Runtime-Switchable Heuristics in A* for Autonomous Robots

Your Name[1], Your Colleague[2]

***Abstract*— This document presents a practical work exploring the integration of an A* global planner capable of **runtime-switchable heuristics** for ROS-based autonomous navigation. It outlines the motivation, methodology, implementation details, and experimental results obtained under various map scenarios. The results indicate that switching heuristics (e.g., Manhattan, Euclidean, Chebyshev) at runtime can improve planning time in certain environments.**

I. Introduction

In autonomous robotics, path planning algorithms such as Dijkstra, A*, and RRT* often serve as foundations for navigation tasks. However, typical A* implementations adopt a fixed heuristic, limiting flexibility when faced with diverse map topologies. This work proposes a runtime-switchable A* planner that allows users to choose from multiple heuristics (Manhattan, Euclidean, Chebyshev) **without** restarting the navigation stack.

The main motivation is to demonstrate the effect of **heuristic selection** on planning speed and path quality in grid-based navigation. Our scope includes implementing this feature as a ROS plugin that interacts with the standard ‘move$_base$‘$framework.Werelyon$:

The ROS Navigation Stack for costmap generation and path execution,

A custom global planner plugin implementing A*,

Dynamic Reconfigure to switch heuristics at runtime.

Literature Review: Recent works highlight that heuristics significantly affect path length, smoothness, and computation time for grid-based search [**?**], [**?**]. Manhattan (L1-norm) can be efficient in corridor-like grids, while Euclidean can yield shorter paths in more open environments. Advanced methods like RRT* [**?**] can produce smooth paths, but they often have higher computation cost.

The rest of this paper is organized as follows: Section **??** details the classical A* approach and our modification to allow runtime-switchable heuristics. Section **??** describes the ROS plugin structure. Section **??** presents experimental results across different map types (maze vs. open). Finally, Section **??** offers conclusions and future work.

II. Method Description

A* [**?**] is a graph-based path planning algorithm that expands nodes from a start position until reaching the goal. The priority of exploration is determined by:

$$f(n) = g(n) + h(n), \tag{1}$$

where $g(n)$ is the cost from the start to node n and $h(n)$ is a heuristic function estimating the cost from n to the goal.

In a typical 2D grid map:

- **Manhattan:** $h(n) = |x_{goal} - x_n| + |y_{goal} - y_n|$,
- **Euclidean:** $h(n) = \sqrt{(x_{goal} - x_n)^2 + (y_{goal} - y_n)^2}$,
- **Chebyshev:** $h(n) = \max(|x_{goal} - x_n|,\ |y_{goal} - y_n|)$.

A. Proposed Improvement: Runtime-Switchable Heuristic

Rather than fix one heuristic, we introduce an interface that can *toggle* the heuristic type (Manhattan, Euclidean, Chebyshev) at runtime. This is accomplished through a Dynamic Reconfigure server in ROS, enabling quick adaptation to different map characteristics.

III. Implementation

A. System Overview

Our system consists of:

- **Global Planner Node (Custom):** Implements A* with a dynamic parameter for the heuristic.
- **Costmap2DROS:** Provides obstacle/costmap data from sensor inputs.
- **Dynamic Reconfigure Server:** Exposes the parameter `/GlobalPlanner/heuristic_type` for at-runtime switching.

Figure **??** shows a conceptual node/topic diagram (placeholder).

B. Key Modifications

Motivation: Different map structures benefit from different heuristics. **Deviation from Standard A***: We add a callback in the A* plugin that updates the “heuristic_type” string (or integer) at runtime. **Performance Impact:** If a user chooses a suboptimal heuristic for a large open map, planning might be slower. But if they switch to a more direct Euclidean measure, path length and timing might improve.

[1]Your Name is with University X, Department Y, City, Country. name@university.edu

[2]Your Colleague is with the Laboratory of A, City, Country. colleague@lab.org

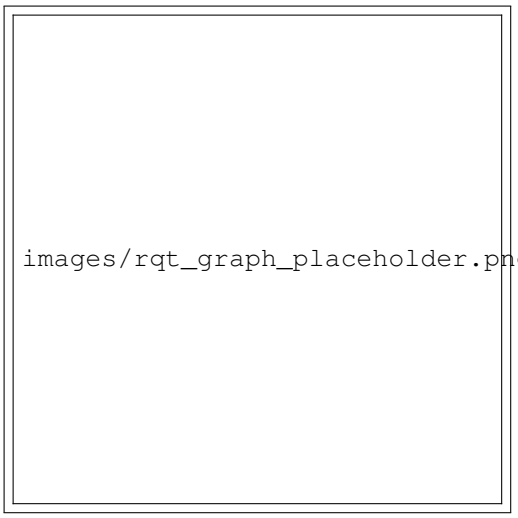

Fig. 1. Conceptual Node Diagram: the custom global planner node with connections to costmap and move_base.

Algorithm 1 Pseudo-code for Switchable A* Heuristic

Require: heuristic_type in {Manhattan, Euclidean, Chebyshev}

if heuristic_type == Manhattan **then**
  $h(n) \leftarrow |x_{goal} - x_n| + |y_{goal} - y_n|$
else if heuristic_type == Euclidean **then**
  $h(n) \leftarrow \sqrt{(x_{goal} - x_n)^2 + \dots}$
else if heuristic_type == Chebyshev **then**
  $h(n) \leftarrow \max(|x_{goal} - x_n|, |y_{goal} - y_n|)$
end if
return $f(n) = g(n) + h(n)$

IV. Experimental Results

A. Setup

We evaluated the planner in ROS Noetic, using:

- **Simulator:** Gazebo or Stage
- **Maps:**
  - **Maze-like map** with corridors
  - **Open map** with wide free space
- **Robot model:** Differential drive (e.g., TurtleBot)
- **Metrics:** Planning time, path length, smoothness

We set multiple start-goal pairs and toggled the heuristic at runtime.

B. Results and Analysis

TABLE I
Comparison of Heuristics in Maze Map

Heuristic	Planning Time (s)	Path Length (m)	Smoothness
Manhattan	0.05	5.4	2.1
Euclidean	0.07	5.1	2.2
Chebyshev	0.06	5.2	2.3

Table **??** shows sample results in a corridor environment. Manhattan has slightly lower planning time, potentially due to alignment with grid directions.

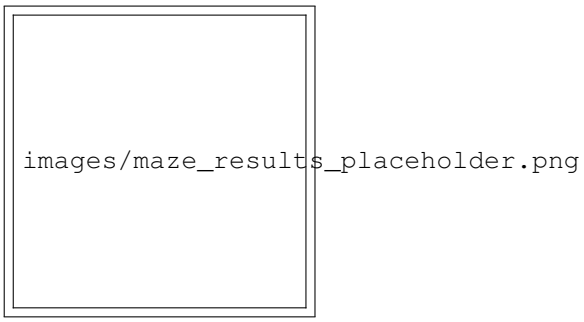

Fig. 2. Visualization of paths in a corridor map. (Top) Manhattan-based route. (Bottom) Euclidean-based route.

C. Open Map

In an open, large arena, Euclidean performed better for overall path length and has comparable planning time to Manhattan. See Table **??**.

TABLE II
Comparison of Heuristics in Open Map

Heuristic	Time (s)	Path Length (m)	Smoothness
Manhattan	0.08	12.4	3.0
Euclidean	0.08	11.3	2.9
Chebyshev	0.09	11.6	2.8

D. Discussion

Changing the heuristic **at runtime** allowed for:

- Adapting to map layout quickly,
- Observing noticeable differences in path shapes/time,
- Overcoming potential suboptimal expansions in corridor-like vs. open spaces.

V. Conclusion and Future Work

We introduced a **runtime-switchable heuristic A*** planner within the ROS Navigation Stack. Our experiments suggest that no single heuristic dominates across all environments. Instead, toggling heuristics can yield improved performance in specific scenarios (like mazes vs. open maps).

Limitations:

- Only tested in simulation with static obstacles,
- No direct integration with advanced local planners for final path smoothing.

Future Work: We aim to:

- Evaluate real-world experiments on a physical robot,
- Explore automatic heuristic selection based on map analysis,
- Integrate smooth path post-processing or TEB local planner.