PhD Viva Voce Preparation Guide

Dynamic Pathfinding for Autonomous Systems:
An Efficient Grid-Map Framework for Classical Search

Elshahed Amr Moustafa Mohamed Aly Elsayed

Doctor of Philosophy | Universiti Sains Malaysia | 2026

Comprehensive Q&A guide with 60 possible viva questions and detailed answers

SECTION 1: OPENING & GENERAL QUESTIONS

Q1: Can you summarize your thesis in 3-5 minutes?

My thesis addresses the problem of making pathfinding faster for autonomous systems in biosecurity-sensitive environments such as agricultural facilities, healthcare settings, and ports.

The core idea is simple: instead of letting a search algorithm explore the entire grid map, I confine it to a narrow "corridor" drawn along the straight line between start and goal. This is the **Incremental Line Search (ILS)** framework.

The flow is:
1. Draw a Bresenham line from start to goal.
2. Build a corridor of fixed width around that line.
3. Run any classical search algorithm (A*, Dijkstra, BFS, DFS, or Best-First) inside that corridor only.
4. If no path is found, widen the corridor and retry.

Results on 6,000 synthetic 200x200 grids showed an average **87.31% reduction in execution time** and **71.44% reduction in node expansions**. Path optimality was preserved for optimal algorithms.

My second contribution, **Adaptive ILS (AILS)**, uses integral images to estimate local obstacle density and adjusts the corridor width at each point -- narrow in open areas, wider near obstacles. AILS achieved 51-56% node reduction on 200x200 grids, scaling to **76.8% on 500x500 grids**. The Predictive strategy achieved **99.8% optimality**.

The two methods are complementary: ILS excels on uniform-density environments, while AILS provides robustness across heterogeneous obstacle layouts.

Q2: What motivated you to choose this research topic?

Three things came together:

1. Real-world need: Autonomous systems in biosecurity settings -- drones surveying contamination zones, robots in quarantine areas, vehicles in ports -- need to compute paths quickly and safely. Standard algorithms like A* can be too slow on large or cluttered grids.

2. A gap in the literature: Jump Point Search achieves dramatic speedups but only works on uniform-cost grids. For risk-annotated grids (where each cell has a different cost reflecting danger level), there was no equivalent corridor-based approach.

3. Practical observation: In most navigation scenarios, the optimal path does not deviate far from the straight line between start and goal. This geometric insight suggested that confining search to a narrow band could work well in practice.

Q3: What is your original contribution to knowledge?

Contribution 1 -- Incremental Line Search (ILS): A general framework that wraps any classical search algorithm inside a corridor constraint. Unlike JPS, ILS works with any edge-cost model. It achieved 87.31% execution-time reduction and 71.44% node reduction on average. For non-optimal algorithms like DFS, ILS also dramatically improved path quality (up to 93.74% path-length reduction).

Contribution 2 -- Adaptive ILS (AILS): Uses integral images for O(1) local density estimation and constructs a variable-width corridor. Three strategies (Base, Standard, Predictive) are selected automatically. The Predictive strategy achieved 62.2% time improvement with 99.8% optimality.

Q4: Why is this research significant?

1. Computational efficiency: ILS and AILS enable real-time pathfinding on large grids using commodity hardware, without GPU acceleration.

2. Generality: Unlike JPS (uniform costs only) or hierarchical methods (require preprocessing), ILS/AILS work with any search algorithm and any cost model, with no static preprocessing.

3. Broader applicability: While biosecurity is the motivating context, the techniques apply to warehouse automation, agricultural robots, search-and-rescue, and any grid-based planning scenario.

SECTION 2: RESEARCH DESIGN & METHODOLOGY

Q5: Why did you use synthetic grids instead of real-world benchmarks like Moving AI?

Synthetic grids were a deliberate methodological choice:

1. Precise control: Synthetic grids let me control exactly two variables -- obstacle density and topology -- while holding everything else constant.

2. Statistical power: I generated 6,000 maps for ILS (2,000 per density) and hundreds of configurations for AILS.

3. Five controlled topologies: Random, Clustered, Maze, Room, and Open patterns cover a wide range of structural characteristics.

4. Real-world validation: I did test on a satellite-derived grid (DS4) to show ILS generalizes beyond synthetic conditions.

5. Moving AI benchmarks are identified as the immediate next step for external validation.

Q6: Why did you choose the Bresenham line as the corridor axis?

1. It is the best discrete approximation of the straight line using only integer arithmetic.

2. It runs in O(L) time -- essentially free compared to the search itself.

3. In most practical navigation scenarios (open fields, warehouses, agricultural rows), the optimal path stays close to the straight line.

The limitation is that when the optimal path deviates significantly (maze/room patterns), the Bresenham reference becomes a poor approximation. This is acknowledged as a limitation.

Q7: How do you justify using five different algorithms?

1. Generality claim: By showing ILS works across optimal (A*, Dijkstra, BFS) and non-optimal (DFS, Best-First) algorithms, I demonstrate corridor restriction is a general-purpose wrapper.

2. Different insights: Optimal algorithms showed ILS preserves optimality. Non-optimal algorithms revealed an unexpected bonus -- DFS path length dropped by up to 93.74%.

3. Practical relevance: In resource-constrained embedded systems, simpler algorithms like BFS or DFS may be preferred due to lower memory requirements.

Q8: Explain the preprocessing pipeline.

Nine standardized steps:
Step 1: Image acquisition (or procedural generation)
Step 2: Greyscale conversion
Step 3: Binary thresholding (threshold = 128)
Step 4: Grid construction (pixels to vertices, edges based on connectivity)
Step 5: Obstacle density verification (within 1% of target)
Step 6: Start and goal assignment
Step 7: Reachability check (BFS)

Step 8: Integral image computation (AILS only, O(|V|))
Step 9: Output preprocessed grid

Each step has a clear purpose. The pipeline ensures all grids undergo identical processing.

Q9: How does the integral image enable O(1) density queries?

An integral image (summed-area table) stores cumulative sums. For each cell (x,y), I(x,y) = sum of all obstacle values in the rectangle from (0,0) to (x,y).

To find the obstacle count in any rectangular window:
count = I(x2,y2) - I(x1-1,y2) - I(x2,y1-1) + I(x1-1,y1-1)

This takes exactly 4 lookups and 3 arithmetic operations -- O(1) regardless of window size. Building the integral image takes O(|V|) time (one pass). After that, every density query is O(1).

Q10: Why three corridor strategies? Why not always use Predictive?

Base (fixed-width): Used when Bresenham line is obstacle-free. No density computation needed -- cheapest.

Standard (density-adaptive): Used when obstacles exist but density changes gradually.

Predictive (gradient-enhanced): Used when density changes rapidly. Widens corridor BEFORE dense regions.

Why not always Predictive? It adds gradient computation overhead. On obstacle-free lines, Base is sufficient and faster. The automatic selection ensures each query gets the cheapest sufficient strategy.

Q11: What is the time complexity of ILS and AILS?

ILS: Bresenham: O(L). Corridor: O(L*w). Search: O(|C|*log|C|) for A*. Worst case (full expansion): same as unconstrained O(N log N). Best case: O(L*w*log(L*w)) -- much smaller.

AILS: Integral image: O(|V|). Density queries: O(1) each, O(L) total. Corridor: O(sum of r(p)^2). Search: O(|C_a|*log|C_a|). Fallback: O(boundary*delta_r) per expansion.

Key insight: |C| << |V| in practice, so effective complexity is much lower than full-grid search.

Q12: Why paired t-tests and Cohen's d? Are these appropriate?

Paired t-tests: Each map was run with both standard and corridor-based algorithms. Pairing removes between-map variance. Normality verified with Shapiro-Wilk (W > 0.97, p > 0.10).

Cohen's d: With 2,000 maps per density, almost any tiny difference becomes statistically significant. Cohen's d tells whether the difference MATTERS:
- AILS-Base vs A*: d = 0.82 (large effect)
- AILS-Adaptive vs A*: d = 0.76 (medium-to-large effect)

For multi-group comparisons, one-way ANOVA with Tukey HSD correction was used.

SECTION 3: ILS-SPECIFIC QUESTIONS

Q13: How does ILS preserve path optimality?

Within the corridor, A*, Dijkstra, and BFS retain all original guarantees. The only modification is filtering out cells outside the corridor. Within the corridor, algorithms work exactly as normal.

The path returned is **optimal within the corridor**. If the corridor contains the globally optimal path (which it does in most cases at 10-25% density), the result is also globally optimal.

BFS and Dijkstra returned identical discrete path costs. A* with line-of-sight post-processing produced shorter Euclidean paths (69.54-86.37% improvement) while discrete optimality was preserved.

The incremental expansion provides a safety net: if the initial corridor misses the optimal path, widening will eventually include it.

Q14: Why does ILS improve DFS path quality so dramatically (up to 93.74%)?

Unconstrained DFS explores depth-first -- it can chase a single branch all the way to a distant corner before backtracking. The resulting path can be absurdly long.

The corridor completely changes DFS behavior. Instead of 40,000 cells to wander through (200x200 grid), DFS is funneled into 2,000-5,000 corridor cells. The worst-case path within that corridor is inherently much shorter.

This was an unexpected but valuable finding -- ILS acts as an **implicit quality guide** for non-optimal algorithms.

Q15: What happens when the initial corridor doesn't contain a valid path?

ILS fallback: Width incremented by delta_w, corridor rebuilt, search restarted. Continues until path found or corridor = full grid.

AILS fallback: More efficient local expansion. BFS from corridor boundary adds cells within Chebyshev distance delta_r. Only boundary expanded, not entire corridor.

This ensures **completeness**: if a path exists, it will be found. The cost of fallback is the main reason performance degrades on high-density environments.

Q16: How did you choose the initial corridor width?

ILS: w_0 = floor(gamma * min(H,W)). Proportional sizing ensures corridor scales with grid size.

AILS: r_min=2 (default), r_max=ceil(0.1*min(H,W)). The ablation study confirmed:
- (r_min=2, r_max=ceil(0.1*min(H,W))) achieved 99.8% optimality with minimal overhead
- Smaller radii: faster but lower optimality (94.3%)
- Larger radii: perfect optimality but slower

Q17: The ILS results exceeded your hypothesized 40-70% reduction. Why?

The hypothesis (RH1) predicted 40-70% reductions. Actual: 87.31% (time) and 71.44% (nodes).

Conservative prediction was based on literature for corridor methods. Actual results exceeded because:
1. The corridor was effective at ALL tested densities (10-30%)

2. Best-First responded especially well (95.52% at 10%)
3. Bresenham line was a better approximation than anticipated

Having predictions exceeded is positive -- the hypothesis served its purpose of providing a testable prediction.

SECTION 4: AILS-SPECIFIC QUESTIONS

Q18: Explain the density-adaptive radius formula.

r(p) = r_min + floor((r_max - r_min) * sigma(p)^alpha)

- sigma(p): local obstacle density at point p (via integral image, O(1))
- r_min (default 2): minimum radius for obstacle-free regions
- r_max (default ceil(0.1*min(H,W))): maximum radius for fully blocked regions
- alpha (default 1.0): controls how aggressively radius responds to density

When sigma=0: r(p)=r_min (narrow). When sigma=1: r(p)=r_max (widest). alpha<1: wide early. alpha>1: narrow longer. Ablation showed alpha=1.0 optimal: 98.7% optimality.

Q19: Why does AILS have higher execution time than A* on grids smaller than 300x300?

AILS has fixed overhead: integral image O(|V|), per-point density queries, hash-set assembly, corridor membership checks.

On small grids, search is already fast (A* takes 0.95ms on 50x50). AILS overhead (3.94ms) exceeds total search time. Node savings (5.1%) too small to compensate.

At 300x300, crossover: AILS 6.3% faster (29.61ms vs 31.62ms) with 65.5% fewer nodes. On 500x500: 76.8% fewer nodes. In C++, crossover would occur at smaller grids.

Q20: Why does AILS fail on Maze, Room, and Clustered patterns?

Root cause: optimal path deviates significantly from the Bresenham reference line.

Maze (50.5% density): path must follow winding passages. Bresenham cuts through walls.
Room (90.9% density): path threads through narrow doorways that don't align with line. AILS 116x slower.
Clustered (27% density): large clusters force path around them. 70x slower.

Fundamental issue: AILS's corridor is anchored to a straight-line approximation. When the environment requires winding/detouring, this breaks down. Explicitly acknowledged as a limitation.

Q21: Explain the Predictive strategy and why it achieves 99.8% optimality.

Predictive adds density gradient: r(p) = r_min + floor((r_max - r_min) * (sigma(p) + beta*|grad sigma(p)|)^alpha)

When gradient is large, obstacle concentration is CHANGING rapidly. Predictive widens the corridor BEFORE reaching the dense region.

Most suboptimality comes from the corridor being too narrow when hitting a dense region. Predictive avoids this by preemptively widening, so the optimal path is already inside the corridor. The 0.2% non-optimal cases are instances where gradient was not a reliable predictor.

Q22: How does automatic strategy selection work?

Single scan of Bresenham line at initialization:
1. Compute sigma(p) and gradient for all p on the line
2. If sigma(p)=0 for ALL points --> Base (cheapest)

3. Else if max|gradient| < 0.1 --> Standard
4. Else --> Predictive

Zero-cost in practice: density computations are already needed for corridor construction. Selection logic adds only a max-reduction over gradient values.

SECTION 5: RESULTS & STATISTICAL ANALYSIS

Q23: Walk us through the key numerical results.

ILS Results (DS1, 6000 maps, 200x200):
- Average time reduction: 87.31% | Node reduction: 71.44%
- Best single: Best-First at 10% density -- 95.52% time reduction
- DFS path improvement: up to 93.74% | All $p < 0.05$

AILS Results (DS2/DS3):
- Node reduction 200x200: 51-56% ($d=0.76$-0.82, $p<0.001$)
- Node reduction 500x500: 76.8% | Crossover: ~300x300
- Predictive: 62.2% time improvement, 99.8% optimality

Ablation: Optimal defaults: r_min=2, r_max=ceil(0.1*min(H,W)), alpha=1.0, omega=3

Q24: Why do improvements decrease as obstacle density increases?

At higher densities:
1. More corridor expansions triggered -- dense obstacles block paths within initial corridor
2. Corridor fraction of grid increases
3. Paths deviate more from straight line

A* time improvement: 94.81% at 10% --> 82.77% at 30%. Even at 30%, improvements stayed above 80% for A*, DFS, and Best-First. Best at 10-25% density -- common in outdoor robotics and warehouses.

Q25: The ILS and AILS experiments used different hardware. How can you compare them?

ILS: Apple M1 MacBook Air (8GB). AILS: Intel i7-12700K (64GB DDR5). Absolute times NOT directly comparable.

All cross-study comparisons use RELATIVE, hardware-independent metrics:
- Percentage improvement (relative to baseline on SAME hardware)
- Node reduction (completely hardware-independent)
- Corridor efficiency, optimality rate

This is explicitly acknowledged as a limitation. Node count comparisons are always valid.

Q26: Why didn't you test on grids larger than 500x500?

1. Trend was clear: 5.1% (50x50) to 76.8% (500x500) -- consistent upward trend
2. Crossover already captured at 300x300
3. Python overhead on very large grids would obscure algorithmic benefits
4. 200x200 to 500x500 covers many real-world scenarios

C++ reimplementation for larger-scale testing identified as future work.

Q27: No formal sub-optimality bound -- isn't that a significant weakness?

Recognized limitation, not fatal:

Why no bound: Optimality gap is instance-dependent. AILS corridor is non-convex. Worst-case gives vacuous bound.

Why not fatal:
1. Empirically 99.8% optimal (Predictive), paths within 1-3% when not exact
2. Fallback ensures full-grid search if needed
3. Weighted A* also lacks tight practical bounds
4. Future work: instance-specific or probabilistic bounds

SECTION 6: LITERATURE & THEORETICAL QUESTIONS

Q28: How does ILS compare to Jump Point Search (JPS)?

JPS: Exploits path symmetry. Prunes intermediate nodes. 10-100x speedup. RESTRICTED to uniform-cost grids. Modifies A* internal logic.

ILS: Exploits geometric proximity to straight line. Restricts entire search to corridor. Any cost model. Any algorithm as wrapper.

The two are compatible: JPS could serve as base algorithm inside ILS corridor on uniform grids. ILS fills the gap JPS leaves: risk-annotated grids where costs vary.

Q29: How does your work relate to D* Lite and LPA*?

D* Lite/LPA* are **incremental replanning** methods -- maintain search trees across episodes, repair solutions when environment changes. But full-grid memory, no scope constraint.

ILS/AILS are **search-space restriction** methods -- constrain WHERE to look within a single query.

Complementary: D* Lite inside AILS corridor = memory savings + incremental repair. ILS restricts spatial scope; D* Lite restricts temporal scope.

Q30: Why didn't you use learning-based approaches?

1. **Formal guarantees:** Classical search provides provable optimality. Neural heuristics may violate admissibility.
2. **Generalization:** Learned models may fail in novel settings. ILS/AILS work on any grid without training.
3. **Interpretability:** Classical algorithms are fully traceable -- important for safety certification.
4. **No training data needed:** Works out of the box.

A hybrid approach (learned corridor axis, classical search within) is an interesting future direction.

Q31: What is the relationship between your work and Theta*?

ILS borrows line-of-sight POST-PROCESSING from Theta*. After finding a path, if a vertex's grandparent has clear line of sight, the intermediate parent is removed.

Key difference: Theta* modifies A*'s internal expansion logic. ILS applies post-processing AFTER the standard search -- so it works with ANY algorithm (including DFS and BFS).

Observed improvements (69.54-86.37%) consistent with Theta* literature.

Q32: How does your work address the two research gaps?

Gap 1: No corridor-constrained search for risk-annotated grids. JPS needs uniform costs, subgoal methods need static preprocessing.
--> **ILS** fills this: any cost model, no preprocessing.

Gap 2: No adaptive search-scope mechanism for dynamic replanning. D* Lite maintains full-grid structures.
--> **AILS** fills this: dynamically adjusts corridor width based on local density.

Gap 1 --> O1 --> ILS. Gap 2 --> O2 --> AILS.

SECTION 7: BIOSECURITY APPLICATION QUESTIONS

Q33: How exactly does pathfinding relate to biosecurity?

1. **Physical navigation:** Drones/robots navigate agricultural facilities, quarantine areas, ports, healthcare settings where biosecurity risks exist.

2. **Risk-aware pathfinding:** Paths must minimize exposure to biological hazards. ILS/AILS support weighted cost models: cost(n,n') = dist(n,n') + lambda * r(n').

3. **Real-time response:** When contamination detected, autonomous systems need to replan quickly. 87% time reduction enables faster response.

4. **Port security:** Ports are critical biosecurity nodes -- entry points for biological threats.

Q34: Your experiments don't include actual biosecurity scenarios. How do you justify the framing?

1. The thesis develops GENERAL-PURPOSE techniques. Biosecurity provides MOTIVATION and CONTEXT.
2. Any biosecurity environment can be represented as an occupancy grid with risk annotations -- my algorithms work on this abstraction.
3. DS4 demonstrates real-world applicability on satellite-derived grid.
4. Density ranges tested (10-25%) match real biosecurity environments.
5. Biosecurity is the motivating USE CASE, not the experimental testbed. Full biosecurity evaluation is future work.

Q35: How would ILS/AILS handle dynamic risk maps?

1. **ILS:** Re-run from scratch with updated grid. Fast enough (87% reduction) for moderate update frequencies.
2. **AILS:** Re-compute integral image O(|V|), rebuild corridor. Naturally responds to new distribution.
3. **Combined with D* Lite:** Repair only affected plan portions within corridor.
4. Designed for "moderate, piecewise-static dynamics" -- realistic for biosecurity where updates come from lab tests (hours) or sensor readings (minutes).

SECTION 8: LIMITATIONS & FUTURE WORK

Q36: What are the main limitations of your work?

1. **Overhead on small grids:** AILS slower than A* below ~300x300.
2. **High-density & structured environments:** Degrades above 30% density; poor on maze/room/clustered.
3. **No formal sub-optimality bound:** 99.8% empirical but no worst-case guarantee.
4. **Synthetic benchmarks:** External validation on Moving AI needed.
5. **Different hardware:** Cross-study uses relative metrics only.
6. **Parameter dependence:** No automatic tuning mechanism.

Q37: If you had another year, what would you do?

1. Moving AI benchmark evaluation
2. C++ implementation (push crossover to smaller grids)
3. Formal sub-optimality analysis
4. Combine with D* Lite for dynamic replanning
5. Multi-agent pathfinding extension
6. Learned corridor axis prediction
7. Hardware deployment on actual robots/drones
8. Automatic parameter tuning

Q38: Would results be different in C++?

Algorithmic results (node counts, corridor sizes, optimality rates): **identical** -- language-independent.

Timing results: Much faster absolute times. Time-efficiency crossover would shift to smaller grids. Performance gap at small sizes would narrow.

This is why I report both timing metrics (implementation-dependent) and node counts (implementation-independent) -- node counts are the true measure of algorithmic efficiency.

Q39: How would you extend to 3D?

1. **3D Bresenham:** Generalizes naturally to 3D.
2. **3D corridor:** Becomes a tube. Density window becomes a cube. Integral image becomes 3D summed-volume table.
3. **Savings scale better:** Corridor volume = O(L*r^2) vs grid = O(N^3). Even larger fraction savings.
4. Relevant for UAV navigation, underwater vehicles, surgical robotics.

SECTION 9: CHALLENGING QUESTIONS

Q40: Isn't a corridor-based approach just a heuristic hack?

I would push back on "hack":
1. **Formal definition:** Corridor rigorously defined (Definition 3.1). Adaptive radius has clear mathematical structure.
2. **Completeness:** Fallback expansion guarantees path is found if one exists.
3. **Within-corridor optimality:** Provably optimal within the corridor for optimal algorithms.
4. **Geometric justification:** Bresenham line is the optimal discrete straight-line approximation.
5. **Systematic evaluation:** 5 algorithms, 3 densities, 8 grid sizes, 5 topologies, rigorous statistics.

Q41: Your method fails on maze/room patterns. Doesn't that severely limit applicability?

1. Target domain: Outdoor robotics, warehouses, agricultural fields, ports. These are open/random patterns at 10-25% density -- exactly where ILS/AILS excels. Mazes (50-90% density) are not typical.

2. Algorithm selection: A well-designed system characterizes the environment and selects appropriately. For mazes, use A*. For open environments, use ILS/AILS.

No single algorithm dominates all scenarios. The value is providing a superior tool for a practically important class of environments.

Q42: Why should we care about DFS with ILS?

1. **Generality demonstration:** Proves corridor is a general-purpose wrapper.
2. **Theoretical insight:** Revealed the corridor acts as an implicit quality guide -- would not emerge from testing only optimal algorithms.
3. **Resource-constrained systems:** DFS uses O(d) memory vs O(b^d) for BFS/A*. With ILS, DFS becomes viable on constrained platforms.
4. **Completeness:** Including all algorithms prevents cherry-picking.

Q43: The 87.31% time reduction seems too good. Could there be a bug?

Safeguards:
1. **Paired comparison:** Same map, same machine, same session.
2. **Consistent metrics:** Time (87.31%) aligned with nodes (71.44%).
3. **Statistical validation:** $p < 0.05$ across 2,000 maps per density.
4. **Expected trend:** Improvements decrease with density -- not arbitrary.
5. **Median of three runs.**
6. **Geometric reasoning:** At 10% density, corridor covers <10% of grid. Searching 10% of space naturally yields ~90% savings.

Q44: Why not compare against Contraction Hierarchies or HPA*?

1. **Correct baseline:** ILS/AILS modify how classical algorithms explore. Right comparison is "same algorithm with vs without corridor."
2. **Different categories:** CH/HPA* need expensive offline preprocessing. ILS/AILS are online with no preprocessing.
3. **Different use cases:** Static map + many queries --> preprocessing wins. Dynamic map + single queries --> ILS/AILS more suitable.
4. A fair comparison requires same language, same hardware -- identified as future work.

SECTION 10: PUBLICATION & CONTRIBUTION QUESTIONS

Q45: What papers have you published from this thesis?

Two papers:
1. Elshahed (2025) - ILS paper: Incremental Line Search framework on DS1/DS4. Covers Objective O1.
2. Elshahed (2025) - AILS paper: Adaptive ILS framework on DS2/DS3. Covers Objective O2.

Referenced as [Elshahed2025ILS] and [Elshahed2025AILS] throughout.

Q46: How does your work advance the field beyond incremental improvements?

1. **New paradigm:** Corridor-constrained search for non-uniform-cost grids did not exist. Occupies a new point in the design space.
2. **Algorithm-agnostic wrapper:** Novel idea that corridor restriction can wrap ANY search algorithm.
3. **Unexpected finding:** Path-quality improvement for non-optimal algorithms (DFS 93.74%) not anticipated by prior work.
4. **Practical impact:** Real-time pathfinding on commodity hardware for grid sizes that previously required more power.

SECTION 11: TECHNICAL DEEP-DIVE QUESTIONS

Q47: Explain Bresenham's algorithm and why integer arithmetic matters.

Computes discrete cells approximating a straight line using ONLY integer addition/subtraction.

Steps along major axis, maintains error term, adjusts minor axis when error exceeds 0.5.

Why integer: Faster than floating-point (especially embedded systems). Deterministic (no rounding errors). Output is discrete grid cells -- maps directly to grid representation. Runs in O(L) time.

Q48: What is Chebyshev distance and why use it?

d(a,b) = max(|a_x - b_x|, |a_y - b_y|)

Measures minimum king-moves on a chessboard. On 8-connected grids, minimum steps from a to b equals Chebyshev distance. Corridor boundary becomes a square band -- aligns with grid structure, efficient to compute. For 4-connected grids, Manhattan distance used instead.

Q49: Detail the ablation study results.

Radius (r_min, r_max): (1,5): 94.3% opt, 8.2ms. (2,10): 99.8% opt, 10.3ms. (2,15): 100% opt, 12.1ms. Default balances both.

Window omega: 3x3: 45.2%. 5x5: 58.4%. 7x7: 62.2% (best). 9x9: 61.8%. 11x11: 59.1%. Classic bias-variance.

Alpha: 0.5: 96.8%. 1.0: 98.7% (best). 1.5: 97.2%. 2.0: 93.4%.

Strategy: Base: 35.2%/89.4%. Standard: 55.8%/96.7%. Predictive: 62.2%/99.8% (winner).

Q50: What is Cohen's d and why is it important?

d = (mean1 - mean2) / pooled_std. Measures PRACTICAL significance.

|d|<0.2: negligible. 0.2-0.5: small. 0.5-0.8: medium. >=0.8: large.

With large samples, tiny differences become statistically significant (p<0.05). Cohen's d tells you if the difference MATTERS. AILS-Base vs A*: d=0.82 (large, practically meaningful).

SECTION 12: BROADER & PHILOSOPHICAL QUESTIONS

Q51: What have you learned from doing this PhD?

1. **Simple ideas can be powerful:** The corridor concept is conceptually simple but remarkably effective.
2. **Rigorous evaluation matters:** The difference between "seems to work" and "here is exactly how much" is what makes a contribution.
3. **Knowing limitations is valuable:** Characterizing failures is as important as showing successes.
4. **Complementary methods beat silver bullets:** ILS and AILS are additions to the toolbox, not replacements.

Q52: If you could start over, what would you do differently?

1. Start with C++ from the beginning
2. Include Moving AI benchmarks from the start
3. Unified hardware platform
4. Explore D* Lite combination earlier
5. More real-world data alongside synthetic

That said, the research trajectory made sense: ILS first (proof of concept), then AILS (extension), then analysis.

Q53: How would you explain your thesis to a non-technical person?

Imagine driving from home to the airport. You could explore every street in the city -- or you could focus on roads roughly in the airport's direction.

My thesis does the same for robots: draw a straight line from A to B, only look at a narrow band around it. This makes pathfinding ~87% faster. If the band is too narrow, it automatically widens.

My second innovation makes the band smart: wider near obstacles, narrow in open space.

SECTION 13: RAPID-FIRE QUESTIONS

Q54: What is the single most important result?

ILS achieving 87.31% average execution time reduction across five algorithms while preserving path optimality. This demonstrates the core contribution: corridor-based restriction is simple, general, and dramatically effective.

Q55: What contribution will still matter in 10 years?

The IDEA that corridor restriction can be applied as a general-purpose wrapper around ANY search algorithm. Specific algorithms evolve, but the principle of confining search to a geometrically motivated subspace is a lasting conceptual contribution.

Q56: If a referee disagrees with your biosecurity framing?

The biosecurity framing is MOTIVATIONAL, not experimental. The algorithms are general-purpose grid-based pathfinding methods. The technical contribution stands independently of the application context.

Q57: What is the failure mode?

REPEATED CORRIDOR EXPANSION on environments where optimal path deviates far from the Bresenham line. The algorithm never produces a WRONG answer, but it can be very slow (116x for room patterns). It always finds a valid path or correctly reports no path exists.

Q58: How do you ensure reproducibility?

1. Fixed random seeds (documented in logs)
2. Deterministic algorithms
3. Median of three timing runs
4. Explicit hardware/software specs
5. Standardized 9-step preprocessing pipeline

Q59: Practical deployment path?

Step 1: C++ reimplementation. Step 2: Moving AI validation. Step 3: ROS integration. Step 4: Gazebo/AirSim simulation. Step 5: Field trials on robots/drones. Step 6: D* Lite integration for dynamic replanning.

Q60: Summarize your thesis in one sentence.

I developed two corridor-based pathfinding techniques -- ILS and AILS -- that dramatically reduce computation for grid-based navigation by confining search to a narrow, optionally density-adaptive band around the straight line between start and goal, achieving up to 87% time reduction while preserving path quality.