CSci 4511 Writing 4

Baanee Singh & Masooma Sayeda

April 25th 2025

Introduction to Robot Path Planning

Robot Path Planning is a crucial domain in robotics and artificial intelligence (AI), tasked with determining optimal routes for autonomous systems to reach a destination without collisions. Its applications span a wide range of fields, from industrial automation to self-driving vehicles, all of which rely heavily on real-time decision-making and spatial reasoning. Effective path planning must account for both static and dynamic environments. In static environments, all obstacles are known beforehand and remain unchanged, whereas dynamic environments involve moving or unpredictable obstacles that require real-time updates and re-planning.

The primary goals of robot path planning include minimizing travel distance and time, ensuring collision-free navigation, and optimizing computational efficiency, particularly in scenarios demanding real-time performance. Among the various algorithms developed, the A* algorithm is widely regarded as one of the most efficient and optimal heuristic-based search methods.

A* Algorithm in Static Environments

Gigras and Gupta (2012)[1] emphasize the effectiveness of A* in structured, static environments, where complete map information is available and path recalculations are unnecessary. The algorithm is optimal, finding the shortest path under certain conditions and complete, guaranteeing that a solution will be found if one exists, assuming an admissible heuristic. In grid-based indoor environments, commonly used heuristics such as Euclidean or Manhattan distance can significantly influence both time and path quality.

Limitations and Enhancements of A*

Despite its strengths, A* becomes less suitable in dynamic or large-scale environments, where changes require complete re-planning. This makes the algorithm computationally expensive and potentially inefficient for real-time applications. To address these limitations, researchers have explored more adaptive strategies. For instance, Gilbert and Johnson (1985)[2] introduced a method for planning robot movement that helps the robot avoid obstacles while still taking the best possible path. Instead of handling obstacle avoidance separately, they make it part of the main planning process by using distance functions. These functions continuously measure the robot's proximity to obstacles, allowing smoother and more efficient motion. Though originally applied to robotic arms, their method applies to mobile robots navigating indoor spaces. When integrated with algorithms like A* or D*, their approach enhances both path safety and realism by enabling smoother curves and avoiding abrupt movements.

Building on the foundational A* algorithm, Fu et al. (2018)[3] proposed an enhanced A* algorithm tailored for industrial robots, incorporating dynamic weighting in heuristics to improve success rates and reduce path length in cluttered factory layouts. Their method introduces a pre-processing stage that checks for a direct, collision-free path to the goal before searching nearby nodes and a post-processing stage that removes unnecessary points to shorten the path. Their approach demonstrated a higher success rate and shorter paths compared to traditional A*, making it suitable for robotics in constrained industrial settings. However, these improvements come at the cost of increased complexity, leading to a trade-off between accuracy and speed, especially relevant for real-time or embedded systems, where approximate solutions may be preferable.

An additional consideration discussed in "A Guide to Heuristic-based Path Planning"[4] is how various search algorithms can be adapted or extended using domain-specific heuristics or memory-efficient structures. For A*, this includes weighted A* variants and memory-bounded search strategies, which aim to maintain near-optimal paths while significantly reducing runtime and space requirements. These adaptations make A* more viable for larger environments or devices with limited onboard resources.

Introduction to D* Lite Algorithm

Dynamic environments further complicate path planning by introducing challenges such as moving obstacles, incomplete maps, and unexpected changes. Traditional A* algorithms struggle in such settings due to the need for complete replanning. Addressing this gap, D* Lite was developed as a re-planning algorithm designed to adapt to changes efficiently. Unlike A*, D* Lite retains previous search information and updates only the affected areas when the environment changes. This feature makes it particularly well-suited for real-time navigation in partially known or evolving environments.

Setiawan et al. (2014)[5] compared the performance of A* and D* Lite in both simulation and real-world tests using a mobile robot (AGV). Their findings revealed that while A* could be faster in obstacle-free environments due to its simplicity, D* Lite outperformed it when dealing with dynamic obstacles. The ability to update paths incrementally allowed for smoother adjustments and improved navigation efficiency in larger or more complex environments.

This is echoed in the experimental work by Pandu Sandi Pratama[6], who evaluated both algorithms on a differential-drive AGV platform. Their results emphasized the flexibility of D* Lite in responding to changes and suggested it as the preferred choice for applications involving frequent re-mapping or unpredictable elements, such as delivery or maintenance robots.

Foundations, Enhancements, and Variations of D* Lite

Further validating the advantages of D* Lite, Bagad et al. (2024)[7] demonstrated that the algorithm significantly reduces the computation time by updating only the nodes affected by the changes rather than recalculating the entire path. Their experiments on grid maps ranging from 100×100 to 500×500 cells showed that D* Lite maintained high success rates and delivered near-optimal paths with faster replanning—an essential quality for real-time applications such as warehouse robotics and indoor mobile navigation.

Koenig and Likhachev (2002)[8] provided a solid theoretical foundation for D* Lite, emphasizing its simplified structure compared to its predecessor, Focused D*. The algorithm's clean design avoids layered conditions and uses consistent rules for tie-breaking, making it easier to implement and expand. They also highlighted that D* Lite can accommodate non-admissible heuristics and customizable rules to enhance performance. Their analysis showed that each node is expanded at most twice after a change, proving D* Lite's computational efficiency and robustness—qualities that make it an excellent choice for future research in real-time robotic planning.

Several enhancements to the standard D* algorithm have also been proposed to improve its practical performance. For instance, Guo et al.(2009)[9] extended the direction resolution in D* from 4 or 8 to 16 possible movement directions. This refinement led to smoother and more efficient paths by reducing unnecessary turns and shortening routes—advantages that were confirmed through experiments using the WiRobotX80 mobile robot. The results highlighted the importance of fine-grained direction control in improving motion efficiency and path quality in real-world navigation scenarios.

Similarly, Dakulović and Petrović (2011)[10] introduced the two-way D* (TWD*) algorithm, specifically designed for navigating indoor environments such as buildings. By planning from both the start and the goal simultaneously and incorporating straight-line segments, TWD* generated smoother, more natural paths. The inclusion of safety margins around obstacles further improved collision avoidance. Tested in both simulations and real-world settings, TWD* demonstrated its ability to quickly adapt to changes while maintaining high path efficiency, making it a practical choice for service and delivery robots in indoor settings.

An article by Deshpande et al.[11], investigated enhancements to D* Lite by integrating cost prediction models. Their implementation used prior knowledge of terrain types and incorporated weighted cost adjustments, leading to better energy efficiency for wheeled robots operating in industrial areas. Their findings further support the use of D* Lite in energy-sensitive applications and reinforce the value of combining path-planning algorithms with environmental semantics.

Conclusion and Project Relevance

Heuristic algorithms like A* and D* Lite are central to robot path planning. A* is effective in static, predictable settings, whereas D* Lite thrives in dynamic, partially known environments. Variants and improvements continue enhancing efficiency, success rates, and applicability in real-world scenarios. Recent studies have combined simple reactive navigation with Q-learning to help robots move more effectively in unknown areas. This mix helps the robot react quickly while also learning better paths over time. It works in both 2D and 3D spaces, like for ground robots and drones, and has been shown in tests to handle both still and moving obstacles well. [12].

Our project leverages both algorithms to develop and evaluate efficient path-planning solutions for indoor robot navigation within a 2D grid. By simulating real-world challenges, we aim to highlight trade-offs in performance, speed, and obstacle avoidance, reinforcing the broader importance of path planning in autonomous systems.

References

[1] Y. Gigras and K. Gupta, "Artificial intelligence in robot path planning," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 2, pp. 471–474, 2012.

[2] E. Gilbert and D. Johnson, "Distance functions and their application to robot path planning in the presence of obstacles," *IEEE Journal on Robotics and Automation*, vol. 1, no. 1, pp. 21–30, 1985.

[3] B. Fu *et al.*, "An improved a* algorithm for the industrial robot path planning with high success rate and short length," *Robotics and Autonomous Systems*, vol. 106, pp. 26–37, 2018.

[4] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Proceedings of the ICAPS 2005 Workshop on Planning under Uncertainty for Autonomous Systems*, (Monterey, California, USA), 2005.

[5] Y. Setiawan, P. Pratama, S. Jeong, V. Duy, and S. Kim, "Experimental comparison of a* and d* lite path planning algorithms for differential drive automated guided vehicle," in *AETA 2013: Recent Advances in Electrical Engineering and Related Sciences* (I. Zelinka, V. Duy, and J. Cha, eds.), vol. 282 of *Lecture Notes in Electrical Engineering*, pp. 551–558, Springer, 2014.

[6] P. Pratama, *Experimental Comparison of A* and D* Lite Path Planning Algorithms for Differential Drive Automated Guided Vehicle*, pp. 555–564. Springer, 01 2013.

[7] Bagad, Dahatonde, Gagare, Athare, and Ghule, "Optimizing pathfinding in dynamic environments: A comparative study of a* and d-lite algorithms," in *2024 IEEE Pune Section International Conference (PuneCon)*, (Pune, India), pp. 1–11, 2024.

[8] S. Koenig and M. Likhachev, "D* lite," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 476–483, AAAI Press, 2002.

[9] J. Guo, L. Liu, Q. Liu, and Y. Qu, "An improvement of d* algorithm for mobile robot path planning in partial unknown environment," in *2009 Second International Conference on Intelligent Computation Technology and Automation*, (Changsha, China), pp. 394–397, 2009.

[10] M. Dakulović and I. Petrović, "Two-way d* algorithm for path planning and replanning," *Robotics and Autonomous Systems*, vol. 59, no. 5, pp. 329–342, 2011. Special Issue ECMR 2009.

[11] S. Deshpande, A. K. Kashyap, and B. K. Patle, "A review on path planning ai techniques for mobile robots," *Robotic Systems and Applications*, vol. 3, pp. 27–46, jun 2023.

[12] J. Zhang, "Ai based algorithms of path planning, navigation and control for mobile ground robots and uavs," 2021.

Contributions: Masooma Sayeda (all parts expanded and written, found bibliographic references, wrote citations in overleaf, proofread overleaf), Baanee Singh (all parts expanded and written, started ideas and outline, editions, found bibliographic references, conversion to overleaf, proofread overleaf)

Note: Used Grammarly and Overleaf Built in AI for grammar purposes and sentence structure.