Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2025)

Generic Guard AI in Stealth Game with Composite Potential Fields

Kaijie Xu, Clark Verbrugge

Department of Computer Science, McGill University
Montreal, Quebec, Canada
kaijie.xu2@mail.mcgill.ca, clump@cs.mcgill.ca

Abstract

Guard patrol behavior is central to the immersion and strategic depth of stealth games, while most existing systems rely on hand-crafted routes or specialized logic that struggle to balance coverage efficiency and responsive pursuit with believable naturalness. We propose a generic, fully explainable, training-free framework that integrates global knowledge and local information via Composite Potential Fields, combining three interpretable maps—Information, Confidence, and Connectivity—into a single kernel-filtered decision criterion. Our parametric, designer-driven approach requires only a handful of decay and weight parameters—no retraining—to smoothly adapt across both occupancy-grid and NavMesh-partition abstractions. We evaluate on five representative game maps, two player-control policies, and five guard modes, confirming that our method outperforms classical baseline methods in both capture efficiency and patrol naturalness. Finally, we show how common stealth mechanics—distractions and environmental elements—integrate naturally into our framework as sub modules, enabling rapid prototyping of rich, dynamic, and responsive guard behaviors.

Introduction

Stealth games center on the core interactions where players aim to remain undetected by or strategically overcome patrolling guards. Guard patrol behavior is therefore important to the stealth experience, directly influencing perceived challenge, strategic depth, and narrative immersion (Isla 2005; Al Enezi and Verbrugge 2020). In current games, these patrols are usually costly hand-scripted and highly repetitive with low maintainability—enemies follow fixed routes or simple state-machine routines that are easily memorized and hard to adapt to new environments. This over predictability undermines gameplay: when guards consistently fall for the same distractions or never vary their routes, expert players feel they are exploiting a script rather than outsmarting a dynamic adversary. Previous analysis confirms that in many stealth titles, AI agents stick to predictable patterns with minimal response to player actions (Isla 2013). The result is a loss of immersion and reduced player engagement as the illusion of intelligence breaks down.

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Prior attempts to improve guard patrols span various approaches, including probabilistic models to track player positions (Isla 2006, 2013; Hladky and Bulitko 2008; Al Enezi and Verbrugge 2021), dynamic coverage techniques that avoid repetitive paths (Xu, Tremblay, and Verbrugge 2014; Al Enezi and Verbrugge 2020), and reinforcement learning for adaptive multi-agent behaviors (Chia 2022). While these methods address predictability, responsiveness, or coverage, each has trade-offs: probabilistic approaches can be computationally heavy or overly specialized, dynamic coverage sacrifices pursuit efficiency, and learned methods lack interpretability, complicating design. Hence, there remains a clear need for a unified, explainable, and designer-friendly solution that effectively balances exploration, pursuit responsiveness, and natural guard behavior.

To mitigate these limitations, we propose a generic, explainable, and training-free guard AI framework based on *composite potential fields*. This framework integrates three interpretable maps—*information*, *confidence*, and *connectivity*—into a combined decision-making process controlled by simple, designer-tunable parameters. By adjusting these parameters, designers can effortlessly achieve diverse, adaptive guard behaviors without extensive retraining across both occupancy-grid and NavMesh-partition abstractions.

We evaluate our approach across multiple representative game environments, player behaviors, and guard strategies, confirming significant improvements over classical baselines in both patrol efficiency and realism. Our results indicate that our framework achieves strong performance and enables easy incorporation of common stealth mechanics such as distractions, decoys, and environmental interactions, promoting richer, more dynamic gameplay. This work not only addresses immediate challenges in stealth game AI but also offers a basis for future innovations in believable and adaptable common virtual agents. Our key contributions are:

- A generic, explainable guard AI based on composite potential fields integrating global and local information.
- A kernel-filtered decision process enabling effective balance between exploration and pursuit without training.
- Detailed experimental analysis showing better performance in diverse scenarios.
- Straightforward extensions to incorporate common stealth game mechanics in the same unified framework.

Background

Guard Behavior in Games

Guard behavior shapes realism, difficulty, and immersion in stealth games (Al Enezi and Verbrugge 2023a,b). Traditional FSMs and fixed patrol routes are easy to implement but produce predictable and repetitive patterns, enabling players to quickly exploit their limitations. This predictability reduces the challenge and breaks the illusion of intelligent adversaries, weakening long-term engagement (Isla 2005, 2013).

Dynamic, adaptive approaches have been proposed to overcome these shortcomings. Isla's (2006, 2013) occupancy maps guide search by maintaining a probability distribution over the player's likely positions, and Hladký and Bulitko (2008) employ probabilistic inference to anticipate future player movements. Environmental skeletons support coordinated multi-agent search when the player is hidden (Al Enezi and Verbrugge 2021). Al Enezi and Verbrugge's (2020, 2023a) "staleness" metric—where each region's score grows with time since its last visit—drives guards toward under-patrolled areas, balancing control and exploration. We adopt this staleness model as a strong, interpretable baseline in our study. Other methods include generative optimization for guard placement (Xu, Tremblay, and Verbrugge 2014), behavior trees for context-aware transitions, and AI personalities for varied NPC traits (Cernák and Lianoudakis 2021). Deep reinforcement learning coordinates complex ambushes (Chia 2022), but its policies are opaque, computationally expensive, and hard to retune.

Thus, the challenge remains to develop guard AI systems that balance adaptiveness, coverage efficiency, and natural responsiveness, without becoming overly predictable, opaque, or resource-intensive. Our work directly addresses this gap by proposing a generic, explainable, training-free framework using composite potential fields.

Potential Fields in AI

Artificial Potential Fields (APFs), originally introduced in robotics (Khatib 1986), provide a clear, real-time way to steer agents through complex spaces. In PFs, agents are represented as particles influenced by attractive forces from goals and repulsive forces from obstacles. The combination of these forces produces a vector field guiding agents smoothly toward objectives while avoiding collisions (Samodro, Puriyanto, and Caesarendra 2023). PFs have been extensively used in robotics for tasks like obstacle avoidance, path planning, and group coordination. However, traditional PFs can encounter issues such as local minima, where conflicting forces cause agents to become temporarily trapped between multiple influences. Solutions include heuristic methods, stochastic perturbations, and memory-based techniques (Alarabi et al. 2024).

Game AI has adopted the same idea for varied tasks. Beyond fundamental NPC pathfinding and recreating pedestrian social forces (Helbing and Molnar 1995), these field-based techniques have been used in Real-Time Strategy games for coordinating group movements (Hagelbäck and Johansson 2008; Danielsiek et al. 2008) and enabling micro control such as kiting (Uriarte and Ontanón 2012). Influence maps have also been combined with machine learning to analyze game states in StarCraft (Sánchez-Ruiz and Miranda 2017). More recently, PFs are utilized through multi-agent reinforcement learning for generative subgoal orientation (Li et al. 2024). The clear potential of PFs to inform agent behavior across various game scenarios motivates our approach. We focus these concepts by developing *composite potential fields* designed for the complex decision-making of guards in stealth games.

Methods

In this section, we introduce our level abstractions, the composite potential fields, kernel-filtered decision process, and adaptive weights that together govern guard navigation.

Environment Abstractions

Occupancy Grid We discretize the level into an $r \times c$ array of square cells c_{ij}. Both player and guards occupy exactly one cell at a time. At each decision tick, an agent may move to one of its four orthogonal neighbors (up, down, left, right), provided that the cell is walkable. The agent's new position is snapped to the center of the target cell, and its grid coordinates are updated accordingly.

Partition Nodes We build a graph $G = (V, E)$ from Unity's NavMesh triangulation: each triangle's centroid is a node $v \in V$, and edges connect nodes sharing a mesh edge. Guards and player are represented by NavMeshAgent components. When selecting a target node, the agent uses Unity's pathfinder to move continuously along the mesh until the centroid is reached. This leads to smooth, obstacle-aware pathfinding over the partition graph.

Composite Potential Fields

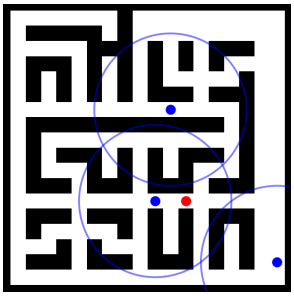
Our framework combines three distinct but complementary potential fields—Information, Confidence, and Connectivity—to guide guard behavior. Figure 1 provides a snapshot of these individual fields operating within a handcrafted test maze, illustrating their respective contributions to the overall decision-making landscape. The player is represented by a red dot, and guards by blue dots. The subsequent paragraphs detail the formulation and role of each field.

Information Field $I(n)$ The Information Field, $I(n)$, represents the "scent" of the player's presence, guiding guards toward potential areas of interest. At each update step, existing $I(n)$ values are first temporally decayed:

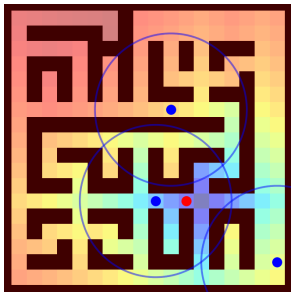
$$I(n) \leftarrow \rho\, I(n),$$

where $\rho \in (0, 1)$ is the temporal decay rate, shared with the Confidence Field, causing the influence of old information to gradually wane over time if not refreshed. The primary update to this field occurs whenever any guard first detects the player at node p_t (the red dot in Figure 1a). At this point, $I(n)$ is reset to 0 for *all* nodes, and a negative-valued potential, indicating strong attraction, is propagated from p_t via Breadth-First Search (BFS). This newly propagated potential, visualized in Figure 1b, spatially decays with distance:

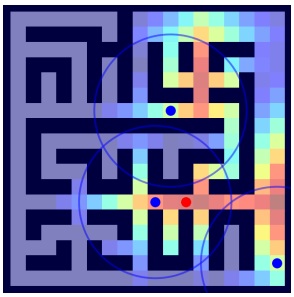
$$I(n) = -I_0\, \gamma_I^{d(p_t,n)}, \quad \forall\, n \in V,$$



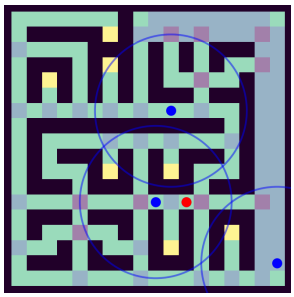
(a) Base Maze Layout with Agents



(b) Information Field $I(n)$ Example



(c) Confidence Field $C(n)$ Example



(d) Connectivity Field $N(n)$ Example

Figure 1: Visualization of the individual components of the Composite Potential Field within our test maze. (a) The base maze layout showing the player (red dot) and three guards (blue dots with vision/influence radii). (b) The Information Field, emanating from the player's current (or last known) position (red dot), creates a strong attractive potential (e.g., represented by cooler colors like blue) that diminishes with distance. (c) The Confidence Field, showing areas influenced by guard presence. Higher confidence (e.g., warmer colors) indicates recently patrolled zones, making them less immediately attractive for revisits by other guards, thus encouraging broader coverage. (d) The static Connectivity Field, where structurally significant areas are highlighted. For instance, more open or connected cells might be represented by one range of values (e.g., lighter green), while bottlenecks or dead-ends (yellowish patches in this example) are differentiated, influencing patrol routes based on their assigned weight.

where $I_0 > 0$ is the initial magnitude of attraction, $\gamma_I \in (0, 1)$ is the spatial decay factor, and $d(p_t, n)$ is the graph or grid distance from p_t to node n. This event-driven update, combined with continuous temporal decay, ensures that guards are strongly guided towards fresh intelligence, while outdated clues diminish in relevance.

Confidence Field $C(n)$ The Confidence Field represents how well areas are currently or have recently been covered by guards. As with the Information Field, existing $C(n)$ values are temporally decayed at each update using the rate ρ. Subsequently, for each guard currently at position g_t (blue dots in Figure 1a), a positive potential (representing an injection of confidence or "patrolled" status) is added to all its neighbor nodes, decaying with distance from the guard:

$$C(n) \leftarrow \rho\, C(n) + \sum_{g \in \text{Guards}} \left(C_0\, \gamma_C^{\,d(g_t, n)} \right),$$

where $C_0 > 0$ is the per-guard injection magnitude—i.e., the confidence boost a single guard deposits at its node—and $\gamma_C \in (0, 1)$ is the spatial distance-decay factor for confidence. Figure 1c illustrates how guard presence (blue dots) creates regions of higher confidence (warmer colors), effectively discouraging immediate revisits to already covered areas and promoting patrol diversity. Thus, each node's confidence reflects a combination of its prior (decayed) coverage and current guard influences.

Connectivity Field $N(n)$ This field is computed once from the static map topology to quantify the local structural properties of each node n, as depicted in Figure 1d:

$$N(n) = \kappa - |\text{Nbr}(n)|,$$

where $\text{Nbr}(n)$ is the set of adjacent (neighboring) nodes to n, and κ is a constant, typically set to the maximum degree observed in the graph (or a fixed value like 4 for grid environments). Nodes with fewer neighbors, such as those in dead-ends or narrow corridors (potentially shown as distinct patches, e.g., yellowish areas in Figure 1d), thus receive higher $N(n)$ values. This field allows guards to prioritize or deprioritize such structural features based on the sign and magnitude of their weights in the composite potential, influencing path choices related to exploration or strategic positioning.

Composite Potential $P(n)$ At each decision point, a guard computes the *Composite Potential* $P(n)$ for all its candidate next nodes. This scalar value is a weighted linear combination of the three individual fields:

$$P(n) = w_I(t)\, I(n) + w_C(t)\, C(n) + w_N(t)\, N(n),$$

where $w_I(t)$, $w_C(t)$, and $w_N(t)$ are the dynamically adjusted weights for the Information, Confidence, and Connectivity fields, respectively. This combined potential $P(n)$ then serves as the input to the kernel-filtered decision process, which ultimately determines the guard's next movement target. The dynamic interplay of these fields and their weights allows the AI to fluidly transition between behaviors such as aggressive pursuit, systematic area patrolling, and strategic exploration of key topological features.

Kernel-Filtered Graph Decision

Candidate Set Selection Before evaluating potential, each guard at current node n_t assembles a local candidate set $\mathcal{C}(n_t)$:

- **Grid Environment** $\mathcal{C}(n_t)$ comprises the four orthogonal neighbor cells, ensuring one-step moves.
- **Partition Environment**: To enable smoother, multi-hop planning, $\mathcal{C}(n_t)$ is primarily constructed by performing a BFS from the guard's current node n_t. This search expands until the accumulated path length to potential candidates reaches a predefined lookahead threshold R_{edge}

(e.g., typically set to 2-3 times the average inter-node distance, tuned empirically to balance foresight and computational cost). The nodes reached at this threshold form the primary candidates. If this BFS-based search yields no viable candidates within the R_{edge} limit (e.g., in very confined spaces or near the global boundary of the explorable area), the candidate set $\mathcal{C}(n_t)$ defaults to include only the directly adjacent (neighboring) nodes to n_t.

This constrained set focuses computation on reachable, locally meaningful waypoints.

Kernel-Filtered Evaluation and Target Selection Each candidate $x \in \mathcal{C}(n_t)$ is scored by a kernel-smoothed potential

$$K(x) = \frac{\sum_{y \in R_\delta(x)} \lambda^{d(x,y)}\, P(y)}{\sum_{y \in R_\delta(x)} \lambda^{d(x,y)}},$$

where:

- $R_\delta(x) = \{\, y \mid d(x,y) \le \delta \}$ is the local neighborhood,
- $d(\cdot,\cdot)$ is grid or graph distance,
- $\lambda \in (0,1)$ controls spatial decay,
- $P(y)$ is the composite potential at node y.

This operation computes a weighted average of P over all nodes within radius δ, suppressing narrow, localized spikes (high-frequency noise) while preserving broader, low-frequency gradients. The guard then selects

$$n_{t+1} = \arg\min_{x \in \mathcal{C}(n_t)} K(x)$$

as its next destination. The selection of the candidate node minimizing (K(x)) directs the guard towards regions with strong attractive signals (e.g., high negative information potential from the player) while balancing the influence of repulsive fields (e.g., high confidence in recently patrolled areas) and topological preferences, thereby achieving a decision that blends pursuit, coverage, and strategic navigation.

Adaptive Weight Scheduling

Weights are updated globally each frame:

$$w_I(t) = \begin{cases} w_I^{\max}, & \text{if player detected,} \\ \max\big(w_I^{\text{base}},\, w_I(t-1) - \delta \Delta t\big), & \text{otherwise.} \end{cases}$$

Thus, $w_I(t)$ is maximized upon player detection; otherwise, it temporally decays towards w_I^{base} by $\delta_w \Delta t$ each update. The remaining budget $1 - w_I(t)$ is split between $w_C(t)$ and $w_N(t)$ according to their predefined base ratio, enabling smooth oscillation between *alert* and *patrol* modes without an explicit FSM.

Experiments

Overview

We evaluate our guard AI under two scenarios with distinct objectives. The *Capture Experiment* assesses short-term pursuit efficiency under heightened difficulty, while the *Fixed-Time Experiment* measures long-term patrol stability in a standard setting. All tests combine two environment abstractions (Grid, Partition), five guard control modes (PotentialField, RandomWalk, FSM, Staleness, Cheat), two player policies (MaxMin, SafetyGraph), and five distinct levels.

Scenario Details

We configure two distinct experiment scenarios with different difficulty levels and duration settings, designed to evaluate complementary aspects of guard AI performance.

Capture Experiment This scenario emphasizes short-term pursuit efficiency under heightened challenge. Players move at 120% of their base speed, while guards' view range is reduced to 80% of their base range, making the player more agile and harder to detect. Player health decreases by one per second for each guard whose attack range encompasses the player. Each trial in this experiment concludes either when the player's health reaches zero (i.e., captured) or after a maximum duration of 60 seconds. This setup measures how rapidly and reliably different guard AI strategies can detect and neutralize a challenging target under pressure.

Fixed-Time Experiment In contrast, this scenario focuses on long-term patrol stability and sustained engagement in a standard gameplay setting. Players and guards operate at 100% base speed and 100% base view range. Each simulation trial runs for a fixed duration of exactly 60s, regardless of the player's health status or whether a capture occurs. The objective here is to assess whether guard strategies maintain effective, evenly distributed patrol patterns and consistent pursuit capabilities over an extended period.

Guard Control Modes

As the internal decision logic of commercial guards is closed-source, we approximate them with simplified baselines. We evaluate five guard-AI methods:

- **Potential Field**: Our proposed method combines dynamic information, confidence, and connectivity fields into a kernel-filtered potential to guide guard movement. Guards continuously adapt their paths to pursue the player and efficiently patrol the area.
- **Random Walk**: Each guard randomly selects one reachable neighboring node or cell at each decision tick, disregarding previous states or player positions.
- **Finite State Machine (FSM)**: Guards switch explicitly between predefined states (Patrol, Investigate, Chase, Return; see Table 2 for details). They patrol random paths until detecting a player, triggering investigation and pursuit behaviors based on set thresholds.
- **Staleness-based FSM**: Guards maintain a "staleness" value (time since last patrol) for each node. The FSM selects the next patrol location based on nodes with the highest staleness, ensuring frequent visits to neglected areas while still enabling standard Investigate–Chase transitions when encountering the player.
- **Cheat**: Guards directly follow the shortest possible path toward the player's current location, serving as an idealized baseline for theoretical optimal performance.

Player Policies

We implement two player-control strategies——MaxMin, a simple yet effective greedy policy, and SafetyGraph, a longer-horizon planning approach—serving as reproducible proxies for human players to evaluate guard effectiveness:

Level Name	NavMesh Properties				Grid Properties		
	Triangles (Count)	Min Area (world units²)	Max Area (world units²)	Avg Area (world units²)	Dimensions (Rows × Cols)	Walkable (0s) (Count (%))	Non-walkable (1s) (Count (%))
Arkham	140	0.1111	20.5000	2.8789	28 × 26	548 (75%)	180 (25%)
Dishonored	238	0.0833	15.3472	2.0090	26 × 39	726 (72%)	288 (28%)
Miami	178	0.0417	9.2083	1.1891	26 × 21	377 (69%)	169 (31%)
Test (Maze)	357	0.0025	6.9525	0.2220	20 × 20	206 (52%)	194 (48%)
TLOU	437	0.0017	2.5359	0.3008	14 × 30	255 (61%)	165 (39%)

Table 1: Characteristics of the five test levels used in the experiments, detailing NavMesh properties (total triangles, min/-max/avg triangle area) and Grid properties (dimensions, count and percentage of walkable (0) vs. non-walkable (1) cells).

- **MaxMin Policy**: At each decision step, the player agent simulates moves several steps ahead, evaluating each potential path by calculating the minimum distance to any guard along that path. The agent chooses the move that maximizes this minimum distance, continuously aiming to stay as far as possible from all guards.
- **Safety Graph Policy**: We construct a "safety graph" by computing two sets of shortest-path distances: from each guard to every node (estimating guard arrival time T_g), and from the player to every node (player arrival time T_p). Each node's safety is quantified by $S = T_g - T_p$, indicating the time margin before potential capture. The player then plans paths using A* search, penalizing paths passing through nodes with lower safety margins, thus preferring routes with lower capture risk.

Test Levels

We test on five maps: a handcrafted Maze ("Test"), Arkham Asylum's Medical Facility ("Arkham"), Dishonored 2's Grand Palace ("Dishonored"), Hitman 2's Miami ("Miami"), and a Seattle Bridge scene from The Last of Us 2 ("TLOU"). Each layout is approximated by a binary 0/1 grid (for the Grid environment) or baked into a NavMesh triangulation (for the Partition environment). Key characteristics of these levels are summarized in Table 1.

The selected levels present a diverse set of challenges for robust AI evaluation. Key factors include overall scale (from the compact "TLOU" grid to the larger "Dishonored"), NavMesh granularity (e.g., "Test (Maze)" and "TLOU" feature dense NavMeshes with numerous small triangles, implying intricate paths, contrasting with "Arkham"'s sparser, larger triangles), and spatial density (e.g., the "Test (Maze)" offers a near 50/50 split of walkable/non-walkable grid cells, indicating a dense, confined structure, while "Arkham" and "Dishonored" provide more open environments with 72-75% walkable area). This diversity in size, navigational complexity, and pathway constructiveness allows for a comprehensive assessment of how different guard AI, especially our Potential Field method, generalize and adapt their behaviors across varied spatial configurations.

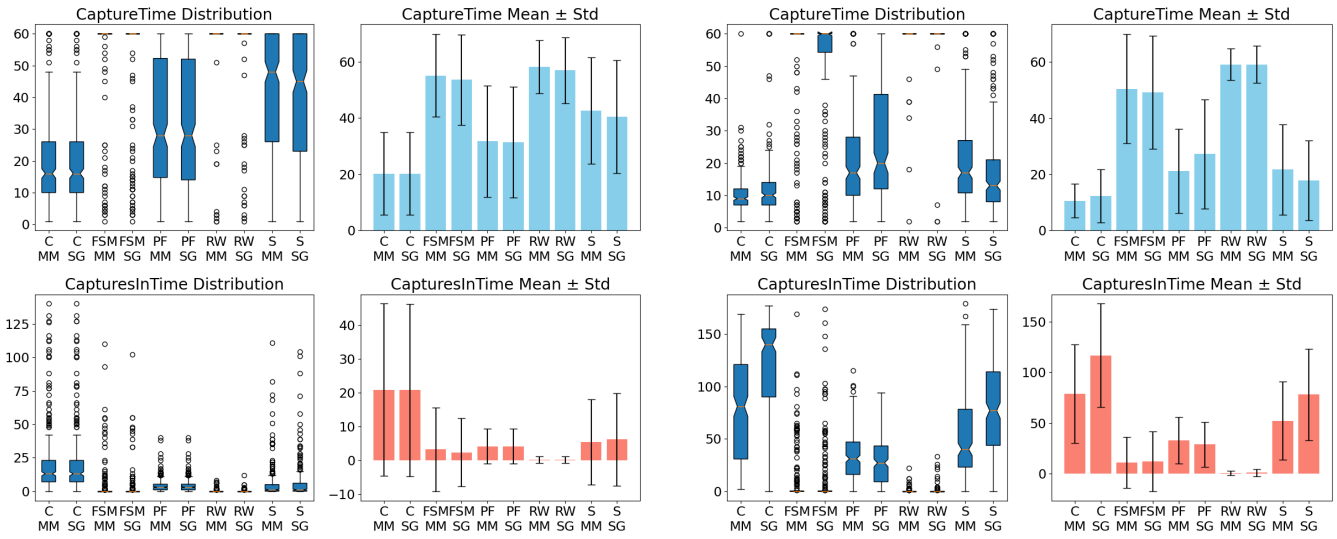
Implementation Details and Parameters

Our experiments were conducted using a specific set of parameters for the proposed Potential Field method, as well as for the baseline FSM and Staleness-based FSM methods. Key parameters are summarized in Table 2. Player agents were initialized with 3 health points (HP). For each experimental repeat, the player was initialized at a grid cell or NavMesh node chosen to maximize its minimum shortest-path distance to any guard, with the additional constraint that the starting position was not within any guard's initial line of sight. This was intended to provide a fair and consistently challenging starting configuration. The simulation update interval for AI decisions was set to 1 second.

For each (environment, guard mode, player policy, level) combination we run $R = 50$ repeats with independent random seeds. We record per-trial metrics—capture time or count, coverage rate, average guard–player distance, first detection time, backtracking count—and export Visit/Attack heatmaps as CSV for post hoc analysis.

Category	Parameter	Value
Potential Field Parameters (Our Method)		
	Information Init (I_0)	5.0
	Confidence Init (C_0)	1.0
	Information Decay (γ_I)	0.9
	Confidence Decay (γ_C)	0.8
	Field (Temporal) Decay (ρ)	0.95
	Kernel Lambda (λ)	0.8
	Kernel Neighborhood (δ)	3
Dynamic Weight Parameters (Our Method)		
	Basic Info Weight (w_I^{base})	0.4
	Basic Conf Weight (w_C^{patrol})	0.5
	Basic Conn Weight (w_N^{patrol})	0.1
	Max Info Weight ($w_I^{\max}$)	0.9
	Weight Decay Rate (δ_w)	0.05
FSM Parameters (Baseline)		
	Info Thresh	1
	T Investigate	3s
	T Lose Sight	2s
	Chase Max Time	10s
	Return Max Time	5s
	Decision Interval	1s
General Simulation		
	Staleness Increment	1
	Player Health	3 HP
	AI Update Interval	1s

Table 2: Key experimental parameters for different AI components and simulation settings.



(a) Grid Environment

(b) Partition Environment

Figure 2: Aggregated performance across all levels and player policies. Abbreviations: C = Cheat, MM = MaxMin, SG = SafetyGraph, PF = PotentialField, RW = RandomWalk, S = Staleness.

Results

We evaluate guard AI performance using several key metrics, each highlighting distinct aspects of effectiveness and behavior quality:

- **CaptureRate** (Capture Exp.): Proportion of successful captures, measuring primary objective achievement.
- **CaptureTime** (Capture Exp.): Time until capture in successful trials; lower values indicate faster pursuit.
- **CapturesInTime** (Fixed-Time Exp.): Total player captures, reflecting sustained engagement capability.
- **CoverageRate**: Percentage of unique map areas visited by guards, indicating patrol thoroughness.
- **AvgGuard–PlayerDist**: Average closest distance between a guard and the player, reflecting pursuit closeness.
- **FirstDetectTime**: Time until the player is first detected, measuring guard vigilance and responsiveness.
- **BacktrackCount**: Frequency of guards immediately revisiting recent locations (e.g., within the last $N = 5$ steps); lower values suggest more naturalistic and less redundant patrol paths.

Together, these metrics provide a multifaceted view of guard performance, from immediate threat response to sustained environmental control.

Aggregated Performance Analysis

Figure 2 and Table 3 summarize results over all five levels and both player policies, for the Grid and Partition variants. We compare our Potential Field method against the idealized Cheat baseline and three realistic baselines: FSM, RandomWalk, and Staleness.

Capture Experiment In the Grid environment (Fig. 2(a), top), PF achieves a *Capture Rate* of 0.80 with mean *Capture Time* 31.7 ± 19.7s, significantly faster than FSM (55.1 ± 14.7s, rate 0.12), RandomWalk (58.2 ± 9.5s, rate 0.04) and Staleness (42.7 ± 18.9s, rate 0.56). Although Cheat remains the lower bound (20.3 ± 14.7s, rate 0.95), PF closes over half that gap compared to classical baselines, confirming that the combination of information, confidence and connectivity fields yields much more effective pursuit under pressure.

In the Partition environment (Fig. 2(b), top), PF captures at 21.1 ± 15.0s with rate 0.93, again outperforming FSM (50.5 ± 19.5s, rate 0.22), RandomWalk (59.2 ± 5.5s, rate 0.03), and even Staleness (21.7 ± 16.2s, rate 0.91). The decaying confidence map prevents guards from clustering prematurely and fosters broad flanking, offering both speed and robustness in complex navmesh layouts.

Fixed-Time Experiment Over 60s in the Grid variant (Fig. 2(a), bottom), PF records 4.18 ± 5.24 captures, coverage 0.945 ± 0.097, average guard–player distance 15.17 ± 5.26m, and first detection at 15.16 ± 15.07s. Staleness FSM slightly exceeds PF in total captures (5.45 ± 12.69) but at the cost of heavy backtracking (9.14 ± 13.67 vs. PF's 2.94 ± 3.06), indicating less efficient routing and more repetitive motion.

In the Partition long-run (Fig. 2(b), bottom), PF yields 32.79 ± 22.97 captures, coverage 1.72 ± 0.23, distance 9.02 ± 4.76m, and detection at 9.95 ± 15.04s. Staleness FSM captures more (52.17 ± 38.85) and covers slightly more area (1.72 ± 0.22) but incurs extremely high backtracking (165.86 ± 25.95), reflecting an unnatural "vacuum-cleaner" revisit pattern. PF strikes a balance: distributed patrols that still converge efficiently on the player without excessive zig-zagging.

		Capture Exp		Fixed-Time Exp				
GuardMode	PlayerMode	CaptureRate	CaptureTime	CapturesInTime	CoverageRate	AvgGuard–PlayerDist	FirstDetectTime	BacktrackCount
Grid Environment								
Cheat	MaxMin	0.95	20.25 ± 14.66	20.85 ± 25.51	0.76 ± 0.16	8.26 ± 2.99	11.46 ± 9.97	15.60 ± 29.36
Cheat	SafetyGraph	0.95	20.23 ± 14.68	20.84 ± 25.52	0.75 ± 0.16	8.27 ± 2.99	11.36 ± 9.96	15.60 ± 29.36
FSM	MaxMin	0.12	55.10 ± 14.68	3.23 ± 12.37	0.61 ± 0.15	17.80 ± 5.86	47.46 ± 22.71	53.34 ± 14.19
FSM	SafetyGraph	0.15	53.60 ± 16.08	2.38 ± 10.08	0.61 ± 0.14	18.29 ± 5.96	45.03 ± 23.97	54.65 ± 14.16
PotentialField	MaxMin	**0.80**	$\mathbf{31.70 \pm 19.72}$	4.18 ± 5.24	$\mathbf{0.94 \pm 0.10}$	$\mathbf{15.17 \pm 5.26}$	$\mathbf{15.16 \pm 15.07}$	$\mathbf{2.94 \pm 3.06}$
PotentialField	SafetyGraph	**0.80**	$\mathbf{31.39 \pm 19.78}$	4.19 ± 5.22	$\mathbf{0.94 \pm 0.10}$	$\mathbf{15.18 \pm 5.25}$	$\mathbf{15.10 \pm 15.06}$	$\mathbf{2.94 \pm 3.06}$
RandomWalk	MaxMin	0.04	58.18 ± 9.48	0.20 ± 0.96	0.61 ± 0.14	17.90 ± 5.80	46.18 ± 23.18	56.48 ± 12.96
RandomWalk	SafetyGraph	0.08	56.92 ± 11.72	0.19 ± 0.94	0.62 ± 0.15	18.23 ± 5.96	45.45 ± 23.19	55.45 ± 13.15
Staleness	MaxMin	0.56	42.65 ± 18.90	$\mathbf{5.45 \pm 12.69}$	0.94 ± 0.08	15.74 ± 4.30	17.29 ± 17.02	9.14 ± 13.67
Staleness	SafetyGraph	0.62	40.51 ± 20.08	$\mathbf{6.18 \pm 13.67}$	0.94 ± 0.08	15.72 ± 4.53	15.35 ± 16.26	9.48 ± 13.49
Partition Environment								
Cheat	MaxMin	1.00	10.61 ± 6.00	78.81 ± 48.61	1.21 ± 0.23	3.30 ± 1.09	4.18 ± 3.13	44.20 ± 48.64
Cheat	SafetyGraph	0.98	12.33 ± 9.53	116.76 ± 51.15	1.18 ± 0.20	3.00 ± 1.43	4.88 ± 4.00	97.29 ± 49.90
FSM	MaxMin	0.22	50.45 ± 19.46	10.77 ± 25.12	0.99 ± 0.27	14.85 ± 5.05	44.86 ± 23.69	64.22 ± 29.66
FSM	SafetyGraph	0.27	49.17 ± 20.13	12.07 ± 29.66	0.98 ± 0.26	14.31 ± 5.16	46.39 ± 23.49	61.72 ± 30.65
PotentialField	MaxMin	**0.93**	$\mathbf{21.10 \pm 14.99}$	32.79 ± 22.97	$\mathbf{1.72 \pm 0.23}$	9.02 ± 4.76	9.95 ± 15.04	$\mathbf{16.69 \pm 13.89}$
PotentialField	SafetyGraph	0.82	27.21 ± 19.36	28.78 ± 22.19	$\mathbf{1.73 \pm 0.21}$	9.62 ± 3.90	$\mathbf{9.43 \pm 13.58}$	$\mathbf{19.43 \pm 15.11}$
RandomWalk	MaxMin	0.03	59.17 ± 5.54	0.42 ± 2.10	0.99 ± 0.26	15.12 ± 4.75	45.86 ± 23.44	76.60 ± 15.07
RandomWalk	SafetyGraph	0.02	59.16 ± 6.59	0.72 ± 3.76	0.99 ± 0.24	14.47 ± 4.77	46.20 ± 22.96	75.54 ± 14.17
Staleness	MaxMin	0.91	21.72 ± 16.15	$\mathbf{52.17 \pm 38.85}$	1.72 ± 0.22	$\mathbf{8.58 \pm 3.78}$	$\mathbf{9.35 \pm 9.26}$	165.86 ± 25.95
Staleness	SafetyGraph	**0.95**	$\mathbf{17.90 \pm 14.24}$	$\mathbf{78.08 \pm 45.30}$	1.61 ± 0.23	$\mathbf{8.14 \pm 4.37}$	9.77 ± 9.55	165.86 ± 25.95

Table 3: Summary of CaptureRate/CaptureTime and all Fixed-Time metrics for both Grid and Partition environments. All values are mean $\pm$ std (two decimals).

Analysis of PF's Behavior The confidence decay in PF encourages guards to spread out and maintain coverage, while the information field quickly refocuses effort once the player is detected—this leads to high capture efficiency in the short-run and plausible patrols over time. This emergent "encircle-and-squeeze" pattern yields fast first captures yet avoids over-concentration, explaining why PF trails Staleness on the *CapturesInTime* count but still posts better coverage and fewer backtracks. As Table 3 shows, PF also slightly outperforms Staleness in *AvgGuard–Player Distance* (Grid: 15.17 vs. 15.74m; Partition: 9.02 vs. 8.58m) and *First Detect Time* (Grid: 15.16 vs. 17.29s; Partition: 9.95 vs. 9.35s) while maintaining the lowest *Backtrack Count*. More aggressive tuning (e.g. boosting information weight upon sighting and slowing its decay) could further tighten pursuit, but naturalistic guard motion is favored to avoid the extreme clustering of Cheat or the repetitive loops of Staleness FSM. Overall, PF significantly outperforms classical baselines and approaches the theoretical limit set by Cheat, across both environments and both experimental conditions.

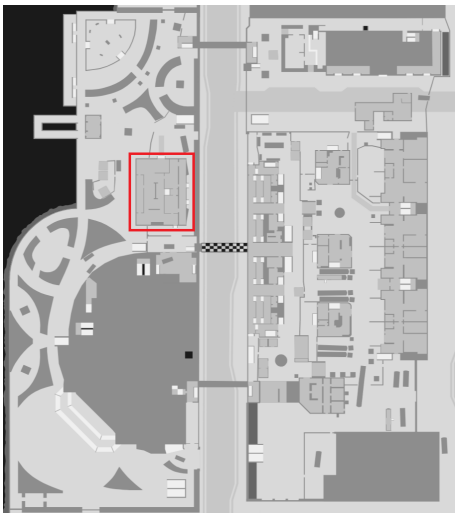
In the Partition environment, uneven NavMesh triangles impose extra replanning overhead, slightly slowing capture times; nonetheless, PF still performs competitively even on Unity's default mesh, confirming its robustness. By contrast, Staleness patrols are largely unaffected by mesh quality. We center our discussion on the MaxMin policy—its straightforward "maximize distance" rule leads to stable, interpretable behavior—whereas SafetyGraph trials occasionally stall in dense triangulations (i.e. in the partition environment, the abundance of near-equivalent shortest paths forces repeated tie-breaking, and our Euclidean heuristic provides little guidance, leading to prolonged A* expansions), as shown by its higher variability in key metrics in Table 3, so we report but do not specifically analyze those results here.

Heatmap Case Study: Miami Level

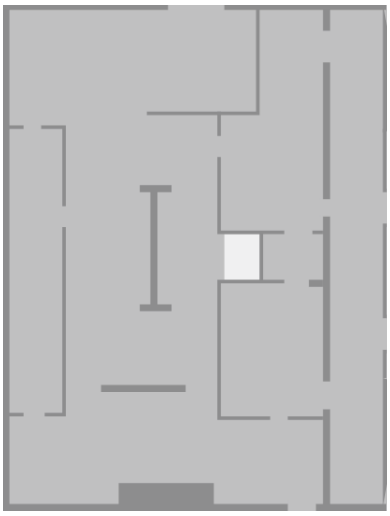
To ground our aggregate results in spatial context, we first present the Miami level geometry used in both experiments (Fig. 3), and then overlay guard *visit* and *attack* heatmaps (Fig. 4). Figure 3a shows the entire Miami level. In the Grid variant (Fig. 3b), walkable areas are rasterized into a 26×21 cell grid; in the Partition variant (Fig. 3c), we triangulate that same area and use triangle centroids as nodes. By keeping geometry identical, we isolate the effect of our composite-field AI on both substrates.

Visit Patterns As shown in Fig. 4a, the *Cheat* and *Staleness* baselines concentrate nearly all visits along a few major corridors, neglecting side passages. *FSM* and *RandomWalk* improve but still waste effort in dead-end branches. In contrast, our PF variants (MaxMin and SafetyGraph) have a more uniform coverage: main routes receive strong attention—mirroring Cheat's axes—while secondary corridors and junctions are also patrolled regularly. This validates PF's ability to avoid both the tunnel-vision of *Cheat*/*Staleness* and the aimless wandering of simpler baselines.

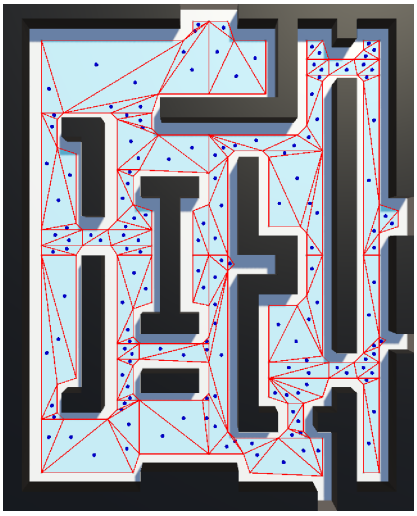
Attack Locations Figure 4b reveals that PF's capture spots align with *Cheat* hotspots—validating its aggressive pursuit—yet PF also secures additional flanking captures in side passages that *Cheat* rarely uses. *FSM* and *RandomWalk*



(a) Full Miami layout (used for both Grid and Partition).



(b) 26 × 21 occupancy grid extract.



(c) NavMesh partition graph (blue=centroids, red=edges).

Figure 3: **Miami map assets** (a) The full layout. (b) Discretized grid for the Grid experiments. (c) NavMesh-derived graph for the Partition experiments.

produce sparse, rare attacks, while *Staleness* seldom captures outside core corridors. Overall, PF not only tracks the player reliably but also leverages distributed coverage to corral the player from multiple approaches, combining strategic focus and spatial control.

Extensible Stealth-Mechanism Simulation

To show the adaptability of our composite potential-field framework, we manually implemented and briefly validated seven common stealth-game mechanics. Each uses simple extensions to our Information and Confidence fields; no large-scale automated tests were performed, only hand-guided simulations to confirm plausibility.

Footstep Noise We maintain a temporary Information field I_{temp} in addition to the global I. Whenever the player sprints, we inject at location x_s:

$$I_{\text{temp}}(y) \;+\!= A_n\,\alpha^{\,d(x_s,y)}, \quad A_n = 3.0,\ \alpha = 0.6,\ d \le 5.$$

Each second $I_{\text{temp}}(y) \leftarrow 0.8 \cdot I_{\text{temp}}(y)$. In our manual simulation, guards briefly veer toward the sprint path and then resume patrol as I_{temp} decays below 0.1.

Thrown Decoy On throw at position x_t, we spawn a one-off impulse in I_{temp} identical to Footstep Noise, then let it decay. We gave the player a 10 s cooldown and allowed placement within a Manhattan radius of 4 cells (avoiding walls). Guards reliably investigate the landing spot before refocusing on normal patrol.

Static Decoy We precompute a persistent "static" Information map I_{static} by seeding

$$I_{\text{static}}(y) = A_d\,\beta^{\,d(x_d,y)}, \quad A_d = 5.0,\ \beta = 0.8,$$

over the entire grid or graph at game start. Once the player interacts at x_d, we add I_{static} into the global I (with full-map BFS), causing guards to converge. After the first guard arrives and "disarms" the decoy for 3 s, we remove I_{static} and observe guards return to baseline patrol.

Corpse Discovery When the player "kills" a guard at x_c, we create another static map I_{corpse} with

$$I_{\text{corpse}}(y) = A_c\,\gamma^{\,t}\,\delta^{\,d(x_c,y)}, \quad A_c = 4.0,\ \gamma = 0.95/\text{s},\ \delta = 0.7.$$

Only after a guard reaches x_c and spends 10 s "inspecting" do we remove I_{corpse}. In our test, this triggered realistic crowding around the body before gradual dispersal.

Lighting Effects In designated "dark" cells we override the global Confidence-decay $\lambda_C = 0.95$ by a faster rate $\lambda'_C = 0.80$. That is,

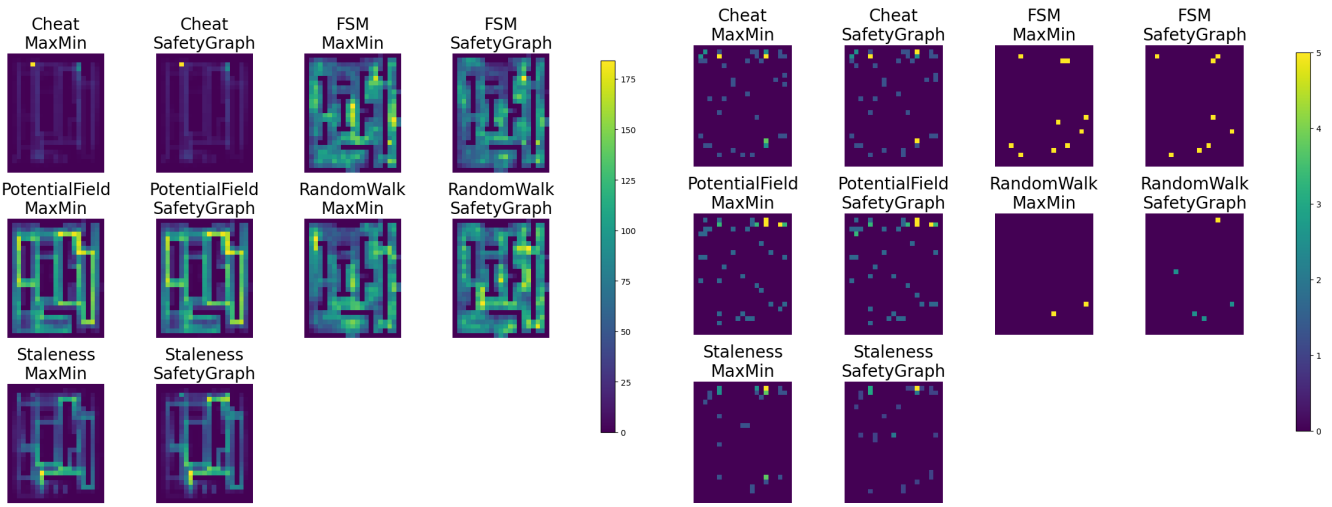
$$C(y) \leftarrow \begin{cases} 0.80\,C(y), & y \in \text{dark} \\ 0.95\,C(y), & \text{otherwise.} \end{cases}$$

Manual trials showed guards re-visit shadowed areas more frequently, simulating cautious search under low light.

Weather Effects When "raining," we halve all footstep amplitudes ($A_n = 1.5$) and increase their decay to $\alpha = 0.7$. Guards in our simulation detected only very close footsteps, modeling sound dampening by rain.

Concealment (Grass, Smoke) Cells tagged as "concealing" apply both Lighting-style confidence decay ($\lambda'_C = 0.80$) and reduce each guard's view range r to $0.7r$. We observed that guards paused at smoke boundaries, hesitated longer before entering tall grass zones, and exhibited believable uncertainty.

Because these extensions merely adjust field-update parameters and injection rules, they slot seamlessly into our existing AI loop. Although we only performed hand-guided playtests to verify each behavior, the framework clearly supports rapid prototyping of diverse stealth mechanisms without specific scripting or new state machines.



(a) Visit Heatmaps — Miami

(b) Attack Heatmaps — Miami

Figure 4: **Visit and Attack Heatmaps** on the Miami level (Grid environment). Brighter cells indicate higher visit or capture counts across all guard–player policy combinations.

Conclusion

In this paper, we introduced a novel, training-free, and explainable framework for generic guard AI in stealth games, centered around the concept of Composite Potential Fields. Our approach integrates three interpretable maps—Information, Confidence, and Connectivity—into a unified, kernel-filtered decision criterion, enabling guards to dynamically balance broad patrol coverage with responsive pursuit of the player. The parametric, designer-driven nature of our system allows for intuitive tuning and adaptation across different environmental abstractions, namely occupancy grids and NavMesh partitions, without the need for retraining.

Our experiments, conducted across five diverse game levels, two player policies, and compared against four distinct baseline guard behaviors, indicate the efficacy of our proposed Potential Field method. The results confirm that PF significantly outperforms classical FSM and Staleness-based approaches in terms of capture efficiency, particularly under high-pressure scenarios. Furthermore, our method achieves superior patrol naturalness, evidenced by higher coverage rates and substantially lower backtracking counts, indicating more plausible and less repetitive guard movements. The heatmap analyses provided qualitative validation of PF's ability to achieve both focused pursuit and comprehensive environmental coverage.

Beyond core patrol and pursuit, we proved the adaptability of our framework by extending it to simulate a variety of common stealth game mechanics, such as footstep noise, decoys, corpse discovery, and environmental effects like lighting and concealment. These extensions were achieved through simple parameter adjustments and additions to the existing field update rules, highlighting the framework's capacity for adaptive prototyping of dynamic guard behaviors.

Limitations Despite these contributions, our work has limitations. The current parameter tuning for the adaptive weights and field dynamics, while effective, was performed manually and could potentially be optimized further, perhaps through a designer-assisted search or limited learning process for specific game styles. While the extensibility demonstrations were plausible, they were validated through hand-guided simulations rather than large-scale quantitative testing, and we have not evaluated how they impact human players in actual gameplay. Additionally, the current model primarily focuses on individual guard decision-making; explicit coordination strategies for multiple guards, beyond emergent behaviors from shared field information, remain an area for deeper exploration. Our Partition implementation also relies on Unity's default NavMesh triangulation—easy to reproduce but vulnerable to uneven mesh quality.

Future works Future work will focus on several key directions. First, we plan to investigate methods for semi-automated tuning of the parameters to better match specific designer goals or desired difficulty curves. Second, we aim to conduct formal user studies with human players to assess perceived naturalness, challenge, and immersion compared to existing AI techniques. Third, exploring better multi-guard coordination strategies built upon the composite potential field foundation is a promising avenue. This could involve guards sharing information more explicitly or taking on specialized roles within the composite field framework. Finally, we intend to apply and evaluate our framework in more complex, large-scale 3D environments with richer stealth mechanics and to evaluate alternative partition strategies—such as level skeletons or optimized mesh segmentation—to further enhance field fidelity and consistency.

We believe our Composite Potential Field framework offers a valuable contribution towards creating more intelligent, believable, and adaptable AI agents in stealth games, providing designers with tunable tools to enhance player immersion and strategic engagement.

References

Al Enezi, W.; and Verbrugge, C. 2020. Dynamic guard patrol in stealth games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16, 160–166.

Al Enezi, W.; and Verbrugge, C. 2021. Skeleton-based multi-agent opponent search. In *2021 IEEE Conference on Games (CoG)*, 1–8.

Al Enezi, W.; and Verbrugge, C. 2023a. Evaluating player experience in stealth games: Dynamic guard patrol behavior study. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 175–184.

Al Enezi, W.; and Verbrugge, C. 2023b. Investigating the influence of behaviors and dialogs on player enjoyment in stealth games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 19, 166–174.

Alarabi, S.; Lei, T.; Santora, M.; Luo, C.; and Sellers, T. 2024. Multi-robot path planning using potential field-based simulated annealing approach. In *Unmanned Systems Technology XXVI*, volume 13055, 102–117. SPIE.

Cernák, M.; and Lianoudakis, O. 2021. *The Tower: Design of Distinct AI Personalities in Stealth-Based Games Modeling personalities with believable and consistent behaviour for NPCs in games using stealth gameplay style*. Master's thesis, Uppsala University.

Chia, A. 2022. The artist and the automaton in digital game production. *Convergence*, 28(2): 389–412.

Danielsiek, H.; Stuer, R.; Thom, A.; Beume, N.; Naujoks, B.; and Preuss, M. 2008. Intelligent moving of groups in real-time strategy games. In *2008 IEEE Symposium On Computational Intelligence and Games*, 71–78. IEEE.

Hagelbäck, J.; and Johansson, S. J. 2008. Using multi-agent potential fields in real-time strategy games. In *Seventh International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 12-16, 2008, Estoril*, 631–638.

Helbing, D.; and Molnar, P. 1995. Social force model for pedestrian dynamics. *Physical review E*, 51(5): 4282.

Hladky, S.; and Bulitko, V. 2008. An evaluation of models for predicting opponent positions in first-person shooter video games. In *2008 IEEE Symposium On Computational Intelligence and Games*, 39–46. IEEE.

Isla, D. 2005. Handling Complexity in the Halo 2 AI System. Presented at the Game Developers Conference. GDC Talk.

Isla, D. 2006. Probabilistic target tracking and search using occupancy maps. *AI Game Programming Wisdom*, 3: 379–388.

Isla, D. 2013. Third Eye Crime: Building a stealth game around occupancy maps. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 9, 206–206.

Khatib, O. 1986. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1): 90–98.

Li, S.; Jiang, H.; Liu, Y.; Zhang, J.; Xu, X.; and Liu, D. 2024. Generative subgoal oriented multi-agent reinforcement learning through potential field. *Neural Networks*, 179: 106552.

Samodro, M. M. J.; Puriyanto, R. D.; and Caesarendra, W. 2023. Artificial potential field path planning algorithm in differential drive mobile robot platform for dynamic environment. *International Journal of Robotics and Control Systems*, 3(2): 161–170.

Sánchez-Ruiz, A. A.; and Miranda, M. 2017. A machine learning approach to predict the winner in StarCraft based on influence maps. *Entertainment Computing*, 19: 29–41.

Uriarte, A.; and Ontanón, S. 2012. Kiting in RTS games using influence maps. In *Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment*, volume 8, 31–36.

Xu, Q.; Tremblay, J.; and Verbrugge, C. 2014. Generative methods for guard and camera placement in stealth games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 10, 87–93.