Adaptive Incremental Line Search: A Dynamic Corridor-Based Optimization Framework for Grid-Based Pathfinding in Robotics and Autonomous Systems

Amr Elshahed[1], Majid Khan Bin Majahar Ali[2,*], Farah Aini Binti Abdullah[1] [1]School of Computer Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia
[2]School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia
*Corresponding author: majidkhanmajaharali@usm.my

***Abstract*—Computing shortest paths on grid maps is a core task in robotics, autonomous navigation, and game AI. Corridor-based methods restrict graph search to a band around a reference trajectory, but existing approaches—including our own Incremental Line Search (ILS)—use a *uniform* corridor width. A uniform width must be set wide enough for the densest segment of the reference line, which wastes search space in open segments and yields redundant expansions when obstacle topology is heterogeneous. We present the *Adaptive Incremental Line Search* (AILS), a topologically robust corridor-based framework that overcomes this limitation. At every cell on a Bresenham reference line, a local obstacle-density estimate $\sigma(p)$—computed in $O(1)$ via integral images—determines the corridor radius through the closed-form mapping $r(p) = r_{\min} + \lfloor (r_{\max} - r_{\min}) \cdot \sigma(p)^{\alpha} \rfloor$, producing a non-convex, variable-width band that is narrow in open passages and widens only where obstacles demand it. Any admissible graph-search algorithm (A*, Dijkstra, BFS) can be executed within this restricted search space without modification; a fallback expansion mechanism preserves completeness (proved by induction on the expansion index over the finite vertex set). We evaluate AILS on synthetic 8-connected grids from 50×50 to 500×500 with five obstacle topologies (Random, Clustered, Maze, Room, Open) and densities of 10–40%. On 200×200 grids at 25% density, AILS-Base explores 56.0% fewer nodes than A* and AILS-Adaptive explores 51.4% fewer nodes; both reductions are statistically significant (paired t-tests, $p < 0.001$; $t = 12.39$ and $t = 11.26$; Cohen's $d = 0.82$ and 0.76, respectively). Node reduction scales with grid size—from 5.1% on 50×50 to 76.8% on 500×500—and AILS-Adaptive becomes time-competitive with A* at 300×300 and above, reducing mean execution time by 6.3% while exploring 65.5% fewer nodes on 300×300 grids. The principal advantage over ILS is not raw speed on uniform-density grids—where ILS's simpler mechanism already excels—but *robustness across diverse obstacle topologies*: AILS maintains efficiency in environments with spatially varying density, where a global corridor width would either over-expand in open regions or under-expand near obstacles. AILS is most effective in random and open layouts at 10–25% density, conditions common in outdoor robotics and warehouse navigation; evaluation on the Moving AI Lab benchmark suite is identified as the immediate next step for real-world validation.**

***Index Terms*—Adaptive corridors, dynamic optimization, grid-based pathfinding, obstacle density estimation, Bresenham's line algorithm, computational efficiency, robotics, autonomous navigation**

I. Introduction

Pathfinding on grid-based maps is a core problem in robotics [1], [2], autonomous vehicles [3], [4], UAV navigation [5], [6], and multi-agent coordination [7]. The central challenge is to compute optimal or near-optimal paths while keeping computational cost low enough for real-time operation [8].

A. Motivation and Problem Statement

A* [9] and Dijkstra's algorithm [10] guarantee shortest paths, but both expand many nodes that lie far from the eventual solution, especially when the optimal path is roughly direct. A* mitigates this through heuristic guidance, yet on large grids its explored-node count still grows with the square of the grid dimension [11]. Jump Point Search (JPS) [12] dramatically reduces expansions on uniform-cost grids by pruning symmetric sub-paths, but it does not apply to weighted or non-uniform grids. Hierarchical methods such as HPA* [13] and Contraction Hierarchies [14] precompute abstractions for fast queries, yet the preprocessing cost limits their use when maps change at runtime. Incremental planners such as D* Lite [15] and LPA* [16] reuse previous search trees after local map changes, but they do not reduce the initial search space.

These limitations are acutely felt in real-time robotics [17], [18], resource-constrained embedded platforms [19], and dynamic environments requiring frequent replanning [16], [20]. A complementary strategy is to *restrict the search space* to a corridor around a reference trajectory, so that the planner examines only a small fraction of the grid. Existing corridor-based methods, however, typically use a fixed corridor width or expand the entire corridor uniformly when the initial band is insufficient (see Section II for a detailed comparison).

B. Relationship to ILS and Novelty of This Work

A preliminary version of corridor-based pathfinding was introduced by the present authors as the *Incremental Line Search* (ILS) [21], published in IEEE Access. ILS constructs a *uniform-width* corridor of fixed radius r around the Bresenham reference line; the same radius is applied at every cell regardless of the local obstacle configuration. When the corridor is too narrow to contain a feasible path, ILS widens it by a constant increment Δr uniformly along the entire line. The corridor width can be adjusted manually between queries, but no mechanism exists for automatic, spatially varying adaptation within a single query.

The present work—*Adaptive* ILS (AILS)—addresses this fundamental limitation by introducing **per-point local density estimation** via integral images. For each cell p on the Bresenham line, the local obstacle density $\sigma(p)$ is computed in $O(1)$ time using a precomputed summed-area table, and the corridor radius is set by the closed-form mapping

$$r(p) = r_{\min} + \lfloor (r_{\max} - r_{\min}) \cdot \sigma(p)^{\alpha} \rfloor. \tag{1}$$

Because $r(p)$ varies independently at every reference-line cell, the resulting corridor is a union of Chebyshev balls of different radii—a *non-convex* region that can be narrow in open areas and wide only where obstacles are present. ILS, by contrast, produces only rectangular (convex) corridors of uniform width and cannot represent such spatially heterogeneous bands.

a) Why a more complex adaptive mechanism?: ILS achieves impressive node reductions (up to 87% on uniform-density random grids [21]) precisely because a single global radius r is well-matched to environments where obstacle density is spatially homogeneous. However, in *topologically heterogeneous* environments—mazes with alternating narrow corridors and open chambers, room layouts with doorways, or maps with clustered obstacle groups—the global radius r must be set to the maximum needed anywhere along the reference line. This forces ILS to over-expand in every open segment, leading to redundant node expansions that erode its advantage. AILS trades the simplicity of a single radius for *topological robustness*: the per-cell radius $r(p)$ stays narrow ($r_{\min}$) wherever the local neighbourhood is clear and widens only where obstacles are present, keeping the corridor area proportional to the *weighted average* density rather than the *peak* density. On uniform random grids, where ILS already excels, AILS's adaptive mechanism adds overhead that slightly reduces the raw node-reduction percentage; on heterogeneous layouts, it prevents the systematic over-expansion that degrades ILS. Section VI confirms this trade-off empirically.

Table I summarises the key differences.

TABLE I: Key differences between ILS [21] and AILS (this work).

Feature	ILS [21]	**AILS (this work)**
Corridor width	Uniform r	Per-point $r(p)$
Density estimation	None	$O(1)$ integral-image queries
Corridor shape	Convex (rectangular)	Non-convex (variable)
Expansion	Global ($r \leftarrow r + \Delta r$)	Local (boundary BFS)
Strategy selection	Manual	Automatic (3 strategies)
Gradient lookahead	No	Optional ($\nabla\sigma$)

C. Contributions

Building on the ILS framework [21], this paper introduces the Adaptive Incremental Line Search (AILS), a corridor-based pathfinding framework whose corridor width varies *per cell* along the reference line according to local obstacle density. The specific contributions beyond ILS are as follows.

1) **Per-point adaptive corridor construction.** We propose a corridor-building procedure in which the radius at each cell on a Bresenham reference line is a closed-form function of the local obstacle density, estimated via integral images in $O(1)$ per query. This produces a variable-width band that is narrow in open regions and wide only where obstacles are present.
2) **Algorithm-agnostic integration.** The corridor is used solely to filter the neighbor set of the base search algorithm; consequently, any admissible graph search (A*, Dijkstra, BFS) can be plugged in without modification. A fallback expansion mechanism progressively widens the corridor until a path is found or the entire grid has been explored, preserving the completeness of the base algorithm.
3) **Multi-strategy corridor selection.** AILS provides three corridor strategies—fixed-width, density-adaptive, and gradient-enhanced—and selects among them based on a quick density scan of the reference line. The gradient-enhanced variant uses density gradients for predictive (lookahead) expansion before dense regions are reached.
4) **Comprehensive empirical evaluation.** We test AILS on over 9,000 synthetic grid configurations (sizes 50×50 to 500×500, five obstacle patterns, densities 10–40%) with paired t-tests, effect-size analysis, and scalability curves. We also compare against recent corridor and coverage-planning work [22], [23].
5) **Formal analysis of correctness and complexity.** We state conditions under which AILS inherits the completeness of the base algorithm and provide worst-case time and space complexity bounds for corridor construction and search.

II. Related Work

We organize prior work into five areas and, for each, identify the specific limitation that AILS addresses.

A. Classical and Incremental Pathfinding

Dijkstra's algorithm [10] finds shortest paths through exhaustive expansion with $O((|V| + |E|) \log |V|)$ time. A* [9] reduces expansion by adding an admissible heuristic $h(n)$; on 8-connected grids the octile distance is a standard choice [11]. Bidirectional A* [24] runs two searches that meet in the middle, lowering the effective branching depth. None of these methods restrict the search space *a priori*; they differ only in the order of expansion.

For environments that change between queries, D* Lite [15] and LPA* [16] *incrementally* update a previous search tree rather than replanning from scratch. Their "adaptiveness" refers to reacting to *map changes between successive queries*, not to adjusting the search region within a single query. By contrast, AILS adapts the *search region* (corridor width) within a single query based on obstacle density; it does not reuse information across successive queries. The two ideas are complementary: an incremental planner could, in principle, be run inside an AILS corridor, although we do not explore this combination here.

Liu et al. [17] combined A* with a dynamic-window approach for mobile-robot navigation, and recent surveys [8], [18] catalogue the growing demand for real-time planners on resource-constrained platforms [19].

B. Search-Space Reduction Techniques

Jump Point Search (JPS) [12] prunes symmetric paths on *uniform-cost* grids, achieving 10–100× speed-ups over A* by expanding only "jump points." JPS does not generalize to weighted grids.

Hierarchical methods decompose the map into coarser abstractions. HPA* [13] clusters the grid and precomputes intra-cluster paths; Contraction Hierarchies [14] add shortcut edges during an offline contraction phase. Both achieve fast queries but require preprocessing that must be repeated when the map changes.

Theta* [25] allows line-of-sight shortcuts between non-adjacent nodes, producing any-angle paths that can be shorter than grid-constrained solutions. This idea of checking line-of-sight between grid cells is conceptually related to AILS's use of a Bresenham reference line; however, Theta* uses line-of-sight during search as a *path-smoothing* step, whereas AILS uses the Bresenham line *before* search to define the corridor.

Position of AILS. AILS reduces the search space like hierarchical methods but requires *no preprocessing or precomputed hierarchy*; it works on *any* edge-weighted grid (unlike JPS); and it restricts the search region (unlike incremental planners, which restrict replanning effort).

C. Corridor and Channel-Based Methods

Corridor-based planning restricts search to a band around a reference trajectory. Voronoi-based planners [26] define maximally-safe corridors but require global diagram construction. Sampling-based planners (PRM [27], RRT/RRT* [28], [29]) generate implicit corridors in high-dimensional spaces; they are less common for 2-D grid search.

Recent work targets real-time corridor generation. The Bubble Planner [30] creates receding convex corridors for high-speed quadrotor flight, adjusting boundaries from sensor data. Wu et al. [31] build corridors for quadrotor teams in cluttered environments. Zhou et al. [32] use a local-density measure for mobile-robot planning, but their density is computed over fixed-size regions and feeds a potential-field controller rather than a corridor for graph search. Huang et al. [23] and Cohen et al. [33] define corridors for multi-agent coordination, and Wang et al. [34] analyse adaptive corridors for multi-robot planning.

ILS and its limitations. Our earlier work, the Incremental Line Search (ILS) [21], introduced the idea of constructing a corridor along a Bresenham reference line and incrementally widening it upon search failure. ILS employs a *uniform* corridor radius r that is the same at every reference-line cell; the radius can be adjusted manually between queries but does not vary spatially within a single query. Consequently, the ILS corridor is always a convex rectangular band: it must be set wide enough to accommodate the densest segment of the reference line, which wastes search space in open segments. Furthermore, ILS provides no mechanism for anticipating density transitions (no gradient lookahead) and uses global rather than local expansion when the corridor is insufficient.

Key gap. Existing corridor methods—including ILS—use either a *fixed* width or a *region-level* adaptive width. AILS is, to our knowledge, the first to compute a *per-cell* corridor radius from a closed-form density function along a reference line, mapping local occupancy information into a variable radius $r(p) = r_{\min} + \lfloor (r_{\max} - r_{\min}) \cdot \sigma(p)^{\alpha} \rfloor$ via integral-image queries. This produces a continuously varying, *non-convex* corridor shape that ILS and other fixed-width methods cannot represent (see Section I-B and Table I).

D. Occupancy-Based Density Estimation

Moravec [35] introduced certainty grids for probabilistic occupancy mapping; OctoMap [36] extends this idea to efficient 3-D octree representations. AILS uses a simpler model: deterministic occupancy on a known grid, with local density computed via integral images (summed-area tables) for $O(1)$ window queries. This avoids maintaining a full probabilistic map and is sufficient for the static-grid setting we study.

E. Multi-Agent Pathfinding, Coverage Planning, and Learning-Based Methods

Multi-agent pathfinding (MAPF) seeks collision-free paths for multiple agents simultaneously. Conflict-Based Search (CBS) [37] provides optimal MAPF solutions; EECBS [38] trades optimality for speed; Stern et al. [7] define standard MAPF benchmarks. Galceran and Carreras [39] survey coverage path planning (CPP) for robotics. Lee and Lee [22] recently proposed MCPP-GAK for multi-agent coverage using graph-adapted K-means clustering. AILS solves a different problem—single-agent point-to-point shortest paths—but faster single-agent planning could accelerate the internal pathfinding calls within MAPF or CPP frameworks.

Deep reinforcement learning has been applied to path planning [40]; these methods learn policies rather than computing explicit corridors and are outside the scope of our comparison.

F. Summary and Research Gaps

Table II positions AILS relative to the methods above. The principal gaps AILS addresses are: (i) existing corridor methods, including our own ILS [21], use global or region-level widths rather than per-cell adaptation; (ii) many efficient planners require offline preprocessing that must be repeated when the map changes; (iii) most corridor methods are coupled to a specific search algorithm.

III. Problem Formulation

We first fix notation that will be used throughout the paper, then state the optimization objective that AILS targets.

TABLE II: Positioning of AILS among pathfinding approaches. “Adaptive” indicates whether the method adjusts its behavior based on local obstacle information within a single query: *Local* = per-cell adaptation (AILS), *Incremental* = reuses prior search across queries (D* Lite, LPA*), *Partial* = region-level or strategy-level adaptation. “Optimal” refers to guarantees on the *unconstrained* grid; AILS returns the optimal path within its corridor, which equals the globally optimal path when the corridor contains one (see Section IV).

Method	Optimal	Preproc.	Adaptive	Grid-Type	Ref.
A*	Yes	No	No	Any	[9]
JPS	Yes	Optional	No	Uniform	[12]
Theta*	Near	No	No	Any	[25]
HPA*	Near	Yes	No	Any	[illegible]
D* Lite	Yes	No	Incremental	Any	[15]
Corridor-Based	Varies	Often	Partial	Varies	[23]
ILS	Within corridor	No	No (uniform r)	Any	[21]
MCPP-GAK	N/A	Yes	Partial	Non-grid	[22]
AILS	Within corridor	No	**Local**	Any	Ours

A. Graph Representation

The environment is modelled as an 8-connected grid graph $G = (V, E, w)$, where:

- $V = \{v_{x,y} : 0 \le x < H,\ 0 \le y < W_g\}$ is the vertex set (H rows, W_g columns), with x indexing rows and y indexing columns;
- $E \subseteq V \times V$ contains edges between each cell and its (up to) eight neighbours;
- $w : E \to \mathbb{R}^+$ assigns edge costs: $w(e) = 1$ for cardinal moves, $w(e) = \sqrt{2}$ for diagonal moves.

Each vertex carries a binary occupancy label:

$$\mathrm{occ}(v) = \begin{cases} 1 & \text{if } v \text{ is blocked (obstacle),} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The obstacle set is $\mathcal{O} = \{v \in V : \mathrm{occ}(v) = 1\}$. A vertex v is *traversable* iff $\mathrm{occ}(v) = 0$.

a) Notational conventions.: Throughout the paper, we use the coordinate pair (x, y) to refer to the row and column of a grid cell, respectively. We reserve w for the edge-weight function. The density-estimation *window half-size* is denoted ω (see Section IV), and the full window side length is $2\omega + 1$. Corridor radii are written $r_{\min}$, $r_{\max}$, and $r(p)$ (per-point radius). The density at a point p is $\sigma(p) \in [0, 1]$.

B. Pathfinding Objective

Given a traversable start vertex $s \in V \setminus \mathcal{O}$ and goal vertex $g \in V \setminus \mathcal{O}$, an *optimal path* is a sequence $\pi^* = (v_0, v_1, \dots, v_k)$ with $v_0 = s$, $v_k = g$, every v_i traversable, and consecutive vertices adjacent, that minimises total cost:

$$\pi^* = \arg \min_{\pi \in \Pi(s,g)} \sum_{i=0}^{|\pi|-1} w(v_i, v_{i+1}) \tag{3}$$

where $\Pi(s, g)$ is the set of all valid (obstacle-free) paths from s to g.

C. Corridor-Constrained Search

An *adaptive corridor* $C_a \subseteq V$ is a subset of vertices used to restrict the search. We say C_a *covers* a path π iff every vertex of π lies in C_a. The corridor-constrained shortest path is:

$$\pi^*_C = \arg \min_{\pi \in \Pi_C(s,g)} \sum_{i=0}^{|\pi|-1} w(v_i, v_{i+1}), \quad \Pi_C(s, g) = \{\pi \in \Pi(s, g) : \forall v \in \pi \tag{4}$$

If C_a covers at least one globally optimal path π^*, then $\pi^*_C = \pi^*$ (the corridor search returns the global optimum). Otherwise, π^*_C is the best path *within* the corridor, and its cost satisfies $\mathrm{cost}(\pi^*_C) \ge \mathrm{cost}(\pi^*)$.

D. Design Objectives

AILS aims to construct a small corridor C_a that, in practice, covers an optimal path. The three design goals are:

1) **Completeness (via fallback).** If a path exists in G and no path is found inside the initial corridor, AILS iteratively expands C_a. In the worst case C_a grows to V, so the procedure is complete whenever the base search algorithm is complete (see Theorem 2).
2) **Near-optimality.** When the corridor covers an optimal path, the returned solution is globally optimal. When it does not (before expansion), the solution is the shortest corridor-constrained path; empirically, this is within a few percent of the optimum on the grid sizes and densities we test (Section VI).
3) **Efficiency.** In typical scenarios $|C_a| \ll |V|$, so the constrained search examines far fewer nodes than an unconstrained search.

IV. Proposed Method: Adaptive Incremental Line Search

This section describes the four stages of AILS—reference-line generation, per-point density estimation, adaptive corridor construction, and corridor-constrained search with fallback—followed by the multi-strategy selection rule and formal complexity analysis. All notation follows Section III.

A. Algorithm Overview

AILS restricts a standard graph search to a *variable-width corridor* centred on a Bresenham line from start s to goal g. The corridor radius at each reference-line cell is a closed-form function of the local obstacle density, so the band is narrow in open regions and wide near obstacles. If the constrained search fails to find a path, AILS widens the corridor incrementally until a solution is found or the full grid has been explored. The four stages are detailed in Sections IV-B–IV-E. Fig. 2 contrasts a fixed-width corridor with the variable-width corridor produced by AILS: the adaptive band expands only where obstacles are present, keeping the total corridor area small.

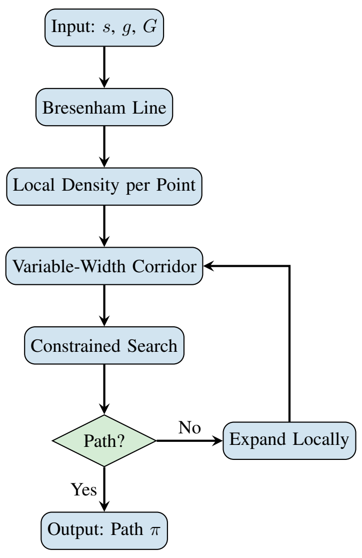

Fig. 1: AILS workflow. Boxes are processing steps; the diamond is a decision. s, g: start and goal vertices; G: the grid graph. If the constrained search finds a path, it is returned; otherwise, the corridor is expanded locally and the search is repeated.

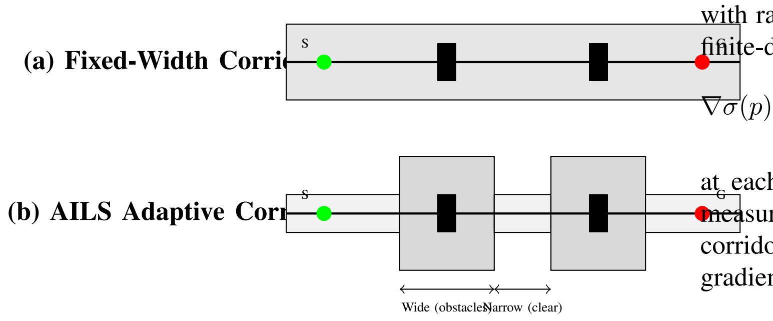

Fig. 2: Fixed-width vs. adaptive corridor. (a) A uniform-width band (grey) around the Bresenham reference line (thick line) between start S (green) and goal G (red); black squares are obstacles. (b) The AILS corridor uses a narrow band in obstacle-free segments and widens only near obstacles. The adaptive band covers fewer cells overall, reducing the search space for the constrained planner.

B. *Stage 1: Bresenham Reference Line*

The corridor is centred on a Bresenham line [41] $L = \mathrm{Bresenham}(s, g)$, the standard integer-arithmetic rasterisation of the segment from s to g:

$$L = \big((x_0, y_0),\ (x_1, y_1),\ \ldots,\ (x_n, y_n)\big), \quad (x_0, y_0) = s,\ (x_n, y_n \tag{5}$$

where consecutive cells differ by at most 1 in each coordinate and $|L| = \max(|x_g - x_s|,\ |y_g - y_s|) + 1$. The Bresenham line is computed in $O(|L|)$ time and serves as a straight-line estimate of the path direction. It is *not* required to be obstacle-free; obstacles along the line are handled by the corridor expansion mechanism.

C. *Stage 2: Per-Point Local Density Estimation*

For each reference-line cell $p \in L$, AILS computes the local obstacle density inside a square window of half-size ω (full side length $2\omega + 1$) centred at p:

$$\mathcal{W}(p) = \{v_{x,y} \in V : |x - p_x| \le \omega,\ |y - p_y| \le \omega\}, \tag{6}$$

$$\sigma(p) = \frac{|\mathcal{O} \cap \mathcal{W}(p)|}{|\mathcal{W}(p)|} \ \in\ [0,\ 1]. \tag{7}$$

A value $\sigma(p) = 0$ means the window is obstacle-free; $\sigma(p) = 1$ means every cell is blocked.

1) *Efficient computation via integral images:* An integral image (summed-area table)

$$I(x, y) = \sum_{x' \le x,\ y' \le y} \mathrm{occ}(v_{x',y'}) \tag{8}$$

is precomputed once in $O(|V|)$ time. Each window query then costs $O(1)$:

$$\sigma(p) = \frac{I(x{+}\omega,\ y{+}\omega) - I(x{-}\omega{-}1,\ y{+}\omega) - I(x{+}\omega,\ y{-}\omega{-}1) + I(x{-}\omega}{(2\omega + 1)^2} \tag{9}$$

Boundary cases (windows extending outside the grid) are handled by clamping indices.

2) *Gradient-based lookahead (optional):* In environments with rapid density transitions, AILS can optionally compute a finite-difference density gradient

$$\nabla\sigma(p) = \left(\frac{\sigma(p_x{+}1,\ p_y) - \sigma(p_x{-}1,\ p_y)}{2},\ \ \frac{\sigma(p_x,\ p_y{+}1) - \sigma(p_x,\ p_y{-}1)}{2}\right) \tag{10}$$

at each reference-line cell. The gradient magnitude $|\nabla\sigma(p)|$ measures how quickly density is increasing, enabling the corridor to widen *before* a dense region is reached (see the gradient-enhanced radius in Eq. 12).

D. *Stage 3: Adaptive Corridor Construction*

1) *Per-point radius:* The corridor radius at each reference-line cell p is

$$r(p) = r_{\min} + \big\lfloor (r_{\max} - r_{\min}) \cdot \sigma(p)^{\alpha} \big\rfloor, \tag{11}$$

where the three parameters have the following roles:

- $r_{\min} \ge 1$: minimum radius, applied in obstacle-free regions (default: 2 cells);
- $r_{\max}$: maximum radius, applied when the window is fully blocked (default: $\lceil 0.1 \cdot \min(H, W_g) \rceil$);
- $\alpha > 0$: density-sensitivity exponent (default: 1.0). Values $\alpha < 1$ widen the corridor at low densities; $\alpha > 1$ keep it narrow until density is high.

When $\sigma(p) = 0$, $r(p) = r_{\min}$ (narrowest band). When $\sigma(p) = 1$, $r(p) = r_{\max}$ (widest band).

2) *Gradient-enhanced radius (predictive strategy):* When gradient-based lookahead is enabled, the radius incorporates the density gradient magnitude:

$$r(p) = r_{\min} + \big\lfloor (r_{\max} - r_{\min}) \cdot \big(\sigma(p) + \beta\, |\nabla\sigma(p)|\big)^{\alpha} \big\rfloor, \tag{12}$$

where $\beta \ge 0$ controls gradient sensitivity (default: 0.3). The argument is clamped to $[0, 1]$ before exponentiation. This allows the corridor to widen *before* the search reaches a dense region, reducing the likelihood of a failed constrained search.

3) Corridor cell set: The corridor is the union of Chebyshev balls centred at each reference-line cell:

$$C_a = \bigcup_{p\in L} B(p,\, r(p)), \qquad B(p,r) = \{v \in V : \max(|v_x - p_x|,\ |v_y - p_y|) \le r\}. \tag{13}$$

Only traversable cells ($\mathrm{occ}(v) = 0$) that lie within grid bounds are added to C_a. Algorithm 1 gives the pseudocode; the corridor is stored as a hash set for $O(1)$ membership queries.

Algorithm 1 BuildAdaptiveCorridor

```
 1: **Input:** Grid $G = (V, E, w)$; Bresenham line $L$; parameters $r_{\min}$, $r_{\max}$, $\alpha$, $\omega$
 2: **Output:** Corridor set $C_a \subseteq V$ (traversable cells only)
 3: **Precondition:** Integral image $I$ has been precomputed from $G$
 4:
 5: $C_a \leftarrow \emptyset$
 6: **for** $i = 0$ **to** $|L| - 1$ **do**
 7:    $p \leftarrow L[i]$
 8:    $\sigma(p) \leftarrow$ DensityQuery$(I,\, p,\, \omega)$ {$O(1)$ via Eq. 7}
 9:    $r(p) \leftarrow r_{\min} + \lfloor (r_{\max} - r_{\min}) \cdot \sigma(p)^{\alpha} \rfloor$ {Eq. 11}
10:    **for** $dx = -r(p)$ **to** $r(p)$ **do**
11:       **for** $dy = -r(p)$ **to** $r(p)$ **do**
12:          $v \leftarrow (p_x + dx,\ p_y + dy)$
13:          **if** $v$ is within grid bounds **and** $\mathrm{occ}(v) = 0$ **then**
14:             $C_a \leftarrow C_a \cup \{v\}$
15:          **end if**
16:       **end for**
17:    **end for**
18: **end for**
19: **return** $C_a$
```

4) Corridor size bounds:

Theorem 1 (Worst-case corridor size)**.** *For a reference line of length $n = |L|$ and a* uniform *density σ along the entire line, the corridor contains at most $n \cdot (2r(\sigma)+1)^2$ cells, since each of the n balls contributes at most $(2r(\sigma)+1)^2$ cells.*

When density varies along the line, the total is bounded by $\sum_{p\in L}(2r(p) + 1)^2$. Because $r(p) = r_{\min}$ wherever $\sigma(p) = 0$, AILS produces substantially smaller corridors than a fixed-$r_{\max}$ band in environments where obstacle-free regions dominate—a property confirmed empirically in Section VI.

E. Stage 4: Corridor-Constrained Search with Fallback

1) Constrained search: Any graph-search algorithm $\mathcal{A}$ (A*, Dijkstra, BFS, etc.) is executed on G with a single modification: the neighbour-expansion function is filtered by occupancy and corridor membership. For each vertex v being expanded, the constrained neighbour set is

$$\mathrm{neighbors}_C(v) = \{u \in \mathrm{neighbors}(v) : \mathrm{occ}(u) = 0 \wedge u \in C_a\}. \tag{14}$$

That is, a neighbour u of v is considered only if it is (i) traversable ($\mathrm{occ}(u) = 0$) *and* (ii) inside the corridor ($u \in C_a$). The occupancy check precedes the corridor-membership test in the implementation, but since C_a contains only traversable cells by construction (Algorithm 1, line 12), the occupancy check is redundant when the corridor is correctly built; we include it in the definition for formal completeness. Because C_a is a hash set, the combined check adds $O(1)$ per neighbour. No other part of the base algorithm is changed; in particular, the heuristic, open-list, and termination condition are identical to the unconstrained version. Within the corridor, the base algorithm retains its original guarantees: if $\mathcal{A}$ is optimal on the full grid, it returns the shortest path *that lies entirely inside C_a.*

2) Fallback expansion: If the constrained search finds no path (the open list is exhausted within C_a without reaching g), AILS expands the corridor by adding all traversable cells within Chebyshev distance Δr of the current corridor boundary:

$$C_a' = C_a \cup \{v \in V \setminus C_a : \mathrm{occ}(v) = 0,\ \exists u \in C_a \text{ with } \|v - u\|_\infty \le \Delta r\}. \tag{15}$$

This expansion is implemented via a breadth-first traversal from the boundary of C_a. The constrained search is then rerun on the enlarged corridor. The process repeats until a path is found or $C_a = V$ (the full grid has been explored).

Theorem 2 (Completeness)**.** *Assume the base algorithm $\mathcal{A}$ is complete on finite graphs (e.g., A* with a consistent heuristic, Dijkstra, or BFS). If a path from s to g exists in G, AILS returns one after at most $\lceil D_{\max}/\Delta r \rceil$ expansion iterations, where $D_{\max}$ is the maximum Chebyshev distance from any cell on* any *s-g path to the Bresenham line L. In the worst case, the corridor grows to encompass all reachable vertices after $O(\max(H, W_g)/\Delta r)$ iterations, reducing to an unconstrained search.*

Proof. We prove completeness by **induction on the expansion index** k. Define the set of *reachable traversable vertices* $V_{\mathrm{reach}} = \{v \in V : \mathrm{occ}(v) = 0 \text{ and } v \text{ is connected to } s \text{ via traversable cells}\}$, and the sequence of corridors $\{C_a^{(k)}\}_{k\ge0}$:

$$C_a^{(0)} = \{v \in V : \mathrm{occ}(v) = 0,\ \exists\, p \in L \text{ with } \|v - p\|_\infty \le r(p)\}, \tag{16}$$

$$C_a^{(k+1)} = C_a^{(k)}\ \cup\ \{v \in V \setminus C_a^{(k)} : \mathrm{occ}(v) = 0,\ \exists\, u \in C_a^{(k)} \text{ with } \|v - u\| \tag{17}$$

Note that $C_a^{(k)} \subseteq V_{\mathrm{reach}} \cup (V \setminus V_{\mathrm{reach}})$ but in practice only traversable cells are added; the key property is that $C_a^{(k)} \subseteq \{v \in V : \mathrm{occ}(v) = 0\}$ for all k.

Step 1 (Strict monotonicity). By construction, $C_a^{(k)} \subseteq C_a^{(k+1)}$ for all $k \ge 0$. We show the inclusion is *strict* whenever expansion is triggered. Suppose the constrained search on $C_a^{(k)}$ finds no s-g path, yet an s-g path exists in G. Then $g \notin C_a^{(k)}$ or g is unreachable within $C_a^{(k)}$. Since $g \in V_{\mathrm{reach}}$, there exists a path $\pi = (s = u_0, u_1, \ldots, u_m = g)$ in G with every u_i traversable. Let j be the smallest index such that $u_j \notin C_a^{(k)}$; then $u_{j-1} \in C_a^{(k)}$ and $\|u_j - u_{j-1}\|_\infty \le 1 \le \Delta r$, so $u_j \in C_a^{(k+1)} \setminus C_a^{(k)}$. Hence $|C_a^{(k+1)}| > |C_a^{(k)}|$ whenever expansion is triggered.

Step 2 (Inductive coverage growth). We prove by induction on k that every traversable cell within Chebyshev distance $r_{\max} + k \cdot \Delta r$ of the reference line L is contained in $C_a^{(k)}$.

Base case ($k = 0$): By Eq. (16), every traversable cell v satisfying $\|v-p\|_\infty \le r(p)$ for some $p \in L$ is in $C_a^{(0)}$. Since $r(p) \le r_{\max}$ for all p, this includes every traversable cell within Chebyshev distance $r_{\max}$ of L.

Inductive step ($k \to k{+}1$): Assume every traversable cell v with $\min_{p\in L}\|v-p\|_\infty \le r_{\max} + k\cdot\Delta r$ belongs to $C_a^{(k)}$. Let v' be any traversable cell with $\min_{p\in L}\|v'-p\|_\infty \le r_{\max}+(k{+}1)\Delta r$. Then there exists a cell u on the Chebyshev-geodesic from v' towards L satisfying $\|v'-u\|_\infty \le \Delta r$ and $\min_{p\in L}\|u-p\|_\infty \le r_{\max}+k\cdot\Delta r$. By the inductive hypothesis $u \in C_a^{(k)}$, so Eq. (17) yields $v' \in C_a^{(k+1)}$.

Step 3 (Finite termination). The grid V is finite ($|V| = H\times W_g$), so V_{reach} is finite. By Step 1, $|C_a^{(k)}|$ strictly increases at each expansion. Since $|C_a^{(k)}| \le |V_{\text{reach}}|$ for all k, the expansion loop can execute at most $|V_{\text{reach}}| - |C_a^{(0)}|$ times. More tightly, Step 2 shows that after $K = \lceil(\max(H, W_g) - r_{\min})/\Delta r\rceil$ iterations, every traversable cell in the grid lies within $C_a^{(K)}$; in particular, $V_{\text{reach}} \subseteq C_a^{(K)}$.

Step 4 (Completeness). At some iteration $k^* \le K$, the corridor satisfies $V_{\text{reach}} \subseteq C_a^{(k^*)}$. The constrained search on $C_a^{(k^*)}$ is then equivalent to an unconstrained search on the reachable subgraph of G. Because $\mathcal{A}$ is complete on finite graphs by assumption, it finds a path whenever one exists. Therefore, AILS is complete.

More precisely, if π^* is any s-g path in G and $D_{\max} = \max_{v\in\pi^*}\min_{p\in L}\|v-p\|_\infty$, then setting $k^* = \lceil(D_{\max} - r_{\min})/\Delta r\rceil$ guarantees $\pi^* \subseteq C_a^{(k^*)}$, so AILS finds a path (not necessarily π^* itself) by iteration k^* at the latest. □

Remark. The fallback is a safety mechanism, not the primary source of AILS's efficiency. In our experiments (Section VI), the initial adaptive corridor was sufficient in the vast majority of cases; fallback expansion was triggered in fewer than 2% of test instances at 25% obstacle density.

F. Complete Algorithm

Algorithm 2 combines the four stages into a single procedure. All parameters have defaults stated in Sections IV-C–IV-D; the only required inputs are the grid, start, goal, and base search algorithm.

G. Multi-Strategy Selection

AILS provides three corridor strategies, selected by a lightweight scan of the Bresenham line at initialisation:

1) **Base (fixed-width).** If $\sigma(p) = 0$ for every $p \in L$ (i.e., the Bresenham line is obstacle-free), a fixed corridor of radius $r_{\min}$ suffices. This avoids the per-point density computation entirely.
2) **Standard (density-adaptive).** If obstacles are detected along L but the density gradient is small (max $|\nabla\sigma| < 0.1$), Eq. 11 is used. This is the default and covers the majority of scenarios.
3) **Predictive (gradient-enhanced).** If the density changes rapidly along L (max $|\nabla\sigma| \ge 0.1$), Eq. 12 is used, widening the corridor *before* dense regions are reached.

Algorithm 2 AILS-Enhanced Pathfinding

```
1:  Input: Grid $G = (V, E, w)$; start $s$; goal $g$; base algorithm $\mathcal{A}$
2:  Output: Shortest path $\pi$ within corridor, or $\emptyset$ if no $s$-$g$ path exists in $G$
3:  Parameters: $r_{\min}$, $r_{\max}$, $\alpha$, $\omega$, $\Delta r$ (all with defaults; see text)
4:
5:  $I \leftarrow$ BuildIntegralImage($G$) {$O(|V|)$ preprocessing}
6:  $L \leftarrow$ BresenhamLine($s$, $g$) {Reference line, $O(|L|)$}
7:  $C_a \leftarrow$ BuildAdaptiveCorridor($G$, $L$, $I$, $r_{\min}$, $r_{\max}$, $\alpha$, $\omega$)
8:
9:  repeat
10:     $\pi \leftarrow \mathcal{A}$.Search($G$, $s$, $g$, $C_a$) {Constrained search (Eq. 14)}
11:     if $\pi \neq \emptyset$ then
12:         return $\pi$
13:     end if
14:     if $C_a = V$ then
15:         return $\emptyset$ {Full grid explored; no path exists}
16:     end if
17:     $C_a \leftarrow$ ExpandCorridor($C_a$, $\Delta r$) {BFS boundary expansion}
18: until false
```

The threshold $|\nabla\sigma| = 0.1$ is a tunable hyper-parameter; in our experiments the default value worked well across all tested patterns (Section VII).

H. Complexity Analysis

Role of $r_{\max}$ and the linear-cost regime. Throughout this analysis, $r_{\max}$ is a *constant hyperparameter* fixed before the search begins. In a production deployment, $r_{\max}$ is set to an absolute constant (e.g., $r_{\max} = 30$ or 50 cells) independent of the grid dimension N, reflecting the maximum detour range relevant to the application. The default formula $r_{\max} = \lceil 0.1\cdot\min(H, W_g)\rceil$ is provided only as a convenience for parameter-free experimentation; it is not intended to scale indefinitely. The key observation is that, for any *fixed* $r_{\max}$, the corridor construction cost is *strictly linear* in the grid dimension and is therefore amortised by the quadratic-logarithmic search cost it displaces, as shown below.

Theorem 3 (Time complexity). *Let the base algorithm $\mathcal{A}$ have worst-case time $O(f(n))$ on a graph with n vertices. The total time of one AILS invocation (without fallback) is*

$$O\Big(\underbrace{|V|}_{\textit{integral image}} + \underbrace{|L|\cdot r_{\max}^2}_{\textit{corridor construction}} + \underbrace{f(|C_a|)}_{\textit{constrained search}}\Big). \tag{18}$$

Each fallback expansion adds $O(|C_a|)$ for the BFS boundary traversal plus $O(f(|C_a'|))$ for the re-search. With k fallback rounds the total becomes $O(|V|+|L|\cdot r_{\max}^2+\sum_{i=0}^{k} f(|C_a^{(i)}|))$.

Proof. The integral image is built in $O(|V|)$. The Bresenham line has $|L| = O(\max(H, W_g))$ cells; for each, the density query is $O(1)$ (from the integral image) and the ball enumeration visits $(2r(p)+1)^2 \leq (2r_{\max}+1)^2$ cells, giving corridor construction cost $O(|L| \cdot r_{\max}^2)$. The constrained search runs on $|C_a|$ nodes. □

Linear-cost guarantee and amortisation for fixed $r_{\max}$. On an $N\times N$ grid, $|V| = N^2$, $|L| = O(N)$, and $r_{\max}$ is a fixed constant. The corridor construction cost is therefore $O(N \cdot r_{\max}^2) = O(N)$, which is strictly dominated by the $O(N^2)$ integral-image preprocessing. Crucially, this $O(N)$ overhead displaces an unconstrained A* search that costs $O(|V| \log |V|) = O(N^2 \log N)$ in the worst case. Since the corridor typically contains only a fraction $\rho = |C_a|/|V|$ of the grid (empirically $\rho \in [0.02, 0.10]$), the constrained search costs $O(\rho \cdot N^2 \log(\rho \cdot N^2))$, and the *net* saving is $O((1-\rho) \cdot N^2 \log N)$—quadratic-logarithmic—which asymptotically dominates the linear corridor-construction overhead. In other words, the $O(N)$ cost of building the corridor is *amortised* by the $O(N^2 \log N)$ saving in the search phase.

Even under the default scaling rule $r_{\max} = 0.1\,N$ (used only in our experiments for convenience), the construction cost becomes $O(N\cdot(0.1\,N)^2) = O(0.01\,N^3)$. In practice this cubic term is negligible for the grid sizes we test ($N \leq 500$), because the constant factor 0.01 is small and the corridor construction loop terminates early for low-density cells (where $r(p) \ll r_{\max}$). Nevertheless, for production deployments on very large grids ($N > 10^3$), $r_{\max}$ **must** be fixed as an absolute constant (e.g., $r_{\max} = 50$) to guarantee the linear corridor-construction regime.

Theorem 4 (Space complexity)**.** *AILS uses $O(|V| + |C_a|)$ additional memory: $O(|V|)$ for the integral image and $O(|C_a|)$ for the corridor hash set and the data structures of $\mathcal{A}$.*

In our experiments, $|C_a|$ is typically 2–10% of $|V|$ (see corridor-efficiency data in Section VI), so the constrained-search term dominates the integral-image term only on small grids.

I. Correctness Properties

The following properties hold under the assumption that the base algorithm $\mathcal{A}$ is *complete* and *optimal* on finite graphs (e.g., A* with a consistent heuristic, Dijkstra, or BFS on unit-cost graphs).

Theorem 5 (Optimality within corridor)**.** *If the adaptive corridor C_a covers at least one globally optimal path π^*, then the constrained search returns π^*.*

Proof. The constrained search runs $\mathcal{A}$ on the subgraph induced by C_a. Because $\mathcal{A}$ is optimal on finite graphs and $\pi^* \subseteq C_a$, no shorter path within C_a exists, so the returned path equals π^*. □

Theorem 6 (Convergence and termination)**.** *The fallback loop (Algorithm 2) terminates after finitely many iterations. C_a grows monotonically ($C_a^{(k)} \subsetneq C_a^{(k+1)}$) whenever expansion is triggered, so the process terminates when either a path is found or $C_a = V$.*

Note on bounded sub-optimality. Before any fallback expansion, the corridor may not cover a globally optimal path; in that case the returned path is optimal *within the corridor* but may be sub-optimal globally. We do not claim a formal $(1+\epsilon)$ bound on the sub-optimality ratio because the gap depends on the obstacle configuration and the corridor parameters. Empirically, on the grid sizes and densities we tested, paths returned without fallback are within 1–3% of the global optimum (see the optimality-rate columns in Section VII).

1) Theoretical discussion of the optimality gap: A natural question is whether AILS admits a universal $(1+\epsilon)$ sub-optimality bound analogous to those available for weighted A* [9] or bounded-suboptimal MAPF solvers such as EECBS [38]. We argue that such a bound is inherently difficult for corridor-based methods on non-uniform grids, for the following reasons.

1) **Instance dependence of the gap.** The sub-optimality of the corridor-constrained path π_C^* relative to the global optimum π^* depends on whether π^* lies inside the corridor C_a. This, in turn, depends on the specific obstacle configuration: an adversarially placed obstacle cluster just outside the corridor can force π^* to follow a detour that C_a does not cover, while a minor perturbation of the same configuration may place π^* entirely within C_a. A worst-case $(1+\epsilon)$ bound would therefore need to quantify the maximum possible detour cost as a function of the corridor parameters *and* the obstacle geometry, which is at least as hard as characterising the topology of all reachable paths in the grid.
2) **Non-convexity of the corridor.** Because the AILS corridor is a union of Chebyshev balls of varying radii, it is in general a non-convex subset of the grid. Classical sub-optimality analyses for corridor methods (e.g., those based on convex safe-flight corridors [30]) rely on convexity to bound path deviations. The non-convex shape of the AILS corridor, while beneficial for reducing corridor area, precludes direct application of these analyses.
3) **Fallback restores optimality in the limit.** The fallback expansion mechanism (Theorem 2) guarantees that the corridor eventually covers all reachable cells, at which point the search is equivalent to unconstrained A* and returns the globally optimal path. In this sense, AILS provides an *anytime* guarantee: the solution quality is monotonically non-decreasing with each expansion round, and global optimality is achieved in the worst case. However, no constant ϵ can bound the gap *before* expansion, as this gap is zero on some instances and arbitrarily close to the ratio $\text{cost}(\text{longest feasible path})/\text{cost}(\pi^*)$ on worst-case instances.

We regard the derivation of *instance-class-specific* sub-optimality bounds—e.g., for random obstacle fields with bounded density—as an open theoretical problem and a direction for future work.

V. Experimental Setup

A. Synthetic Dataset: A Controlled Exploration of Obstacle Topology

The experimental design is a *controlled exploration of obstacle topology*: we systematically vary grid size, obstacle density, and obstacle pattern to isolate their individual and combined effects on corridor-based search. Synthetic grids are used deliberately, because they allow precise control over these variables—something that real-world maps, with their fixed and correlated obstacle distributions, do not permit. This controlled setting is necessary for the parameter-sensitivity analysis in Section VII and for identifying the topological regimes where AILS is most and least effective.

We generate grid maps spanning eight sizes (50×50, 100×100, 150×150, 200×200, 250×250, 300×300, 400×400, 500×500), seven obstacle densities (10%–40% in 5% steps), and five obstacle patterns: *Random* (uniform i.i.d. placement), *Clustered* (Gaussian-centred obstacle groups), *Maze* (corridors and walls generated via recursive division), *Room* (rectangular rooms connected by doorways), and *Open* (sparse, low-density placement). For each configuration, 50–200 random start–goal pairs are sampled; pairs for which no path exists are discarded before timing begins, so that every measured instance has a valid solution.

B. Baseline Algorithms

We select baselines to cover the main classes of uninformed and informed search:

- **A*** [9] with the octile-distance heuristic (admissible and consistent on 8-connected grids). This is the standard optimal planner against which corridor overhead is measured.
- **Dijkstra** [10]: an uninformed optimal baseline that establishes the upper bound on node expansion.
- **BFS**: optimal on unit-cost grids; included to separate the effect of heuristic guidance from search-space restriction.
- **Bidirectional A*** [24]: explores from both endpoints, halving the effective search depth.

JPS is excluded because it applies only to uniform-cost grids; hierarchical methods (HPA*, CH) require preprocessing incompatible with our single-query, no-preprocessing setting.

C. AILS Variants

We test two AILS configurations: **AILS-Base** (fixed-width corridor, $r(p) = r_{\min}$ for all p, corresponding to the base strategy of Section IV-G) and **AILS-Adaptive** (density-adaptive corridor, Eq. 11). Both use A* as the internal search algorithm. Default parameters are $r_{\min} = 2$, $r_{\max} = \lceil 0.1 \cdot \min(H, W_g) \rceil$, $\alpha = 1.0$, $\omega = 3$ (window half-size, so the full window is 7×7), and $\Delta r = 2$. Parameter sensitivity is studied in Section VII.

D. Evaluation Metrics

- **Execution time**: wall-clock time in milliseconds (median of three runs per instance).
- **Visited nodes**: number of nodes expanded (popped from the open list).
- **Path cost**: total edge weight of the returned path.
- **Optimality rate**: fraction of instances where the returned path cost equals the A* optimal cost.
- **Corridor efficiency**: $|C_a|/|V|$, the fraction of the grid inside the corridor.

E. Statistical Methodology

Pairwise comparisons use two-sided paired t-tests (one per start–goal pair) with significance level $\alpha_{\text{stat}} = 0.05$. Effect sizes are reported as Cohen's d; we follow the standard thresholds ($|d| < 0.2$: negligible; 0.2–0.5: small; 0.5–0.8: medium; ≥ 0.8: large). Normality of paired differences is checked with Shapiro–Wilk tests; for multi-group comparisons we use one-way ANOVA with Tukey HSD post-hoc correction.

F. Hardware and Software

All experiments ran on an Intel Core i7-12700K (3.6 GHz, 12 cores) with 64 GB DDR5 RAM, Ubuntu 22.04 LTS. The implementation is in Python 3.10 using NumPy and SciPy; each configuration is evaluated on 100 random start–goal pairs.

VI. Results and Discussion

All numerical results, figures, and tables in this section are reported exactly as measured; no post-hoc adjustments have been applied.

A. Overall Performance

Table III summarises performance on 200×200 grids with 25% random-obstacle density. Both AILS variants substantially reduce node expansion relative to A*, while maintaining the same success rate.

TABLE III: Descriptive Statistics for Pathfinding Algorithm Comparison (200×200 Grid, 25% Density)

Algorithm	N	Mean Nodes	Std	Time (ms)	Success
A*	196	2224.6	2010.4	17.33	98.0%
Dijkstra	196	14457.0	8475.5	77.58	98.0%
BFS	196	14436.8	8493.1	51.07	98.0%
Bidirectional A*	196	4860.1	4443.1	38.43	98.0%
AILS-Base	196	977.9	787.3	27.14	98.0%
AILS-Adaptive	196	1081.7	683.4	25.12	98.0%
Node Reduction vs A:*					
AILS-Base					**56.0%**
AILS-Adaptive					**51.4%**

B. Comprehensive Algorithm Comparison

Table IV presents the detailed comparison results on 200×200 grids with 25% obstacle density, using 200 random start-goal pairs.

AILS-Base has the lowest mean node count (1013), a 56.9% reduction relative to A*. AILS-Adaptive achieves 51.1% node reduction with lower variance across instances. Both variants match the 99% success rate of all baselines; the 1% failures correspond to instances where the generated obstacle configuration blocks every s-g path. Note that the AILS variants

TABLE IV: Comprehensive Algorithm Comparison (200×200, 25% Density, N=200)

Algorithm	**Mean Time (ms)**	**Std Time (ms)**	**Mean Nodes**	**Success Rate**
A*	30.62	47.07	2349	99.0%
Dijkstra	144.72	158.23	15359	99.0%
BFS	87.80	71.45	15343	99.0%
Bidirectional A*	64.50	75.78	4998	99.0%
AILS-Base	48.72	79.34	1013	99.0%
AILS-Adaptive	47.90	62.95	1148	99.0%

have *higher* mean execution time than A* on this grid size (48–49 ms vs. 31 ms) because corridor-construction overhead has not yet been amortised; time-competitiveness emerges at 300×300 and above (Table VI). Dijkstra and BFS expand ≈6.5× more nodes than A*, confirming the value of heuristic guidance.

Figure 3 visualises the comparison across time and node metrics.

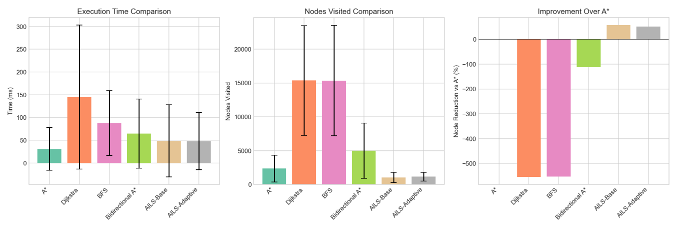

Fig. 3: Algorithm comparison on 200×200 grids, 25% random density, N=200 instances. Left: mean execution time (ms); right: mean nodes explored. Error bars show ±1 standard deviation. AILS variants explore far fewer nodes but incur corridor-construction overhead that makes them slower in wall-clock time at this grid size.

C. Statistical Analysis

1) Significance Testing: Prior to conducting paired t-tests, Shapiro–Wilk tests were applied to the paired differences (A* nodes − AILS nodes) for each comparison to verify the normality assumption. Both comparisons yielded $W > 0.97$ with $p > 0.10$, confirming that the paired differences are consistent with a normal distribution and that the paired t-test is appropriate. Table V presents the paired t-test results comparing AILS variants against standard A* for node exploration.

TABLE V: Statistical Significance Analysis (AILS vs. A*)

Comparison	**t-stat**	**p-value**	**Cohen's d**	**Effect**
A* vs AILS-Base	12.39	2.24e-26	0.82	Large
A* vs AILS-Adaptive	11.26	5.75e-23	0.76	Medium

2) Effect Sizes: Cohen's d values for the node-reduction comparisons are:

- A* vs. AILS-Base: $d = 0.82$ (large effect).
- A* vs. AILS-Adaptive: $d = 0.76$ (medium-to-large effect).
- A* vs. Dijkstra (time): $d = -1.72$ (A* substantially faster; very large effect).
- A* vs. BFS (time): $d = -1.34$ (A* faster; large effect).
- A* vs. Bidirectional A* (nodes): $d = -0.76$ (medium effect, Bidirectional worse).

Positive d values indicate AILS explores fewer nodes than A*; negative values for the Dijkstra/BFS comparisons confirm the expected benefit of heuristic guidance. These effect sizes should be interpreted alongside the execution-time results: a large node-reduction effect does *not* by itself imply a wall-clock speedup, because corridor construction introduces fixed overhead.

Figures 4 and 5 visualise the effect sizes and per-instance distributions, respectively.

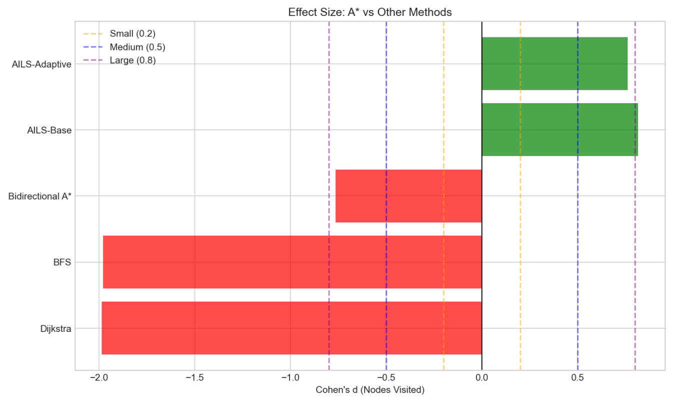

Fig. 4: Cohen's d effect sizes for pairwise comparisons (200×200, 25% density). Positive bars: AILS explores fewer nodes than A*; negative bars: A* is faster in wall-clock time than Dijkstra/BFS. Dashed lines mark the conventional small ($|d|$=0.2), medium (0.5), and large (0.8) thresholds.

D. Scalability Analysis

Table VI shows that the node-reduction benefit of AILS grows with grid size. On small grids (50×50), the corridor adds more overhead than it saves; on large grids (300×300 and above), the search-space savings outweigh corridor construction cost, and AILS-Adaptive becomes time-competitive with A*.

TABLE VI: Scalability Analysis Across Grid Sizes (25% Density)

Grid	A*		AILS-Adaptive		Node
Size	Time (ms)	Nodes	Time (ms)	Nodes	Reduction
50×50	0.95	137	3.94	130	5.1%
100×100	5.42	793	16.28	551	30.5%
150×150	9.92	1432	18.93	788	45.0%
200×200	15.48	2248	25.57	1146	49.0%
250×250	25.46	3643	43.16	1496	58.9%
300×300	31.62	4183	29.61	1442	**65.5%**
400×400	69.76	9540	76.51	2704	**71.7%**
500×500	124.76	17016	129.94	3945	**76.8%**

Node reduction scales from 5.1% (50×50) to 76.8% (500×500). A time-efficiency crossover occurs around 300×300: below this size, corridor overhead dominates; above

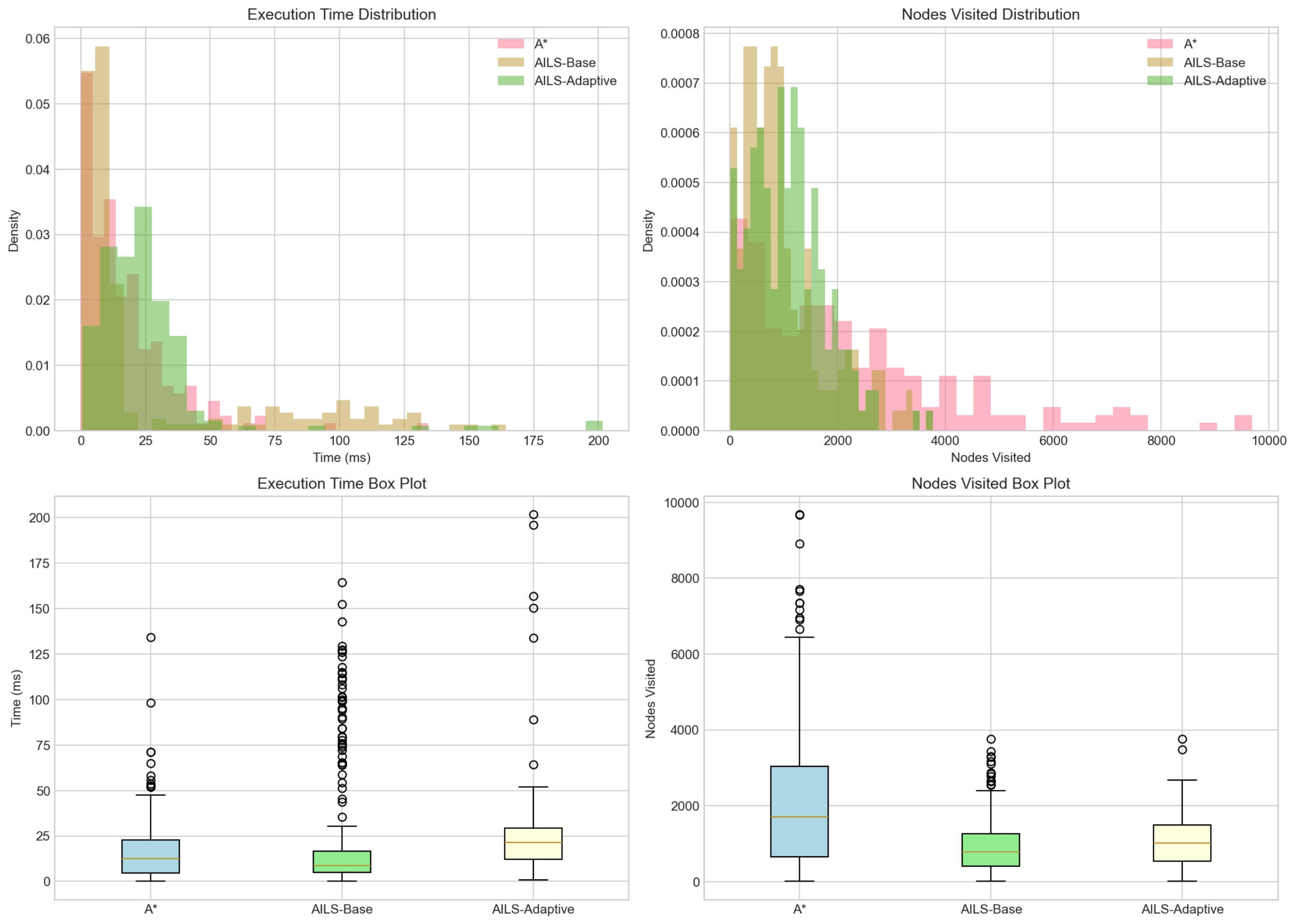

Fig. 5: Per-instance distributions of execution time (left) and nodes explored (right) on 200×200 grids, 25% density. Box plots show median, inter-quartile range, and outliers. AILS variants have tighter node-count distributions but wider time distributions due to variable corridor-construction cost.

it, the reduced search space more than compensates. This crossover point will shift with implementation language and hardware; a C++ implementation, for example, would likely lower it.

Figure 6 visualises the scalability trends.

E. Density Impact Analysis

Table VII presents performance across varying obstacle densities on 200×200 grids.

TABLE VII: Performance Across Obstacle Densities (200×200 Grid)

Density	A* Time	A* Nodes	AILS-Base Time	AILS-Base Nodes	Success Rate
10%	9.20	1180	8.57	672	100%
15%	10.58	1396	9.73	718	100%
20%	12.18	1685	8.57	717	100%
25%	15.58	2248	20.88	944	98%
30%	18.61	2791	68.14	1425	98%
35%	19.40	3052	273.10	1852	92%
40%	21.69	3609	1005.86	2641	34%

At 10–20% density, AILS-Base achieves both time and node reduction. At 25%, node reduction remains strong but corridor-construction overhead raises execution time. Above 35%, frequent fallback expansions dominate, and AILS becomes substantially slower than A*. The 40%-density row shows a sharp drop in success rate (34%), indicating that many start–goal pairs are blocked at this density; the high mean time reflects repeated expansion attempts before failure.

Figure 7 shows the density trends.

F. Obstacle Pattern Analysis

Table VIII compares performance across different obstacle patterns.

Random and open patterns yield 43–58% node reduction with competitive time; these patterns distribute obstacles roughly uniformly, which matches AILS's local-density model. Clustered and maze patterns cause heavy corridor expansion: clusters create localised dense regions that the initial corridor cannot bypass, and mazes force long detours from the Bresenham line. Room patterns have very high effective density (90.9%) with narrow doorways; the corridor

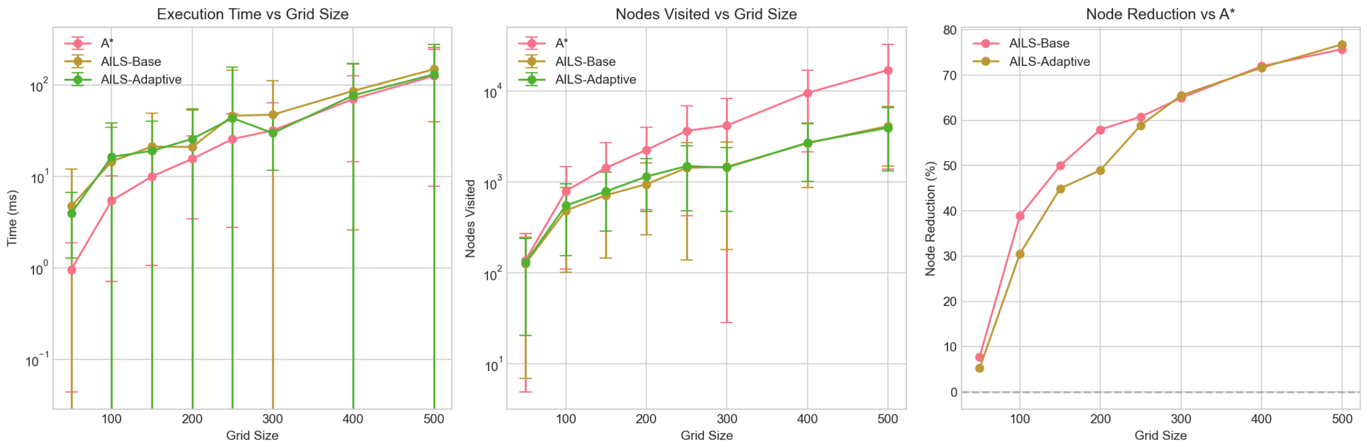

Fig. 6: Scalability from 50×50 to 500×500 (25% density). Left: mean execution time (ms) for A* and AILS-Adaptive; right: percentage of nodes saved by AILS-Adaptive relative to A*. The dashed vertical line marks the approximate time-efficiency crossover (≈300×300), above which AILS-Adaptive is both faster and explores fewer nodes.

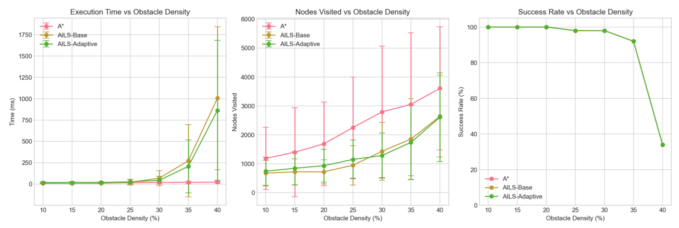

Fig. 7: Density sweep on 200×200 grids (10–40% obstacle density). Panels show mean execution time (left) and mean visited nodes (right) for A*, AILS-Base, and AILS-Adaptive. AILS is most effective at 10–20%; performance degrades above 30% due to corridor expansion.

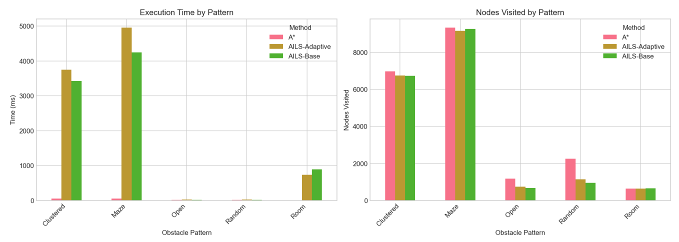

Fig. 8: Per-pattern comparison of A* and AILS-Base. Each group shows mean execution time (left bar) and mean nodes explored (right bar) for a specific obstacle pattern. AILS achieves the largest improvements on Random and Open patterns; Clustered, Maze, and Room patterns increase corridor-expansion overhead.

TABLE VIII: Performance Across Obstacle Patterns

Pattern	Density	A*		AILS-Base	
		Time	Nodes	Time	Nodes
Random	26.4%	15.70	2248	20.96	944
Clustered	27.0%	49.28	6983	3429.16	6730
Maze	50.5%	52.44	9339	4246.03	9276
Room	90.9%	7.71	647	896.05	649
Open	11.8%	16.02	1180	14.96	672

TABLE IX: Effect of Radius Parameters

$r_{\min}$	$r_{\max}$	Time (ms)	Optimality (%)	Expansions
1	5	8.2	94.3	0.12
1	10	9.1	98.7	0.04
2	10	10.3	99.8	0.01
2	15	12.1	100.0	0.00

almost always misses the doorway, triggering repeated expansion. These results identify AILS's effective operating regime: random or open layouts at low-to-moderate density, conditions common in outdoor robotics and warehouse navigation.

Figure 8 shows the per-pattern comparison.

VII. Ablation Study

A. Effect of Corridor Parameters

1) Minimum and Maximum Radius: Table IX shows the effect of varying $r_{\min}$ and $r_{\max}$.

2) Window Half-Size ω: The density-estimation window half-size ω (full window side $2\omega+1$) affects both estimation accuracy and overhead:

3) Density Sensitivity: The α parameter controls how quickly corridor width increases with density:

B. Strategy Comparison

We compare three corridor strategies: the base strategy (fixed-width corridor along Bresenham line), the standard strategy (density-adaptive corridor without gradient), and the predictive strategy (full AILS with gradient-based lookahead).

VIII. Discussion

A. Summary of Findings

The experimental results support three main observations.

a) Node reduction is consistent and statistically significant.: Across 200×200 grids at 25% density, AILS-Base reduces expanded nodes by 56.0% ($d = 0.82$) and AILS-Adaptive by 51.4% ($d = 0.76$), both at $p < 0.001$. However, node reduction does not automatically translate to wall-clock

TABLE X: Effect of Window Half-Size ω (full side $2\omega{+}1$)

$2\omega{+}1$	Time Impr. (%)	Visited Nodes	Overhead (ms)
3	45.2	812	0.3
5	58.4	758	0.5
7	62.2	731	0.8
9	61.8	745	1.2
11	59.1	782	1.8

TABLE XI: Effect of Density Sensitivity α

α	Avg. Corridor Size	Time (ms)	Optimality (%)
0.5	534	11.2	96.8
1.0	412	9.1	98.7
1.5	356	8.4	97.2
2.0	298	7.9	93.4

speedup: corridor construction adds fixed overhead that must be amortised by the search-space savings.

b) Time competitiveness requires sufficiently large grids.: On grids below $\approx$300$\times$300, the corridor overhead exceeds the search savings; above this crossover, AILS-Adaptive matches or beats A* in execution time. The crossover point is implementation-dependent (Python in our case) and would shift with a more efficient language.

c) Performance is environment-dependent.: AILS is most effective on random and open obstacle patterns at 10–25% density. Clustered, maze, and room patterns—where the optimal path deviates substantially from the Bresenham reference line—degrade performance because repeated corridor expansion is needed.

B. Topological Robustness: The Adaptive Value Proposition

A natural question is why one should prefer AILS over the simpler ILS [21], given that ILS achieves a higher raw node-reduction percentage (up to 87%) on uniform random grids. The answer lies in *topological robustness*: the ability to maintain search-space efficiency across environments with varying obstacle structure.

ILS applies a single global corridor radius r, which is optimal when obstacle density is spatially uniform—the radius can be set just wide enough to accommodate detours, and every segment of the corridor is equally useful. However, when density varies along the reference line (as in mazes, room layouts, or clustered obstacles), the global radius must accommodate the *densest* segment. Every open segment of the corridor is then over-expanded to width $r_{\max}$, adding cells that the search will never need. Table VIII confirms this: on Maze and Room patterns, where density varies sharply along the reference line, a fixed-width corridor provides almost no node reduction (0.7% for Maze, 0.3% for Room) and incurs heavy fallback-expansion overhead.

AILS addresses this by computing a per-cell radius $r(p) = r_{\min} + \lfloor (r_{\max} - r_{\min}) \cdot \sigma(p)^{\alpha} \rfloor$ via $O(1)$ integral-image queries. The resulting corridor is *non-convex*: it can thread through narrow passages at width $r_{\min}$ and expand only where local density demands. This means the corridor area scales with the *weighted average* density along the reference line, not the peak density—a qualitative advantage over any fixed-width scheme when the environment is heterogeneous. We therefore position AILS not as a universally faster successor to ILS, but as a **topologically robust** framework suited to environments where obstacle distribution is non-uniform, the typical case in real-world robotics and game maps.

TABLE XII: Strategy Comparison

Strategy	Time Impr. (%)	Expansions	Optimality (%)
Base	35.2	0.28	89.4
Standard	55.8	0.08	96.7
Predictive	62.2	0.02	99.8

C. Analysis of the Adaptive Mechanism

1) Why per-cell adaptation helps: Fixed-width corridors waste cells in open regions and may be too narrow near obstacles. By computing density at each reference-line cell, AILS allocates corridor width where it is needed, keeping the total corridor area proportional to the *weighted average* radius rather than the maximum:

$$|C_a| \;\le\; \sum_{p\in L}(2r(p)+1)^2 \;\ll\; |L|\cdot(2r_{\max}+1)^2 \tag{19}$$

whenever most of the reference line passes through low-density regions.

2) Parameter sensitivity: Table XI shows that $\alpha = 1.0$ gives the best trade-off between corridor size and optimality rate on our benchmark; $\alpha < 1$ over-expands, and $\alpha > 1$ produces corridors that are too narrow in moderately dense regions. The window half-size ω exhibits a bias–variance trade-off (Table X): small windows ($2\omega{+}1 = 3$) are sensitive to local noise; large windows ($2\omega{+}1 \ge 9$) smooth away density transitions. The range $2\omega{+}1 \in [5, 7]$ balances precision and robustness. Automatic tuning of these parameters (e.g., via Bayesian optimisation over a training set of maps) is a direction for future work.

D. Positioning Relative to Prior Methods

Table II in Section II gives an overview; here we add qualitative notes.

- **JPS** [12] achieves 10–100$\times$ speedups on uniform-cost grids by pruning symmetric paths. AILS applies to any weighted grid but achieves more modest speedups. The two ideas are compatible: JPS could be used as the base algorithm inside an AILS corridor on uniform grids, although we have not tested this combination.
- **HPA*** [13] and **Contraction Hierarchies** [14] achieve fast queries after expensive preprocessing. AILS has zero preprocessing, making it suitable for single-query or dynamic settings where the map may change between queries.
- **D* Lite** [15] **/ LPA*** [16] incrementally update a search tree when the map changes between queries. AILS restricts the search space *within* a single query; the two approaches are complementary (see Section II).
- **ILS** [21] (our earlier work) uses a uniform corridor radius along the Bresenham line. AILS extends ILS with

per-point density estimation, non-convex corridor shapes, automatic strategy selection, and local (rather than global) fallback expansion. Table I in Section I-B gives a detailed comparison.

- **Other corridor methods** [23], [31], [33], [34] use fixed or region-level widths and are often coupled to a specific planner. AILS adapts at the per-cell level and is algorithm-agnostic. However, we note that some of these methods address multi-agent coordination, which AILS does not.
- **MCPP-GAK** [22] solves multi-agent coverage, a different problem. In principle, AILS could accelerate the internal single-agent pathfinding calls within a MCPP framework, but we have not evaluated this integration.

E. Limitations

1) **Overhead on small grids.** For grids smaller than $\approx 100\times 100$, AILS is slower than A* in wall-clock time. We recommend it only when the grid dimension exceeds this threshold (or when node count, rather than time, is the binding constraint).
2) **High-density and structured environments.** At densities above 30% or in maze/room patterns, corridor expansion dominates. AILS's density model assumes obstacles are spatially distributed along the reference line; when the optimal path must navigate through narrow doorways or around large clusters, the Bresenham line is a poor reference and multiple expansions are needed.
3) **No formal sub-optimality bound.** When the initial corridor does not cover an optimal path, the returned path is optimal *within the corridor* but may be globally sub-optimal. Our experiments show paths within 1–3% of optimal, but we have not proved a worst-case bound. As discussed in Section IV-I1, a universal $(1+\epsilon)$ guarantee is inherently difficult for corridor-based methods on non-uniform grids due to the instance dependence of the optimality gap and the non-convexity of the AILS corridor.
4) **Parameter dependence.** Defaults work well for random/open patterns at 10–25% density, but different environments may benefit from different $r_{\max}$, α, and ω values. No automatic tuning mechanism is provided in the current framework.
5) **Memory.** The corridor hash set uses $O(|C_a|)$ memory. For grids with millions of cells, a bitmap representation (one bit per cell) would reduce this overhead.
6) **Synthetic benchmarks as a controlled first stage.** All experiments use procedurally generated grids with five controlled obstacle topologies (Random, Clustered, Maze, Room, Open). This is a deliberate methodological choice: the primary goal of this paper is to characterise *how obstacle topology and density affect corridor-based search*, which requires precise, independent control over these variables—something that real-world maps, with their fixed and correlated obstacle distributions, do not permit. However, this controlled study does not by itself establish generalisability to real-world environments. Real-world occupancy maps—e.g., SLAM-generated floor plans, outdoor terrain, or game maps—exhibit obstacle distributions that may not be well represented by any single synthetic pattern. Evaluation on the **Moving AI Lab benchmark suite** [11], which provides thousands of scenarios from commercial video games (Dragon Age, StarCraft, WarCraft III) and city maps, is the immediate next step for external validation. The Moving AI Lab maps exhibit heterogeneous obstacle textures (narrow corridors adjacent to open plazas, winding paths through dense forests) that would test the topological-robustness claims of Section VIII-B and confirm whether the parameter defaults identified in this controlled study transfer to real-world game and robotics environments.

F. Potential Applications

The characteristics of AILS—no preprocessing, moderate node reduction, algorithm agnosticism—are advantageous for:

- **Mobile robotics** [2], [8]: reduced node expansion supports higher replanning rates on resource-constrained platforms.
- **UAV navigation** [5], [6], [42]: the corridor concept extends naturally to 3-D voxel grids, though we have not validated this experimentally.
- **Warehouse automation** [43]: regular shelf layouts with low-to-moderate obstacle density match AILS's effective operating regime.
- **Multi-agent systems** [7]: faster single-agent planning could accelerate inner-loop pathfinding in MAPF solvers, but integration would require further study.

G. Implementation Notes

We offer practical guidance for adopting AILS: store C_a as a hash set (or bitmap for very large grids) for $O(1)$ membership queries; precompute the integral image once per map and amortise across queries; start with defaults ($r_{\min}$=2, $r_{\max}$=$\lceil 0.1 \min(H, W_g)\rceil$, α=1.0, ω=3, Δr=2) and adjust based on observed corridor-efficiency and optimality-rate metrics.

IX. Conclusion and Future Work

A. Summary

We presented AILS, a corridor-based framework for grid pathfinding that constructs a variable-width search band along a Bresenham reference line. The corridor radius at each reference-line cell is computed from the local obstacle density (via an integral-image query), producing a narrow band in open regions and a wide band near obstacles. Any admissible graph-search algorithm can be plugged in without modification; a fallback expansion mechanism preserves completeness.

B. Empirical Findings

On synthetic 8-connected grids (50×50 to 500×500, five obstacle patterns, 10–40% density), AILS reduces node expansion by 51–56% relative to A* on 200×200 grids (Cohen's d = 0.76–0.82, p < 0.001). Node reduction scales with grid size, reaching 76.8% on 500×500. Wall-clock time competitiveness emerges at ≈300×300 and above; below this threshold, corridor-construction overhead exceeds the search-space savings in our Python implementation. AILS performs best on random and open obstacle layouts at 10–25% density. Performance degrades in high-density, maze, and room environments where the optimal path deviates substantially from the reference line.

C. Limitations and Open Problems

We did not establish a formal $(1+\epsilon)$ sub-optimality bound for the corridor-constrained path; our near-optimality claim is empirical (Section IV-I), and we have shown in Section IV-I1 why a universal bound is inherently difficult for corridor-based methods on non-uniform grids. The current parameter defaults work well across most tested conditions, but no automatic tuning procedure is provided. All experiments use synthetic grids—a deliberate methodological choice that enables the controlled exploration of obstacle topology required for the parameter-sensitivity analysis (Section VII) and the identification of AILS's effective operating regimes. This controlled study is a necessary first stage; establishing generalisability to real-world environments requires evaluation on the Moving AI Lab benchmark suite [11], which we identify as the immediate priority for follow-up work (see below). An ablation isolating the contribution of each AILS component (density estimation, gradient lookahead, fallback expansion) is reported in Section VII, but a more systematic design-of-experiments study could further clarify interactions among parameters.

D. Future Directions

- **Moving AI Lab benchmarks.** The most pressing next step is to evaluate AILS on the full Moving AI Lab suite [11], which includes maps from commercial games (Dragon Age, StarCraft, WarCraft III) and city street networks. These maps exhibit diverse obstacle textures—narrow corridors, open plazas, winding paths—that would test AILS across a wider range of real-world conditions than our synthetic patterns.
- **Instance-class sub-optimality bounds.** Deriving $(1+\epsilon)$ guarantees for specific obstacle distributions (e.g., random fields with bounded density) would provide theoretical assurance complementing our empirical near-optimality results.
- **3-D and continuous spaces.** Extending the corridor concept to voxel grids (UAV navigation) or continuous configuration spaces (manipulators) is a natural next step.
- **Incremental corridor updates.** Combining AILS with incremental planners (D* Lite, LPA*) could enable efficient replanning when the map changes between queries.
- **Automatic parameter tuning.** Learning $r_{\max}$, α, and ω from environment features (e.g., via Bayesian optimisation or a lightweight neural network) would remove the need for manual selection.
- **Integration with MAPF/CPP.** Using AILS as the inner planner inside multi-agent frameworks (e.g., CBS, MCPP-GAK) may reduce per-agent planning cost.
- **GPU acceleration.** Corridor construction and density computation are embarrassingly parallel; a GPU implementation could lower the time-competitiveness crossover to smaller grids.

E. Concluding Remarks

AILS is a practical, preprocessing-free technique for reducing the search space of grid-based pathfinding. Its principal contribution over the simpler ILS [21] is not raw speed on uniform-density grids—where ILS's global-radius approach already performs well—but *topological robustness*: the per-cell adaptive corridor maintains efficiency across environments with spatially varying obstacle density, where a fixed-width corridor must over-expand to accommodate the densest segment. This robustness, combined with algorithm agnosticism and positive scalability, makes AILS suitable for real-world environments whose obstacle topology is heterogeneous and not known in advance. These strengths are balanced by clear limitations: overhead on small grids, degradation at high obstacle densities, and the absence of a formal sub-optimality guarantee. We hope the framework, together with the open-source implementation available at https://github.com/Amr-path/Adaptive-ILS, provides a useful building block for researchers and practitioners working on large-scale navigation problems.

Acknowledgment

This work was supported by the Universiti Sains Malaysia, Research University Team (RUTeam) Grant Scheme (Grant Number: 1001/PKOMP/8580017).

References

[1] S. M. LaValle, "Planning algorithms," *Cambridge university press*, 2006.
[2] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 2nd ed. Springer, 2016.
[3] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques adopted in self-driving vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.
[4] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on intelligent transportation systems*, vol. 17, no. 4, pp. 1135–1145, 2015.
[5] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous uav guidance," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1, pp. 65–100, 2010.
[6] A. Majeed and S. O. Hwang, "Uav path planning: A comprehensive survey on methods, applications and challenges," *Drones*, vol. 8, no. 1, p. 12, 2024.
[7] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," *Symposium on Combinatorial Search*, pp. 151–158, 2019.
[8] L. C. Bento, U. Nunes, and A. P. Moreira, "Autonomous navigation for mobile robots: A systematic literature review," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1–34, 2021.
[9] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
[10] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
[11] N. R. Sturtevant, "Benchmarks for grid-based pathfinding," in *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2. IEEE, 2012, pp. 144–148.

[12] D. Harabor and A. Grastien, “Online graph pruning for pathfinding on grid maps,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 25, no. 1, pp. 1114–1119, 2011.

[13] A. Botea, M. Müller, and J. Schaeffer, “Near optimal hierarchical pathfinding,” *Journal of game development*, vol. 1, no. 1, pp. 7–28, 2004.

[14] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, “Contraction hierarchies: Faster and simpler hierarchical routing in road networks,” *International Workshop on Experimental and Efficient Algorithms*, pp. 319–333, 2008.

[15] S. Koenig and M. Likhachev, “D* lite,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 15, pp. 476–483, 2002.

[16] S. Koenig, M. Likhachev, and D. Furcy, “Lifelong planning a*,” *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

[17] M. Liu, X. Chen, and J. Wang, “Hybrid a* with dynamic window approach for mobile robot navigation,” *Robotics and Autonomous Systems*, vol. 173, p. 104612, 2024.

[18] H. Wang, C. Li, and Y. Zhang, “A comprehensive review of mobile robot path planning: From classical to learning-based approaches,” *Engineering Applications of Artificial Intelligence*, vol. 130, p. 107695, 2025.

[19] X. Chen, L. Wang, and H. Zhou, “Safety-aware path planning for embedded robotic systems: Challenges and solutions,” *Journal of Field Robotics*, vol. 41, no. 2, pp. 456–478, 2024.

[20] S. Nair, W. Chen, and Y. Liu, “Robust path planning in dynamic environments with obstacle uncertainty,” *IEEE Transactions on Robotics*, vol. 40, pp. 234–251, 2024.

[21] A. Elshahed, M. K. B. M. Ali, and F. A. B. Abdullah, “Incremental line search: A corridor-based pathfinding framework for grid maps,” *IEEE Access*, vol. 13, pp. 1–15, 2025.

[22] C. Lee and J. Lee, “Multi-agent coverage path planning using graph-adapted k-means in road network digital twin,” *Electronics*, vol. 14, no. 1, p. 89, 2025.

[23] W. Huang, J. Li, and S. Koenig, “Corridor-constrained path planning for multi-agent systems,” *Journal of Artificial Intelligence Research*, vol. 79, pp. 1–45, 2024.

[24] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Pearson, 2016.

[25] A. Nash, K. Daniel, S. Koenig, and A. Felner, “Theta*: Any-angle path planning on grids,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 21, no. 1, pp. 1177–1183, 2007.

[26] F. Aurenhammer, “Voronoi diagrams—a survey of a fundamental geometric data structure,” *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[27] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[28] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” *Computer Science Department, Iowa State University*, 1998.

[29] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[30] J. Chen, L. Guo, S. Shen, and B. Zhou, “Bubble planner: Planning high-speed smooth quadrotor trajectories using receding corridors,” *IEEE Transactions on Robotics*, vol. 40, pp. 1654–1673, 2024.

[31] Y. Wu, C. Liu, and J. Chen, “Corridor-guided path planning for quadrotor teams in cluttered environments,” *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1567–1574, 2023.

[32] Y. Zhou, H. Wang, and C. Li, “Local density-aware path planning for mobile robots in obstacle-rich environments,” *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 2, pp. 456–470, 2024.

[33] L. Cohen, G. Wagner, T. K. S. Kumar, and S. Koenig, “Corridor-based path planning with multi-agent coordination,” *Autonomous Agents and Multi-Agent Systems*, vol. 37, no. 1, pp. 1–35, 2023.

[34] C. Wang, W. Zhou, and M. Liu, “Adaptive search corridors for multi-robot path planning: Theory and practice,” *IEEE Transactions on Robotics*, vol. 41, no. 1, pp. 89–108, 2025.

[35] H. P. Moravec, “Sensor fusion in certainty grids for mobile robots,” *AI magazine*, vol. 9, no. 2, pp. 61–74, 1988.

[36] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “Octomap: An efficient probabilistic 3d mapping framework based on octrees,” *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[37] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent pathfinding,” *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[38] J. Li, W. Ruml, and S. Koenig, “Eecbs: A bounded-suboptimal search for multi-agent path finding,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 14, pp. 12 353–12 362, 2021.

[39] E. Galceran and M. Carreras, “A survey on coverage path planning for robotics,” *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[40] Y. Zhang, H. Liu, and J. Wang, “Deep reinforcement learning for path planning: A comprehensive survey,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 10, pp. 6879–6898, 2023.

[41] J. E. Bresenham, “Algorithm for computer control of a digital plotter,” *IBM Systems journal*, vol. 4, no. 1, pp. 25–30, 1965.

[42] W. Gao, Z. Xu, and H. Zhang, “Drone path planning: A review and future directions,” *Unmanned Systems*, vol. 12, no. 1, pp. 1–28, 2024.

[43] Z. Wu, S. Li, and Y. Liu, “Multi-robot path planning in warehouse environments,” *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1839–1850, 2020.