Proceedings of CONF-MLA 2025 Symposium: Applied Artificial Intelligence Research
DOI: 10.54254/2755-2721/2025.BJ26887

Implementation of D Lite Algorithm for Dynamic Pathfinding in a Street Environment*

Boxi Zhao

Santa Clara University, Santa Clara, USA
BZhao2@scu.edu

Abstract. This project implements the D* Lite algorithm for dynamic pathfinding in a simulated urban street environment featuring realistic autonomous vehicle constraints. The AV, modeled as a 2×3 grid-aligned car, navigates forward through a narrow 10×100 grid with both static and probabilistically generated dynamic obstacles, such as pedestrians crossing at designated intervals. The simulation integrates a forward safety buffer of three grid cells, equivalent to the vehicle's length to ensure safety, which allows the AV to anticipate and react to potential threats before they intersect its path. The system computes planning and safety masks at each timestep, which enable real-time halting and re-routing behavior in response to environmental changes. While the safety radius improves collision avoidance, D* Lite remains a geometric planner without predictive capabilities. Rare collisions can still occur due to dynamic obstacle motion between sensing and actuation. Results of this research demonstrate that the algorithm supports low-latency decision-making and efficient re-planning under uncertainty. The project validates the algorithm's practical utility for simulating AV navigation in simplified yet dynamic scenarios.

Keywords: D* Lite algorithm, dynamic pathfinding, autonomous vehicles, collision avoidance, real-time re-planning

1. Introduction

This project implements the D* Lite algorithm to simulate realistic autonomous vehicle navigation in a dynamic street environment populated with both static and dynamic obstacles. The primary goal is to develop a robust path-finding solution that can adapt efficiently to environmental changes without requiring replanning from scratch at every time step. The simulation incorporates realistic driving constraints, such as restricted movement directions and vehicle dimensions, and models urban dynamics, including parked vehicles and randomly crossing pedestrians.

2. Literature review

Path planning in dynamic, partially known environments has been widely studied. Such problems have evolved from early heuristic searches to modern incremental and anytime replanning methods. The foundational A* algorithm introduced heuristics to guide search toward the goal, which significantly reduces node expansions compared to classical dynamic programming [1]. However, in dynamic environments where obstacles appear or costs change, A* must be rerun from scratch, which is computationally inefficient.

To address this, Lifelong Planning A* (LPA*) was introduced. LPA* reuses previous search results by maintaining both g-values and one-step lookahead rhs-values. This enables localized repairs when the map changes [2]. LPA* laid the foundation for D* Lite, which simplifies LPA* by reversing the search direction and retaining only the states whose values become inconsistent, thereby achieving fast re-planning suitable for real-time navigation [3].

The original D* algorithm [4] demonstrated the potential of real-time optimal path planning under changing costs, inspiring extensions such as Field D*, which interpolates over grid costs to generate smoother and more realistic paths [5]. Another major variant, Anytime Dynamic A*, combines anytime planning with dynamic replanning. This enables quick initial solutions that improve as time allows [6].

Surveys on incremental heuristic search in reference [7] clarify the conditions under which incremental approaches outperform full replanning, which emphasizes factors such as heuristic consistency and efficient priority queue updates. In the context of autonomous driving, recent work has highlighted the integration of global geometric planners like D* Lite with motion prediction layers to handle moving obstacles, ensuring safety while maintaining efficiency [8, 9].

This project builds on that body of research by implementing D* Lite in a constrained street environment with moving pedestrians, applying a safety buffer tailored to vehicle dimensions to balance collision avoidance and efficiency.

3. Problem formulation

The environment is a 10-row by 100-column grid representing a linear street section. The lower portion of the grid (rows 5 to 9) simulates the parked vehicle lane and is entirely prohibited to the AV. The upper portion (rows 0 to 4) comprises the drivable lanes. The AV is represented as a 2×3 block (2 rows high, 3 columns wide). The AV can only move forward, forward-left, or forward-right, representing realistic kinematic constraints that disallow reversing or lateral motion.

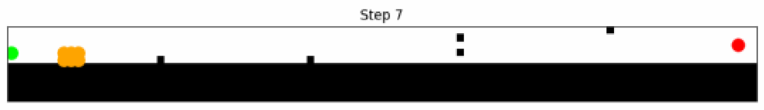

Figure 1. Grid structure

Dynamic obstacles are introduced in the form of vertical pedestrian movements, which simulate people crossing the road at designated crosswalks. These dynamic obstacles require the algorithm to support incremental replanning as the environment evolves during the simulation.

The AV (orange) starts from the green node and plans toward the red goal. Black squares represent static and dynamic obstacles, with the lower half fully blocked as parked car lanes.

4. Environment generation

The environment is procedurally generated to reflect both structural and temporal dynamics. A forward-only 8-connected graph is created for the top half of the grid to accommodate the AV's movement constraints. The static obstacles are implemented as the entire bottom half of the grid (rows 5 to 9), permanently blocked to the agent.

Dynamic obstacles simulate pedestrians crossing at four predetermined columns: 20, 40, 60, and 80. At each of these columns, at each of the 60 simulation timesteps, there is a 10% probability that a pedestrian begins to cross. Once triggered, a pedestrian moves downward through rows 0 to 4 over successive timesteps. Each step in their motion adds a new cell in the vertical column, resulting in a continuous movement effect simulating real pedestrian crossing behavior.

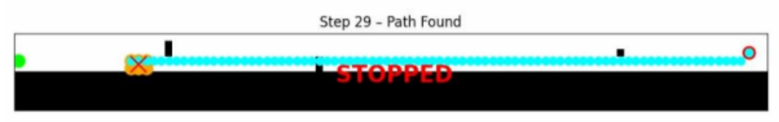

Figure 2. The AV initially halts when it detects a pedestrian within the safety radius (red STOPPED)

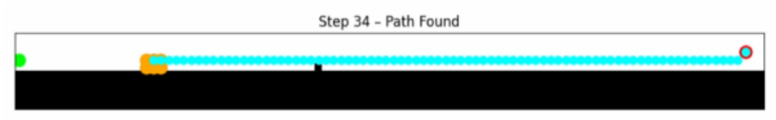

Figure 3. The path is updated as pedestrians appear at designated crossing points, requiring re-planning

These two figures illustrate the initialized simulation environment and how the vehicle interacts with dynamic obstacles.

5. Algorithm implementation

Path planning is handled by the D* Lite algorithm, a heuristic-based incremental search algorithm designed for environments with dynamic changes. The D* Lite algorithm evolves from a progression of pathfinding strategies, beginning with classical dynamic programming. Traditional dynamic programming precomputes optimal paths from all states to the goal, but it becomes computationally expensive for large environments and requires complete recomputation upon any environmental change.

This limitation is mitigated by A*, which introduces heuristics to direct search from start to goal, which reduces the number of expanded nodes. However, if new obstacles appear, A* must re-run entirely. LPA* improves on this by maintaining g and rhs values for each node by localized updates when the map changes. D* Lite simplifies LPA*by reversing the search direction and retain only the states whose values become inconsistent. This enables rapid re-planning in dynamic environments, making it possible for real-time applications like autonomous navigation.

In this implementation, each node in the planning graph represents the top-left corner of the AV's 2×3 grid footprint. Only the node that the full car block fits within bounds and does not collide with any obstacle at that timestep is valid. The AV is limited to move in three directions: forward, forward-left, and forward-right, represented as (0,1), (-1,1), and (1,1). These movements mirror the car's limited maneuverability and ensure physically plausible motion.

The algorithm uses a Manhattan distance heuristic to estimate cost-to-go values and prioritize node expansions. It maintains g-values, the actual cost to reach a state, and rhs-values, one-step lookahead estimates. A priority queue is used to expand nodes based on a composite key derived from these values and the heuristic. Instead of recalculating the entire path, the planner selectively updates only affected nodes and their successors when a new obstacle appears. This makes the algorithm efficient in environments with frequent changes, such as randomly appearing pedestrians.

This incremental approach enables the AV to adapt fluidly to changes in the environment, re-routing efficiently without pausing for full recomputation. The D* Lite algorithm thus serves as a powerful solution for path planning in dynamic, partially known settings.

6. Safety and motion planning

To replicate real-world AV safety behavior, the algorithm integrates a safety buffer of three grid cells in front of the AV. This buffer serves as a conservative safety margin to prevent the vehicle from advancing into areas that are too close to obstacles, even if those obstacles are not yet directly in the car's next immediate move. The choice of a three-cell buffer is directly informed by the AV's physical dimensions: the vehicle occupies a 2×3 grid footprint, with its length spanning three columns in the forward direction. By matching the buffer distance to the car's length, the AV effectively maintains a "car-length" of foresight. This reflects realistic stopping intuition and ensures that the vehicle has adequate space to react before a collision becomes unavoidable.

At each timestep, the simulation computes two types of masks. First, a planning mask identifies whether a move is immediately valid based on current obstacle positions. Second, a safety mask projects the vehicle's motion envelope forward by three cells, detecting any static or dynamic object that might soon enter the AV's path. This layered sensing mechanism allows the AV to anticipate and respond conservatively to pedestrians or other obstacles.

If the planned path is blocked by a newly appeared pedestrian or if any part of the front safety zone is obstructed, the AV halts and triggers re-planning. The algorithm ensures the vehicle does not proceed unless a clear and safe forward trajectory is available. This behavior models defensive driving, where a human driver would yield or stop even before an obstacle directly blocks the lane.

The safety buffer proves critical in minimizing risk in dynamic environments. It allows the AV to anticipate potential threats before they intersect the path of motion, functioning as an early warning zone. Smaller buffers, such as 1 or 2 cells, were tested but found insufficient in avoiding close encounters, especially under sudden pedestrian movement. Larger buffers, such as bigger than 4 cells, while safer, resulted in overly cautious behavior, frequent halts, and inefficient path execution. The three-cell configuration offered the best balance between safety and responsiveness.

That said, it is important to recognize that while the safety radius enhances reactivity, D Lite does not guarantee collision-free paths. It is a geometric planner that adapts to changes in obstacle configuration but lacks predictive capabilities. It does not account for the motion of dynamic obstacles over time, nor does it model the AV's velocity, inertia, or braking dynamics. Consequently, certain failure cases still occur. For instance, if a pedestrian spawns within the buffer zone after a move is committed, the AV may not stop in time. Such timing mismatches can result in unintended collisions despite the presence of a safety margin.

In simulation, rare failures were observed when pedestrians appeared during the narrow interval between planning and execution. These outcomes underscore the limitations of reactive-only planning in dynamic environments. Future work may explore velocity-aware planning, obstacle motion forecasting, or integration with time-parameterized motion planners that model the AV's physical dynamics more faithfully.

7. Results

The simulation produces an animated visualization in the form of a GIF, illustrating the agent's behavior across time as it navigates through the dynamic street environment. The AV successfully adapts to random pedestrian appearances, occasionally waiting and rerouting when necessary to maintain safety.

Quantitatively, the simulation tracks the total number of steps taken and execution time for the entire pathfinding session. These metrics demonstrate that the D* Lite algorithm efficiently handles dynamic replanning requirements, supporting low-latency decision-making even in environments with non-deterministic obstacle appearance.

8. Conclusion

This project validates the practicality of D* Lite for dynamic, real-world-inspired AV path planning. The simulation accurately models realistic driving constraints, including limited maneuverability, large vehicle size, and proactive safety precautions. The environment incorporates randomly appearing pedestrians with specific probabilistic behavior, and the algorithm proves capable of adapting without full re-computation.

Future extensions may include activating the departing car logic, incorporating more complex pedestrian models with trajectory prediction, increasing the number of crosswalks, or expanding the environment into more complex topologies (e.g., intersections, two-way streets). These would bring the simulation closer to real-world urban scenarios and further test the robustness of incremental planning under uncertainty.

References

[1] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, 4(2), 100–107. https: //doi.org/10.1109/TSSC.1968.300136
[2] Koenig, S., & Likhachev, M. (2004). Lifelong planning A*. Artificial Intelligence, 155(1–2), 93–146. https: //doi.org/10.1016/j.artint.2003.12.001
[3] Koenig, S., & Likhachev, M. (2002). D* Lite. In Proceedings of the AAAI Conference on Artificial Intelligence (pp. 476–483). AAAI Press.
[4] Stentz, A. (1994). The D* algorithm for real-time planning of optimal traverses (Technical Report CMU-RI-TR-94-37). Carnegie Mellon University, Robotics Institute.
[5] Ferguson, D., & Stentz, A. (2005). Field D*: An interpolation-based path planner and replanner. In Proceedings of the 12th International Symposium on Robotics Research (ISRR) (pp. 239–253). Springer.
[6] Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., & Thrun, S. (2008). Anytime dynamic A*: An anytime, replanning algorithm. Artificial Intelligence, 172(14), 1613–1643. https: //doi.org/10.1016/j.artint.2007.12.009
[7] Koenig, S. (2004). Incremental heuristic search in AI. AI Magazine, 25(2), 99–112. https: //doi.org/10.1609/aimag.v25i2.1776
[8] Xia, T., Li, H., & Liu, Y. (2024). A survey of autonomous vehicle behaviors: Trajectory planning and risk estimation. Sensors, 24(12), 4215. https: //doi.org/10.3390/s24124215
[9] Hamidaoui, M., et al. (2025). Survey of autonomous vehicles' collision avoidance techniques. Sensors, 25(3), 1021. https: //doi.org/10.3390/s25031021