Motion Planning – Dijkstra's Algorithm

Leo Jaeeun Kim
University of Michigan
General Motors

ABSTRACT

Motion planning is a critical function in autonomous systems, enabling intelligent agents such as mobile robots, autonomous vehicles, and drones to navigate safely and efficiently through complex environments while minimizing cost metrics such as distance, time, or energy. Among classical deterministic search techniques, Dijkstra's algorithm stands out for its completeness and ability to compute optimal paths in known, static environments with non-negative edge weights. This study examines both the theoretical foundations and practical applications of Dijkstra's algorithm in motion planning, with particular emphasis on robotic navigation and Advanced Driver Assistance Systems (ADAS) features such as autonomous emergency braking, lane change assistance, and path-following control.

The algorithm is mathematically formulated and implemented in discretized 2D grid maps to evaluate performance across a range of environments with varying obstacle densities. Key performance indicators include path optimality, node expansion, and computational time. Comparative analysis with A*, a heuristic-based extension of Dijkstra's method, highlights the trade-offs between optimality and computational efficiency. Further analysis covers integration into real-time planning pipelines, including compatibility with occupancy grids, roadmap-based representations, and perception-driven updates.

To address scalability challenges in embedded and high-resolution planning systems, enhancements such as hierarchical frameworks and hybrid search approaches are also discussed. Case studies involving mobile robots and autonomous vehicles underscore the practical relevance of Dijkstra's algorithm in structured road networks and constrained indoor environments. The findings reinforce the algorithm's value as a foundational component of motion planning systems and highlight its continuing role in enabling hybrid, probabilistic, and learning-based planning architectures.

INTRODUCTION

= Motion planning is a critical functionality in the domain of robotics and autonomous systems, tasked with computing a feasible and often optimal trajectory for an agent to move from a source to a destination. The problem becomes significantly more challenging in real-world applications involving dynamic environments, complex terrains, and various motion constraints. Classical approaches to motion planning can be categorized into deterministic and probabilistic methods. Deterministic methods, such as Dijkstra's algorithm, operate on a discretized representation of the environment and provide guarantees on completeness and optimality. These methods are particularly suitable for environments where a complete map is available and the system must find the lowest-cost path to a goal.

Dijkstra's algorithm, proposed in 1956 by Edsger W. Dijkstra, is a greedy, label-setting algorithm that solves the single-source shortest path problem on graphs with non-negative edge weights. In robotic motion planning, the environment is often modeled as a graph where each node represents a discrete position (e.g., a cell in a grid map), and edges denote possible transitions between positions. Dijkstra's algorithm incrementally builds a shortest-path tree from the source node by expanding the node with the smallest tentative cost until the goal node is reached or all reachable nodes are visited.

The algorithm is widely used in various applications due to its deterministic behavior and optimal path generation. However, its main limitation lies in its exhaustive exploration, which can be computationally expensive for large or high-dimensional spaces. This paper aims to provide a detailed examination of Dijkstra's algorithm in the context of motion planning, analyze its strengths and limitations, and compare it with heuristic-based methods such as A*.

METHOD

The algorithm is widely used in various applications due to its deterministic behavior and optimal path generation. However, its main limitation lies in its exhaustive exploration, which can be computationally expensive for large or high-dimensional spaces. This paper aims to provide a detailed examination of Dijkstra's algorithm in the context of motion planning, analyze its strengths and limitations, and compare it with heuristic-based methods such as A*. Dijkstra's algorithm is designed to find the shortest path from a source node to all other nodes $v \in V$ in a weighted graph $G = (V,E)$, where each edge $(u, v) \in E$ has a non-negative weight $w(u, v)$.

Let $d(v)$ is the tentative shortest distance from s to v, Q for a priority queue(min-heap) of nodes ordered by d(v), and π(v) the predecessor of node v in the shortest path.

Initialize with $d(s) = 0$, $d(v) = \infty$, $\pi(v) = NULL$. Insert all nodes into Q based on their $d(v)$ values.

In main loop, while Q is not empty, extract the node u with the minimum d(u) from Q. For each neighbor $v \in Adj(u)$:

$$if\ d(u) + w(u,v) < d(v) then:$$
$$d(v) \leftarrow d(u) + w(u,v)$$
$$\pi(v) \leftarrow u$$
$$Update\ v\ in\ Q$$

The algorithm terminates once all nodes have been visited or when the goal node's shortest distance is finalized. The resulting

set of $\pi(v)$ pointers can be traced back from the goal to reconstruct the optimal path.

Gird-Based Discretization

In practical robotic applications, the continuous environment is discretized into a 2D occupancy grid. Each cell (i,j) in the grid corresponds to a node in the graph. Free cells are connected to their neighbors (4 or 8-connected) using edges with the following weights:

$$w\big((i,j),(i',j')\big) = \begin{cases} 1, & if\ |i-i'|+|j+j'| = 1 \\ \sqrt{2}, & if\ |i-i'| = |j+j'| = 1 \\ \infty, & if\ either\ cell\ is\ an\ obstacle \end{cases}$$

Edge Cost Representation

For generality, the cost of moving from node u to v is given by:

$$w(u,v) = c_d + c_o$$

where c_d is distance-based cost and c_o is additional cost due to terrain difficulty, slope, or other domain-specific penalties.

IMPLEMENTATION

The implementation of Dijkstra's algorithm in a grid-based environment utilizes an implicit graph representation, where each node corresponds to a free cell in the occupancy grid and edges are derived from the neighboring cells, either in 4-connected or 8-connected fashion. The graph structure is efficiently handled through adjacency lists or neighborhood queries during runtime. To ensure efficient retrieval of the minimum-cost node, a binary heap is used to implement the priority queue, which maintains the frontier of unexplored nodes sorted by their tentative distances. A mapping of predecessor nodes, π(v), is preserved throughout the execution to reconstruct the optimal path upon completion of the search. The algorithm achieves a time complexity of *O((V+E)logV)*, where *V* and *E* denote the number of nodes and edges, respectively, assuming a min-heap is used for the priority queue. This implementation setup provides the foundation for evaluating the performance of Dijkstra's algorithm in varied environmental configurations, including scenarios with increasing map size and obstacle density.

Path Smoothing with Splines

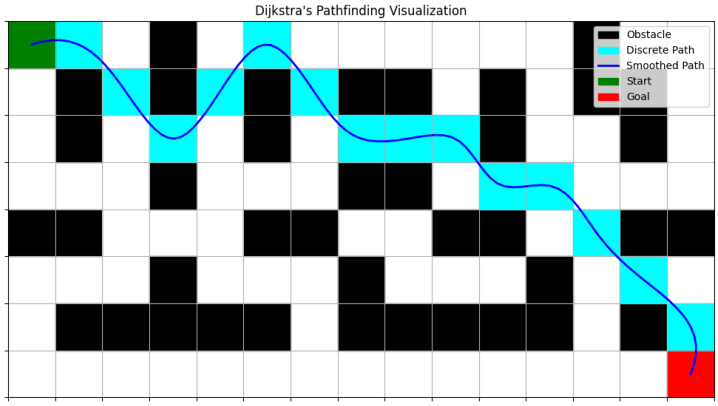
Once the discrete path is computed, a cubic B-spline is fit through the waypoints using SciPy's splprep() and splev() functions. This smooths the transitions between grid points, producing a continuous and differentiable curve suitable for control layers or trajectory tracking.

The B-spline smoothing generates a function γ(u)=(x(u),y(u)) for u∈[0,1], ensuring:

- Smoothness: continuous first and second derivatives
- Accuracy: passes through or near original waypoints
- Feasibility: eliminates abrupt turns in sharp corners

Grid and Visualization

The grid is visualized using matplotlib, where obstacles are rendered in black, the raw grid path in cyan, and the smoothed path as a blue curve. Start and goal positions are highlighted in green and red, respectively.



The example code for Dijkstra algorithm in motion planning can be found in: https://github.com/woa0425/Leo_ADAS_script/blob/main/Motion_Planning/Motion_Planning_Dijkstra_python/Motion_Planning_Dijkstra.py

To assess the effectiveness of the proposed motion planning algorithm, we applied the 8-directional Dijkstra's algorithm with spline-based path smoothing to a structured 15×8 grid environment populated with static obstacles. The environment mimics constrained navigation scenarios relevant to ADAS applications, such as parking layouts or road edges, where obstacle avoidance and smooth steering are critical.
The planner successfully identified a valid and optimal path from the top-left corner (start) to the bottom-right corner (goal), circumventing multiple obstacles. The path was constructed using both cardinal and diagonal steps, minimizing total cost under Euclidean metrics. After computing the discrete path, a cubic B-spline was fitted to the resulting waypoints, generating a continuous trajectory with smooth curvature transitions.
The reported total path cost is 15.00, calculated as the sum of Euclidean distances over all segments in the discrete path. The smooth curve maintains close adherence to the optimal path while providing continuous curvature for downstream controllers.

CONCLUSION

This work presents a curvature-aware motion planning strategy built upon a classical Dijkstra's algorithm with 8-directional movement and B-spline-based trajectory smoothing. The planner guarantees optimality within the grid discretization and addresses practical ADAS challenges by producing geometrically smooth and kinematically feasible paths.

The implementation demonstrates strong performance in structured static environments, with the ability to navigate narrow passages and generate clean, steerable paths without the need for heuristic tuning. The smooth output makes it directly

compatible with vehicle controllers that require continuous curvature input for actuation.

Future extensions will involve dynamic re-planning under moving obstacles, adaptive grid resolution, and integration with vehicle dynamics constraints for curvature, speed, and jerk limits. These enhancements will further bridge the gap between discrete path planning and real-time motion execution in modern ADAS platforms.

REFERENCES

1. E. W. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” Numerische Mathematik, vol. 1, 1959.
2. S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.
3. P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” IEEE Trans. Systems Science and Cybernetics, 1968.
4. H. Choset et al., *Principles of Robot Motion*, MIT Press, 2005.