Electrician : Jurnal Rekayasa dan Teknologi Elektro
ELECTRICIAN
Vol. 20 No. 01, e-ISSN:2549-3442 http://dx.doi.org/10.23960/elc.v20.n1.3002

Adaptive Path Planning Using Depth-First Search in Environments with Dynamic Obstacle

Ni Putu Devira Ayu Martini[1*], Muchamad Oktaviandri[2], Elvi Armadani[3]

[1]Electrical Engineering, Faculty of Engineering, Universitas Pembangunan Nasional Veteran Jakarta
[2]Mechanical Engineering, Faculty of Engineering, Universitas Pembangunan Nasional Veteran Jakarta
[3]Industrial Engineering, Faculty of Engineering, Universitas Pembangunan Nasional Veteran Jakarta

Received: 01-12-2025
Accepted: 27-01-2026

Keywords:
Autonomous Navigation;
Depth-First Search;
Dynamic Obstacle;
Path Planning.

Corespondent Email:
deviraayum@upnvj.ac.id

Abstract. *Depth-First Search (DFS) is one of the classical algorithms used for robotic path planning due to its simplicity and deterministic exploration strategy. However, its performance can degrade significantly in environments where obstacles move or appear unexpectedly. This study analyzes the behavior of the DFS algorithm under both static and dynamic obstacle conditions using a grid-based simulation. Three primary performance indicators were evaluated: total path length, turning complexity, and overall path efficiency. In static environments, where all obstacle positions were known beforehand, DFS demonstrated stable and efficient navigation, achieving approximately 117% shorter paths, 42% higher efficiency, and 133% fewer turning maneuvers compared to scenarios with dynamic obstacles. When unexpected obstacles were introduced, the robot frequently performed backtracking and route replanning, which increased the total travel distance and the number of directional changes, ultimately lowering navigation efficiency. Despite these challenges, the DFS algorithm was still capable of reaching the goal after multiple replanning steps. These findings highlight both the robustness and limitations of DFS and suggest that integrating adaptive sensing or hybrid algorithms could improve performance in unpredictable, dynamic environments.*

1. INTRODUCTION

Path planning plays a central role in autonomous robotic navigation because it determines how a robot selects a safe and efficient route from its starting position to a specified destination [1], [2]. Classical graph-search algorithms—most notably Dijkstra's algorithm, A*, and Depth-First Search (DFS)—have provided the foundation for many navigation systems thanks to their well-understood properties and predictable performance in structured environments [3], [4]. These methods, however, were originally developed with the assumption of a static environment, where all obstacles are known and remain fixed. In practical field deployments, this assumption is rarely valid. Robots operating in warehouses, hospitals, or outdoor public spaces must often cope with dynamic conditions such as moving pedestrians, shifting cargo, or objects that suddenly appear in the workspace [5], [6]. Such uncertainties introduce significant challenges because a robot must be capable of updating its planned path in real time without incurring excessive computational overhead [7].

To address these challenges, recent studies have proposed a variety of strategies for dynamic path planning. Incremental graph methods like D* Lite allow rapid re-planning when local changes occur, while reinforcement-learning approaches and motion-prediction techniques enable a robot to anticipate and avoid moving obstacles [8]–[10]. Despite these advances, classical algorithms such as DFS remain relevant as baseline methods. Their deterministic behavior, conceptual simplicity, and ability to exhaustively explore a search

space make them valuable both for education and for benchmarking more sophisticated planners [11], [12]. At the same time, it is widely recognized that DFS was not originally designed to handle environments where the map changes during execution. Evaluating its performance under these conditions is therefore critical to understanding its inherent limitations and to identifying simple adaptations—such as real-time backtracking or incremental re-routing—that can extend its practical utility [13]–[15].

Building on this perspective, the present study systematically examines DFS performance in a grid-based simulation under two distinct scenarios. The first scenario represents a static map with only known obstacles, while the second introduces dynamic obstacles that appear unexpectedly as the robot moves. Quantitative metrics—including path length, number of turns, and overall route efficiency—are used to measure how these environmental changes affect the effectiveness of DFS. By comparing outcomes across these scenarios, the analysis provides insight into how a straightforward search algorithm behaves when confronted with real-world uncertainty and offers guidance for augmenting classical planners to meet the demands of adaptive navigation in autonomous robotic systems.

2. LITERATURE REVIEW

Autonomous navigation has become one of the core challenges in robotics, as it enables robots to move safely and efficiently in environments that may change over time. The primary difficulty lies in allowing robots to perceive their surroundings, plan feasible trajectories, and make decisions under uncertainty while avoiding collisions [16], [17]. Path planning in static environments is relatively straightforward, but in real-world conditions—such as warehouses, factories, or crowded spaces—robots must deal with dynamic and unpredictable obstacles that continuously modify the feasible navigation space [18], [19]. Therefore, achieving a balance between computational efficiency, path optimality, and adaptability remains a central focus of recent research [20].

Traditional path planning algorithms such as Dijkstra's, A*, and Dynamic Programming (DP) have been the foundation of robotic navigation for decades. These algorithms provide deterministic and optimal solutions when the environment is fully known [21], [22]. The A* algorithm, for example, uses a heuristic cost function to improve efficiency compared to the exhaustive search of Dijkstra's algorithm. Similarly, DP guarantees an optimal policy by decomposing global problems into smaller subproblems [23]. However, these methods assume static maps and tend to perform poorly when obstacles move or appear unexpectedly [24]. To address this, incremental algorithms such as D* and D* Lite were developed to replan paths dynamically with reduced computational overhead [25].

At the same time, sampling-based algorithms—including Rapidly-exploring Random Tree (RRT) and Probabilistic Roadmap (PRM)—have been used for high-dimensional and partially known environments due to their scalability and probabilistic completeness [26]. Yet, they often generate non-smooth or suboptimal trajectories, especially under time-varying conditions. To improve local responsiveness, hybrid approaches that combine global planners (e.g., A*, D* Lite) with local obstacle avoidance methods such as the Dynamic Window Approach (DWA) and Artificial Potential Field (APF) have been proposed [27], [28]. These frameworks enable robots to avoid dynamic obstacles in real time while maintaining smooth global trajectories.

Recent studies have integrated machine learning and optimization techniques to further enhance adaptability. Deep reinforcement learning (DRL) has been applied to navigation tasks to allow robots to learn obstacle-avoidance behaviors autonomously in dynamic environments [29]. Optimization-based algorithms such as Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) have also been used to refine path smoothness and convergence under motion uncertainty [30], [31]. In addition, new frameworks like Conflict-Based Search (CBS) combined with D* Lite [32] and the Strategy-based Dynamic Object Velocity Space (S-DOVS) [33] explicitly model obstacle motion to support long-range, safe navigation in dynamically changing spaces.

Despite the growing popularity of adaptive and learning-based algorithms, Depth-

First Search (DFS) remains an important baseline for studying fundamental pathfinding behavior. Its deterministic structure and low computational cost make it suitable for analyzing algorithmic robustness and efficiency under varying map conditions [34]. Previous research indicates that while DFS performs well in static environments, it struggles to adapt when faced with changing obstacles due to frequent backtracking and lack of predictive mechanisms [35]. However, these studies often evaluate DFS only under semi-dynamic or static scenarios, without fully considering the temporal behavior of obstacles.

In contrast, the present research specifically focuses on DFS performance under obstacle dynamics, where obstacles not only appear or disappear but also move unpredictably during navigation. This represents a more realistic and complex challenge compared to previous studies, as obstacle motion directly influences path length, turning complexity, and overall efficiency. By analyzing DFS behavior in such dynamically changing environments, this study aims to provide new insights into the robustness and limitations of classical search algorithms, and to identify potential pathways for improving adaptive navigation through hybrid or sensing-enhanced strategies.

3. RESEARCH METHODS

The method used in this study is illustrated in Figure 1, which presents the workflow of a navigation system based on the Depth-First Search (DFS) algorithm in a dynamic environment. The process begins with the initialization of the start position, goal position, and a map containing static obstacles. The robot then moves along the path points generated by the DFS algorithm while continuously monitoring for dynamic obstacles along its route. When a new obstacle is detected, the system performs *replanning* to find an alternative path until it realigns with the original trajectory. This process iteratively continues until the robot successfully reaches the goal position with an optimal path that adapts to environmental changes.

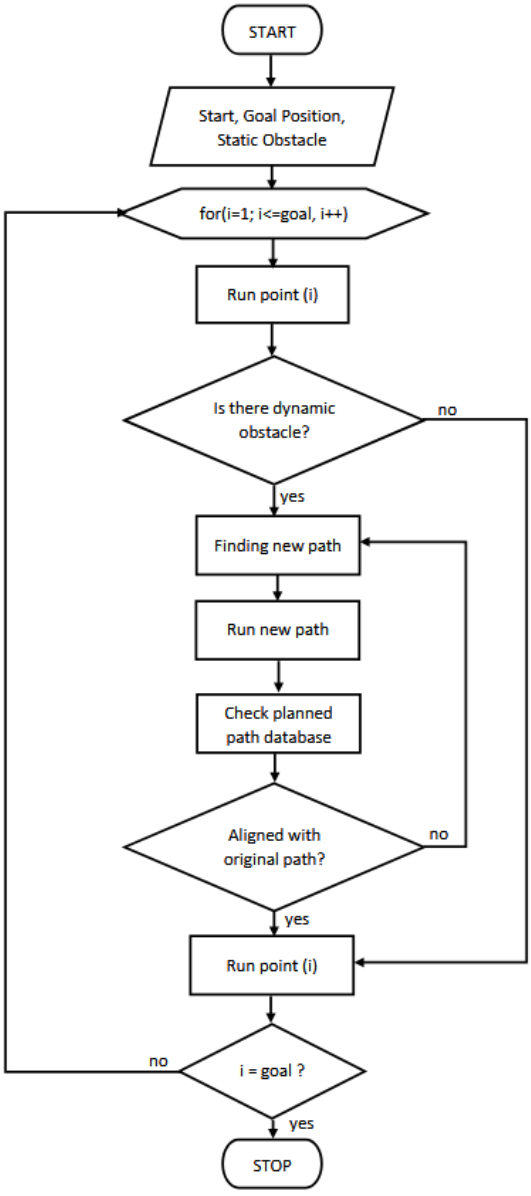


Figure 1. Flowchart of Navigation System

3.1. Depth-First Search

DFS operates by moving as deeply as possible along one branch before backtracking to explore alternative paths. Starting from an initial node, the algorithm systematically visits adjacent nodes, marking each as *visited* to prevent revisiting or creating loops within the search process. In the illustrated traversal as shown in Figure 2, DFS begins from the source node and proceeds recursively to its neighboring nodes, prioritizing depth over breadth. Once a node has no unvisited neighbors, the algorithm backtracks to the most recent branching point to continue exploring remaining unexplored edges. This depth-

oriented exploration results in a traversal pattern that closely follows the structure of a spanning tree, as shown in the diagram.

The process effectively demonstrates how DFS constructs a search tree that represents the order of node discovery and backtracking. Such a traversal strategy is particularly advantageous in applications like pathfinding, connectivity testing, and maze solving, where exploring one possible route to completion before trying alternatives provides computational simplicity and deterministic outcomes. The example in the figure also highlights how DFS systematically records both *tree edges* (paths of discovery) and *back edges* (links to previously visited nodes), forming the foundation for more advanced algorithms in robotic navigation, topological sorting, and cycle detection.

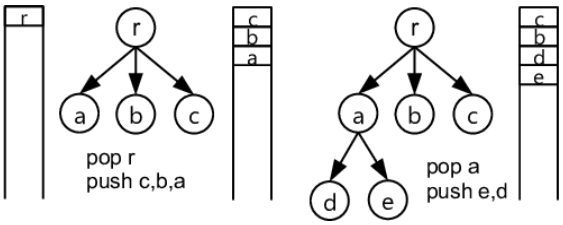


Figure 2**.** Diagram Depth First Search Transversal [16]

3.2. Dynamic Environment

The navigation experiments were conducted on a two-dimensional 7×7 times 7×7 grid where each cell represents a unit workspace for the mobile robot. Obstacles within this environment were classified into two types—static and dynamic—according to their movement characteristics and the availability of prior information.

Static obstacles refer to fixed physical structures whose positions are known before the path-planning process begins. Typical examples include walls, cabinets, or other immovable objects that permanently occupy specific coordinates in the map. Because their locations are predetermined, these obstacles can be incorporated into the initial occupancy grid, allowing the Depth-First Search (DFS) algorithm to avoid them from the start.

Dynamic obstacles, by contrast, represent entities whose presence and positions are not known in advance. They may appear or move unpredictably while the robot is navigating. Examples include humans walking through a corridor, pets crossing a room, or objects that are accidentally moved into the robot's path. Such obstacles are introduced into the grid during execution, forcing the robot to sense the blockage in real time and trigger a replanning procedure.

This experimental setup captures the contrast between predictable, static barriers and unexpected, moving elements that characterize real-world environments, enabling evaluation of how DFS handles sudden changes and maintains a viable route toward the goal.

3.3. Evaluation Metrics

The performance of the Depth-First Search (DFS) path planner was assessed using three quantitative indicators: path length**,** path complexity, and path efficiency.

Path length (L_p) describes the total distance traveled from the start node S to the goal node G as shown in Equation 1. Because the workspace is a uniform grid and movement is limited to the four cardinal directions, the distance between adjacent cells is one unit. Thus the total length is the number of consecutive cells visited along the final route,

$$L_p = \sum_{i=1}^{N-1} d(P_i, P_{i+1}) \quad \cdots (1)$$

Where P_i is the i-th cell in the final path and d is the Manhattan distance (equal to 1 for adjacent cells).

Path complexity (C_t) measures the number of heading changes, which reflects the maneuvering effort required by a mobile robot as shown in Equation 2. A turn is counted whenever the direction between two consecutive segments differs from the previous segment,

$$C_t = \sum_{i=2}^{N-1} \delta(\theta_i - \theta_{i-1}) \quad \cdots (2)$$

where θ_i is the heading of segment $P_{i-1} \rightarrow P_i$ and $\delta(x) = 1$ if $x \neq 0$ (direction change) and 0 otherwise.

Path efficiency (E_p) expresses how close the obtained route is to the theoretical optimum as

shown in Equation 3. Let L_{min} be the shortest possible distance between S and G determined, for example, by a breadth-first search on the same grid. The efficiency is defined as:

$$E_p = \frac{L_{min}}{L_p} \times 100\% \quad \cdots (3)$$

4. DATA AND ANALYSIS

The path planning shown in Figure 3 uses Depth-First Search (DFS) on a 7×7 grid without obstacles. The start point **S** is at coordinate (0,0) and the goal **G** at (6,2), marked in green. With no barriers to avoid, DFS moves through the grid in its typical deep-search pattern, visiting almost every cell. The numbered steps from 0 to 48 give a path length of 49 points and 12 direction changes. The long, looping route reflects how DFS explores thoroughly before finally reaching the goal.

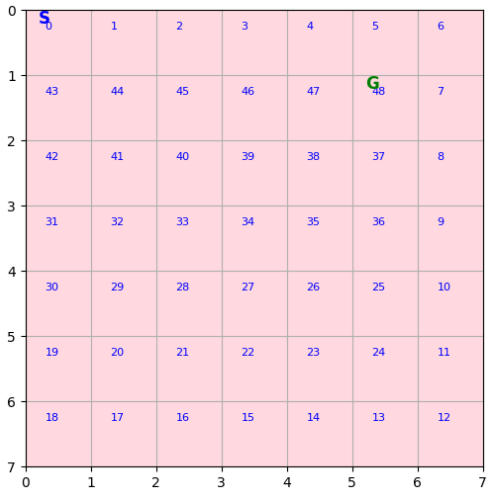
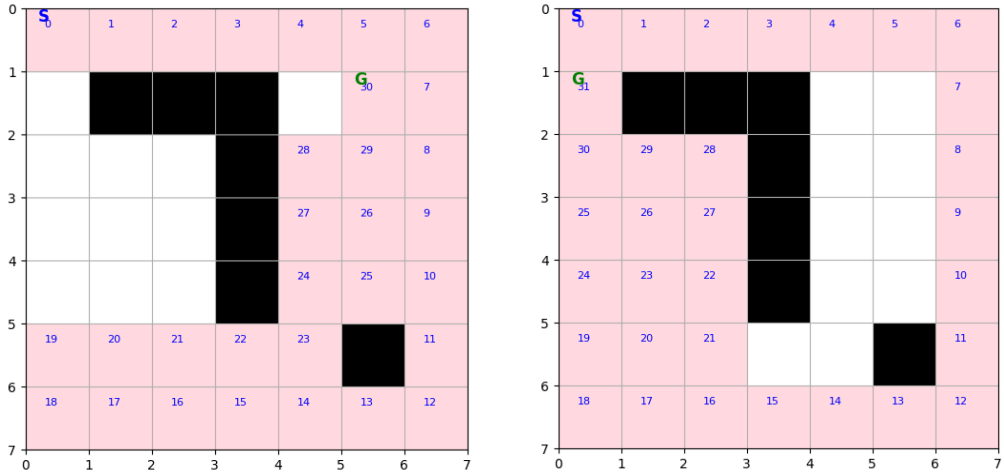


Figure 3. Navigation Environment



(a) (b)

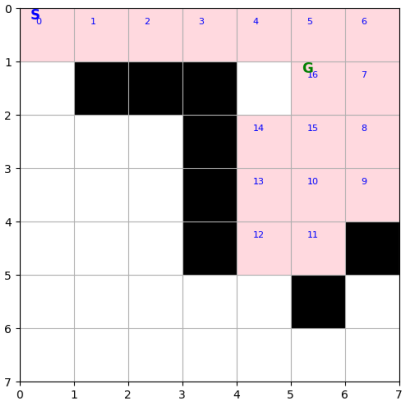
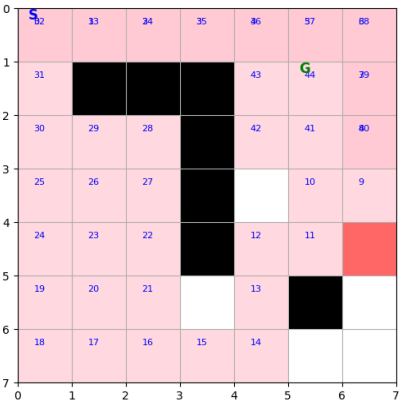
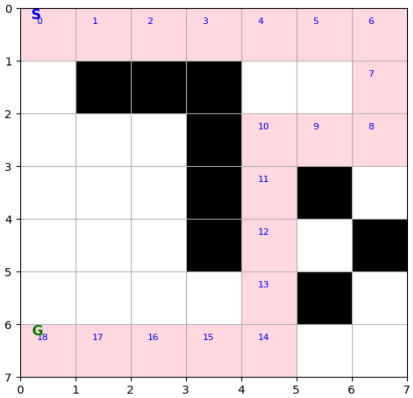
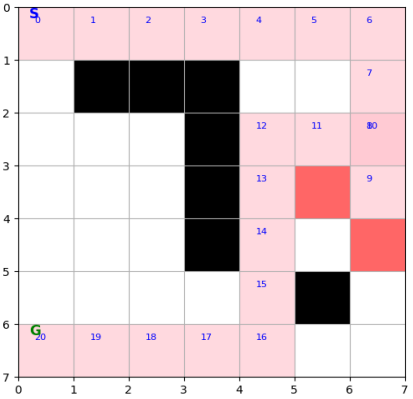
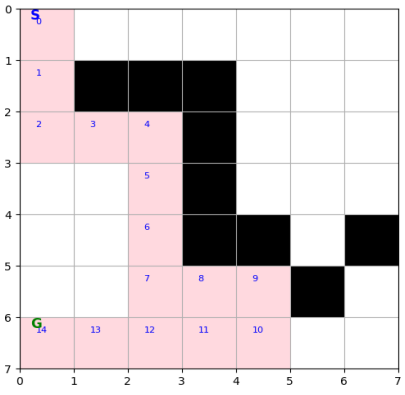
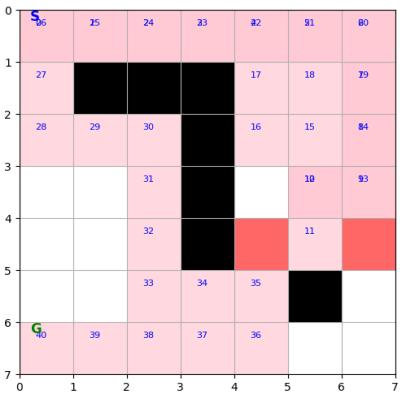
(a) Goal Point in (5, 1) (b) Goal Point in (0, 1)

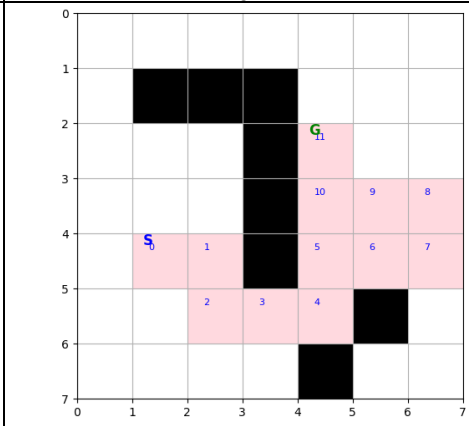
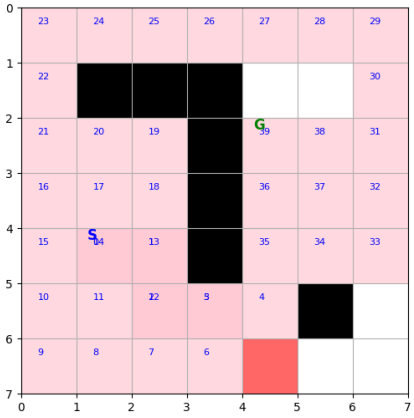
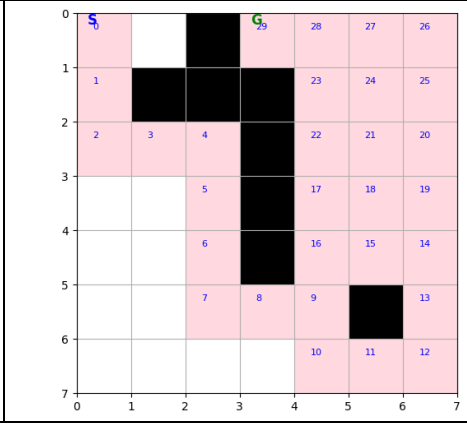
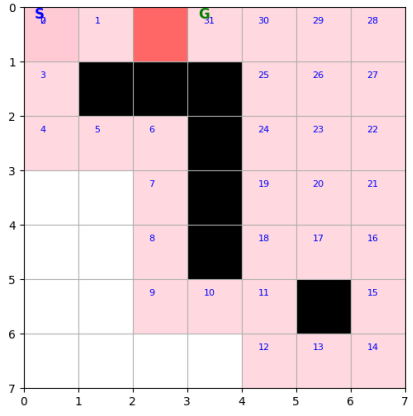
Figure 4. Path Planning with Static Obstacle

The Figure 4 presents two grid-based simulations of Depth-First Search (DFS) path planning in environments containing static obstacles. Both grids measure 7 × 7 and display black squares as fixed barriers, with the start point **S** located at the top-left corner (0,0). Figure (a) sets the goal at coordinate (5,1). DFS explores one branch as deeply as possible before backtracking, which leads to a winding route around the obstacles. The final path covers 31 cells and requires 11 direction changes, reflecting how DFS systematically explores until the goal is reached, even when detours are necessary. Figure (b) moves the goal to (0,1), creating a different search pattern. Despite the goal's proximity to the start, the algorithm still traverses a large portion of the grid because it follows depth-first exploration before eventually backtracking toward the target. This run yields a 32-cell path, also with 11 turns, and a calculated path efficiency of 6.25 %, showing how DFS can produce long, low-efficiency routes when early branches lead away from the goal. The two subfigures illustrate a key property of classical DFS: it does not optimize for shortest distance but instead pursues each path to its limit before retreating, which can result in lengthy, complex

trajectories when static obstacles block more direct options. The Table 1 summarizes five trials of path planning on the same 7×7 grid using a Depth-First Search (DFS) approach, comparing environments with only static obstacles and those with both static and dynamic obstacles. Each row represents one trial. In the static-only column, the DFS algorithm follows a single pre-planned route from start to goal without interruption, giving path lengths between roughly 12 and 29 cells.

Table 1. Variation Coordinate of Dynamic Obstacle in Map

Exp	Path without Dynamic Obstacle	Path with Dynamic Obstacle
1		
2		
3		

Exp	Path without Dynamic Obstacle	Path with Dynamic Obstacle
4		
5		

When dynamic obstacles are introduced, the values in the right-hand column show clear changes. Unexpected blocks force the robot to stop, detect the obstacle, and replan, which increases the total number of cells visited and the number of turns. For example, in the first trial the path length rises from 17 to about 45 cells and the turn count jumps to 20, cutting efficiency from 41 % to about 15 %. Similar effects appear in the remaining trials: some show efficiency dropping to around 17 % or 20 % when multiple dynamic obstacles appear, while others with fewer disturbances still experience noticeable decreases in efficiency and more turns compared with their static counterparts. Across all five rows, the table highlights a consistent pattern: the presence of dynamic obstacles leads to longer and more complex routes and lower efficiency. These results illustrate the sensitivity of a classical DFS planner to real-time environmental changes and emphasize the need for adaptive or hybrid strategies when operating in environments where obstacles may emerge unexpectedly.

The data show in Fig 5 that when only static obstacles are present, the path length remains relatively short. In these cases the DFS algorithm can plan the entire route in advance, because all obstacles are known from the start. With a complete map, the search systematically explores and records a valid path while avoiding blocked cells, so the robot moves efficiently toward the goal without unnecessary detours. Dynamic obstacles present a different challenge. These obstacles are not included in the initial map and only become known when the robot is directly in front of them.

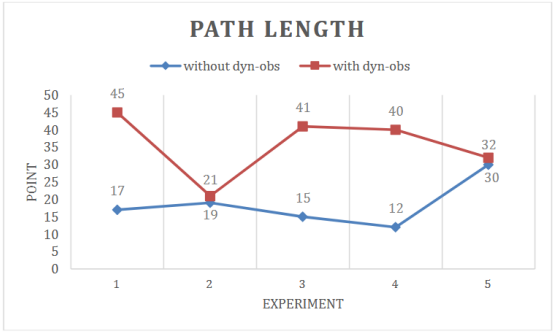


Figure 5. Path Length

Until that moment, the robot continues along its original route as if the next cell is free. Once the new obstacle is detected, the robot must stop, mark the cell as blocked, and backtrack to find an alternative route. On average, the paths taken without dynamic obstacles are about 117.7 % shorter than those taken when dynamic obstacles appear, meaning the system travels more than twice the distance when it must replan on the fly.

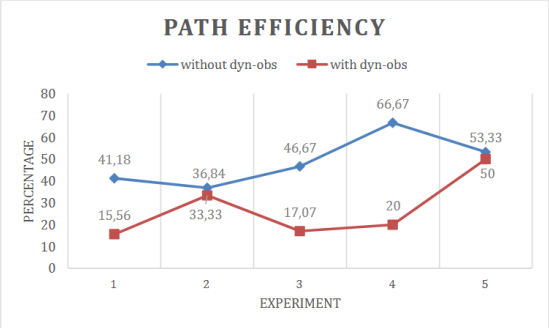


Figure 6. Path Efficiency

Path efficiency expresses how close the traveled route is to the shortest possible path between the start and the goal, stated as a percentage. Higher values indicate that the robot followed a path nearly as short as the optimal route. As shown in Fig 6, when only static obstacles were present, the system produced efficiencies ranging from about 36 % to 67 %. Because all obstacles were known from the start, the algorithm could explore and lock in a complete route that avoided barriers from the outset, allowing the robot to move with relatively few detours. In contrast, the runs with dynamic obstacles show markedly lower efficiencies, in several cases dropping below 20 %. Dynamic obstacles are not part of the initial map and are only sensed when the robot reaches them. The robot therefore continues along its planned path until it encounters an unexpected barrier, at which point it must stop, backtrack, and compute an alternate route. These mid-course corrections extend the total travel distance and reduce the ratio of the traveled path to the optimal path. The average efficiency without dynamic obstacles is about 42.28 % higher than with dynamic obstacles. This difference demonstrates that a system planner can navigate more directly and economically when all obstacles are known in advance, whereas unexpected dynamic barriers force additional movement that lowers path efficiency and increases the time to reach the goal.

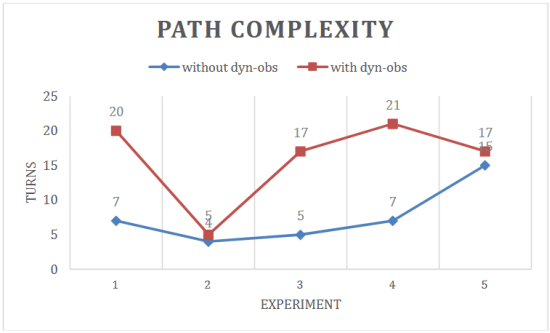


Figure 7. Path Complexity

Path complexity, expressed as the number of directional changes along a route, increased substantially when dynamic obstacles were introduced. As shown in Figure 7, the average increase of roughly 132.8 %. This outcome reflects the behavior of the Depth-First Search algorithm: when an unexpected obstacle blocks the planned path, the algorithm must backtrack and explore alternative branches, naturally creating more frequent changes of direction than a pre-planned path that avoids known static obstacles from the start. In practical mobile-robot navigation, a higher number of turns can reduce positional accuracy because each maneuver introduces acceleration, deceleration, and potential wheel slip, all of which accumulate odometry error. More turns also lengthen travel time, raise energy consumption, and make motion control more difficult, particularly on slippery surfaces or at higher speeds. To limit these effects, early detection of obstacles, path-smoothing techniques, and hybrid planning methods that combine heuristic guidance with reactive adjustments are useful, while robust localization approaches such as IMU or LiDAR-based sensor fusion help maintain accurate positioning even when path complexity increases.

5. CONCLUSION

The Depth-First Search (DFS) algorithm operates efficiently in environments with only static obstacles, since all obstacle positions are known in advance and can be incorporated into the initial route planning. Average performance gains in the static-only scenario compared with

the dynamic-obstacle scenario are approximately:

- Path length improved by about 117 %,
- Path efficiency higher by about 42 %,
- Path complexity (turns) lower by about 133 %.

When navigating without dynamic obstacles, the robot follows a shorter path, makes fewer turns, and achieves higher overall path efficiency. Increased turning and extended travel distance can raise the risk of navigation errors in real applications, such as wheel slip or odometry drift.

In environments with dynamic obstacles, the robot encounters unexpected barriers that are sensed only when directly approached. This triggers backtracking and re-planning, resulting in longer paths and more directional changes.

ACKNOWLEDGMENT

This research was supported by the Research-Based SINTA Grant (RISTA) organized by the Institute for Research and Community Service (LP2M). The authors gratefully acknowledge this financial support, which made the study possible.

REFERENCES

[1] N. AbuJabal et al., "A Comprehensive Study of Recent Path-Planning Techniques for Dynamic Environments," *Sensors*, vol. 24, no. 4, pp. 1–23, 2024.

[2] L. Liu et al., "Path planning techniques for mobile robots: A review of recent advances," *Information Sciences*, vol. 640, pp. 119–138, 2023.

[3] Y. Lin et al., "Efficient TD3-based path planning of mobile robots in dynamic environments," *Scientific Reports*, vol. 15, 2025.

[4] C.-H. Wang et al., "Multi-robot path planning in online dynamic obstacle environments using adaptive algorithms," *Discover Computing*, vol. 5, 2025.

[5] Y. Jiang et al., "Dynamic path planning for electric autonomous vehicles," *Robotics and Autonomous Systems*, vol. 172, 2023.

[6] B. Tao et al., "Deep reinforcement learning-based local path planning in dynamic environments," *J. King Saud Univ.–Computer and Information Sciences*, vol. 36, no. 5, 2024.

[7] J. Liu et al., "Path Planning for Robotic Manipulators in Dynamic Scenes Using Improved Heuristic Search," *IEEE Trans. Mechatronics*, vol. 29, no. 3, 2024.

[8] K. Shi et al., "Multi-robot dynamic path planning with spatiotemporal constraints," *Int. J. Advanced Robotic Systems*, vol. 22, 2025.

[9] S. Banik et al., "Path Planning Approaches in Multi-robot Systems: A Review," *Engineering Reports*, vol. 7, 2025.

[10] X. Wu et al., "Obstacle avoidance optimization and path replanning in dynamic environments," *Scientific Reports*, vol. 13, 2023.

[11] K. C. Ugwoke et al., "Simulation-based review of classical and heuristic path planning methods," *Robotics and Autonomous Systems*, vol. 166, 2025.

[12] "Real-time Trajectory Replanning for Dynamic Obstacles," *ACM SIGGRAPH Asia Proceedings*, 2022.

[13] "Graph-based Path Planning with Dynamic Obstacle Constraints," *arXiv preprint arXiv:2503.01234*, 2025.

[14] L. Di et al., "Trajectory tracking and obstacle avoidance in dynamic environments," *PLOS ONE*, vol. 20, 2025.

[15] K. Almazrouei et al., "Dynamic Obstacle Avoidance and Path Planning through Reinforcement Learning," *Applied Sciences*, vol. 13, no. 7, 2023.

[16] Cai, H., Chen, J., & Xu, S. (2020). *Mobile Robot Path Planning in Dynamic Environments: A Survey*. *arXiv preprint* arXiv:2006.14195.

[17] Liu, Q., Zhang, D., & Zhao, H. (2023). *Path Planning of Mobile Robots in Dynamic Environments Based on Analytic Geometry and Cubic Bézier Curve with Three Shape Parameters*. *Expert Systems with Applications*, 235, 120223.

[18] Wu, J., Li, S., & Chen, Y. (2023). *Deep Reinforcement Learning for Adaptive Robot Path Planning in Dynamic Environments*. *IEEE Access*, 11, 134502–134516.

[19] Alenezi, F., & Kim, M. (2024). *Deep Reinforcement Learning-Based Local Path Planning in Dynamic Environments for Mobile Robots*. *Ain Shams Engineering Journal*, 15(3), 343–356.

[20] Thakur, A., & Pandey, M. (2024). *Review of Autonomous Navigation Techniques in Dynamic Environments*. *Sensors*, 24(15), 5462.

[21] Zhang, Y., & Wang, T. (2022). *Global Path Planning for Mobile Robots Based on*

Improved A. Expert Systems with Applications*, 190, 116263.

[22] Dijkstra, E. W. (1959). *A Note on Two Problems in Connection with Graphs*. *Numerische Mathematik*, 1(1), 269–271.

[23] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

[24] Wu, X., & Chen, L. (2024). *Dynamic Programming for Mobile Robot Path Planning and Tracking in Static and Dynamic Environments*. *Robotics and Autonomous Systems*, 180, 104864.

[25] Yao, J., Liu, Y., & Tan, H. (2023). *Conflict-Based Search with D* Lite Algorithm for Path Planning in Unknown Dynamic Environments*. *Computers & Electrical Engineering*, 112, 108902.

[26] Li, B., Zhao, H., & Sun, W. (2021). *RRT-Based Path Planning for Mobile Robots in Complex Dynamic Environments*. *Applied Sciences*, 11(9), 4192.

[27] Fox, D., Burgard, W., & Thrun, S. (1997). *The Dynamic Window Approach to Collision Avoidance*. *IEEE Robotics & Automation Magazine*, 4(1), 23–33.

[28] Xu, R., Zhao, Q., & Zhang, H. (2024). *Obstacle Avoidance Path Planning Based on Improved Artificial Potential Field for Mobile Robots*. *Mechanical Sciences*, 14, 87–99.

[29] Ren, Y., & Wu, J. (2021). *Reinforcement Learning for Path Planning in Time-Varying Obstacle Fields*. *Robotics and Autonomous Systems*, 141, 103790.

[30] Tang, Y., & Guo, Z. (2024). *Dynamic Path Planning for Mobile Robots Using PSO Optimization in Dynamic Obstacle Environments*. *Journal of Intelligent Systems*, 33(4), 665–681.

[31] Dorigo, M., & Stützle, T. (2020). *Ant Colony Optimization: Recent Advances and Applications*. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(8), 3032–3045.

[32] Nguyen, T., & Huynh, N. (2023). *Computational Trade-offs in Real-Time Robot Replanning Systems*. *Sensors*, 23(15), 7034.

[33] Chen, K., Zhang, W., & Luo, D. (2023). *Long-Range Navigation in Complex and Dynamic Environments with Full-Stack S-DOVS*. *Applied Sciences*, 13(15), 8925.

[34] Hassan, A., Rahman, M., & Lee, H. (2024). *Analysis of Pathfinding Algorithms for Mobile Robots Movement*. *Lecture Notes in Networks and Systems*, Springer, Singapore.

[35] Orozco, P., & Torres, J. (2025). *Adaptive Sensing Strategies for Robust Autonomous Navigation in Dynamic Environments*. *Robotics and Autonomous Systems*, 178, 104612.

[36] Yoshizoe, Kazuki & Terada, Aika & Tsuda, Koji. (2015). Redesigning pattern mining algorithms for supercomputers. 10.48550/arXiv.1510.07787.