

Hybrid A* Path Planning Performance Report

Overview

This program implements a Hybrid A* search algorithm for non-holonomic robot path planning in a 2D occupancy grid. The planner operates in a continuous state space (x,y,ψ,κ) while maintaining discrete buckets for efficient closed-set management. Performance was evaluated under a controlled experiment where the search was allowed to expand at least 100,000 nodes, enabling a consistent comparison of runtime and computational behavior across trials.

Experimental Setup

Search Configuration

- Minimum node expansions: 100,000
- Maximum node expansions: 350,000
- Motion primitive step size: 0.5 m
- Steering samples: 11 discrete steering angles per expansion
- Forward and reverse motion: enabled, with reverse penalty
- Collision checking: continuous arc sampling with polygonal footprint validation

The planner terminates once the goal is reached and the minimum node expansion threshold is satisfied, ensuring stable timing measurements independent of early goal discovery.

Timing Results

Across multiple runs, the planner demonstrated the following performance characteristics:

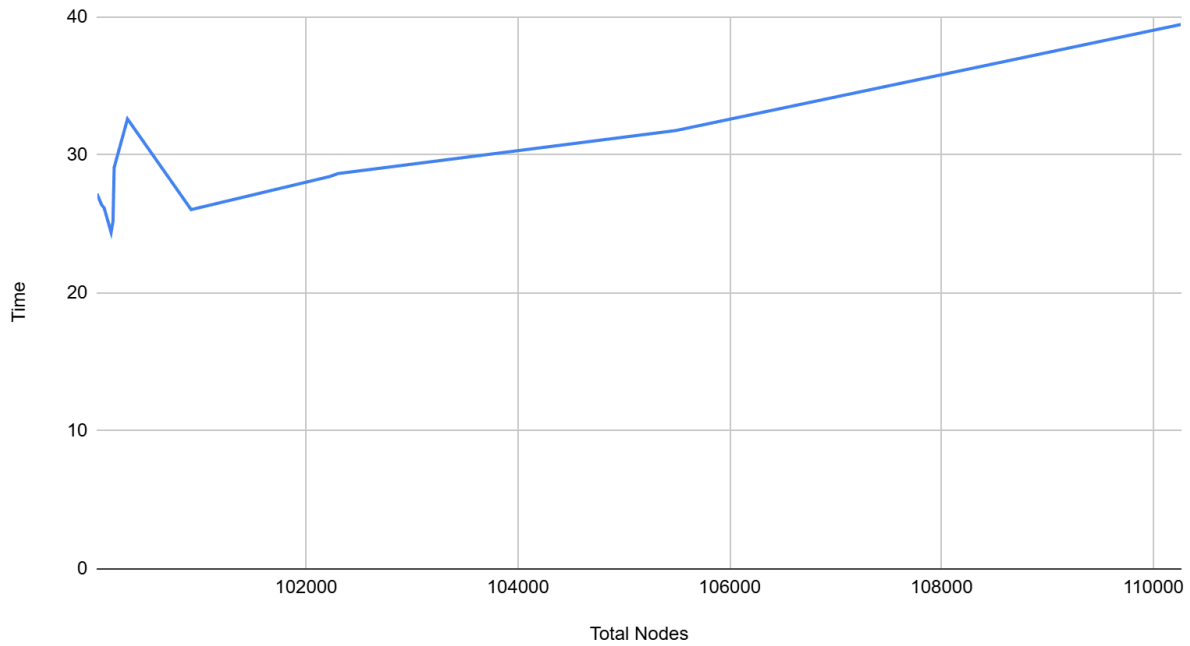
Metric	Value
Average nodes expanded	~101,851
Average runtime	~28.76 seconds
Observed runtime range	~24.4 s – 39.4 s

The runtime scales approximately linearly with node expansions once the minimum threshold is exceeded. Variance across runs is primarily attributable to:

- Randomized obstacle placement in the grid
- Differences in collision rejection frequency
- Heuristic guidance effectiveness near the goal region

The runtime vs. node expansion plot shows a generally monotonic increase, indicating that per-node expansion cost remains stable and predictable.

Hybrid A* Node Expansion with respect to Time



Hardware Description

All experiments were conducted on a ASUS ROG Strix G15 workstation:

- CPU: Intel® Core™ i9-13900H
- Architecture: Single-threaded execution (no parallelism)
- Acceleration: No GPU or SIMD acceleration used
- Operating mode: Python interpreter, CPU-bound

The absence of parallelization or hardware acceleration means the reported results reflect a conservative baseline. Significant speedups are expected with multithreading or C++/CUDA ports.

Algorithmic Optimizations

Several optimizations were incorporated to improve performance while preserving correctness:

1. Discretized State Hashing

States are discretized using configurable resolution buckets in (x, y, ψ, κ) , reducing closed-set growth while avoiding excessive aliasing. This prevents redundant expansions without sacrificing solution quality.

2. Dual Heuristic Strategy

The heuristic combines:

- Holonomic distance with obstacles, computed via a 2D Dijkstra distance transform
- Non-holonomic analytical cost, accounting for heading and curvature mismatch

The maximum of the two heuristics is used, maintaining admissibility while providing strong guidance in cluttered environments.

3. Precomputation

Both holonomic and non-holonomic heuristic components are precomputed before the search begins, removing repeated expensive calculations from the main loop.

4. Continuous Collision Checking Along Arcs

Rather than checking only discrete poses, collisions are evaluated by sampling intermediate poses along each motion arc. This reduces false-negative collisions while keeping the sample count fixed and bounded.

5. Obstacle Proximity Cost

A distance transform based obstacle penalty biases the planner toward safer corridors without introducing hard constraints, improving solution robustness in dense environments.

Observations and Limitations

- Primary bottleneck: Collision checking dominates runtime due to repeated polygon–grid overlap tests.
- Scalability: Runtime grows linearly with node expansions; memory usage grows proportionally with explored states.
- Determinism: Minor runtime variance exists due to randomized grid generation and search ordering.