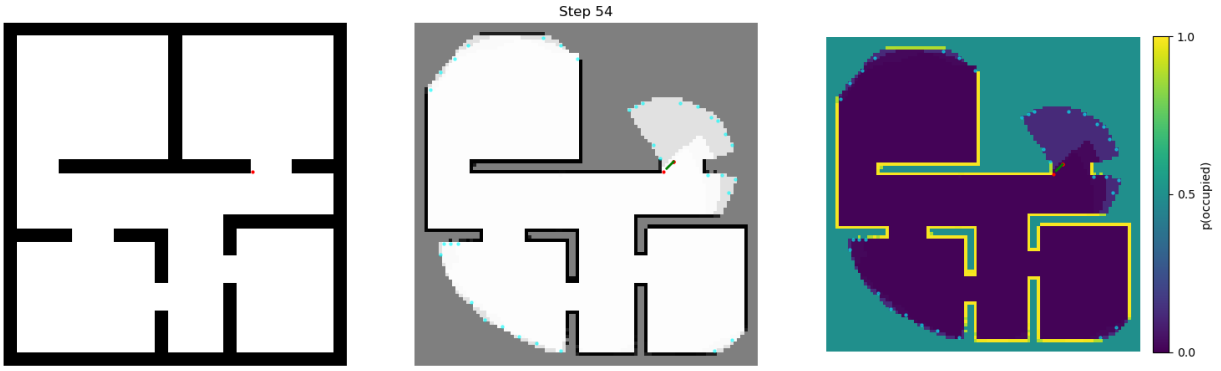


Figure 2.1: Exploration progress at a certain steps. (**Left**) Ground-truth map with robot position (red dot). (**Middle**) Current explored occupancy grid and visible frontiers (cyan). (**Right**) Estimated occupancy probabilities p(occupied), where brighter regions indicate higher occupancy likelihood.

tainty if the robot were to observe from that pose. Concretely, we predict which cells would become visible via occlusion–aware ray–casting on the current map and estimate the drop in uncertainty (entropy) between current beliefs and the hypothetical posterior,

$$\mathrm{IG} \;=\; \mathbb{E}[H(\text{before}) - H(\text{after})]\,,$$

then trade this information gain against travel cost to prefer "information per meter/time." Intuitively, the next move is the one that turns the most *gray/unknown* cells into confident free/occupied cells for the least effort.

A standard choice for $H(\cdot)$ is *Shannon entropy* [4], widely used to compute IG with occlusion–aware visibility and to balance IG with path cost [21]. *Rényi entropy* generalizes Shannon with an order γ, providing a single knob that shifts emphasis between rare and common outcomes and couples map and pose uncertainties through γ–dependent weighting [3, 22]. Larger γ yields more conservative (risk-averse) behavior; smaller γ encourages probing uncertain regions. This flexibility avoids hand-mixing disparate terms, though performance depends on choosing γ appropriately across environments.

More recently, *behavioral entropy* (BE) composes Shannon with Prelec's probability weighting to model nonlinear, human-like risk perception using parameters (α, β) [6, 7]. BE has been shown to satisfy the fundamental Shannon–Khinchin axioms while offering a strictly larger range of sensitivity/perceptiveness than Shannon/Rényi, thus spanning a broader family of exploration attitudes [6]. In practice, the chief computational cost in these methods stems not from the entropy algebra but from evaluating many candidates (rays × traversed cells) and from repeatedly estimating path cost; careful candidate pruning and coarse-to-fine visibility can help. A remaining gap is *adaptivity*: both Rényi's γ and BE's (α, β) are often tuned offline and may be brittle under domain shifts [22].

2.3. Reinforcement Learning for Exploration

RL offers a data-driven route to non-myopic exploration by optimizing long-horizon returns that mix information gain and robot motion costs [8]. A common and practical design is *planner-in-the-loop* RL: the high-level policy selects targets (e.g., frontiers), while in low-level, a classical planner handles motion execution. This separation yields better safety and sample efficiency than end-to-end motor commands. Deep RL has been used to learn goal selection with entropy-shaped rewards, demonstrating improved coverage-time and reduced backtracking relative to purely greedy selection [9]. Surveys further chart challenges such as reward shaping, exploration–exploitation trade-offs, credit assignment, and sim-to-real transfer [10].

RL with Information-Theoretic Signals. Several systems embed Shannon or Rényi information gains directly in the RL reward or as intrinsic motivation, combining map-uncertainty reduction with traversal penalties and safety terms [21–23]. Despite these advances, most RL formulations treat entropy parameters as fixed hyperparameters; online *learning* of γ or (α, β) remains underexplored.

2.4. Positioning and Open Problems

The literature suggests a progression: geometry-only frontiers → entropy-guided utilities → RL policies with intrinsic information gain. Open problems particularly relevant to this thesis include:

- **Adaptive uncertainty modeling.** Learn to *tune* entropy parameters online (Rényi's γ or BE's (α, β)) so that risk attitude matches environment phase and mission goals, instead of grid-searching fixed values [6, 22].

2.5. Summary

Classical frontier heuristics remain strong baselines but are hand-tuned,heuristic but myopic; entropy-based utilities bring principled uncertainty reduction yet depend on sensitive parameters; RL can learn long-horizon decisions but typically assumes a fixed uncertainty model. This thesis targets their intersection: a planner-compatible RL framework that *adapts* risk perception online—instantiating information-theoretic frontier selection with BE and learning to tune its parameters during exploration [6, 9].

3 Preliminaries

The problem setup and notation can be summarized as followed: a single ground robot explores a planar, static environment represented as a 2-D occupancy grid $\mathcal{M} \in \{\text{free}, \text{occupied}, \text{unknown}\}^{H \times W}$. We first outline the overall exploration workflow, then derive the mathematics of frontier detection and behavioral entropy–based information gain. The path planning algorithm applied will also be briefly mentioned.

3.1. Generic Frontier-based Exploration Workflow (Overview)

A canonical frontier-based exploration loop is adopted to connect sensing, mapping, and decision-making. At each time t, the workflow has already shown in figure 2.2:

1. **Sense & Localize:** acquire scan z_t; estimate pose x_t via Rao-Blackwellized Particle Filter - Simultaneous Localization and Mapping (RBPF-SLAM) (Sec. 3.2).
2. **Map Update:** update the occupancy grid $\mathcal{M}_t$ in log-odds and export a thresholded grid $\hat{\mathcal{M}}_t$ for downstream modules. (Sec. 3.2)
3. **Frontier Detection:** extract unknown–free boundaries and cluster into frontier regions (Sec. 3.3).
4. **Scoring & Selection:** evaluate each candidate with an information-gain based utility and select the best one.
5. **Planning & Control:** plan (e.g., global A* with a local controller), execute, rescan, and repeat.

3.2. Occupancy grid and Uncertainty

We fuse each incoming range/depth scan z_t with the robot pose x_t in an RBPF-SLAM pipeline [24–26]. The map $\mathcal{M}$ is a binary occupancy grid with per-cell Bernoulli variables $m_i \in \{0, 1\}$ and beliefs

$$p_i^{(t)} \triangleq \Pr\big(m_i = 1 \,\big|\, z_{1:t}, x_{1:t}\big). \tag{3.1}$$

(1) Inverse sensor model (ray casting). For every cell i intersected by a beam at time t, we assign a *single-scan* occupancy likelihood $\tilde{p}_i^{(t)} \triangleq p(m_i = 1 \mid z_t, x_t)$ via a standard piecewise inverse model:

$$\tilde{p}_i^{(t)} = \begin{cases} p_{\text{occ}}, & \text{if } i \text{ is the cell at a valid hit,} \\ p_{\text{free}}, & \text{if } i \text{ lies along the free ray} \\ & \quad \text{before the hit (or up to max range),} \\ p_0, & \text{otherwise.} \end{cases} \tag{3.2}$$

where $p_{\text{occ}} > \frac{1}{2}$, $p_{\text{free}} < \frac{1}{2}$, and $p_0 \approx \frac{1}{2}$ is the uninformative prior. Cells not touched by any beam at time t keep their previous belief.

(2) Log-odds update. Per-cell beliefs are maintained in log-odds, $\ell_i^{(t)} \triangleq \log \frac{p_i^{(t)}}{1-p_i^{(t)}}$. The update then becomes additive:

$$\ell_i^{(t)} \;=\; \text{clip}\Big(\ell_i^{(t-1)} \;+\; \log\frac{\tilde{p}_i^{(t)}}{1-\tilde{p}_i^{(t)}} \;-\; \ell_0 \;,\; \ell_{\min}, \ell_{\max}\Big), \tag{3.3}$$

where $\ell_0 = \log \frac{p_0}{1-p_0}$ and $\text{clip}(\cdot, \ell_{\min}, \ell_{\max})$ optionally caps log-odds to avoid numerical saturation. Probabilities are recovered via

$$p_i^{(t)} \;=\; \sigma\big(\ell_i^{(t)}\big) \;=\; \frac{1}{1 + e^{-\ell_i^{(t)}}}. \tag{3.4}$$

(3) Thresholded grid for downstream modules. For real-time navigation interfaces (e.g., GMapping/ROS), we also export a discrete map by thresholding:

$$\hat{m}_i \;=\; \begin{cases} 1, & p_i^{(t)} \geq \tau_{\text{occ}} \quad \text{(occupied)}, \\ 0, & p_i^{(t)} \leq \tau_{\text{free}} \quad \text{(free)}, \\ 0.5, & \text{otherwise} \quad \text{(unknown)}, \end{cases} \tag{3.5}$$

with $\tau_{\text{free}} < \frac{1}{2} < \tau_{\text{occ}}$ (a margin helps reduce flickering near 0.5).

Uncertainty for exploration. Thresholding in (3.5) discards confidence. For exploration, we retain the full beliefs $p_i^{(t)}$ (or $\ell_i^{(t)}$) internally and quantify residual global uncertainty with the (Shannon) map entropy

$$H(m_i) = -p_i^{(t)} \log p_i^{(t)} - (1 - p_i^{(t)}) \log(1 - p_i^{(t)}). \tag{3.6}$$

$$H_{\text{map}}^{(t)} = \sum_i H(m_i) \tag{3.7}$$

whose expected reduction (information gain) guides frontier exploration policy.

3.3. Frontier definition

As mentioned in 3.2, occupancy grid mapping offers an explicit representation of the surrounding environment. The spatial distribution of a certain area can be represented with a certain value which describes the occupancy probability. Formally, if the environment is represented as an occupancy grid (with each cell marked as free = 0, occupied = 1, unknown = 0.5, and any value in between to take uncertainties into consideration), a cell c is defined as a frontier cell if and only if c is free and at least one neighboring cell (in the grid adjacency) is unknown.[27] Equivalently, frontiers are the "regions on the border between space known to be open and unexplored space."[17] In practice, detecting these frontiers can be viewed as an edge-detection problem on the occupancy grid: one finds the boundary between known free space and unseen territory by labeling any free cell adjacent to an unknown cell as a frontier edge. Clusters of adjacent frontier cells form frontier regions, and any such region exceeding a minimum size (on the order of the robot's footprint) qualifies as a navigable frontier. Figure 3.1 illustrates this concept on a 2D grid map: free-space cells (white) bordering unknown cells (gray) are identified as frontier edges, which are then grouped into frontier regions (each region's centroid marked with a crosshair). These frontier regions represent candidate goals for further exploration.

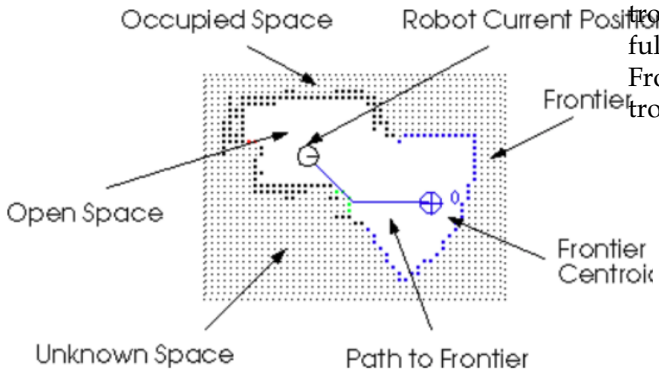


Figure 3.1: Illustration of frontier-based exploration. Occupied cells (black) mark known obstacles; free cells (white) mark known free space; unknown cells (gray) mark unexplored areas. The robot at position(circled) plans a path (solid blue line) to the centroid of the chosen frontier (dashed blue boundary) on the boundary between free and unknown space.[28]

3.4. Information theory

Generalized entropy measures provide a quantitative framework for evaluating uncertainty and offers a heuristic algorithm for frontier exploration. The classic Boltzmann–Gibbs–Shannon (BGS) entropy, introduced by Shannon [4] and further developed by Gibbs, quantifies uncertainty using a logarithmic function of the probability distribution, serving as a fundamental tool in information theory and robotics. Derived from this origin shannon entropy, behavioral entropy offers another measurement and will be discussed in section 3.4.1.

Shannon Entropy

Shannon Entropy: Shannon's entropy provides a baseline measure of information. For a discrete random variable X with outcomes x_i and probabilities p_i, it is defined as:

$$H_S(p) = -\sum_i p_i \ln p_i \tag{3.8}$$

Generalized entropy measures extend this form by relaxing the strong additivity axiom while retaining continuity, maximality, and expansibility [29]. Any admissible generalized entropy can be written as

$$H_g(p) = \sum_i g(p_i), \tag{3.9}$$

where g is continuous, concave, and satisfies $g(0) = 0$. In an occupancy grid map context, each cell's occupancy probability (e.g., p for occupied vs $1-p$ for free) carries an entropy

$$H_S = -(p\log(p) + (1-p)\log(1-p)) \tag{3.10}$$

Hence, unknown regions (e.g. $p \approx 0.5$) have high entropy, meaning highest information potential, whereas fully known cells ($p \to 0$ or 1) have zero entropy. Frontier-based exploration can thus use Shannon entropy as an information-theoretic utility.

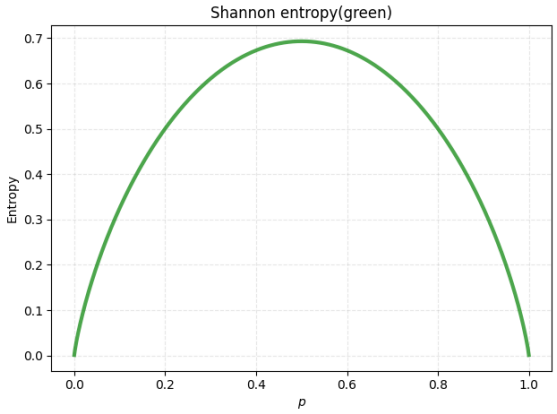


Figure 3.2: Shannon entropy(green) within different binary probability.

For each cells in the environment, a prior value of 0.5 is assigned originally and therefore a unknown high global entropy exists. With the update of occupancy grid the robot selects frontiers (boundaries between

known and unknown space) that promise the greatest expected entropy reduction. In practice, one computes the expected information gain of visiting a frontier, often by summing the entropies of currently unseen cells that would become observable from that viewpoint[11]. The robot then drives toward the frontier that maximizes this expected Shannon entropy reduction (sometimes normalized by travel cost)[6]. This strategy encourages the robot to reduce uncertainty fastest by mapping high-entropy (high-uncertainty) areas first. Classic exploration algorithms using occupancy grids employ Shannon entropy in this manner to efficiently expand the known map while keeping the robot localized. For example, Yamauchi's original frontier method sought unknown-space boundaries, and later extensions added information metrics: Bourgault et al. (2002) first introduced information-based exploration using Shannon entropy of map cells to evaluate frontiers [17, 21]. Shannon entropy remains a fundamental solution for exploration, as it provides a single unified measure of map uncertainty[22].

3.4.1. Behavioral entropy

Extending from generalized entropies, behavior entropy has been introduced to integrate aspects of human risk perception into uncertainty quantification. It is an information-theoretic measure recently proposed by Suresh et al. that integrates principles from behavioral economics into robot exploration[6]. The core idea is to replace the "rational" probability p in Shannon's formula with a perceived probability $w(p)$ given by a human-inspired probability weighting function. Specifically, BE composes Shannon's entropy with Prelec's weighting function[7]. Prelec's model mathematically captures the non-linear perception of probabilities observed in human decision-making.

Given a probability distribution $p = (p_1, p_2, \ldots, p_M)$, behavioral entropy H_B is defined as:

$$H_B(p) = -\sum_{i=1}^{M} w(p_i) \log w(p_i) \tag{3.11}$$

where $w(p)$ is the probability weighting function given by Prelec's model:

$$w(p) = e^{-\beta(-\log p)^{\alpha}}, \quad \alpha, \beta > 0 \tag{3.12}$$

The function 3.12 has two parameters: α controls the curvature (convexity/concavity) of the weighting curve, thereby determining how probabilities are over- or under-weighted relative to their Shannon value, and β determines the fixed point p where $w(p) = p$, which adjusts the overall risk averting/seeking in probability perception. The relationship can be found in figure 3.3. By plugging these distorted probabilities into an entropy formula, BE effectively models an agent that does not treat uncertainty linearly and will be able to mimic human-like biases, potentially leading to more adaptive and efficient decision-making strategies under uncertain conditions.

Suresh et al. prove that behavioral entropy H_B is an admissible generalized entropy: it satisfies the first three Shannon–Khinchin axioms—continuity, maximality at the uniform distribution, and expansibility [6, 29]. On an occupancy grid this has direct operational meaning: axioms-continuity described that the entropy function is continuous. Hence, when sensor updates induce possibility changes, the entropy value can always be computed. Maximality implies that a completely unknown patch remains the most informative target, aligning with outward exploration. Expansibility means masking or adding impossible states (e.g., out-of-bounds cells) does not bias the score.

Via Prelec probability weighting, BE exposes parameters (α, β) that shape risk attitude beyond Shannon/Rényi: $\alpha < 1$ (*uncertainty-averse*) increases sensitivity to mid-range probabilities near 0.5 and favors local cleanup; $\alpha > 1$ (*uncertainty-ignorant*) down-weights small pockets and prioritizes large unexplored regions. Suresh et al. formalize this with *sensitivity* $S(H)$ and *perceptiveness* $P(H)$ and show $P(H_B) > P(H_R) > P(H_S)$, i.e., BE spans a strictly wider behavioral range [6]. They also instantiate BE in a frontier utility with occlusion-aware visibility for robotic exploration; our work goes further by *adapting* (α, β) online during exploration (Sec. 3.4.1).

Fixed point and $\beta(\alpha)$. To preserve maximality at the uniform distribution in the Behavioral family, the paper conditions parameters so that $w(1/M) = 1/M$. For the Bernoulli case (M=2), this yields the closed form

$$\beta(\alpha) = (\log 2)^{1-\alpha} \tag{3.13}$$

which is how β is set in code (so only α is tuned).

3.5. Path Planning Overview (A*)

Once a frontier has been determined, we plan a path to it using a global planner (A*) on an inflated costmap. The traversal cost for a cell c is

$$\begin{aligned} C(c) \;=\; & w_{\text{occ}}\, \mathbb{I}\{\hat{m}_c = 1\}_{\infty} + w_{\text{infl}}\, \text{dist\_to\_obs}(c) \\ & + w_{\text{unk}}\, \mathbb{I}\{\hat{m}_c = 0.5\} + w_{\text{unit}}. \end{aligned} \tag{3.14}$$

and the edge cost is $g(n, n') = C(n')$ (with a $\sqrt{2}$ factor for diagonals). The heuristic h is Euclidean (or octile) distance. A goal candidate is feasible if a collision-free path exists; otherwise the next best candidate is tried. Online re-planning is triggered after sensor updates or when the current plan becomes invalid.

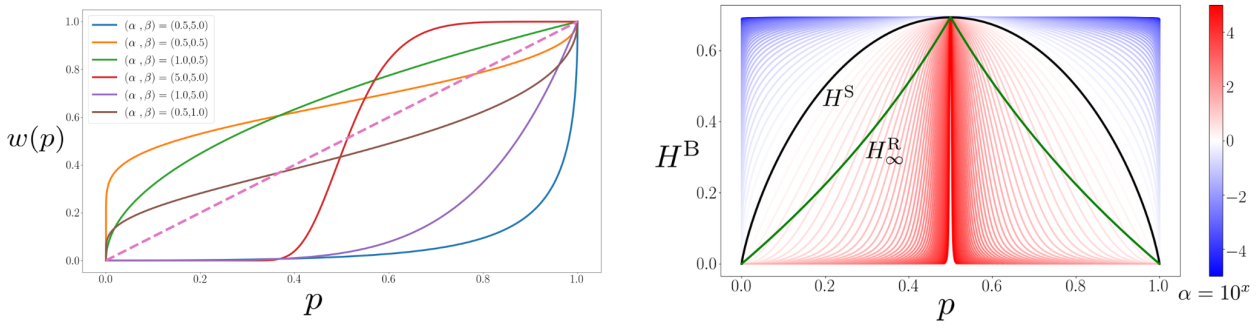


Figure 3.3: Capture from Suresh et al:"(Left) Prelec's probability weighting function with a few parameter choices. Y-axis $w(p)$ indicates the perceived uncertainty associated with probability p (X-axis). Dotted line indicates the unity curve $w(p) = p$. (Right) Behavioral entropy variation with α in log scale for a Bernoulli trial. Black curve indicates Shannon's entropy."[6]

3.6. Assumptions and Scope

Based on the preliminaries explained above, several key assumptions have been made about the environment, the robot platform, the sensor model, the planning framework, and the learning system:

Environment: The operating environment is simulating an indoor layout such as an office area or classrooms. The map assumed to be a two-dimensional static occupancy grid map consisting of discretized cells, with no dynamic obstacles.

Robot: The robot's pose is known at all times (through perfect localization or a reliable state estimator), and its odometry is stable with negligible drift. The robot is equipped with a standard global path planner (e.g., A*) for path planning and has a fixed field-of-view LiDAR with a fixed maximum range, meaning the sensor's coverage area is constant during operation.

Sensors: The robot uses a simulated sensor with 360 and $2m$ of sensor range that produces range scans via ray casting. Sensor readings are updated at discrete time intervals (synchronized with the control loop) rather than continuously, simplifying integration into the planning cycle.

Planning Loop: Exploration operates in cycles where decision-making occurs at frontier selection steps. Once a target frontier is chosen, the subsequent motion execution is assumed to be near-perfect (the robot accurately follows the planned path), and each new LiDAR scan is correctly fused into the global occupancy map without noise, ensuring an accurate and up-to-date map.

Learning System: The environment is modeled as a Markov Decision Process (MDP) under the chosen state representation, meaning the Markov property holds (future states depend only on the current state and action). The robot's observations are partial, but this partial observability is mitigated by incorporating recent observation history or the current map state into the agent's input, so as to approximate a complete state. The agent's action space is discrete, consisting of a set of possible exploration "risk levels" (parameterized by a factor α); each chosen α influences the exploration strategy (e.g., balancing more aggressive versus more cautious exploration).

Exploration Task: The objective of the task is to maximize map coverage while minimizing the path length traveled. Correspondingly, the reward function is shaped to encourage this behavior: the agent receives positive rewards for coverage gain (newly explored area) and for reducing map entropy (uncertainty in the map), guiding the agent toward efficient, informative exploration.

4 Methodology

This chapter addresses the problem of efficient autonomous exploration under uncertainty. The key focus of the method is to train a robot to reach a full coverage in a fast speed. To tackle this, we propose a Behavioral-Entropy-based reinforcement learning framework that learns a risk-attitude parameter α online to guide frontier selection. The method combines classical log-odds map update and A* planning(mention in prelimnary chapter 3) with deep Q-learning, enabling the robot to adapt its exploration strategy between risk-seeking and risk-averse modes according to map progress. The following sections formulate the problem as an MDP, describe the α -action design, and detail the learning pipeline and network architecture. The overall loop can be seen in figure 4.1.

4.1. Problem Formulation

Robotic exploration can be naturally formulated as a Markov Decision Process (MDP), since decision-making is sequential, stochastic, and conditioned on both current knowledge and uncertain future observations. The MDP includes states, actions, transition

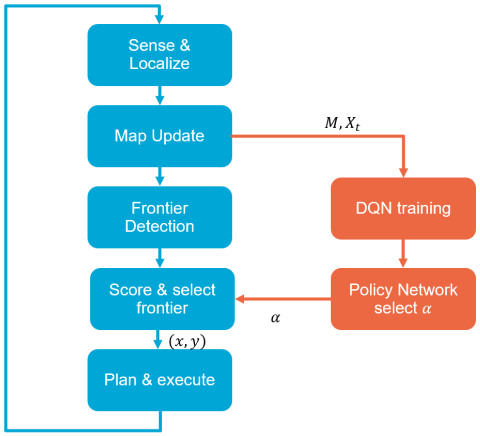


Figure 4.1: Train and α selection pipeline

model, reward, discount factor($\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$). This formulation provides a principled way to handle partial observability: although the true map is unknown, the robot's current belief (its estimated occupancy grid) can serve as the state. In this view, exploration becomes an optimization problem—maximizing the expected cumulative information gain hence a better coverage while accounting for motion cost and uncertainty.

This formulation enables the application of Reinforcement Learning (RL). Unlike classical frontier-based methods that greedily select candidates with fixed heuristics, an RL agent can learn a policy that maximizes exploration speed. In particular, MDP modeling explicitly handles the exploration–exploitation tradeoff: while a greedy strategy may always select the frontier with the largest immediate gain, an optimal policy may prefer frontiers that unlock higher long-term gains (e.g., revealing new regions).

As mentioned in section 3.1, the agent will selected a certain frontier at upper level and send the goal to the path planning and control mechanism to execute the motion. Hence, the time is discretized by two types of steps including motion steps and decision making steps.

State. $S_t \in \mathcal{S}$ includes the occupancy grid $P_t = \{p_i^{(t)}\}$, robot pose x_t, and optional semantic channels such as (entropy, candidate frontier mask)

Action. Instead of making the RL agent directly select a specific frontier cell (which would be a high-dimensional and variable action space), we adopt a discrete action space of risk attitudes denoted by α. Each action corresponds to a choice of α – a parameter in the utility function that reflects the agent's attitude towards uncertainty (risk-seeking vs risk-averse). In practice, we define a small set of representative α values ($\alpha \in 0.2, 5.0$ detailed in **??**) and treat the index of α as the action. The robot then uses the chosen α to evaluate all candidate frontiers and select the best frontier under that risk attitude.

For each frontier candidate, we perform a discrete multi-beam ray-casting simulation(detail in 4.3.2) to obtain its occlusion-aware visibility, which is used to compute information gain(IG). The information gain is based on the behavioral frontier exploration formula [6] and calculate with the selected action $\alpha \in \mathcal{A}$. With utility function 4.4 described detailed in section 3.4.1, the frontier with best heuristic will be selected.

Transition. Once the robot decides the goal, the global planner (A*) takes charge to check the Accessibility and find a proper route leading to the destination.After executing the planned path (or a segment), a new scan z_{t+1} is fused, updating P_{t+1}; frontiers are re-extracted and the loop repeats (workflow in Sec. 3.1).

Reward and objective. In training, we use coverage- and entropy-based rewards. The reward function is aligned with the exploration outcome: coverage rate subtracted by the travel time expended. This reward encapsulates the objective of efficient exploration. The positive reward term indicates that significant new space was mapped relative to cost, whereas a negative reward represnets the robot cost. A typical shaping is

$$r_t = \lambda_C \, \Delta\text{Coverage}_t - \lambda_L \, \Delta\text{steps} \tag{4.1}$$

or:

$$r_t = \lambda_S \, \Delta\big(H_{\text{map}}^{(t)}\big) - \lambda_L \, \Delta\text{steps} \tag{4.2}$$

with success when coverage ≥ 0.95 and no frontier detected anymore. The learning objective is to maximize $\mathbb{E}\big[\sum_{t=0}^{T} \gamma^t r_t\big]$ over policies $\pi(a|s)$.The agent is expected to explore the map in a non-myopic manner, aiming to cover the entire environment as quickly as possible and should be able to achieve full coverage within fewer steps or shorter time.

4.2. Action selection

We computed the two-by-two agreement rate of frontier (frontier) selection under different α based on **1,397** decision frames; the results are shown in Figure 4.2. The heatmap presents two clear structures: one, *low α segment*($\alpha \leq 1.0$) has a high degree of internal consistency; second, the *high α segment* ($\alpha \geq 2.0$) is almost identical across values (the lower right block of the heatmap is close to saturation), i.e., there is significant redundancy in the parameters of this segment. Accordingly, we retain only the two endpoints {α=0.2, α=5.0} as discrete moves: this maximizes the

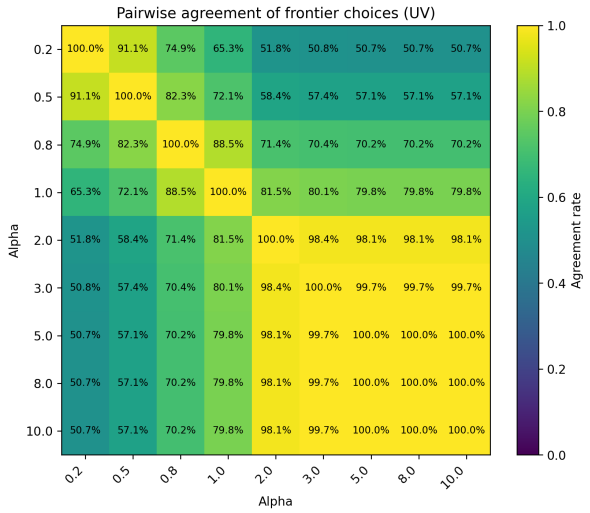


Figure 4.2: Two-by-two agreement rate heatmap for each α frontier selection (brighter colours indicate higher agreement rates), with data aggregated from 1,397 decision frames. It can be seen that the low α (upper left block) is highly internally consistent; the high α (lower right block) is almost "collapsed" to a single choice, suggesting that intermediate values make a limited marginal contribution to the diversity.

separability of the two strategy styles of "conservative vs. aggressive", while avoiding homogeneity and redundancy from intermediate values.

4.3. RL environment setup

To train and evaluate the agent, we developed a simulated Frontier Exploration Environment using a grid-based world simulating an office or school layouts.

4.3.1. Frontier and Candidate Generation

To be more detailed into the algorithm, The map is an occupancy probability grid in $[0, 1]$. Free, unknown, and occupied pixels are separated by simple thresholds. In practice, a pixel is considered free if its value $\leq$ `free_tol` (in this project, default = 35), unknown if it is between `unk_lo` and `unk_hi` (default $[45, 55]$), and occupied if it is $\geq$ `obs_tol` (default ≈ 70). A binary frontier mask is then built by marking a free pixel as a frontier when at least one of its 8-neighbors is unknown. For robustness and collision avoidance, a strict mode removes pixels too close to obstacles, erodes thick edges to a single ring, discards very small components, and optionally removes a narrow border near the map boundary. Thresholds are configured as:

- **Free threshold**: by default, cells with $g \leq 35$ are *free*;
- **Unknown interval**: by default, $g \in [45, 55]$ is *unknown*;
- **Obstacle threshold**: by default, $g \geq 70$ is *occupied*.

Frontier definition. A *frontier* is defined as a *free* cell that has at least one *unknown* cell in its 8-connected (Moore) neighborhood. Implementation proceeds by first thresholding

$$\mathsf{free} := [\, g \leq 35 \,], \qquad \mathsf{unk} := [\, 45 \leq g \leq 55 \,],$$

then counting unknown neighbors by convolving unk with the off-centered 3×3 kernel

$$k_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

and finally intersecting with free:

$$\mathsf{frontier} := \mathsf{free} \wedge \big(\mathrm{conv}(\mathsf{unk}, k_3) \geq 1\big).$$

In the project, this corresponds to a three-step pipeline: (i) `free/unk` thresholding; (ii) neighborhood counting via convolution with k_3; (iii) logical intersection with `free`.

Strict mode (noise/false-positive suppression). To suppress spurious frontiers near obstacles and reduce edge noise, a strict mode is provided via `loose=False`, which applies:

1. **Hard-obstacle adjacency removal:** discard frontier pixels adjacent to *hard obstacles* [$g \geq$ `obs_tol`]; `min_clearance_px` (typically 1–3 px);
2. **Edge thinning:** apply one binary erosion to retain a single-pixel "ring" frontier on thick boundaries;
3. **Small-cluster filtering:** remove connected components with size < `min_frontier_size`;
4. **Border margin removal:** drop a 1 px image margin by default to avoid boundary artifacts.

Default values and switches are exposed via function arguments.

clustering and candidate generation. After finding all the frontiers, the candidates still needs to be trimmed. DBScan(Density-based spatial clustering of applications with noise) is applied to cluster the frontiers nearby and be represent by the centroid/grid selected points.

Summary. The pipeline—*frontier definition* → *strict filtering* → *DBScan clustering* → *representative sampling* yields a coherent candidate-generation chain. It adheres to the standard "free ∧ adjacent-to-unknown" criterion while reducing false positives via clearance/thinning/size constraints, ultimately producing a high-quality, size-controlled candidate set for downstream visibility and information-gain evaluation.

4.3.2. Visibility Model

We adopt the standard *beam-based measurement model* for LiDAR visibility (e.g., [24]). For each candidate frontier f we simulate a set of B beams (uniform in FOV, max range R). Each beam is marched cell-by-cell on the occupancy grid until one of: (i) the range limit is reached, (ii) an *occupied* cell is met (threshold $g \geq$ `obs_tol`), or (iii) the map boundary is crossed. The *visible set* $\mathcal{V}(f)$ is the union of all cells traversed *before the first hit* along all beams (the occluding cell is excluded). We implement the march with an integer Bresenham/DDA-like stepping; the choice of B, FOV, R, and `obs_tol` follows the sensor specification.

4.3.3. Information gain calculation with behavioral entropy

For each grid, the information gain is calculated by behavioral entropy formula with the selected α. For a frontier f,the estimate entropy gain is the summation of per-cell entropies over the occlusion-aware visible set:

$$I_B(f) \;=\; \sum_{c \in \mathcal{V}(f)} H_b\big(p(c)\big) \tag{4.3}$$

With H_B stands for behavioral entropy for each cell

$$H_B(p) = -\sum_{i=1}^{M} w(p_i) \log w(p_i) \tag{4.4}$$

where $w(p)$ is the probability weighting function given by Prelec's model:

$$w(p) = e^{-\beta(-\log p)^{\alpha}}, \quad \alpha, \beta > 0 \tag{4.5}$$

and total information gain is denoted by $I_B(f; \alpha)$.

Utility and frontier ranking. Utilities trade off gain and travel cost,

$$u_B(f, x; M) \;=\; \frac{I_B(f; \alpha)}{|\eta(x, f)|} \tag{4.6}$$

$$f^{\star} \;\in\; \arg\max_{f \in \mathcal{F}} \; u_B(f, x; M) \tag{4.7}$$

where $|\eta(x, f)|$ is the path length from robot pose x to f.

4.3.4. From IG to executed motion.

At each decision step, candidates are sorted by $u_B(f, x; M)$. The global planner (A*) then checks feasibility starting from the top item. If the best candidate is not reachable, the next one in the list is tried until a feasible plan is found. The selected plan is executed for a segment, a new sensor scan is fused, frontiers are re-extracted, and the loop continues. For analysis and supervision, the system also records, for each α in a predefined set, the best utility and the corresponding frontier.

4.4. RL Training Pipeline and Network

The exploration policy is learned with Deep Q-Learning (DQN), augmented by common used DQN extension components including Double Q-learning, Prioritized Replay, and a dueling head.

Background. Q-learning [30] is a model-free, value-based method that learns the optimal state–action value via iterative Bellman updates with learning rate α and discount $\gamma \in [0, 1)$:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\Big]. \tag{4.8}$$

To scale to high-dimensional states, DQN approximates $Q_\theta(s, a)$ with a deep network optimized by the TD loss using a replay buffer D and a periodically updated target network Q_{θ^-}:

$$L(\theta) = \mathbb{E}_{(s,a,r,s') \sim D}\Big[\big(y^{\mathrm{DQN}} - Q_\theta(s, a)\big)^2\Big], \tag{4.9}$$

$$y^{\mathrm{DQN}} = r + \gamma \max_{a'} Q_{\theta^-}(s', a') \tag{4.10}$$

The state s_t encodes the exploration status (processed occupancy grid and robot pose). The action a_t selects a discrete risk-attitude level $\{\alpha_1, \alpha_2, \dots\}$ (see 4.1). After choosing α_t, the environment performs frontier selection and navigation, returns s_{t+1}, and a reward r_t that promotes coverage increase or global entropy reduction and penalizes path length; collisions or getting stuck incur additional penalties. Episodes end on full exploration or a time/budget limit. Training is off-policy with experience replay: transitions (s_t, a_t, r_t, s_{t+1}) are stored and sampled for updates.

Stability and efficiency components.

1. **Double Q-learning** [31]: overestimation is reduced by decoupling action selection (online) and evaluation (target),

$$y^{\mathrm{DDQN}} = r + \gamma\, Q_{\theta^-}\big(s', \arg\max_{a'} Q_\theta(s', a')\big).$$

2. **Prioritized Experience Replay (PER)** [32]: transitions are sampled with probability proportional to the TD-error magnitude, improving sample efficiency (cf. [33]; see §5.3.2).
3. **Dueling architecture** [34]: the head splits into a state-value $V(s)$ and advantages $A(s, a)$ and combines: $Q(s, a) = V(s) + A(s, a) - \frac{1}{|A|}\sum_{a'} A(s, a')$. This component accelerates learning when many actions have similar effects.

CNN grid encoder A three-layer strided CNN, following the widely adopted DQN-style encoder for spatial inputs [33, 35] is employed. An $H \times W$ occupancy grid is formed by stacking $k = 3$ recent frames, $x \in \mathbb{R}^{C_{\text{in}} \times H \times W}$ with $C_{\text{in}} = 2k$ (occupancy probabilities and a robot-position mask). Layers use ReLU and have: (i) 8×8 kernel, stride 4, 64 channels; (ii) 4×4, stride 2, 128 channels; (iii) 3×3, stride 1, 128 channels. For a convolution with kernel size k, stride s, and zero padding b (to avoid overloading p, which elsewhere denotes probability), the spatial size updates as

$$H_{\text{out}} = \left\lfloor \frac{H + 2b - k}{s} \right\rfloor + 1, \qquad W_{\text{out}} = \left\lfloor \frac{W + 2b - k}{s} \right\rfloor + 1. \tag{4.11}$$

With $H = W = 100$ and $b = 0$, sizes evolve $100 \to 24 \to 11 \to 9$, yielding a 128×9×9 tensor that is flattened to a 10,368-dimensional embedding.

Fusion and dueling head. The flattened embedding is mixed by a fully connected layer and split into $V(s)$ and $A(s, a)$ to make robot to better process states that are less associated with actions [34].

Training details. Optimization uses Adam, periodic target updates, gradient clipping at 1.0, ϵ-greedy exploration with annealing, PER with importance-sampling correction, and a discount such as $\gamma = 0.99$.

In conclusion, the Q-network architecture is tailored to the needs of risk-aware exploration. The convolutional frontend processes high-dimensional spatial data into useful features and the dueling head ensures the network can learn complex value distributions for each risk level while efficiently exploring the action space. This architecture, trained with DQN pipeline, forms the core of the methodology, enabling the robot to intelligently decide how to explore unknown environments.

The next chapter will evaluate the performance of this learned policy in simulation, demonstrating the advantages of our risk-aware exploration strategy.

5 Experimental Results

5.1. Experiment setup

We evaluate the performance of the DQN-based risk-aware exploration system in a simulated 2D occupancy grid map environment. Each episode begins with the robot at a random starting pose in an unexplored map. The agent makes exploration decisions at a fixed interval (e.g., each time it finishes reaching a selected frontier). Each episode runs for a maximum number of steps or until the environment is sufficiently explored. For our evaluation, we ran 500 independent episodes for each method.

The map is a 100×100 pixels map. The resolution is 0.1 meters per pixel hence the map is a 10×10square meter office layout. And the selected action space is $\alpha \in [0.2, 5]$.

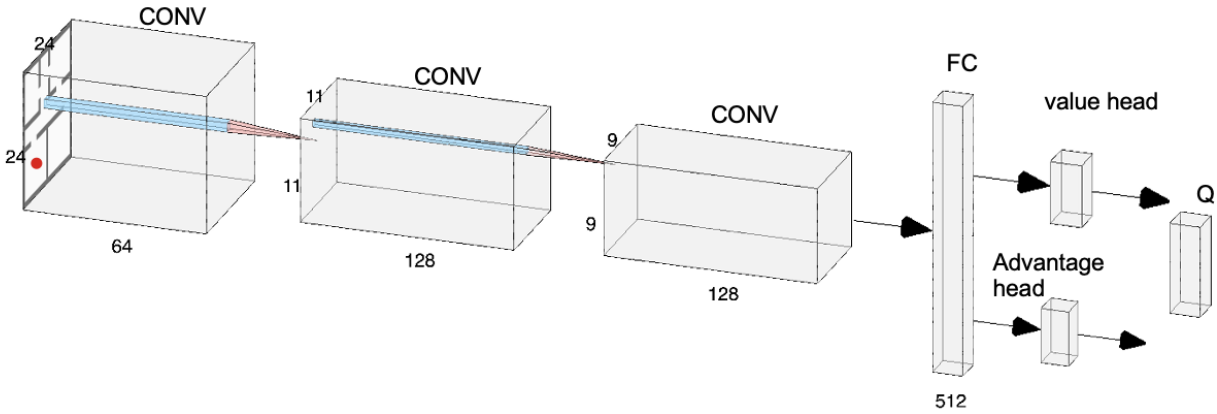


Figure 4.3: Deep Q-Network for risk-aware exploration. Inputs are the occupancy grid and robot pose. A CNN encoder yields a compact state embedding; a dueling (optionally distributional) head outputs $Q(s, a)$ for each discrete risk level.

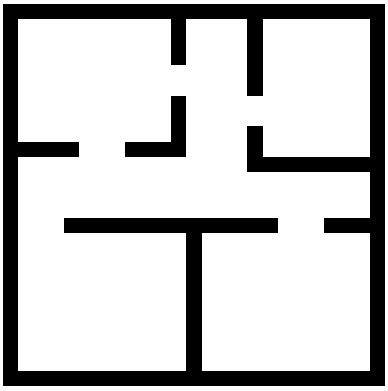


Figure 5.1: Training and evaluation map. Size 100*100 pixels

5.1.1. Baselines and Evaluation Setup

In addition to the learned DQN policy (denoted *model*), we compare against several baseline exploration strategies:

- **Fixed-α (risk parameter)** – Several hand-tuned risk-weighted variants using Prelec's behavioral entropy weighting are evaluated (denoted `hb_0.2hb_0.5`, `hb_0.8`, `hb_2.0`, `hb_3.0hb_5.0`), corresponding to fixed α values in the weighting function.
- **Random-α** – At each decision point, a random α value is chosen from the allowed set. This represents an exploration strategy with no consistent risk preference.
- **Greedy Shannon entropy** – A classical frontier selection method that always selects the frontier maximizing the standard Shannon information gain per travel distance (equivalent to the $\alpha = 1$ case of information gain).
- **Random frontier** – A simple baseline that selects the next exploration frontier uniformly at random, ignoring any information gain metric.
- **Nearest frontier** – A heuristic that always moves to the closest frontier (minimizing travel distance without considering information gain).

We evaluate each method using the following metrics:

- **Final coverage**: the total area of the map covered by the robot at the end of the episode (expressed as a percentage of the environment or in number of grid cells). Higher coverage indicates a more thorough exploration.
- **Total steps to final coverage**: the number of navigation steps taken in the episode. Fewer steps indicate a more efficient exploration trajectory to achieve a given coverage.
- **Total episode return**: the cumulative reward obtained in the episode. In our reward scheme this is proportional to area covered with substracting potential penalties for traveling distance, so a higher return generally reflects greater coverage achieved with fewer penalties.

All methods (including our learned policy and each baseline) were run on the same set of 500 randomized episodes(random starting point) to ensure a fair comparison and conduct a statistical analysis.

5.2. Results

5.2.1. Training Performance

In figure 5.2 and 5.3, the training result is shown. Over the course of 3000 training episodes, the robot's average return steadily increased while the average number of steps required per episode decreased. This indicates that the policy was learning to gather more information (higher returns, correlating with more coverage) in a shorter amount of time. As shown in figure 5.2 in the early training phase the agent often required around 150 steps to reach a given level of coverage, whereas toward the end of training it could achieve comparable coverage in roughly 130–140 steps on average. Similarly, the average episode return improved by approximately 50% from the beginning to the end of training which can be seen in figure 5.3. These trends validate that the learning process produced a more efficient exploration policy.

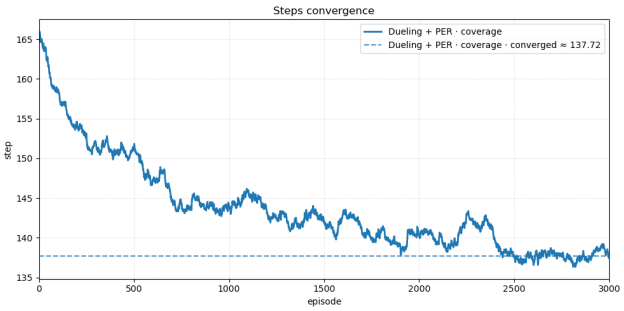


Figure 5.2: Training result: steps(distance travelled) versus episodes

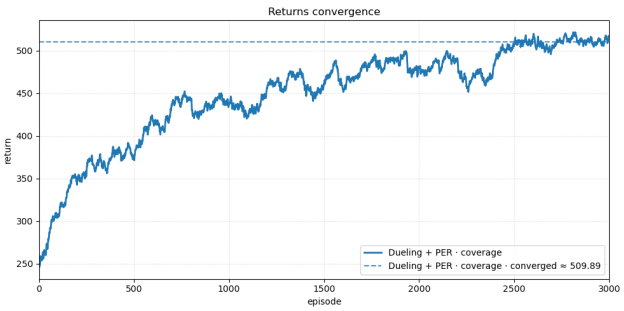


Figure 5.3: Training result: return versus episodes

5.3. Baseline Comparison

Efficiency (steps). Across methods, the learned policy finishes with the lowest average steps (144.3). That is

about 2.0 fewer steps than `alpha_0.5` (1.37%) and 1.6 fewer than `alpha_0.2` (1.10%). Against the heuristics the gap is much wider (32.7% vs. `random_frontier`, 33.6% vs. `nearest_frontier`). In plain terms: the policy tends to pick slightly smarter, lower-detour frontiers throughout an episode rather than relying on an early burst of easy gains.

Return. Average return for `model` is 518.8, essentially tied with `alpha_0.5` (515.6; +0.62%) and below `alpha_0.2` (534.9; 3.01%). Taken together with the lower step count, this yields the strongest *Return/Dec* in Table 5.1: each decision tends to buy a bit more useful progress.

Coverage distributions. On randomized episodes, the time to reach 98% normalized coverage is not only lower on average but also better distributed. The CDFs in Figure 5.3a are visibly left-shifted for `model`: the median sits around $\sim$ 116 steps (vs. $\sim$ 119 for α=1.0), whereas nearest-/random-frontier cluster around $\sim$ 145. The steeper rise in the upper quantiles means fewer runs get stuck cleaning up the last pockets of unknown space.

Figure 5.3b tells a consistent story. At 50% coverage, large-α variants move out quickly (early expansion), and `model` is slightly behind them but already ahead of α=1.0. By 90% the margins are small; by 98% `model` has the lowest median. Importantly, the interquartile width is similar to fixed-α baselines, so the gain does not come with extra variability.

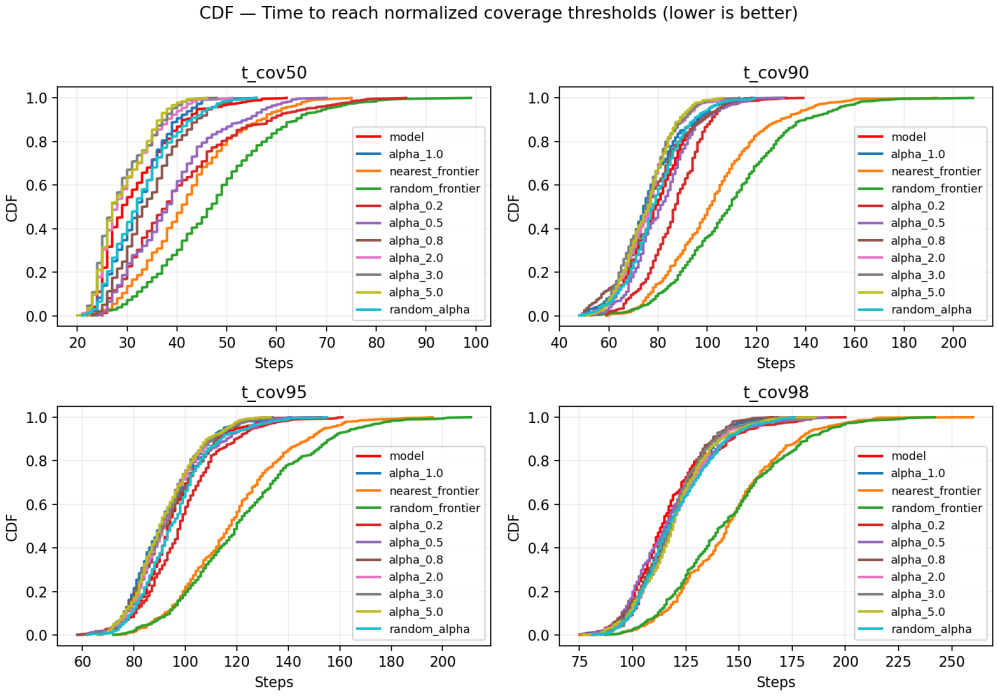
Path and per-decision utility. Secondary metrics in Table 5.1 shows some other advantages of model. Path length stays competitive, while *Entropy gain/Step* and *Return/Decision* are highest for `model`.

If mission time or compute is tight, a policy that reaches high coverage with fewer, more effective decisions is often preferable.

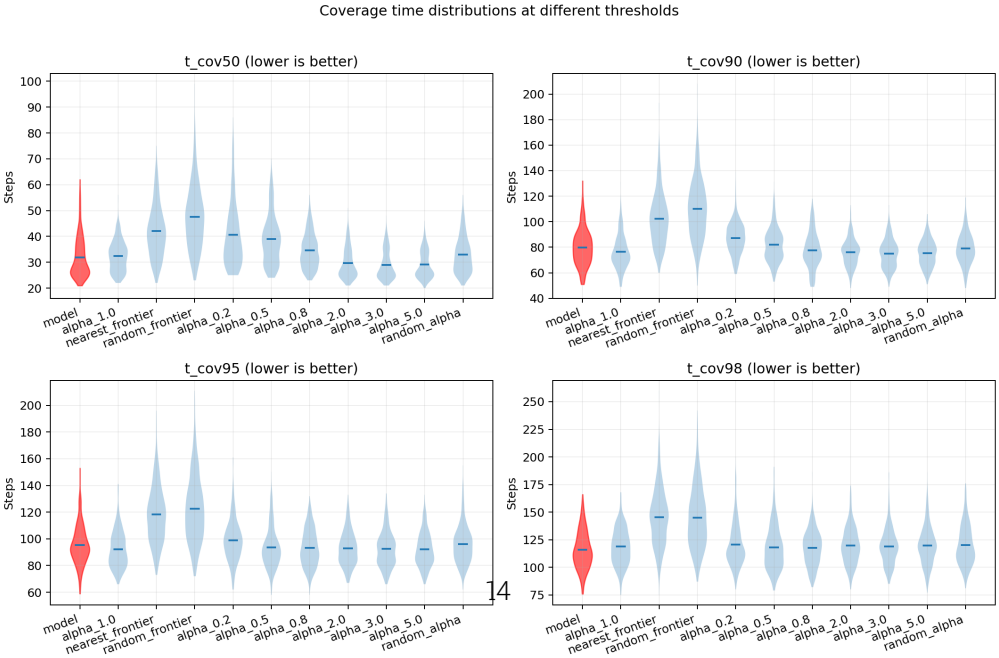
Decision effort vs. progress. High-α baselines use the fewest decision steps (e.g., `alpha_3.0`: 25.05 ± 2.72) yet still need more overall steps to finish. `Model` sits in the middle on decision count (28.00 ± 3.79) but converts those decisions into faster late-stage coverage and better per-decision payoff—consistent with the adaptive risk schedule reported in §5.3.1.

Mode	Return ($\mu \pm \sigma$)	Steps ($\mu \pm \sigma$)	DecSteps ($\mu \pm \sigma$)	Path (m)	Ent/Step (raw)	Return/Dec
model	518.8 ± 177.4	144.3 ± 19.92↓	28.00 ± 3.794	75.68 ± 11.44	**46.18**↑	**18.53**↑
alpha_0.2	534.9 ± 199.0↑	145.9 ± 22.11	33.69 ± 4.363	75.15 ± 12.77↓ ↑	45.54	15.88
alpha_0.5	515.6 ± 200.7	146.3 ± 22.00	31.39 ± 3.640	75.96 ± 12.83	45.38	16.43
alpha_0.8	503.3 ± 163.6	146.7 ± 17.94	29.14 ± 4.059	76.66 ± 10.22	45.59	17.27
alpha_1.0	439.6 ± 175.7	151.5 ± 18.65	26.78 ± 3.740	80.17 ± 10.62	44.14	16.42
alpha_2.0	387.3 ± 191.4	157.0 ± 20.01	25.57 ± 3.095	83.81 ± 11.31	42.43	15.15
alpha_3.0	388.5 ± 188.1	156.8 ± 19.05	25.05 ± 2.723↓	83.99 ± 10.83	42.37	15.51
alpha_5.0	376.1 ± 193.3	158.1 ± 20.68	25.30 ± 2.988	84.66 ± 11.74	42.07	14.87
nearest_frontier	191.0 ± 267.6	177.9 ± 27.04	25.97 ± 2.675	95.99 ± 15.60	37.39	7.35
random_alpha	459.4 ± 193.4	150.1 ± 20.81	28.00 ± 3.180	79.10 ± 11.86	44.23	16.41
random_frontier	206.2 ± 302.8	177.0 ± 30.86	26.14 ± 2.617	95.43 ± 17.95	37.56	7.89

Table 5.1: Compact comparison with highlighted bests. Bold+↑ = max (Return, H/DecStep, Return/Decsteps); Bold+↓ = min (Steps, DecSteps, Path).



(a) CDF of time to reach normalized coverage thresholds (lower is better).



(b) Violin plots of time-to-coverage thresholds (lower is better).

Figure 5.4: Overall results on 500 episodes per method: summary table and distributional analyses.

5.3.1. Adaptive Risk (Alpha) Scheduling

In this experiment, we validate the ability of our learned exploration policy to adapt the risk parameter α over the course of each episode. We analyzed the distribution of α values chosen by the DQN agent at different stages of exploration (by segmenting each episode into coverage intervals).

Figure 5.5 reports, for the learned policy, the *per–step-bin composition* of the risk action, using 10–step bins (1–10, 11–20, . . . , 161–170). Each bar sums to 1 and shows the proportion of conservative ($\alpha \approx 0.2$, blue) versus aggressive ($\alpha = 5$, orange) choices within that bin. Let $p_b(\alpha)$ denote the fraction of steps in bin b on which action α was selected. Three phase-like regimes emerge:

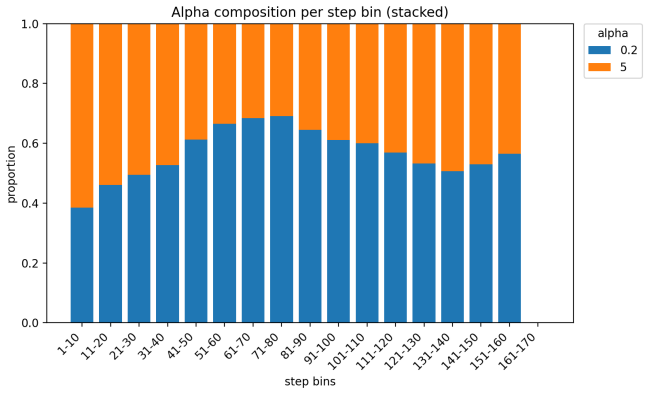


Figure 5.5: Alpha composition per 10-step bin for the learned policy.

Early stage (1–40 steps): open fast with higher risk. The conservative share starts low and rises from ~ 0.38 (1–10) to ~ 0.50 (31–40). The policy thus mixes in more aggressive moves at the beginning, which is consistent with "opening up" the visible space quickly when frontiers are dense and obstacles are less constraining.

Middle stage (40–120 steps): predominantly conservative. From 41–50 to around 71–80 the conservative proportion increases and peaks at $\sim 0.68-0.70$, then remains high ($\sim 0.60-0.63$) through 101–110. This pattern indicates that once the map is partially explored and path costs start to dominate, the policy prefers nearer and more reliable frontiers, reducing detours and failed attempts.

Late stage (120–160+ steps): balanced again, slight re–increase of risk. After the peak, the conservative share gradually declines to $\sim 0.52-0.56$ around 131–150, followed by a mild rebound ($\sim 0.55-0.57$) at 151–160. In practice, the tail of an episode often contains small occluded pockets or narrow passages; a modest increase in aggressive moves can help "break through" these residual bottlenecks. The final bin (161–170) exhibits a similar level, but note that few episodes reach this length (smaller sample size; interpret with caution).

Overall, the policy exhibits an *adaptive risk schedule*: aggressive actions are used to rapidly expand the visible region at the beginning; conservative actions dominate once feasibility and path length become critical; toward the end, the policy returns to a more balanced mix to resolve remaining hard-to-reach areas. This phase-like behavior aligns with the planner-in-the-loop design, where the utility of risk depends on the geometry (frontier density, occlusions) and on the evolving path-cost landscape.

Improvement to be made (i) Because the statistic is computed per *step* within each bin, longer episodes contribute more counts; an episode-weighted variant—first averaging within episode then across episodes—would remove this bias. (ii) Tail bins have fewer samples; adding bin-wise confidence intervals (e.g., Wilson intervals for $p_b(\alpha)$) or a logistic regression of $\Pr(\alpha{=}5 \mid \text{step})$ can quantitatively confirm the early–mid–late trend.

5.3.2. Ablation Study

Figure 5.6 plots the *smoothed* number of steps per episode (lower is better) over training for four configurations.

To isolate the effects of the network head, replay sampling, and reward shaping, four configurations are evaluated under the same environment, action set (discrete α), planner (A*), visibility model, and training schedule:

- **Dueling + PER · coverage**: Dueling DQN head (V/A decomposition with aggregation) and Prioritized Experience Replay (PER) enabled; reward is *coverage-shaped*.
- **Dueling + PER · shannon**: Same as above except the reward uses *Shannon-entropy* reduction instead of coverage-shaped.
- **Only PER · coverage**: Standard (non-dueling) DQN with PER enabled; coverage-shaped reward.
- **Only Dueling · coverage**: Dueling head with uniform replay (PER disabled); coverage-shaped reward.

All other hyperparameters are identical (optimizer, γ, target update, ε-schedule, batch size, seeds), so differences can be attributed to the toggles above.

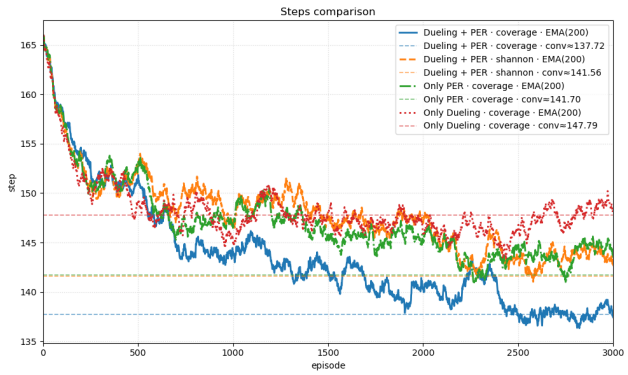


Figure 5.6: Ablation study: different reward mode, and different DQN componet

Ranking and convergence. All methods improve from ∼165 steps at initialization, but they converge to distinct plateaus:

- **Dueling + PER, coverage-shaped reward** (blue) is consistently best, stabilizing around ≈ 137 **steps** by ∼2500 episodes.
- **Dueling + PER, Shannon reward** (orange) converges to ≈ 141 **steps**.
- **Only PER (no Dueling), coverage reward** (green) plateaus at ≈ 141.7 **steps**.
- **Only Dueling (no PER), coverage reward** (red) is worst among the four, at ≈ 147.79 **steps** and shows a slight late-stage drift.

The blue curve separates from the others after ∼600–800 episodes and maintains a steady advantage beyond ∼2000.

Quantitative gaps (late training). Relative to the best of the non-blue baselines, the blue configuration saves ∼5–8 steps (≈3–5%) and ∼8–10 steps (≈5–7%) over the red curve. These differences remain stable in the final training window.

Dueling separates state value $V(s)$ from advantages $A(s,a)$, reducing value-estimation variance when action advantages are small or fluctuate due to feasibility checks, which improves sample efficiency. **PER** (prioritized replay) concentrates updates on large-TD-error transitions (e.g., around A* feasibility changes or risk switches), accelerating learning on the most informative experiences. **Coverage-shaped reward** aligns the learning signal with the operational objective of "fewer steps to explore," providing denser and more directional gradients than a Shannon-only objective.

6 Discussion

This study shows that learning a risk attitude within a classical frontier pipeline yields a faster explorer. Relative to strong fixed-α baselines, the learned policy achieves lower average steps with small but consistent margins, and exhibits a left-shifted distribution for late-stage coverage (e.g., median time-to-98% ≈ **116** vs. α=1.0 ≈ **119**; cf. Figure 5.3a). Per-decision utility is higher (top Ent/Step and Return/Dec in Table 5.1), indicating that each decision tends to contribute more informative progress, especially near the "long tail" of residual pockets.

Behavioral Entropy modulates frontier utilities via the Prelec weighting, where smaller α emphasizes mid-probability regions (beneficial for local clean-up) and larger α emphasizes high-probability mass (beneficial for outward expansion). The learned policy effectively schedules α across the episode—initially favoring expansion, then shifting to clean-up—thereby avoiding stalls observed in fixed-α regimes. Because motion is delegated to A, the network focuses on "which frontier class" rather than "how to traverse," reducing search burden while maintaining planner compatibility.

For missions with tight time or compute budgets (e.g., indoor mapping or valet-parking garage exploration), faster attainment of high coverage thresholds (95–98%) is operationally valuable. The learned policy provides such late-stage advantages while keeping path length competitive. When the mission objective is strictly path-minimal exploration, fixed low α remains a competitive baseline; however, in scenarios prioritizing timely clearance of residual unknowns, the adaptive policy is preferable.

6.1. Limitation

Although the model has been proven as a relatively more effective method for indoor exploration. There are also some shortbacks for either models or for experiments.

Slow computation the learned policy exhibits higher per-step latency than the heuristic because their computational forms differ fundamentally. The heuristic evaluates a closed-form score over M frontier candidates and selects the best, with cost $O(M \log M)$ and $M \ll HW$ (map size $H \times W$). By contrast, the policy performs a full convolutional forward pass over the grid each decision, with complexity $\sum_\ell O(H_\ell W_\ell\, C_\ell^{\text{in}} C_\ell^{\text{out}} k_\ell^2)$, plus encoding and tensor I/O.

Experiment strong related to initial condition : When the episode starts from a side room (left panel of Fig. 5.7), the policy expands outward with fewer revisits, producing a compact trajectory and higher

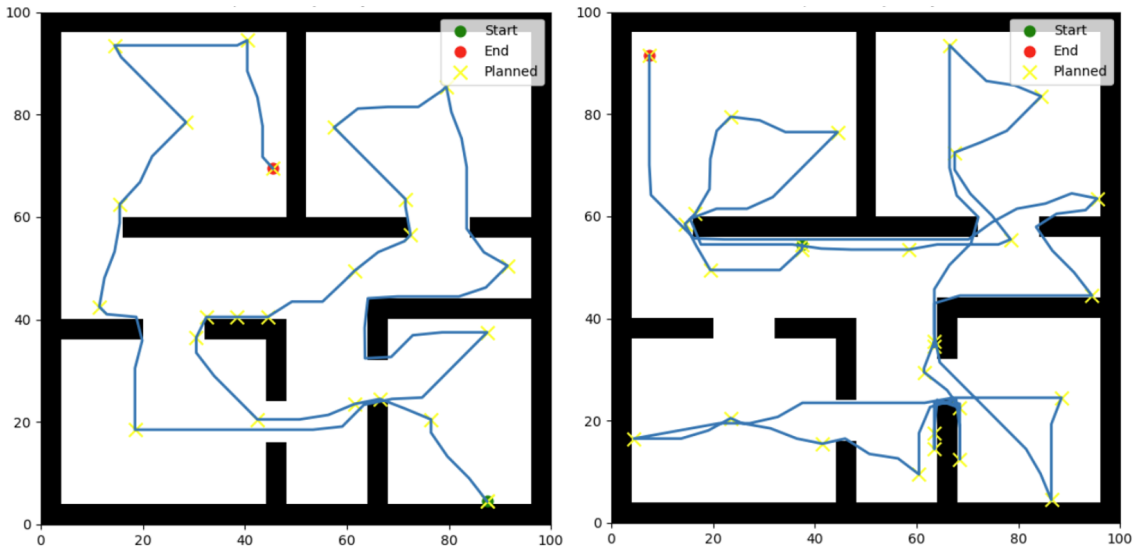


Figure 5.7: Return variance caused purely by start pose. **Left:** starting in a side room encourages radial expansion with few revisits. **Right:** starting on a shared corridor triggers repeated hallway transits between disjoint frontier pockets, wasting steps and rewards.

return. When the episode starts on a shared corridor (right panel), successive replans to reach frontiers located on both sides of the hallway induce repeated back-and-forth transits along the same corridor. These revisitations consume steps without proportional information gain, lowering the dense shaping reward and the final return. This sensitivity demonstrates that identical policies on the same map can yield markedly different returns solely due to the initial location.

6.2. Future work

Three immediate avenues follow from the above. (1) *Risk control*: extend to continuous α and learn a scheduler conditioned on uncertainty maps; connect BE to risk-sensitive RL (e.g., CVaR) for policies with explicit tail guarantees. (2) *Dynamic environments*: integrate short-horizon predictive occupancy (doors/agents) so that α selection accounts for motion and re-occlusion. (3) *Deployment*: validate on real robots and automotive-like platforms (garage exploration for parking), measuring mission-time, energy, and compute budgets end-to-end.

7 Conclusion

This thesis presented a reinforcement-learning-driven framework for autonomous exploration that integrates Behavioral Entropy (BE) into a planner-in-the-loop decision process. By treating the BE risk parameter α as a learnable discrete action rather than a fixed hyperparameter, the system adapts its exploration attitude between risk-seeking and risk-averse modes according to the evolving map state. The learned policy combines the advantages of information-theoretic frontier evaluation and deep Q-learning, yielding a robust, planner-compatible agent capable of balancing information gain against motion effort.

Experiments on synthetic indoor layouts demonstrated that the proposed model achieves faster coverage and a smoother late-stage convergence compared with strong baselines using fixed-α heuristics or classical Shannon utilities. The analysis of action usage revealed an interpretable three-phase risk schedule—aggressive early expansion, conservative mid-stage cleanup, and selective late-stage exploration—consistent with human-like risk adaptation and confirming the utility of behavioral entropy for representing exploration attitude.

Nevertheless, the framework remains limited by its computational cost per decision and sensitivity to initial conditions. Future work could focus on real-world deployment using GPU-accelerated or incremental map encoders, multi-robot coordination, and extension to 3-D or semantic mapping. Incorporating uncertainty in localization and online map updates within the learning loop would further enhance robustness. Overall, the results support the thesis hypothesis that adaptive risk-aware exploration, learned through reinforcement learning, can outperform fixed-parameter strategies while retaining the transparency and safety of classical frontier planning.

References

[1] A. Elfes. "Using occupancy grids for mobile robot perception and navigation". In: *Computer* 22.6 (1989), pp. 46–57. DOI: 10.1109/2.33064. URL: https://ieeexplore.ieee.org/document/30720.

[2] Cyrill Stachniss. *Robotic Mapping and Exploration*. Berlin, Heidelberg: Springer, 2009. ISBN: 978-3-642-01096-5. DOI: 10.1007/978-3-642-01097-2. URL: https://link.springer.com/book/10.1007/978-3-642-01097-2.

[3] Alfréd Rényi. "On Measures of Entropy and Information". In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*. University of California Press, 1961, pp. 547–561. URL: https://projecteuclid.org/proceedings/berkeley-symposium-on-mathematical-statistics-and-probability/Proceedings-of-the-Fourth-Berkeley-Symposium-on-Mathematical-Statistics-and/Chapter/On-Measures-of-Entropy-and-Information/bsmsp/1200512181.

[4] Claude E. Shannon. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423. URL: https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf.

[5] Dirk Holz et al. "Evaluating the Efficiency of Frontier-Based Exploration Strategies". In: *Proceedings of the Joint Conference of the 41st International Symposium on Robotics (ISR) and the 6th German Conference on Robotics (ROBOTIK)*. Munich, Germany, 2010, pp. 36–43. URL: https://www.ais.uni-bonn.de/~holz/papers/holz_2010_isr.pdf.

[6] Aamodh Suresh et al. "Robotic Exploration using Generalized Behavioral Entropy". In: *IEEE Robotics and Automation Letters* 9.9 (2024), pp. 8011–8018. DOI: 10.1109/LRA.2024.3433207. URL: https://doi.org/10.1109/LRA.2024.3433207.

[7] D. Prelec. "The probability weighting function". In: *Management Science* 44.11 (1998), pp. 1589–1605. URL: https://www.jstor.org/stable/2998573?casa_token=5YQEHZSyQZAAAAAA%3AmwFXH1eNZzPT1QXxYwrdDAyDYj6Z6qDinGM2pKgYc75aKSK4NEYF22jFKys0seOewoRTraCa7BZMCm4rFMKGn0S9XhfcMQIrHJVHLmoHMUjwO_ahkSMC.

[8] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. 2nd. MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[9] Haoran Li et al. "Deep Reinforcement Learning-Based Automatic Exploration for Navigation in Unknown Environment". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.6 (2020), pp. 2064–2078. DOI: 10.1109/TNNLS.2019.2927869. arXiv: 2007.11808 [cs.RO]. URL: https://ieeexplore.ieee.org/document/8789673.

[10] Luíza Caetano Garaffa et al. "Reinforcement Learning for Mobile Robotics Exploration: A Survey". In: *IEEE Transactions on Neural Networks and Learning Systems* 34.8 (2023), pp. 3796–3808. DOI: 10.1109/TNNLS.2021.3124466. URL: https://ieeexplore.ieee.org/abstract/document/9612713.

[11] Kenji Leong. "Reinforcement Learning with Frontier-Based Exploration via Autonomous Environment". In: *arXiv preprint* arXiv:2307.07296 (2024). URL: https://arxiv.org/abs/2307.07296.

[12] Liang Lu et al. "Optimal Frontier-Based Autonomous Exploration in Unconstructed Environment Using RGB-D Sensor". In: *Sensors* 20.22 (2020), p. 6507. DOI: 10.3390/s20226507. URL: https://doi.org/10.3390/s20226507.

[13] Anna Dai et al. "Fast Frontier-based Information-driven Autonomous Exploration with an MAV". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9570–9576. DOI: 10.1109/ICRA40945.2020.9196707. URL: https://doi.org/10.1109/ICRA40945.2020.9196707.

[14] Boyu Zhou et al. "FUEL: Fast UAV Exploration using Incremental Frontier Structure and Hierarchical Planning". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 779–786. DOI: 10.1109/LRA.2021.3051563. URL: https://doi.org/10.1109/LRA.2021.3051563.

[15] Armin Hornung et al. "OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees". In: *Autonomous Robots* 34.3 (2013), pp. 189–206. DOI: 10.1007/s10514-012-9321-0.

[16] Andreas Bircher et al. "Receding Horizon "Next-Best-View" Planner for 3D Exploration". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden, 2016, pp. 1462–1468. DOI: 10.1109/ICRA.2016.7487281.

[17] B. Yamauchi. "A frontier-based approach for autonomous exploration". In: *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*. 1997. URL: https://ieeexplore.ieee.org/document/613851.

[18] Jiri Horner. *explore_lite: Lightweight Frontier-Based Exploration*. https://wiki.ros.org/explore_lite. Accessed: 2025-06-24. 2022.

[19] Brian Yamauchi. "Frontier-Based Exploration Using Multiple Robots". In: *Proceedings of the Second International Conference on Autonomous Agents (Agents '98)*. Minneapolis, MN: ACM Press, 1998, pp. 47–53. URL: https://robotfrontier.com/papers/agents98.pdf.

[20] Ludvig Ericson et al. "Information Gain Is Not All You Need". In: *arXiv preprint* arXiv:2504.01980 (2025). URL: https://arxiv.org/abs/2504.01980.

[21] Frédéric Bourgault et al. "Information Based Adaptive Robotic Exploration". In: *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Lausanne, Switzerland: IEEE, 2002, pp. 540–545. ISBN: 0-7803-7398-7. URL: http://ieeexplore.ieee.org/document/1041446.

[22] Henry Carrillo et al. "Autonomous Robotic Exploration Using Occupancy Grid Maps and Graph SLAM Based on Shannon and Renyi Entropy". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 1–8. DOI: 10.1109/ICRA.2015.7139215. URL: https://sites.temple.edu/pdames/files/2016/07/CarilloEtal2015.pdf.

[23] Max Lodel et al. "Where to Look Next: Learning Viewpoint Recommendations for Informative Trajectory Planning". In: *arXiv preprint* arXiv:2203.02381 (2022). URL: https://arxiv.org/abs/2203.02381.

[24] Sebastian Thrun et al. *Probabilistic robotics*. MIT Press, 2005. URL: https://mitpress.mit.edu/9780262201629/probabilistic-robotics/.

[25] Michael Montemerlo et al. "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem". In: *AAAI*. 2003, pp. 593–598. URL: https://cdn.aaai.org/AAAI/2002/AAAI02-089.pdf.

[26] Giorgio Grisetti et al. "Improved techniques for grid mapping with Rao-Blackwellized particle filters". In: *IEEE Transactions on Robotics*. Vol. 23. 1. 2007, pp. 34–46.

[27] Phillip Quin et al. "Approaches for Efficiently Detecting Frontier Cells in Robotics Exploration". In: *Frontiers in Robotics and AI* 8 (2021), p. 616470. DOI: 10.3389/frobt.2021.616470. URL: https://www.frontiersin.org/articles/10.3389/frobt.2021.616470/full.

[28] Anirudh Topiwala et al. "Frontier Based Exploration for Autonomous Robot". In: *arXiv preprint arXiv:1806.03581* (2018). URL: https://arxiv.org/abs/1806.03581.

[29] José M. Amigó et al. "A Brief Review of Generalized Entropies". In: *Entropy* 20.11 (2018), p. 813. DOI: 10.3390/e20110813. URL: https://www.mdpi.com/1099-4300/20/11/813.

[30] Christopher J. C. H. Watkins and Peter Dayan. "Q-learning". In: *Machine Learning* 8 (1992), pp. 279–292. DOI: 10.1007/BF00992698. URL: https://link.springer.com/article/10.1007/BF00992698.

[31] Hado van Hasselt et al. "Deep Reinforcement Learning with Double Q-learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016. URL: https://ojs.aaai.org/index.php/AAAI/article/view/10295.

[32] Tom Schaul et al. "Prioritized Experience Replay". In: *arXiv preprint arXiv:1511.05952* (2015). URL: https://arxiv.org/abs/1511.05952.

[33] Matteo Hessel et al. "Rainbow: Combining Improvements in Deep Reinforcement Learning". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. 2018. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11796.

[34] Ziyu Wang et al. "Dueling Network Architectures for Deep Reinforcement Learning". In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. PMLR, 2016, pp. 1995–2003. URL: https://arxiv.org/abs/1511.06581.

[35] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: *arXiv preprint arXiv:1312.5602* (2013). URL: https://arxiv.org/abs/1312.5602.