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

Department of Mathematics and Computer Science, Santa Clara University, Santa Clara, USA
BZhao2@scu.edu

Abstract. This project uses the D* Lite algorithm to do dynamic pathfinding in a simulated urban street environment, while also considering the movement limits of an autonomous vehicle. In the simulation, the vehicle is represented as a 2×3 car that moves forward on a narrow 10×100 grid. The environment contains both fixed obstacles and moving ones, like pedestrians crossing the street from time to time. To improve safety, a three-cell buffer is placed in front of the vehicle, which is roughly equal to one car length. This helps the vehicle react earlier when something suddenly appears ahead. During each step, the system updates both the planning mask and the safety mask, so the vehicle can either stop or replan its path when necessary. The safety buffer does help lower the chance of collisions, but D* Lite is still only a geometric planning algorithm, so it cannot predict future obstacle movement. Because of that, collisions may still happen in rare cases when an obstacle moves into the path after the vehicle has already sensed the environment. Even so, the results show that the algorithm can respond quickly and handle path planning well in a changing environment. Overall, this project shows that D* Lite is effective for a simplified autonomous driving simulation.

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

1. Introduction

This project uses the D* Lite algorithm to simulate an autonomous vehicle moving through a street environment with both fixed and moving obstacles. The main goal is to build a pathfinding system that can adjust to changes in the environment without having to recalculate the entire path at every timestep. The simulation also takes into account some real-world limits, such as the size of the vehicle and its restricted movement. To make the environment feel more realistic, it includes common street situations like parked cars and pedestrians crossing the road at random times.

2. Literature review

Path planning in dynamic and partially known environments has been studied for a long time. Early research mainly focused on heuristic search methods, and later this gradually developed into incremental and anytime replanning approaches. One of the most well-known methods is A*, which uses heuristics to guide the search toward the goal. Compared with traditional dynamic programming methods, it reduces the number of nodes that need to be expanded [1]. Still, when obstacles or path

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.BJ26887

costs change in a dynamic environment, A* usually has to recompute the whole path from the beginning, which can be inefficient.

To improve this, Lifelong Planning A* (LPA*) was introduced. LPA* reuses information from previous searches by keeping both g-values and one-step-ahead rhs-values for each node. Because of this, it only needs to update the parts of the map affected by changes instead of starting over completely [2]. Based on the same idea, D* Lite was later proposed as a simpler version of LPA*. It reverses the search direction and only updates states whose values become inconsistent, which makes replanning faster and more suitable for real-time environments [3].

Earlier work on the original D* algorithm [4] showed that real-time optimal path planning is possible even when edge costs change. After that, several related methods were developed. For example, Field D* improves path smoothness by interpolating grid costs [5]. Another method, Anytime Dynamic A*, combines anytime planning with dynamic replanning, so it can find an initial solution quickly and then keep improving it over time [6].

A review of incremental heuristic search [7] explains when these methods are more effective than full replanning. Important factors include whether the heuristic is consistent and how efficiently the priority queue can be updated. In autonomous driving, more recent work has looked at combining global planners like D* Lite with motion prediction modules. This makes it easier to deal with moving obstacles while still trying to keep both safety and efficiency [8,9].

This project applies D* Lite to a constrained street environment with moving pedestrians. It also adds a safety buffer based on the size of the vehicle, so the system can better balance collision avoidance and path efficiency.

3. Problem formulation

The environment is modeled as a 10-row by 100-column grid representing a straight section of a street. The lower part of the grid (rows 5 to 9) represents lanes occupied by parked vehicle, and is not accessible to the AV. The upper portion (rows 0 to 4) defines 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 basic vehicle movement constraints. Reverse or purely lateral movements are not allowed.

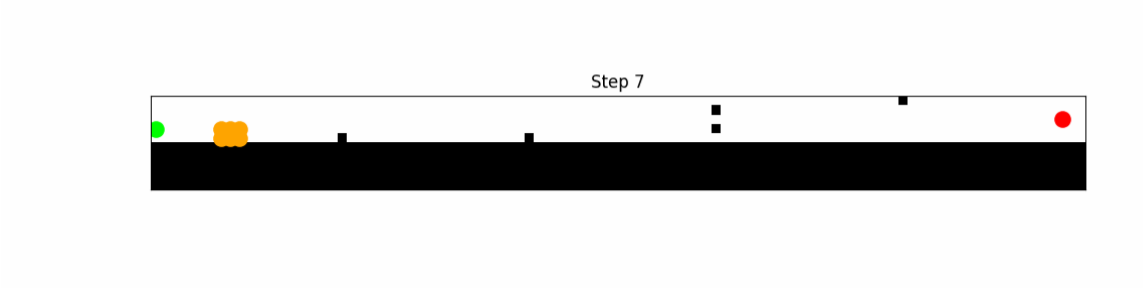

Figure 1. Grid structure

Dynamic obstacles are introduced as vertically moving pedestrians to simulate people crossing the street at fixed crosswalk locations. These moving obstacles require the algorithm to perform incremental replanning as the environment changes over time.

The AV (orange) starts from the green node and moves toward the red goal. Black squares represent both static and dynamic obstacles, and the lower half of the grid remains completely blocked at all time to simulate the lanes for parked vehicles.

DOI: 10.54254/2755-2721/2025.BJ26887

4. Environment generation

The environment is generated procedurally to showcase both the structural layout and the time-based changes. An 8-connected graph designed to move only forward was constructed for the upper half of the grid to align with the movement constraints of autonomous vehicles. Static obstacles are the entire bottom half of the grid (rows 5 to 9), which is permanently blocked.

In this simulation, the dynamic obstacles are pedestrians crossing in four fixed columns: 20, 40, 60, and 80. At every timestep, each column has a 10% chance of spawning a pedestrian. Once activated, the pedestrian moves downward from row 0 to row 4 over successive timesteps. This movement is modeled by adding one occupied cell at a time in the same column, which creates the effect of continuous vertical pedestrian motion.



Figure 2. The AV stopped when it detects a pedestrian within the safety radius

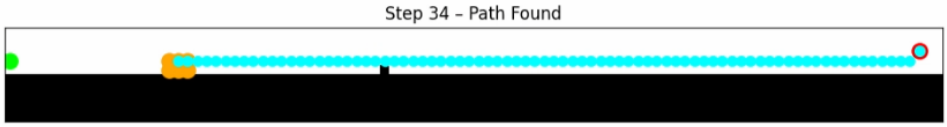

Figure 3. The path is updated whenever pedestrians appear at designated crossing points, which forces the vehicle to replan

These two figures show the initial simulation setup and how the vehicle reacts to dynamic obstacles.

5. Algorithm implementation

The D* Lite algorithm is used for path planning in this system. It is an incremental search algorithm that works well when the environment keeps changing. Earlier methods like traditional dynamic programming could find the best path, but they were expensive to run, especially in large environments. They also had to recompute everything whenever the map changed.

A* improved this by using a heuristic to guide the search, which made it faster. But if a new obstacle appeared, it still had to run again from the beginning. LPA* made this better by updating only the parts of the path affected by changes. D* Lite builds on that idea and makes replanning more efficient. Instead of restarting the whole search, it only updates nodes that are affected, so it works much better in dynamic environments.

In this project, each node represents the top-left corner of the AV's 2×3 footprint. A node is valid only if the whole vehicle stays inside the map and does not collide with obstacles. The vehicle can

DOI: 10.54254/2755-2721/2025.BJ26887

only move straight forward, forward-left, or forward-right, which matches its limited turning ability and makes the movement more realistic.

The algorithm uses Manhattan distance as the heuristic. It also keeps track of g-values and rhs-values to decide which nodes need to be updated. A priority queue is used to choose the next node to expand. When obstacles change, the planner only updates the related nodes instead of recomputing the full path. This makes it much more suitable for situations where pedestrians or other obstacles may appear unexpectedly.

Overall, D* Lite allows the vehicle to react to changes without starting over every time. That makes it a practical choice for dynamic path planning in this simulation.

6. Safety and motion planning

To make the AV behave more safely in a realistic way, the algorithm uses a three-cell safety buffer in front of the vehicle. This gives the car some extra space from obstacles, even if they are not in the very next move. The buffer size comes from the size of the vehicle itself. Since the AV takes up a 2×3 grid and extends three cells forward, the system uses about one car-length of space ahead. This gives the vehicle a better chance to react before a crash becomes unavoidable.

At each step, the simulation creates two masks. The planning mask checks whether the next move is possible based on where obstacles are right now. The safety mask looks a little farther ahead by extending the vehicle's footprint three cells forward, so it can catch objects that may soon move into the path. Working together, these two masks help the vehicle deal with pedestrians and other moving obstacles more carefully.

When a new pedestrian blocks the planned path, or when part of the safety area in front of the vehicle is occupied, the vehicle stops and replans. It only starts moving again when the space ahead is clear. This is similar to defensive driving, where a driver slows down or stops before the situation becomes unsafe.

The safety buffer is important for lowering risk in a changing environment. It gives the vehicle more warning and a little more time to react. Smaller buffers, like one or two cells, were also tested, but they often caused very close calls, especially when pedestrians appeared suddenly. Larger buffers, such as four cells or more, made the vehicle safer, but also made it overly cautious. It stopped too often, and the path became less efficient. From the simulation results, a three-cell buffer gave the best balance between safety and efficiency.

Even with that, the safety buffer cannot avoid every collision. D* Lite is still just a geometric planner. It can respond when obstacles change, but it cannot predict how they will move next. It also does not account for things like speed, inertia, or braking distance. Because of that, some failure cases still happen. For example, if a pedestrian steps into the buffer after the vehicle has already started moving, the AV may not be able to stop in time. So even with a safety margin, collisions can still happen because of timing.

In the simulation, these failures were rare, but they could still be seen. Most of them happened in the short gap between planning and execution. This shows one weakness of purely reactive planning in dynamic environments. A possible next step would be to add motion prediction, velocity-aware planning, or time-based motion planners that better match how real vehicles actually move.

7. Results

The simulation generates an animated GIF that shows how the agent's behavior evolves over time in a dynamic street environment. The AV is capable of adapting to randomly appearing pedestrians and pausing or replanning its route to ensure safe operation.

DOI: 10.54254/2755-2721/2025.BJ26887

In addition to visualization, the simulation records the total number of steps taken and the execution time for each pathfinding run. These data show that the D* Lite algorithm can efficiently handle dynamic replanning. The results also indicate that the system maintains low decision latency even when obstacles appear unpredictably.

8. Conclusion

This project shows that D* Lite can work well for dynamic path planning in a simulated street setting with more realistic vehicle limits. The simulation considers several real driving constraints, such as limited turning ability, the larger size of the vehicle, and the need to keep a forward safety buffer. The environment also includes pedestrians that appear randomly based on set probabilities, which makes the situation less predictable and allows the planner to respond to changes without having to recalculate everything from the beginning.

There is still a lot of aspects to improve in the system. For example, future work could include adding departure vehicle behavior, using more advanced pedestrian models that can predict movement paths, placing more crosswalks in the environment, and building more complicated road layouts like intersections or two-way streets. Changes like these would make the simulation more similar to real urban traffic and would also give more chances to test how well incremental planning works in uncertain situations.

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