# Deterministic Multi-Backend Maze Navigation and Planner Benchmarking for Reproducible Robotics Experiments

Cesar Contreras

***Abstract*—This paper presents a deterministic maze-navigation benchmark stack for reproducible robotics experiments across multiple simulation backends. The implementation unifies planner execution, path validation, benchmarking, and artifact generation in a single repository workflow built around deterministic seeds and explicit benchmark exports. In the executed snapshot, the benchmark evaluates 12 planners on 50 generated $15 \times 15$ mazes with complete trial-level reporting for success, runtime, path length, and expansions. All planners solve all trials; within this static regime, weighted A* has the lowest mean solve time (0.47 ms), while search-effort profiles show non-trivial runtime-expansion tradeoffs across methods. The pipeline also includes deterministic simulation regression checks with fixed screenshot expectations to support repeatability audits. The contribution is a repository-grounded, traceable benchmarking workflow for static grid-maze navigation; claims are intentionally scoped to this benchmark regime and do not assert general superiority in dynamic or kinodynamic navigation domains.**

***Index Terms*—mobile robot navigation, deterministic benchmarking, path planning, reproducibility, simulation**

## I. Introduction

Grid-based planning remains a practical foundation for robotic navigation in constrained environments, and recent surveys show that graph-search, heuristic, and hybrid variants remain competitive baselines in deployed systems [1], [2], [3], [4]. However, reproducible planner comparison is often weakened by inconsistent simulator availability, heterogeneous planner return formats, and evaluation scripts that do not verify geometric path validity. The implementation in this repository is designed to reduce these sources of experimental variance while preserving a fast iteration loop for algorithm development.

The current stack links one command-level entrypoint (`scripts/sim_runner.py`) to a deterministic execution pipeline in `robotics_maze/src/main.py`: argument parsing to `RunConfig`, per-episode seeding, module discovery for maze generator/planner/simulator, and standardized episode logging (`[EP ...]`, `[DONE]`). The runtime is intentionally robust to environment differences by selecting among PyBullet and MuJoCo backends, with deterministic fallback behavior when a backend is unavailable. This design goal is consistent with recent ROS2-centric and distributed robotics frameworks that prioritize portability across local and remote execution environments [5], [6], [7], [8]. This enables the same experiment script to execute on development laptops and CI-like headless settings without rewriting logic.

The project also includes a benchmark harness (`robotics_maze/src/benchmark.py`) that normalizes planner outputs, validates returned paths against the occupancy grid, and ranks planners with a reproducible policy. In the tracked benchmark snapshot (`robotics_maze/results/benchmark_summary.md`), 12 planners are evaluated on 50 backtracker mazes of size $15 \times 15$, with all planners reaching $100\%$ success and `r1_weighted_astar` ranked first by mean solve time and tie-break metrics.

This paper focuses on the implemented system and makes four concrete contributions:

- **Deterministic end-to-end runtime:** a seed-controlled pipeline for maze generation, planning, and episode simulation with stable CLI semantics and per-episode reproducibility.
- **Multi-backend simulation architecture:** an implemented adapter that executes waypoint plans through PyBullet or MuJoCo with explicit backend fallback and URDF fallback handling for robust runtime behavior.
- **Comparable, validated benchmark protocol:** a planner evaluation harness that (i) validates path feasibility, (ii) rotates planner execution order to reduce first-run bias, and (iii) computes comparable-set metrics over mazes solved by all planners.
- **Traceable research artifacts:** committed benchmark reports, deterministic test logs, and screenshot regression outputs that tie manuscript claims to repository evidence.

The remainder of the manuscript details the implemented architecture and protocol, then reports the current benchmark evidence and implications for future planner integration.

## II. Related Work

Recent work on mobile-robot navigation has expanded rapidly, but the space is best understood through five clusters: broad surveys, single-robot planning algorithms, multi-robot exploration, uncertainty-aware planning, and benchmarking/runtime infrastructure.

**Surveys and taxonomies.** Several recent reviews agree that practical navigation systems still combine structured search/planning with learned components rather than replacing one with the other [1], [2], [3], [9], [4], [10], [11]. Across these surveys, the recurring bottlenecks are not only planner

Cesar Contreras is an independent robotics researcher based in California, USA.

optimality, but also integration overhead, runtime stability, and weak cross-paper comparability.

**Single-robot path planning and mapless navigation.** Classical and hybrid planners continue to improve through heuristic design, global-local coupling, and search-policy adaptation [12], [13], [14], [15], [16], [17]. In parallel, deep reinforcement learning has produced strong mapless or partially mapless navigation policies, especially for dynamic obstacles and unknown environments [18], [19], [20], [21], [22], [23], [24], [25], [26], [27]. A consistent tradeoff is that learned planners can improve adaptability, but often require tighter evaluation controls to separate algorithmic gains from simulator- or setup-induced variance.

**Multi-robot exploration and coordination.** Recent methods exploit graph structure, hierarchical decomposition, and decentralized policies to scale exploration under communication limits [28], [29], [30], [31], [32], [33], [34]. These studies show strong progress on coordination efficiency, but they also highlight sensitivity to map generation, rollout ordering, and evaluation protocol assumptions that are frequently underdocumented.

**Planning under uncertainty and risk.** Chance-constrained, belief-space, and risk-aware planners now provide more principled handling of perception and dynamics uncertainty [35], [36], [37], [38], [39], [40], [41], [42]. This line of work improves safety and reliability claims, but it further increases the need for consistent interfaces and validation tooling when comparing heterogeneous planners on shared tasks.

**Benchmarks and runtime frameworks.** Benchmark suites and tooling ecosystems have matured, including Bench-MR, MotionBenchMaker, social-navigation benchmarks, and composable benchmark platforms [43], [44], [45], [46], [47]. Runtime frameworks in ROS2 and cloud/fog settings similarly emphasize modularity and deployment portability [5], [6], [48], [7], [8], [49]. Our system is positioned in this gap between algorithm work and benchmarking infrastructure: instead of proposing a new planner, it provides a deterministic, validated execution-and-evaluation pipeline that makes planner comparisons auditable and reproducible in constrained-maze settings.

## III. Method

### A. Implemented System Architecture

The implemented architecture is a modular runtime centered on `robotics_maze/src/main.py`, with four concrete layers:

1) **Orchestration layer** (`scripts/sim_runner.py`, `main.py`): parses command-line arguments into `RunConfig`, supports optional GUI setup overrides, and executes an episode loop.
2) **Maze and planning layer** (`maze.py`, `planners.py`, `alt_planners/`): generates deterministic mazes and computes paths using baseline and alternative planners.
3) **Simulation layer** (`sim.py`, `robot.py`, `geometry.py`): converts plans to waypoints, instantiates obstacles from maze walls, and executes motion in physics backends.
4) **Evaluation layer** (`benchmark.py`): runs planner trials, validates returned paths, ranks methods, and writes CSV/Markdown artifacts.

For episode $e \in \{1, \ldots, E\}$, `main.py` first computes an episode seed $s_e^{\text{main}} = s_0 + (e-1)$, where $s_0$ is the user-provided base seed. The current deterministic generator adapter then applies an additional episode offset, yielding the effective maze-generation seed $s_e^{\text{gen}} = s_e^{\text{main}} + e = s_0 + 2e - 1$. This manuscript reports the behavior as currently implemented in code.

### B. Maze Generation and Grid Conversion

The maze model (`Maze` dataclass) stores explicit horizontal and vertical wall arrays, start/goal cells, and generation metadata. Two algorithms are implemented: recursive backtracker and randomized Prim. After carving, generation is accepted only if (i) all $W \times H$ cells are reachable from start and (ii) a BFS shortest path exists between start and goal. This provides a deterministic solvable-maze contract before planning is invoked.

For planner compatibility, `benchmark.py` converts wall-based mazes into occupancy grids of size $(2H+1) \times (2W+1)$, where free corridors are explicitly opened between neighboring cells. Start and goal are transformed from cell coordinates into occupancy-grid indices and forced free.

### C. Planner Interface and Normalization

Baseline planners are registered by name in `planners.py` (A*, Dijkstra, BFS, and greedy best-first, plus aliases). Additional methods are loaded from `alt_planners/` through module-symbol mapping. To support heterogeneous planner APIs, `FunctionPlannerAdapter` and benchmark-side normalization convert outputs to a common payload with path plus metrics (e.g., expansions and runtime).

Planner outputs are accepted from multiple schemas (dictionary, tuple, or raw path sequence), then normalized by `_normalize_planner_output`. Path validity is enforced by `_validate_and_measure_path`: endpoints must match benchmark start/goal, all path cells must stay in bounds and collision free, and each segment is checked with Bresenham rasterization before success is credited.

### D. Simulation Backends and Controller Path

`MazeEpisodeSimulator` receives `(maze, plan)` and performs three steps:

1) extract or infer a path (`path`, `waypoints`, tuple payload, or maze fallback),
2) map the path to world-frame waypoints (including occupancy-grid to metric conversion),
3) execute via selected backend (`pybullet`, `mujoco`, or deterministic kinematic fallback).

Backend selection is explicit: `auto` prefers PyBullet when available, then MuJoCo; forced backend requests degrade

gracefully when dependencies are missing. For robot models, custom URDF loading is attempted first, then default fallback (`husky/husky.urdf`, then `r2d2.urdf`) to preserve run continuity. In PyBullet runs, a waypoint follower (`MobileRobotController`) applies heading-aware speed control, differential-drive wheel commands when available, and bounded command slew rates.

### E. Benchmark Protocol and Ranking

Let $\mathcal{P}$ denote the planner set and $\mathcal{M}$ the generated mazes. The benchmark executes every $p \in \mathcal{P}$ on every $m \in \mathcal{M}$, rotating planner order per maze to reduce warm-start/caching bias. Each trial records success, solve time (ms), path length, expansions, and error text. This structure aligns with recent benchmarking toolchains that separate dataset/task generation, planner execution normalization, and metric aggregation [43], [44], [46].

To avoid unfair timing comparisons when planners fail on different mazes, the method computes a shared-success subset:

$$\mathcal{M}_{\text{shared}} = \{m \in \mathcal{M} \mid \forall p \in \mathcal{P},\ p \text{ succeeds on } m\}.$$

Ranking follows the implemented lexicographic policy in `benchmark.py`:

$$\text{rank}(p) \sim \big( -\text{SR}_p,\ T_p^{\text{shared}},\ L_p^{\text{shared}},\ E_p,\ T_p \big),$$

where SR is success rate, $T^{\text{shared}}$ comparable solve time, $L^{\text{shared}}$ comparable path length, $E$ mean expansions, and $T$ mean solve time. The final $T$ term is used as a deterministic tie-break when earlier keys are equal and ensures the same ordering in generated summaries and manuscript tables.

This protocol outputs `benchmark_results.csv` and `benchmark_summary.md`, enabling traceable comparison and direct linkage between manuscript claims and generated artifacts.

## IV. Experimental Setup

This section documents (i) experiments already executed in the current repository snapshot and (ii) planned studies that are intentionally out of scope for the current results.

### A. Executed Experiments

**System and workflow context (executed).** Figure 1 summarizes the deterministic runtime and artifact path used for the reported experiments, from configuration parsing and planner execution through benchmark export. Manuscript-process coordination artifacts are tracked in `coordination/` files and are not treated as primary scientific evidence in the main experimental narrative.

**Planner benchmark (executed).** We evaluate planners with the repository benchmark harness (`robotics_maze/src/benchmark.py`). The harness generates mazes with fixed settings (50 mazes, $15 \times 15$ cells, `backtracker`, base seed 7, per-maze seed $=$ 7 + `maze_index`) and compares all discovered planners. In the committed benchmark artifacts (`robotics_maze/results/benchmark_summary.md` and `robotics_maze/results/benchmark_results.csv`), this corresponds to 12 planners: `astar`, `dijkstra`, `greedy_best_first`, `r1_weighted_astar`, `r2_bidirectional_astar`, `r3_theta_star`, `r4_idastar`, `r5_jump_point_search`, `r6_lpa_star`, `r7_beam_search`, `r8_fringe_search`, and `r9_bidirectional_bfs`. The baseline BFS implementation exists in the planner registry but is intentionally excluded from the benchmark harness default discovered set; all reported rankings and tables use this 12-planner execution set for consistency with committed artifacts.

Per trial, the harness records success, wall-clock solve time (ms), path length, and node expansions. Reported paths are validated against occupancy and bounds constraints, including segment-level collision checks; a trial is marked successful only if both planner output and path validation succeed. Solve time is measured using Python `time.perf_counter()` around each planner invocation and converted to milliseconds. To reduce first-run cache bias, planner execution order is rotated per maze. In the current snapshot, each planner-maze pair is executed once, so timing differences should be interpreted as descriptive. Aggregated ranking follows the executable lexicographic order in `benchmark.py`: success rate, comparable shared-success solve time, comparable shared-success path length, mean expansions, then overall mean solve time.

**Deterministic simulation regression (executed).** In addition to planner benchmarking, the repository executes deterministic simulation checks (`robotics_maze/testing/run_sim_tests.sh`) for three representative planners (`astar`, `weighted_astar`, `fringe_search`) over three episodes (`maze-size`=11, `seed`=42). The same test run invokes the deterministic screenshot pipeline (`robotics_maze/scripts/capture_regression_screensho`) which enforces six expected PNG outputs (three MuJoCo and three fallback renderer images). The latest committed test log reports all deterministic runs and screenshot checks as PASS.

Table I separates executed protocol components (used for all current claims) from planned, not-yet-executed studies.

### B. Planned Future Experiments (Not Yet Executed)

The following experiments are planned and are **not** included in the current result tables:

1) **Cross-algorithm robustness sweep:** expand benchmark runs across `backtracker` and `prim` generators, larger mazes, and broader seed sets to quantify ranking stability.
2) **Dynamic-environment stress tests:** introduce moving obstacles and replanning triggers to evaluate incremental planners under online disturbances.
3) **Post-2021 planner integration study:** add learned local heuristics, Hybrid-A*, and uncertainty-aware planning baselines from the repository shortlist, then rerun the same benchmark protocol for direct comparability.
4) **Backend sensitivity analysis:** run matched scenarios with explicit `pybullet` and `mujoco` settings to isolate

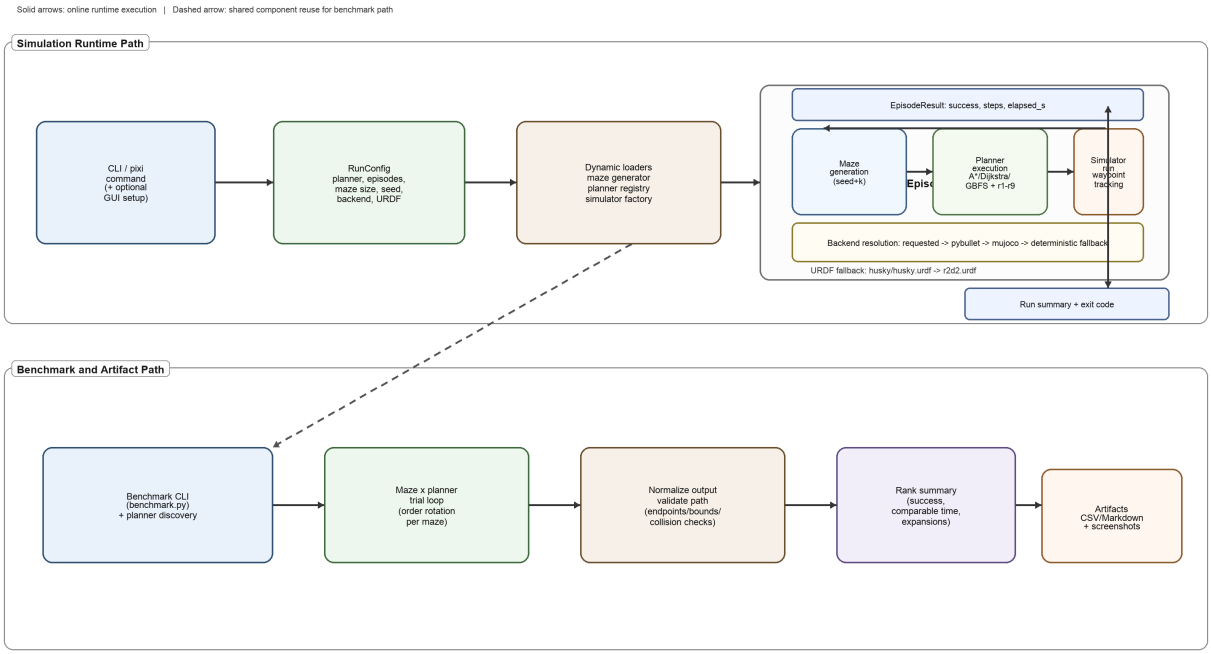


Fig. 1. Deterministic system pipeline used in the executed experiments, including planner benchmarking and artifact emission. The runtime path captures backend fallback behavior (`pybullet` → `mujoco` → deterministic fallback) and the benchmark output branch used for reproducible reporting.

simulator-dependent effects on runtime and path execution behavior.

No quantitative claims from these planned experiments are used in the present manuscript version.

## V. Results

Table II summarizes planner performance on 50 generated $15\times15$ backtracker mazes (seeds 7–56). All 12 planners solved all trials (600/600) with no runtime errors, so differences here primarily reflect computational efficiency rather than reachability.

Figures 2, 3, 4, and 5 provide complementary visual summaries of runtime, runtime uncertainty, search effort, and success rate using the same benchmark snapshot as Table II.

*a) Overall ranking and runtime spread.:* `r1_weighted_astar` is fastest in mean solve time (0.47 ms), followed by `r7_beam_search` (0.57 ms). A second tier—`greedy_best_first`, `r5_jump_point_search`, `r9_bidirectional_bfs`, `r8_fringe_search`, `astar`, and `dijkstra`—is tightly clustered between 0.64 and 0.74 ms (at most +0.27 ms versus the top row). At maze level, `r1_weighted_astar` is fastest on 49/50 mazes, but the best-vs-second-best margin is small (median 0.066 ms; 32/50 mazes within 0.1 ms), indicating limited practical separation among the fastest methods in this setup. This narrow spread is visible in Figures 2 and 3.

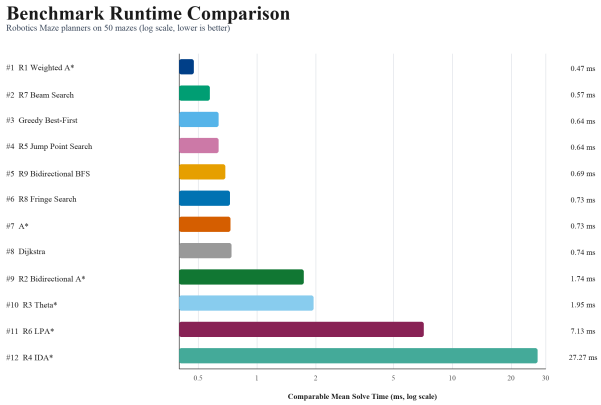


Fig. 2. Mean planner solve time (ms) on a logarithmic scale over 50 benchmark mazes. Lower values indicate faster planning.

*b) Inferential runtime comparison.:* To complement descriptive ranking with paired inference, each planner was compared against `r1_weighted_astar` on the same 50 mazes using exact two-sided paired sign tests with Holm correction (family-wise $\alpha = 0.05$). Effect sizes are reported as paired median runtime deltas ($\Delta$ = comparator − `r1_weighted_astar`, ms) with 95% bootstrap confidence intervals from 40,000 paired resamples. All 11 pairwise comparisons remain significant after Holm correction. For the closest comparator (`r7_beam_search`), the median

TABLE I
REPRODUCIBILITY PROTOCOL, EXPLICITLY SEPARATED INTO EXECUTED VS PLANNED STUDIES.

| Component | Executed protocol (current artifacts) | Planned protocol (not yet executed) |
|---|---|---|
| Entrypoints | `python robotics_maze/src/benchmark.py` for planner benchmarks; `bash robotics_maze/testing/run_sim_tests.sh` for deterministic simulation and screenshot regression. | Add CI jobs that run full benchmark and regression protocols on pinned hardware classes and publish signed artifacts per run. |
| Benchmark workload | 50 mazes, $15 \times 15$, `backtracker`, base seed 7, maze seed = 7+ maze index. Current snapshot evaluates 12 planners. | Scale to larger mazes and multi-generator sweeps (`backtracker` + `prim`) with expanded seed ranges to evaluate ranking stability. |
| Simulation regression workload | 3 planners (`astar`, `weighted_astar`, `fringe_search`), 3 episodes each, `maze-size`=11, `seed`=42, backend preference `auto`. | Matched paired runs with forced `pybullet` and `mujoco`; additional stress scenarios with dynamic obstacles and replanning triggers. |
| Primary metrics | Per trial: success, solve_time_ms, path_length, expansions, and error text on failure. Summary ranking prioritizes success, then comparable shared-success solve time, comparable shared-success path length, mean expansions, then overall mean solve time. | Add variance statistics (median, p95), per-backend deltas, and disturbance-recovery metrics for dynamic scenes. |
| Validity checks | Path-level validation enforces endpoint consistency, in-bounds traversal, and collision-free segments; benchmark marks success only when planner and validator agree. | Add robustness checks for near-collision margins, execution drift under dynamics, and failure-mode taxonomy by planner family. |
| Visual regression | Deterministic screenshot pipeline requires six fixed filenames (3 MuJoCo + 3 fallback). Latest committed run reports PASS on all checks. | Add pixel-diff thresholds and automated anomaly triage reports in CI for each benchmark commit. |

TABLE II
MAIN BENCHMARK RESULTS ON 50 GENERATED $15 \times 15$ BACKTRACKER MAZES. ROWS ARE RANKED BY SUCCESS RATE (DESCENDING), THEN COMPARABLE MEAN SOLVE TIME ON SHARED-SUCCESS MAZES, COMPARABLE MEAN PATH LENGTH ON SHARED-SUCCESS MAZES, MEAN EXPANSIONS, AND OVERALL MEAN SOLVE TIME (ASCENDING). TIME IS REPORTED AS MEAN $\pm$ STANDARD DEVIATION OVER MAZES; MEDIAN AND INTERQUARTILE RANGE (IQR) ARE INCLUDED TO EXPOSE SKEW. LOWER IS BETTER FOR TIME, PATH LENGTH, AND EXPANSIONS.

| Rank | Planner | Success | Time (ms) ↓ | Median [IQR] (ms) ↓ | Path Length ↓ | Expansions ↓ |
|---|---|---|---|---|---|---|
| 1 | R1 Weighted A* | 50/50 | 0.47 ± 0.29 | 0.39 [0.20, 0.71] | 142.72 | 187.16 |
| 2 | R7 Beam Search | 50/50 | 0.57 ± 0.33 | 0.51 [0.26, 0.89] | 142.72 | 198.36 |
| 3 | Greedy Best-First | 50/50 | 0.64 ± 0.38 | 0.50 [0.29, 0.98] | 142.72 | 171.96 |
| 4 | R5 Jump Point Search | 50/50 | 0.64 ± 0.37 | 0.55 [0.27, 0.97] | 142.72 | 57.26 |
| 5 | R9 Bidirectional BFS | 50/50 | 0.69 ± 0.39 | 0.60 [0.31, 1.02] | 142.72 | 201.52 |
| 6 | R8 Fringe Search | 50/50 | 0.73 ± 0.43 | 0.60 [0.32, 1.08] | 142.72 | 190.30 |
| 7 | A* | 50/50 | 0.73 ± 0.44 | 0.61 [0.31, 1.09] | 142.72 | 189.06 |
| 8 | Dijkstra | 50/50 | 0.74 ± 0.45 | 0.65 [0.31, 1.17] | 142.72 | 200.52 |
| 9 | R2 Bidirectional A* | 50/50 | 1.74 ± 0.97 | 1.81 [0.73, 2.64] | 142.72 | 468.02 |
| 10 | R3 Theta* | 50/50 | 1.95 ± 1.17 | 1.60 [0.81, 2.90] | 45.40† | 189.18 |
| 11 | R6 LPA* | 50/50 | 7.13 ± 2.92 | 6.41 [4.44, 9.60] | 142.72 | 295.10 |
| 12 | R4 IDA* | 50/50 | 27.27 ± 26.63 | 13.77 [3.85, 51.48] | 142.72 | 7061.34 |

†Theta* uses any-angle motion, so path-length values are not directly comparable to cardinal-grid planners.

paired delta is 0.095 ms (95% CI [0.057, 0.117]) with `r1_weighted_astar` faster on 50/50 mazes; this indicates a consistent but small absolute gain. Larger separations appear for slower planners (e.g., `r6_lpa_star`: 5.99 ms [5.08, 8.02]; `r4_idastar`: 13.38 ms [7.14, 34.43]).

*c) Search-effort trade-offs.:* `r5_jump_point_search` attains the lowest expansion count (57.26) while remaining in the fast runtime cluster (0.64 ms), reducing expansions by roughly 69.7% relative to baseline `astar` (189.06). `r2_bidirectional_astar` and `r3_theta_star` are slower (1.74 and 1.95 ms). `r6_lpa_star` and `r4_idastar` are clear outliers at 7.13 ms and 27.27 ms, with `r4_idastar` also showing the highest variability (std 26.63 ms; IQR 3.85–51.48 ms) and expansion count (7061.34). The expansion profile in Figure 4 highlights this separation between the efficient frontier (`r5_jump_point_search`) and high-overhead outliers.

*d) Uncertainty and limitations.:* These results are limited to one benchmark regime: 50 mazes, one maze size ($15 \times 15$), one generator (backtracker), and one seed schedule. Timings are wall-clock and mostly sub-millisecond for the fastest methods, so statistical significance should not be conflated with large practical effect size in deployment settings. The benchmark also covers static, fully known mazes only; conclusions may not transfer to dynamic, partially observable, or larger-scale environments. Finally, Theta* uses any-angle motion, so its path-length values are not directly comparable to cardinal-grid planners. The flat success-rate profile in Figure 5 further emphasizes that this dataset mainly distinguishes planners by efficiency rather than solvability.

TABLE III

PAIRED RUNTIME COMPARISONS AGAINST `r1_weighted_astar` ON THE SAME 50 MAZES. POSITIVE $\Delta$ MEANS THE COMPARATOR IS SLOWER. CONFIDENCE INTERVALS ARE PERCENTILE BOOTSTRAP INTERVALS FROM 40,000 PAIRED RESAMPLES (FIXED SEED). $p$-VALUES ARE EXACT TWO-SIDED PAIRED SIGN TESTS WITH HOLM CORRECTION ACROSS 11 COMPARISONS.

| Comparator | Median $\Delta$ (ms) | 95% CI for $\Delta$ (ms) | Slower/Faster (of 50) | Holm-adjusted $p$ |
|---|---|---|---|---|
| R7 Beam Search | 0.095 | [0.057, 0.117] | 50/0 | $1.95 \times 10^{-14}$ |
| Greedy Best-First | 0.113 | [0.091, 0.147] | 49/1 | $9.06 \times 10^{-14}$ |
| R5 Jump Point Search | 0.145 | [0.094, 0.212] | 50/0 | $1.95 \times 10^{-14}$ |
| R9 Bidirectional BFS | 0.193 | [0.134, 0.283] | 50/0 | $1.95 \times 10^{-14}$ |
| R8 Fringe Search | 0.219 | [0.157, 0.334] | 50/0 | $1.95 \times 10^{-14}$ |
| A* | 0.232 | [0.155, 0.349] | 50/0 | $1.95 \times 10^{-14}$ |
| Dijkstra | 0.206 | [0.152, 0.347] | 50/0 | $1.95 \times 10^{-14}$ |
| R2 Bidirectional A* | 1.408 | [0.823, 1.868] | 50/0 | $1.95 \times 10^{-14}$ |
| R3 Theta* | 1.230 | [0.864, 2.112] | 50/0 | $1.95 \times 10^{-14}$ |
| R6 LPA* | 5.991 | [5.078, 8.016] | 50/0 | $1.95 \times 10^{-14}$ |
| R4 IDA* | 13.380 | [7.143, 34.426] | 50/0 | $1.95 \times 10^{-14}$ |

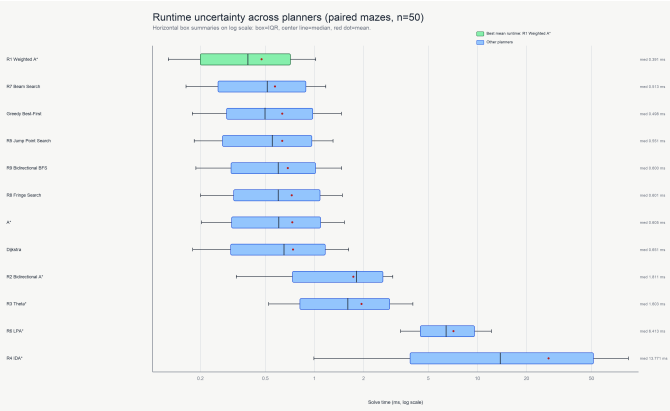


Fig. 3. Runtime uncertainty over the same 50 paired mazes. Horizontal box summaries are shown on a logarithmic scale (box: IQR, center line: median, red dot: mean).

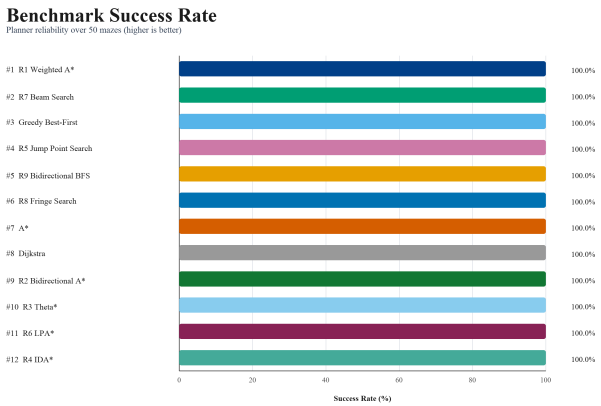


Fig. 5. Planner success rate over 50 mazes. All methods achieve 100% in this static benchmark setting.

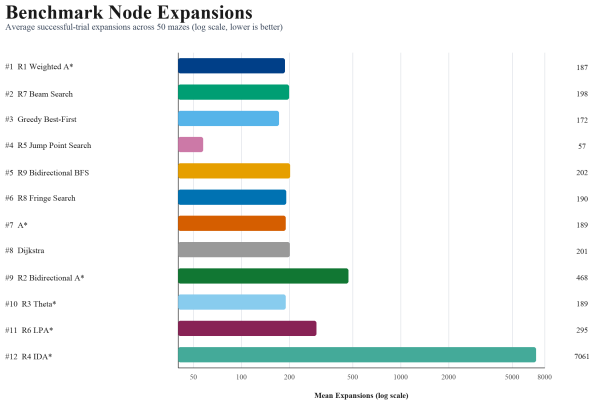


Fig. 4. Mean node expansions on a logarithmic scale for the same benchmark runs. Lower values indicate lower search effort.

## VI. DISCUSSION

The current repository snapshot establishes a reproducible baseline for grid-maze navigation with measurable algorithmic tradeoffs. The benchmark artifact reports 12 planners evaluated on 50 generated $15 \times 15$ mazes, with 100% success for every planner in this static setting. Under the executable ranking policy (success rate, then comparable solve time on shared-success mazes, then comparable path length on shared-success mazes, then mean expansions, then overall mean solve time), `r1_weighted_astar` is first (0.47 ms mean solve time), while `r4_idastar` is last (27.27 ms), showing that admissible but deep iterative search remains expensive for this map class.

### A. What the Current Results Establish

Three descriptive findings are consistent within the present benchmark regime. First, weighted heuristic guidance is already beneficial: `r1_weighted_astar` is approximately 35% faster than baseline `astar` while keeping equivalent mean path length in this dataset. Second, frontier-pruning methods reduce search effort but do not always dominate runtime: `r5_jump_point_search` has the lowest mean expansions (57.26) yet is not the fastest due to additional bookkeeping overhead. Third, algorithm classes optimized for other settings underperform in this static benchmark; for example, `r6_lpa_star` and `r4_idastar` add overhead without gains when maps are regenerated rather than incrementally repaired. These observations are descriptive summaries from the executed benchmark snapshot and are not presented as inferential superiority claims.

Beyond planner speed, the infrastructure contributes practical value. The simulator supports explicit backend selection (`pybullet`, `mujoco`, or `auto`), URDF fallback behavior, and deterministic screenshot regression capture for visual checks. These implementation details improve reproducibility and reduce benchmark fragility across machines.

### B. Implemented Versus Not-Yet-Implemented Methods

**Implemented and used in the reported benchmark.**

- Baseline grid planners: `astar`, `dijkstra`, `greedy_best_first`.
- Alternative planners: `r1_weighted_astar`, `r2_bidirectional_astar`, `r3_theta_star`, `r4_idastar`, `r5_jump_point_search`, `r6_lpa_star`, `r7_beam_search`, `r8_fringe_search`, and `r9_bidirectional_bfs`.
- Runtime stack support: backend auto-resolution, robust URDF fallback, and deterministic screenshot regression scripts.

**Implemented in repository but not part of the current summary table.**

- Additional adapter modules `r11_dijkstra`, `r12_bfs`, and `r13_greedy_best_first` exist, but the current benchmark artifact reports the baseline/alternative set above.

**Not yet implemented (currently documented as future directions).**

- Learned local heuristics for search (LoHA-style A* extensions).
- D* Lite + DWA and AD* + DWA hybrid global-local replanning.
- Heading-aware SE(2) Hybrid-A* / state-lattice planning and IGHA* style incremental hybrid search.
- Guided RRT* variants and uncertainty-aware MPC for dynamic, continuous navigation.

### C. Limitations and Threats to Validity

The present evidence is strong for reproducible static-grid comparison, but limited for broader robotics claims.

- **Task distribution**: all reported mazes are $15 \times 15$ and generated by one algorithm family in one snapshot, so topology diversity is limited.
- **Metric saturation**: with 100% success across planners, failure robustness cannot be ranked; only efficiency metrics differentiate methods.
- **Path comparability**: any-angle methods (e.g., `r3_theta_star`) are not directly comparable to strict lattice paths without additional smoothness/feasibility metrics.
- **Physics realism**: MuJoCo fallback currently executes a simplified waypoint progression, and no dynamic-obstacle uncertainty benchmark is yet integrated.
- **Generalization**: current conclusions are for occupancy-grid planning and do not yet establish superiority in SE(2)/kinodynamic domains.

### D. Aggressive Forward-Looking Hypotheses

*a) H1: Learned local heuristics can outperform fixed heuristics without hurting reliability.:* **Hypothesis:** A LoHA-style heuristic module integrated into A* will reduce mean node expansions by at least 30% relative to `r1_weighted_astar` while preserving at least 99% success. **Risk:** Heuristic overfitting to one maze generator can degrade path quality or fail on held-out topology families. **Validation plan:** implement a plug-in planner with admissible fallback; evaluate on multiple maze generators and sizes with disjoint seeds; require gains on both mean and p95 expansions, with path-length increase bounded to 3%.

*b) H2: Incremental global-local coupling will dominate in dynamic maps.:* **Hypothesis:** D* Lite + DWA (and later AD* + DWA) will reduce replanning latency and collision rate compared with repeated full replanning from `astar` or `r6_lpa_star`. **Risk:** Frequent invalidation can trigger thrashing, and local-control oscillations can erase global-plan gains. **Validation plan:** add controlled dynamic-obstacle perturbations with fixed update rates; track repair latency, collision count, and progress-to-goal; accept only if p95 repair latency and collision rate both improve with no decrease in success.

*c) H3: Heading-aware planning will improve executable trajectory quality.:* **Hypothesis:** SE(2) Hybrid-A* or lattice planning will reduce curvature violations and reverse-maneuver artifacts versus grid-only planners when replayed in physics simulation. **Risk:** State-space explosion can push runtime beyond practical bounds for online use. **Validation plan:** introduce discretized heading bins and motion primitives; benchmark feasibility metrics (curvature, heading discontinuity, control effort) alongside runtime; enforce a runtime budget gate before inclusion in default planner recommendations.

*d) H4: Uncertainty-aware continuous planning can unlock large-map robustness.:* **Hypothesis:** Guided RRT* or uncertainty-aware MPC will provide better safety-efficiency tradeoffs than deterministic grid search on large maps with moving obstacles. **Risk:** Stochastic planners and learned priors may have high variance and weak reproducibility without strict seeding and confidence intervals. **Validation plan:** evaluate at larger scales with many seeds and report median/p95 statistics plus confidence intervals; include ablations that remove learned priors/uncertainty terms to verify causal contribution.

## VII. Conclusion

This manuscript stage delivers a reproducible baseline for maze-navigation planning with explicit evidence from code and benchmark artifacts. In the current snapshot, 12 implemented planners solve all 50 static $15 \times 15$ benchmark mazes, with `r1_weighted_astar` leading runtime and `r4_idastar` showing the highest computational burden. The implemented system is strongest today as a controlled occupancy-grid comparison and simulation harness (backend fallback, URDF robustness, and deterministic visual-regression support), not yet as a fully dynamic or kinodynamically realistic planner stack.

Method scope is therefore explicit: classical and alternative grid-search planners are implemented and benchmarked, while

LoHA-style learned heuristics, D* Lite/AD* + DWA coupling, SE(2) Hybrid-A* or IGHA*, guided RRT*, and uncertainty-aware MPC remain not-yet-implemented research directions. The next milestone is to convert these forward hypotheses into acceptance-tested experiments with dynamic obstacles, larger maps, and feasibility-oriented metrics so future claims are supported by statistically grounded evidence rather than extrapolation.

## APPENDIX A
## ADDITIONAL DETAILS

### A. Notation and Occupancy-Grid Construction

Let the maze have width $W$ and height $H$ in cell coordinates. The benchmark converts each maze to a binary occupancy lattice

$$G \in \{0,1\}^{R\times C}, \quad R = 2H+1, \quad C = 2W+1, \tag{1}$$

where $G_{r,c} = 0$ denotes free space and $G_{r,c} = 1$ denotes blocked space. The cell-to-lattice map used by the implementation is

$$\phi(x,y) = (2y+1,\ 2x+1), \tag{2}$$

so the benchmark start and goal nodes are

$$s = \phi(x_s, y_s), \qquad g = \phi(x_g, y_g). \tag{3}$$

For a lattice node $n = (r,c)$, the neighborhood is either 4-connected or 8-connected:

$$\mathcal{N}_4(n) = \{(r-1,c),(r,c+1),(r+1,c),(r,c-1)\} \cap \Omega, \tag{4}$$

$$\mathcal{N}_8(n) = \mathcal{N}_4(n) \cup \{(r-1,c-1),(r-1,c+1),(r+1,c+1),(r+1,c-1)\} \cap \Omega \tag{5}$$

with $\Omega$ the in-bounds lattice set. The step cost in the baseline planners is

$$c(u,v) = \begin{cases} \sqrt{2}, & \text{if } u \to v \text{ is diagonal,} \\ 1, & \text{otherwise.} \end{cases} \tag{6}$$

Heuristic options are

$$h_{\text{man}}(n,g) = |r - r_g| + |c - c_g|, \tag{7}$$

$$h_{\text{euc}}(n,g) = \sqrt{(r-r_g)^2 + (c-c_g)^2}, \tag{8}$$

$$h_{\text{cheb}}(n,g) = \max\{|r-r_g|, |c-c_g|\}. \tag{9}$$

### B. Planner and Benchmark Objectives

For a path $\pi = (n_0, \ldots, n_K)$ with $n_0 = s$ and $n_K = g$, the weighted path-cost objective is

$$J_{\text{cost}}(\pi) = \sum_{k=1}^{K} c(n_{k-1}, n_k). \tag{10}$$

The baseline priority keys used in the benchmark are:

$$f_{\text{A}*}(n) = g(n) + w\, h(n), \qquad w \geq 1, \tag{11}$$

$$f_{\text{Dijkstra}}(n) = g(n), \tag{12}$$

$$f_{\text{GBFS}}(n) = h(n), \tag{13}$$

and BFS minimizes hop count

$$J_{\text{hop}}(\pi) = K. \tag{14}$$

For A*, the secondary tie key is

$$\tau(n) = \begin{cases} h(n), & \text{low\_h,} \\ -g(n), & \text{high\_g,} \\ 0, & \text{fifo,} \end{cases} \tag{15}$$

and the heap key is $(f(n), \tau(n), t(n))$ where $t(n)$ is a monotonically increasing insertion counter.

*Path validation and measured length:* Given returned path $P = (p_0, \ldots, p_K)$, validity requires:

$$p_0 = s, \quad p_K = g, \quad p_k \in \Omega, \quad G_{p_k} = 0,\ \forall k, \tag{16}$$

and for each segment, every rasterized Bresenham cell is also in-bounds and free. If $\mathcal{B}(p_{k-1}, p_k)$ denotes the rasterized segment cell sequence, the measured benchmark path length is

$$L(P) = \sum_{k=1}^{K} \left(|\mathcal{B}(p_{k-1}, p_k)| - 1\right). \tag{17}$$

*Planner ranking objective:* For planner $p$ over $M$ mazes, with success indicator $I_{p,m} \in \{0,1\}$, solve time $t_{p,m}$, measured length $L_{p,m}$, and expansions $e_{p,m}$:

$$S_p = \frac{1}{M}\sum_{m=1}^{M} I_{p,m}, \qquad \mathcal{M}_p^+ = \{m \mid I_{p,m} = 1\}. \tag{18}$$

Define the shared-success set

$$\mathcal{M}_{\text{shared}} = \{m \mid I_{q,m} = 1,\ \forall q \in \mathcal{P}\}. \tag{19}$$

The comparable means used for ranking are

$$\bar{t}_p^{\,c} = \begin{cases} \frac{1}{|\mathcal{M}_{\text{shared}}|}\sum_{m\in\mathcal{M}_{\text{shared}}} t_{p,m}, & |\mathcal{M}_{\text{shared}}| > 0, \\ \frac{1}{M}\sum_{m=1}^{M} t_{p,m}, & \text{otherwise,} \end{cases} \tag{20}$$

$$\bar{L}_p^{\,c} = \begin{cases} \frac{1}{|\mathcal{M}_{\text{shared}}|}\sum_{m\in\mathcal{M}_{\text{shared}}} L_{p,m}, & |\mathcal{M}_{\text{shared}}| > 0, \\ \frac{1}{|\mathcal{M}_p^+|}\sum_{m\in\mathcal{M}_p^+} L_{p,m}, & \text{otherwise.} \end{cases} \tag{21}$$

With

$$\bar{e}_p = \frac{1}{|\mathcal{M}_p^+|}\sum_{m\in\mathcal{M}_p^+} e_{p,m}, \qquad \bar{t}_p = \frac{1}{M}\sum_{m=1}^{M} t_{p,m}, \tag{22}$$

ranking is lexicographic on

$$\left(-S_p,\ \bar{t}_p^{\,c},\ \bar{L}_p^{\,c},\ \bar{e}_p,\ \bar{t}_p,\ \text{name}_p\right). \tag{23}$$

### C. Local Control Abstraction

*Path-to-waypoint map:* The simulator converts planner outputs to world-frame waypoints. If a path is recognized as cell-grid coordinates $(x_k, y_k)$:

$$w_k = \left((x_k + 0.5)s_c,\ (y_k + 0.5)s_c\right), \tag{24}$$

where $s_c$ is `cell_size`. If recognized as occupancy-lattice coordinates:

$$w_k = \left(0.5\, s_c\, x_k,\ 0.5\, s_c\, y_k\right). \tag{25}$$

Cell-grid waypoints are additionally compressed by removing collinear interior points.

*Waypoint tracking law:* At control step $t$, with robot pose $(x_t, y_t, \psi_t)$ and active waypoint $(x_j, y_j)$:

$$d_t = \sqrt{(x_j - x_t)^2 + (y_j - y_t)^2}, \tag{26}$$

$$\theta_t = \operatorname{atan2}(y_j - y_t,\ x_j - x_t), \qquad e_{\psi,t} = \operatorname{wrap}_{[-\pi,\pi]}(\theta_t - \psi_t). \tag{27}$$

The angular command is the clipped proportional target

$$\omega_t^\star = \arg\min_{|\omega| \le \omega_{\max}} |\omega - k_\psi e_{\psi,t}| = \operatorname{clip}(k_\psi e_{\psi,t}, -\omega_{\max}, \omega_{\max}). \tag{28}$$

The linear target is the largest feasible speed under speed, distance, and braking bounds:

$$v_t^\star = \alpha_t \cdot \max_{v \ge 0} v \quad \text{s.t.} \quad v \le v_{\max}, \quad v \le d_t, \quad v \le \sqrt{2 a_{\text{dec}} d_t}, \tag{29}$$

equivalently

$$v_t^\star = \alpha_t \cdot \min\left\{ v_{\max},\ d_t,\ \sqrt{2 a_{\text{dec}} d_t} \right\}. \tag{30}$$

The heading gate $\alpha_t \in [0, 1]$ is

$$\alpha_t = \begin{cases} 0, & |e_{\psi,t}| \ge \theta_{\text{turn}}, \\ 1, & |e_{\psi,t}| \le \theta_{\text{drive}}, \\ 1 - \sigma\left( \dfrac{|e_{\psi,t}| - \theta_{\text{drive}}}{\theta_{\text{turn}} - \theta_{\text{drive}}} \right), & \text{otherwise}, \end{cases} \tag{31}$$

with smoothstep

$$\sigma(z) = z^2(3 - 2z), \quad z \in [0, 1]. \tag{32}$$

Commands are passed through acceleration/deceleration slew limits with control step $\Delta t$:

$$u_t = u_{t-1} + \operatorname{clip}(u_t^\star - u_{t-1}, -\Delta_u, \Delta_u), \tag{33}$$

where

$$\Delta_u = \begin{cases} a_{u,+} \Delta t, & \operatorname{sgn}(u_t^\star) = \operatorname{sgn}(u_{t-1}) \ \wedge \ |u_t^\star| > |u_{t-1}|, \\ a_{u,-} \Delta t, & \text{otherwise}, \end{cases} \tag{34}$$

applied separately to $u = v$ and $u = \omega$.

Then the turn-rate-dependent linear cap is enforced:

$$\eta_t = \operatorname{clip}\left( \frac{|\omega_t|}{\omega_{\max}}, 0, 1 \right), \qquad v_{\text{turn},t} = v_{\max} \max\left(0.18,\ 1 - 0.68\, \sigma(\eta_t)\right), \tag{35}$$

$$v_t \leftarrow \operatorname{clip}(v_t, -v_{\text{turn},t}, v_{\text{turn},t}). \tag{36}$$

For differential drive with axle track $b$ and wheel radius $r$:

$$v_{L,t} = v_t - \frac{b}{2}\omega_t, \qquad v_{R,t} = v_t + \frac{b}{2}\omega_t, \tag{37}$$

$$\Omega_{L,t} = v_{L,t}/r, \qquad \Omega_{R,t} = v_{R,t}/r. \tag{38}$$

Waypoint index $j$ is advanced when $d_t \le \varepsilon_j$, where $\varepsilon_j$ is the per-waypoint tolerance (larger at the final waypoint).

TABLE IV
REPOSITORY-GROUNDED REPRODUCIBILITY CHECKLIST SNAPSHOT (2026-02-26).

| Item | Status | Evidence |
|---|---|---|
| Environment specification and dependencies | complete | `pixi.toml` (workspace tasks), `robotics_maze/pixi.toml` (Python 3.11, PyBullet, MuJoCo, pytest). |
| CLI entrypoints and run commands | complete | `scripts/sim_runner.py` and root `pixi.toml` task definitions. |
| Physics backend fallback behavior | complete | `robotics_maze/src/sim.py` backend resolution and deterministic fallback branch. |
| Planner benchmark protocol and ranking policy | complete | `robotics_maze/src/benchmark.py`; generated `robotics_maze/results/benchma` |
| Benchmark trial artifacts | complete | `robotics_maze/results/benchmark_resu` and `robotics_maze/results/benchmark_s` |
| Regression screenshot generation | complete | `robotics_maze/scripts/capture_regres` strict expected filename checks. |
| Smoke and regression test logs | complete | `robotics_maze/tests/test_core.py` and `robotics_maze/testing/TEST_RUN_LO` |
| Reference inventory target (≥40 refs from 2021+) | complete | `references.bib` and `coordination/citations_audit.csv` contain populated, post-2021 citation records in the current snapshot. |
| Citation quality target (≥80% peer reviewed) | complete | `coordination/citations_audit.csv` marks peer-review status per entry and is populated beyond header-only state. |
| Figure manifest inventory completeness | complete | `coordination/figure_manifest.csv` lists tracked figure IDs, source paths, and reproducibility fields. |
| Claim-traceability freshness gate | partial | `coordination/claims_traceability.csv` includes verified and pending items; pending rows remain explicit follow-up work. |

### D. Complexity Statements

Let $V = RC$ and $E \le 8V$ for the occupancy lattice graph.

$$T_{\text{A}*/\text{Dijkstra/GBFS}} = O(E \log V), \qquad M_{\text{A}*/\text{Dijkstra/GBFS}} = O(V), \tag{39}$$

because open lists are binary heaps and closed/g-score/came-from are hash maps/sets.

$$T_{\text{BFS}} = O(V + E), \qquad M_{\text{BFS}} = O(V), \tag{40}$$

from deque frontier plus visited/parent maps.

For one returned path $P$:

$$T_{\text{validate}} = O\left( \sum_{k=1}^{K} |\mathcal{B}(p_{k-1}, p_k)| \right) = O(L(P)). \tag{41}$$

For $M$ mazes and $|\mathcal{P}|$ planners:

$$T_{\text{benchmark}} = O\left( \sum_{m=1}^{M} \sum_{p \in \mathcal{P}} \left( T_{\text{plan}}(p, m) + T_{\text{validate}}(p, m) \right) \right). \tag{42}$$

Per control update, command synthesis is constant-time, and actuator dispatch is linear in wheel-joint count $n_w$:

$$T_{\text{control step}} = O(1 + n_w). \tag{43}$$

### E. Reproducibility Checklist

This checklist reflects the repository snapshot date listed in the caption. The coordination artifacts preserve both verified and pending entries so unresolved follow-up tasks remain auditable rather than silently dropped.

## REFERENCES


[1] J. R. Sánchez-Ibáñez, C. J. Pérez-del Pulgar, and A. García-Cerezo, "Path planning for autonomous mobile robots: A review," *Sensors*, vol. 21, no. 23, p. 7898, Nov. 2021. [Online]. Available: http://dx.doi.org/10.3390/s21237898

[2] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, p. 448–468, Aug. 2021. [Online]. Available: http://dx.doi.org/10.3390/vehicles3030027

[3] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion planning and control for mobile robot navigation using machine learning: a survey," *Autonomous Robots*, vol. 46, no. 5, p. 569–597, Mar. 2022. [Online]. Available: http://dx.doi.org/10.1007/s10514-022-10039-8

[4] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, vol. 227, p. 120254, Oct. 2023. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2023.120254

[5] C. Baumann and A. Martinoli, "A modular functional framework for the design and evaluation of multi-robot navigation," *Robotics and Autonomous Systems*, vol. 144, p. 103849, Oct. 2021. [Online]. Available: http://dx.doi.org/10.1016/J.ROBOT.2021.103849

[6] J. He, J. Zhang, J. Liu, and X. Fu, "A ros2-based framework for industrial automation systems," in *2022 2nd International Conference on Computer, Control and Robotics (ICCCR)*. IEEE, Mar. 2022, pp. 98–102. [Online]. Available: http://dx.doi.org/10.1109/ICCCR54399.2022.9790247

[7] K. Chen, M. Wang, M. Gualtieri, N. Tian, C. Juette, L. Ren, J. Ichnowski, J. Kubiatowicz, and K. Goldberg, "Fogros2-ls: A location-independent fog robotics framework for latency sensitive ros2 applications," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2024, pp. 10 581–10 587. [Online]. Available: http://dx.doi.org/10.1109/ICRA57147.2024.10610759

[8] V. Shcherbyna, L. Kastner, D. Diaz, H. G. Nguyen, M. H.-K. Schreff, T. Seeger, J. Kreutz, A. Martban, Z. Shen, H. Zeng, and H. Soh, "Arena 4.0: a comprehensive ros2 development and benchmarking platform for human-centric navigation using generative-model-based environment generation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2025, pp. 9138–9144. [Online]. Available: http://dx.doi.org/10.1109/ICRA55743.2025.11127635

[9] H. S. Hewawasam, M. Y. Ibrahim, and G. K. Appuhamillage, "Past, present and future of path-planning algorithms for mobile robot navigation in dynamic environments," *IEEE Open Journal of the Industrial Electronics Society*, vol. 3, p. 353–365, 2022. [Online]. Available: http://dx.doi.org/10.1109/OJIES.2022.3179617

[10] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," *Engineering Science and Technology, an International Journal*, vol. 40, p. 101343, Apr. 2023. [Online]. Available: http://dx.doi.org/10.1016/j.jestch.2023.101343

[11] J. A. Abdulsaheb and D. J. Kadhim, "Classical and heuristic approaches for mobile robot path planning: A survey," *Robotics*, vol. 12, no. 4, p. 93, Jun. 2023. [Online]. Available: http://dx.doi.org/10.3390/robotics12040093

[12] Z. Jian, S. Zhang, S. Chen, Z. Nan, and N. Zheng, "A global-local coupling two-stage path planning method for mobile robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, p. 5349–5356, Jul. 2021. [Online]. Available: http://dx.doi.org/10.1109/LRA.2021.3074878

[13] C. Li, X. Huang, J. Ding, K. Song, and S. Lu, "Global path planning based on a bidirectional alternating search a* algorithm for mobile robots," *Computers & Industrial Engineering*, vol. 168, p. 108123, Jun. 2022. [Online]. Available: http://dx.doi.org/10.1016/j.cie.2022.108123

[14] W. Chi, Z. Ding, J. Wang, G. Chen, and L. Sun, "A generalized voronoi diagram-based efficient heuristic path planning method for rrts in mobile robots," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 5, p. 4926–4937, May 2022. [Online]. Available: http://dx.doi.org/10.1109/TIE.2021.3078390

[15] Z. Jiang, W. Wang, W. Sun, and L. Da, "Path planning method for mobile robot based on a hybrid algorithm," *Journal of Intelligent & Robotic Systems*, vol. 109, no. 3, Oct. 2023. [Online]. Available: http://dx.doi.org/10.1007/S10846-023-01985-1

[16] L. Han, X. Wu, and X. Sun, "Hybrid path planning algorithm for mobile robot based on a* algorithm fused with dwa," in *2023 IEEE 3rd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA)*. IEEE, May 2023, pp. 1465–1469. [Online]. Available: http://dx.doi.org/10.1109/ICIBA56860.2023.10165386

[17] M. Hu, S. Jiang, K. Zhou, X. Cao, and C. Li, "Improved exponential and cost-weighted hybrid algorithm for mobile robot path planning," *Sensors*, vol. 25, no. 8, p. 2579, Apr. 2025. [Online]. Available: http://dx.doi.org/10.3390/S25082579

[18] J. C. de Jesus, V. A. Kich, A. H. Kolling, R. B. Grando, M. A. d. S. L. Cuadros, and D. F. T. Gamarra, "Soft actor-critic for navigation of mobile robots," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 2, May 2021. [Online]. Available: http://dx.doi.org/10.1007/s10846-021-01367-5

[19] B. Liu, X. Xiao, and P. Stone, "A lifelong learning approach to mobile robot navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, p. 1090–1096, Apr. 2021. [Online]. Available: http://dx.doi.org/10.1109/LRA.2021.3056373

[20] M. Pei, H. An, B. Liu, and C. Wang, "An improved dyna-q algorithm for mobile robot path planning in unknown dynamic environment," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 7, p. 4415–4425, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1109/TSMC.2021.3096935

[21] A. K. Mackay, L. Riazuelo, and L. Montano, "Rl-dovs: Reinforcement learning for autonomous robot navigation in dynamic environments," *Sensors*, vol. 22, no. 10, p. 3847, May 2022. [Online]. Available: http://dx.doi.org/10.3390/S22103847

[22] X. Gao, L. Yan, Z. Li, G. Wang, and I.-M. Chen, "Improved deep deterministic policy gradient for dynamic obstacle avoidance of mobile robot," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 6, p. 3675–3682, Jun. 2023. [Online]. Available: http://dx.doi.org/10.1109/TSMC.2022.3230666

[23] Z. Bai, H. Pang, Z. He, B. Zhao, and T. Wang, "Path planning of autonomous mobile robot in comprehensive unknown environment using deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 11, no. 12, p. 22153–22166, Jun. 2024. [Online]. Available: http://dx.doi.org/10.1109/JIOT.2024.3379361

[24] Y. Yin, Z. Chen, G. Liu, J. Yin, and J. Guo, "Autonomous navigation of mobile robots in unknown environments using off-policy reinforcement learning with curriculum learning," *Expert Systems with Applications*, vol. 247, p. 123202, Aug. 2024. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2024.123202

[25] Y. Hu, S. Wang, Y. Xie, S. Zheng, P. Shi, I. Rudas, and X. Cheng, "Deep reinforcement learning-based mapless navigation for mobile robot in unknown environment with local optima," *IEEE Robotics and Automation Letters*, vol. 10, no. 1, p. 628–635, Jan. 2025. [Online]. Available: http://dx.doi.org/10.1109/LRA.2024.3511437

[26] B. Xue, F. Zhou, C. Wang, M. Gao, and L. Yin, "Robot mapless navigation in vuca environments via deep reinforcement learning," *IEEE Transactions on Industrial Electronics*, vol. 72, no. 1, p. 639–649, Jan. 2025. [Online]. Available: http://dx.doi.org/10.1109/TIE.2024.3404113

[27] W. Zhang, S. Wang, M. Tan, Z. Yang, X. Wang, and X. Shen, "Drl-dclp: A deep reinforcement learning-based dimension-configurable local planner for robot navigation," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 3636–3643, Apr. 2025. [Online]. Available: http://dx.doi.org/10.1109/LRA.2025.3544927

[28] E. Tolstaya, J. Paulos, V. Kumar, and A. Ribeiro, "Multi-robot coverage and exploration using spatial graph neural networks," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sep. 2021, p. 8944–8950. [Online]. Available: http://dx.doi.org/10.1109/IROS51168.2021.9636675

[29] Y. Gao, Y. Wang, X. Zhong, T. Yang, M. Wang, Z. Xu, Y. Wang, Y. Lin, C. Xu, and F. Gao, "Meeting-merging-mission: A multi-robot coordinate framework for large-scale communication-limited exploration," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2022, p. 13700–13707. [Online]. Available: http://dx.doi.org/10.1109/IROS47612.2022.9981544

[30] H. Zhang, J. Cheng, L. Zhang, Y. Li, and W. Zhang, "H2gnn: Hierarchical-hops graph neural networks for multi-robot exploration in unknown environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, p. 3435–3442, Apr. 2022. [Online]. Available: http://dx.doi.org/10.1109/LRA.2022.3146912

[31] Z. Zhang, J. Yu, J. Tang, Y. Xu, and Y. Wang, "Mr-topomap: Multi-robot exploration based on topological map in communication restricted environment," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, p. 10794–10801, Oct. 2022. [Online]. Available: http://dx.doi.org/10.1109/LRA.2022.3192765

[32] A. H. Tan, F. P. Bejarano, Y. Zhu, R. Ren, and G. Nejat, "Deep reinforcement learning for decentralized multi-robot exploration with macro actions," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, p. 272–279, Jan. 2023. [Online]. Available: http://dx.doi.org/10.1109/LRA.2022.3224667

[33] J. Huang, B. Zhou, Z. Fan, Y. Zhu, Y. Jie, L. Li, and H. Cheng, "Fael: Fast autonomous exploration for large-scale environments with a mobile robot," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, p. 1667–1674, Mar. 2023. [Online]. Available: http://dx.doi.org/10.1109/LRA.2023.3236573

[34] Q. Bi, X. Zhang, J. Wen, Z. Pan, S. Zhang, R. Wang, and J. Yuan, "Cure: A hierarchical framework for multi-robot autonomous exploration inspired by centroids of unknown regions," *IEEE Transactions on Automation Science and Engineering*, vol. 21, no. 3, p. 3773–3786, Jul. 2024. [Online]. Available: http://dx.doi.org/10.1109/TASE.2023.3285300

[35] A. Thomas, F. Mastrogiovanni, and M. Baglietto, "Safe motion planning with environment uncertainty," *Robotics and Autonomous Systems*, vol. 156, p. 104203, Oct. 2022. [Online]. Available: http://dx.doi.org/10.1016/J.ROBOT.2022.104203

[36] G. Gutow and J. D. Rogers, "Sparse integration schemes for chance-constrained motion primitive path planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6431–6438, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1109/LRA.2022.3173045

[37] Q. H. Ho, Z. N. Sunberg, and M. Lahijanian, "Planning with simba: Motion planning under uncertainty for temporal goals using simplified belief guides," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2023, pp. 5723–5729. [Online]. Available: http://dx.doi.org/10.1109/ICRA48891.2023.10160897

[38] C. Adu, J. Liu, L. Lymburner, V. Kaushik, L. Trang, and R. Vasudevan, "Radius: Risk-aware, real-time, reachability-based motion planning," in *Robotics: Science and Systems XIX*, ser. RSS2023. Robotics: Science and Systems Foundation, Jul. 2023. [Online]. Available: http://dx.doi.org/10.15607/RSS.2023.XIX.083

[39] Y. K. Nakka and S.-J. Chung, "Trajectory optimization of chance-constrained nonlinear stochastic systems for motion planning under uncertainty," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 203–222, Feb. 2023. [Online]. Available: http://dx.doi.org/10.1109/TRO.2022.3197072

[40] D. Zheng and P. Tsiotras, "Ibbt: Informed batch belief trees for motion planning under uncertainty," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2024, pp. 2403–2409. [Online]. Available: http://dx.doi.org/10.1109/ICRA57147.2024.10610244

[41] A. Theurkauf, J. Kottinger, N. Ahmed, and M. Lahijanian, "Chance-constrained multi-robot motion planning under gaussian uncertainties," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 835–842, Jan. 2024. [Online]. Available: http://dx.doi.org/10.1109/LRA.2023.3337700

[42] R. Takemura and G. Ishigami, "Uncertainty-aware trajectory planning: Using uncertainty quantification and propagation in traversability prediction of planetary rovers," *IEEE Robotics & Automation Magazine*, vol. 31, no. 2, pp. 89–99, Jun. 2024. [Online]. Available: http://dx.doi.org/10.1109/MRA.2023.3341289

[43] E. Heiden, L. Palmieri, L. Bruns, K. O. Arras, G. S. Sukhatme, and S. Koenig, "Bench-mr: A motion planning benchmark for wheeled mobile robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4536–4543, Jul. 2021. [Online]. Available: http://dx.doi.org/10.1109/LRA.2021.3068913

[44] C. Chamzas, C. Quintero-Pena, Z. Kingston, A. Orthey, D. Rakita, M. Gleicher, M. Toussaint, and L. E. Kavraki, "Motionbenchmaker: A tool to generate and benchmark motion planning datasets," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 882–889, Apr. 2022. [Online]. Available: http://dx.doi.org/10.1109/LRA.2021.3133603

[45] J. Karwowski and W. Szynkiewicz, "Srpb: a benchmark for the quantitative evaluation of a social robot navigation," in *2023 27th International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE, Aug. 2023, pp. 411–416. [Online]. Available: http://dx.doi.org/10.1109/MMAR58394.2023.10242422

[46] M. Mayer, J. Kulz, and M. Althoff, "Cobra: A composable benchmark for robotics applications," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2024, pp. 17 665–17 671. [Online]. Available: http://dx.doi.org/10.1109/ICRA57147.2024.10610776

[47] J. Wang, D. Huo, Z. Xu, Y. Shi, Y. Yan, Y. Wang, C. Gao, Y. Qiao, and G. Zhou, "Openbench: A new benchmark and baseline for semantic navigation in smart logistics," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2025, pp. 16 202–16 208. [Online]. Available: http://dx.doi.org/10.1109/ICRA55743.2025.11127476

[48] M. Martini, A. Eirale, S. Cerrato, and M. Chiaberge, "Pic4rl-gym: a ros2 modular framework for robots autonomous navigation with deep reinforcement learning," in *2023 3rd International Conference on Computer, Control and Robotics (ICCCR)*. IEEE, Mar. 2023, pp. 198–202. [Online]. Available: http://dx.doi.org/10.1109/ICCCR56747.2023.10193996

[49] L. Kastner, V. Shcherbyna, H. Soh, G. Truong, D. Anh, T. Kien, T. Seeger, A. Martban, V. Lam, N. Hung, P. Tung, T. An, E. Wiese, and M. Schreff, "Demonstrating arena 5.0: A photorealistic ros2 simulation framework for developing and benchmarking social navigation," in *Robotics: Science and Systems XXI*, ser. RSS2025. Robotics: Science and Systems Foundation, Jun. 2025. [Online]. Available: http://dx.doi.org/10.15607/RSS.2025.XXI.092