2025 IEEE 25th International Symposium on Computational Intelligence and Informatics (CINTI) | 979-8-3315-5291-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/CINTI67731.2025.11311851

A Performance Evaluation of A* and Dijkstra Algorithms with Visual-Based Goal Selection

1st Tuğba Fıçıcı
Computer Engineering Department
Istanbul Sabahattin Zaim University
Istanbul, Türkiye
tugba.ficici@std.izu.edu.tr

2nd Erdal Alimovski
Computer Engineering Department
Istanbul Sabahattin Zaim University
Istanbul, Türkiye
erdal.alimovski@izu.edu.tr

3rd Gökhan Erdemir
Dep. of Engineering Management and Technology
University of Tennessee
Chattanooga, USA
gokhan-erdemir@utc.edu

***Abstract*—Successful operation of autonomous robots in complex environments require both reliable perception and efficient path planing. Reliable perception provide the foundation of scene understanding, which enables robots to understand their environment correctly and operate effectively. This study focuses on two main objectives: generation of a spatial map of the environment and extract the coordinates of bussinesses using LiDAR and visual perception and evaluation-comparison the performance of A* and Dijkstra path planning algorithms based on these perception-driven destination points. As the part of visual perception pipeline, we perform text detection and recognition where the coordinates of bussinesses were extracted. These extracted coordinates then are used as destination points for path planning algorithms. The experiments were conducted in the Gazebo simulation environment using a TurtleBot3 Waffle robot platform. The performance of path planning algorithms is evaluated based on following metrics; distance to the target, time to reach the destination, energy consumption, CPU load, and RAM usage. The experimental results shows that A* outperforms Dijkstra algorithm in terms of shorter travel time, lower energy consumption and CPU usage.**

Keywords— mobile robot, perception, a*, dijkstra

I. Introduction

Over the past century, humans have been trying to develop machines capable of performing human-like tasks in order to facilitate day-to-day tasks [1]. The ambition to build such systems has shaped robotics into a unique domain focused on innovation and replicating human abilities [2]. Beyond their utilization in human-like applications, track inpections, and industrial manufacturing, they also hold great potential for the transport sector as well. In order to make robotics functional in areas like logistics, robust and accurate path planning is one of the most essential requirements. Path planning is fundamental component enabling robots to navigate autonomously [3]. Its primary objective is to compute an optimal, collision-free trajectory from a given start point to the end point, whether the environment is known or unknow [4]. The robots often are subject to dynamic changes, including the apperance of new obstacles and the motion of nearby robots. Therefore, effective path planning must be capable of handling such dynamic changes in the environments [5].

Depending on the underlying optimization approach, global path planning can be categorized into random sampling algorithms, population based intelligence algorithms, graph-based search algorithms, and others. Among random sampling-based algorithms, Rapidly-exploring Trees (RRT) [6] and Probabilistic Roadmaps (PRM) [7] are often utilized two algorithms. The homotopic dynamic AW-PRM algorithm, proposed by Fu [8], employs an adaptive step size strategy to effectively handle different obstacle avoidance scenarios. In addition, Fu in [9] developed a hybrid adaptive obstacle avoidance method called Dynamic Window PRM (DW-PRM), which combines an improved PRM with a Gaussian probability field to enhance the efficiency of dynamic path planning. These methods still encounter limitations due to significant computational demands and inherent stochastic behavior. In other hand, population intelligence based algorithms include techniques such as Genetic Algorithms [10], Ant Colony Optimization (ACO) [11], and Practicle Swarm Optimization (PSO). Niang in [12] combines GA with Q-learning in order to develop a dynamic obstacle avoidance strategy for changing environments. While these algorithms are effective in many cases, they heavily depend on parameter tuning which makes them highly dependent on human expertises. Among all mentioned approaches, graph-based search algorithms are considered as most mature category. When compared with sampling-based methods, graph-based methods provide more comprehensive solutions and can efficiently compute optimal trajectories. Although they offer global optimality, the obtained paths may not always be smooth enough for immediate use in robot motions. Most often utilized graoh-based algorithms are A* [13], Dijkstra [14], Breadth-First Search (BFS) [15], Depth-First Search [16], and others.

Collecting textual information from the environment is crucial for various purposes, including semantic understanding, contextual awareness, and goal-oriented navigation in autonomous robotic systems. Texts such as store signs, street names, warning labels, and directional indicators provide high-level information that can guide robots in decision-making processes. By detecting and recognizing such textual elements, robots can identify potential destinations or areas of interest within their surroundings.

Given the importance of textual information, in this study first we perform text detection to identify relevant texts in the environment, followed by text recognition to extract their semantic content. The locations of the recognized text areas are

converted into spatial coordinates, which are then used as destination points for the robot to navigate toward. Based on destination points, the performance of two classical path planning algorithms A* and Dijkstra is evaluated and compared under different metrics. All experiments are conducted in the Gazebo simulation environment using a TurtleBot platform to ensure controlled testing and realistic robotic behavior.

The rest of the paper is orginized as follows: Section II describes the simulation environment with the robotic platform as well as the perception pipline and applyed techniques, Section III demonstrates the experimental results and section IV concludes the paper.

II. Visual Goal Extraction and Path Planning Techniques

A. Environment Setup in Gazebo, Robotic Platform, RViz

Gazebo is a widely used open-source 3D robotics simulator that enables testing and development of robotic systems in a realistic virtual environment [17]. It provides a robust physics engine, sensor simulation (such as cameras, LiDAR, IMU), and customizable environments, making it suitable for developing and evaluating autonomous navigation, perception, and control algorithms. In Figure 1. designed environment is demonstrated, as a structured urban scene composed of elements commonly found in real-world settings, such as buildings, commercial shops, and banks. To simulate realistic conditions for visual perception tasks, various buildings were annotated with signboards containing textual information. These signs were designed using different fonts, sizes, and styles to introduce visual variability and challenge the robustness of the text detection and recognition modules. In the designed environment the TurtleBot3 Waffle model is utilized as the mobile robot platform due to its compact structure and compatibility with ROS-based navigation and perception tasks. The robot is equipped with a 360-degree LiDAR and a 120-degree field of view camera, making it suitable for both obstacle avoidance, sidewalk following and visual data acquisition.

Fig. 1. Bird's-eye view of the designed environment in Gazebo.

Fig. 2. Utilized Turtlebot3 Waffle robot.

Fig. 3. Samples of signboards and their texture features.

RViz is a widely used 3D visualization tool in the Robot Operating System (ROS) ecosystem. It allows real-time monitoring of various robot-related data, such as LiDAR scans, camera images, generated maps, robot position and orientation, and planned paths. We utilize to used to visualize key sensor data including LiDAR scans (/scan), raw camera images (*/camera/image_raw*), and the generated map (*/map*)

B. Visual Perception and Coordinate Determination

As a visual perception first step, we apply text detection algorithm Efficient and Accurate Scene Text Detector (EAST) [18]. After implementing and comparing the effectiveness of various OCR algorithms in [19], EasyOCR was selected for the text recognition step in this study due to the its effectiveness. In addition to this we apply Sequnce Matcher [20] as a post-procesing step in order to enhance the performance of EasyOCR. After detecting and recognizing bussiness signboards, the center points of the recognized text regions are calculated in image coordinates. To convert the detected text regions into spatial coordinates within the simulation environment, the camera's intrinsic parameters such as focal lengths f_x, f_y and optical center (c_x, c_y) and extrinsic parameters (rotation R and translation T relative to the robot frame) are utilized. Given the fixed mounting height h of the camera and its tilt angle, θ the vertical image coordinate v is used to estimate the distance to the signboard based on a projection onto the ground plane. This process enables accurate localization of signboards on the ground plane, providing reliable target coordinates for navigation and path planning.

Identify applicable funding agency here. If none, delete this text box.

C. A*

The A* algorithm is a widely used path-finding method designed to compute the shortest path between a given start and goal point. It has been applied across various domains, particularly in navigation and mapping systems, where it efficiently determines the optimal route between two locations [21]. The A* algorithm operates based on three key variables:

- G(n): Represents the actual cost incurred from the start node to the current node. It accumulates the traversal cost of all visited nodes along the path.
- H(n): Denotes the estimated cost from the current node to the goal node, commonly referred to as the heuristic value. Since the actual cost to the goal is unknown until the path is completed, this value provides an informed guess based on available data.
- f(n): The total estimated cost of the path through the currentnode, calculated as the sum of the actual cost and the heuristic.

These three components work together to guide the search toward the most promising path.

$$f(n) = G(n) + H(n) \quad (1)$$

The variable $f(n)$ represents the combined cost of the path, integrating both the actual distance traveled from the start and the estimated distance remaining to the goal.

D. Dijkstra

Dijkstra's algorithm constructs a shortest path tree by iteratively selecting the nodes that have the minimum cumulative distance from the source node. In this context, nodes are typically represented by u and v, and the weight or cost between two connected nodes is expressed as $w(u, v)$. The algorithm relies on three key variables to perform its calculations:

- dist: An array that holds the shortest known distance from the source node to every other node in the graph. Initially, $dist(source) = 0$, and $dist(v) = \infty$ for all other nodes v. These values are updated as the algorithm progresses until the minimum distance to each node is determined.
- Q: A priority queue that contains all nodes in the graph. It is used to repeatedly extract the node with the minimum tentative distance. This queue becomes empty once all shortest paths are found.
- S: A set of nodes for which the shortest paths from the source have been finalized. Initially empty, this set gradually grows until it includes all nodes in the graph.

III. EXPERIMENTS

The experiments were conducted on a laptop equipped with an 8-core AMD Ryzen 9 4000 Series processor, 16 GB of RAM, and an NVIDIA GeForce GTX 1660 Ti GPU. All simulations were performed using the ROS 2 middleware in combination with Gazebo and RViz frameworks. Note that the GPU was utilized exclusively for accelerating the simulation environments, whereas the algorithmic computations and experimental evaluations were carried out on the CPU.

Initially, the robot autonomously explored the simulation environment by following sidewalks, guided by LiDAR sensor data to construct the map. During this mapping phase, a visual perception pipeline was employed to detect and recognize bussiness signboards. The spatial positions of these identified bussinesses were then extracted. These positions served as target destinations for the robot, which was navigated using two classical path planning algorithms. The efficiency of each method was evaluated based on five key performance metrics: distance to the target, time to reach the destination (target), energy consumption, CPU load, and RAM usage.

The Figure 4 illustrates the map of the simulation environment generated using the SLAM Toolbox. The white paths in the map represent the sidewalks followed by the robot during the exploration process, while the black lines indicate walls or objects perceived as boundaries. The gray areas correspond to regions that are either unexplored or considered non-traversable.

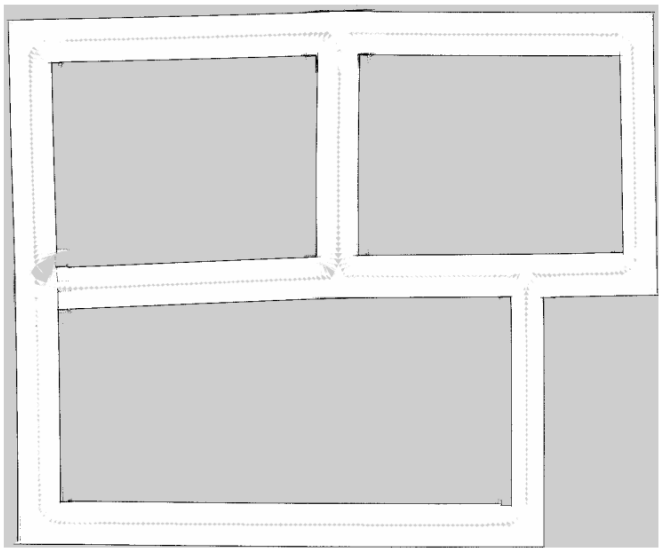

Fig. 4. Generated Map of the simulation environment using SLAM toolbox.

Table I. presents the performance results of the A* algorithm across various destination points. The table includes five key metrics: distance to target (meters), time to reach the target (seconds), energy consumption (joule), CPU usage (percentage), and RAM (percentage) usage. It is observed that bussinesses located closer to the robot, such as Antik Kasap and Vakıfbank, result in shorter travel times and lower energy consumption. This indicates that path length is a critical factor influencing both temporal and energy efficiency. In contrast, destinations like BİM and Ziraat Bankası have longer distances and correspondingly higher time and energy requirements. Central Processing Unit (CPU) and Random Access Memory (RAM) usage remain within a reasonable range across all destinations, but slightly increase for longer and more complex paths.

TABLE I. THE PERFORMANCE OF A* ALGORİTHM FOR EACH DESTİNATİON POİNT.

Bussiness	Dist. to target (m)	Time to reach the target (sec)	Energy Cons. (j)	CPU (%)	RAM (%)
Albaraka	86.85	360	21.74	106	1.8
Antik Kasap	10.08	48	2.48	95	2.1
A101	32.42	132	8.15	95	2.1
BİM	109.99	565	26.96	105	2.1
HalkBank	49.98	238	12.21	90	2.3
Homepark	35.96	195	8.93	88	2.5
İş Bakası	22.63	103	5.60	86	1.4
KuveytTürk	38.44	167	9.57	85	1.4
Ptt	71.18	305	17.73	91	3.6
Vakıfbank	22.34	5	5.56	111	2.3
Yapıkredi	77.53	400	18.67	107	2.2
Ziraat Bankası	98.68	496	23.94	86	2.1

TABLE II. THE PERFORMANCE OF DİJKSTRA ALGORİTHM FOR EACH DESTİNATİON POİNT

Bussiness	Dist. to target (m)	Time to reach the target (sec)	Energy Cons. (j)	CPU (%)	RAM (%)
Albaraka	90.15	388	22.05	126	2.1
Antik Kasap	13.10	53	2.75	110	2.4
A101	28.43	120	8.35	102	2.5
BİM	110.32	568	26.63	113	2.3
HalkBank	52.40	258	12.83	111	2.2
Homepark	32.76	172	8.39	103	2.4
İş Bakası	24.55	113	5.72	99	1.8
KuveytTürk	42.87	17	9.85	102	1.7
Ptt	75.66	340	19.23	104	4.2
Vakıfbank	23.89	96	5.70	133	2.2
Yapıkredi	76.60	398	18.89	126	3.2
Ziraat Bankası	102.52	503	24.21	117	2.4

The Table II. illustrates the results obtained using the Dijkstra algorithm for various destinations. As expected, locations situated at longer distances such as BİM and Ziraat Bankası—caused higher travel time and energy usage. In contrast, nearby places like Antik Kasap and Vakıfbank showed minimal resource demand. CPU and RAM usage varied slightly, with Ptt and YapıKredi showing increased computational load compared to others. Table III. presents a comparative analysis of the overall performance of the Dijkstra and A* algorithms across several key metrics. According to the data, A* consistently outperforms Dijkstra in terms of efficiency. Specifically, the A* algorithm resulted in a shorter path length (54.67 m vs. 56.10 m) and reduced travel time (259.56 s vs. 265.66 s), indicating a more optimal route computation. Additionally, A* demonstrated lower energy consumption (13.43 J compared to 13.71 J), which is critical for resource-constrained robotic systems. Computational resource usage was also more favorable for A*, with noticeably lower CPU (95.54% vs. 112.35%) and RAM (2.15% vs. 2.45%) utilization. These results highlight the A* algorithm's advantage in balancing path optimality and computational efficiency within the simulated navigation environment.

TABLE III. COMPARİSON PERFORMANCE İN AVERAGE OF A* AND DİJKSTRA ALGORİTHMS.

Metrics	A*	Dijkstra
Distance to target (m)	54.64	56.10
Time to reach the destination (sec)	259.56	265.66
Energy consumption (j)	13.43	13.71
CPU (%)	95.54	112.35
RAM (%)	2.15	2.45

As a result of the obtained findings, it can be said that a reliable visual perception approach enables the extraction of meaningful environmental information that can be effectively used for autonomous navigation tasks. Since destination coordinates are derived from detected text regions, all performance metricsare directly influenced by the accuracy of the visual detection process. Any perception failure or misdetection could lead to incorrect target locations, which in turn would cause inefficient route planning and degraded overall system performance.

IV. CONCLUSION

This study demonstrates the integration of visual perception and classical path planning in a simulation environment using ROS 2, Gazebo, and RViz. A LiDAR-based map was generated, and bussiness signboards were detected and recognized through text recognition techniques. The coordinates of these detected texts were then projected into the simulation space and used as destination points for the A* and Dijkstra algorithms. Experimental results show that A* outperformed Dijkstra in terms of distance traveled, time to target, energy consumption, CPU usage, and RAM usage. The success of the entire pipeline relies heavily on accurate visual perception; any error in detection could directly impact navigation efficiency. These findings highlight the importance of combining robust perception systems with path planning algorithms for effective autonomous robot navigation and as well as can be leveraged for various tasks.

For future work, we plan to extend the system by incorporating goal selection strategies and comparing the applied methods with more advanced planners such as D* Lite, RRT*, and hybrid approaches. This will provide practical insights for practitioners and enhance the applicability of autonomous navigation in real-world scenarios.

REFERENCES

[1] Nerlekar, V., Mamtura, T., & Parihar, S. (2022, January). Implementation of A* algorithm for optimal path planning for mobile robots. In 2022 4th International Conference on Smart Systems and Inventive *Technology (ICSSIT)* (pp. 382-390). IEEE.

[2] Gilbert, E., & Johnson, D. (2003). Distance functions and their application to robot path planning in the presence of obstacles. *IEEE Journal on Robotics and Automation*, *1*(1), 21-30.

[3] Zhang, J., Guo, J., Zhu, D., & Xie, Y. (2025). Dynamic path planning fusion algorithm with improved A* algorithm and dynamic window approach. *International Journal of Machine Learning and Cybernetics*, *16*(3), 2057-2071.

[4] Karur, K., Sharma, N., Dharmatti, C., & Siegel, J. E. (2021). A survey of path planning algorithms for mobile robots. *Vehicles*, *3*(3), 448-468.

[5] Anavatti, S. G., Francis, S. L., & Garratt, M. (2015, October). Path-planning modules for Autonomous Vehicles: Current status and challenges. In *2015 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA)* (pp. 205-214). IEEE.

[6] Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., & Teller, S. (2011, May). Anytime motion planning using the RRT. In *2011 IEEE international conference on robotics and automation* (pp. 1478-1483). ieee.

[7] Kuffner, J. J., & LaValle, S. M. (2000, April). RRT-connect: An efficient approach to single-query path planning. In Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065) (Vol. 2, pp. 995-1001). IEEE.

[8] Fu, J., Sun, G., Liu, J., Yao, W., & Wu, L. (2023). On hierarchical multi-UAV dubins traveling salesman problem paths in a complex obstacle environment. *IEEE Transactions on Cybernetics*, *54*(1), 123-135.

[9] Fu, J., Sun, G., Yao, W., & Wu, L. (2022). On trajectory homotopy to explore and penetrate dynamically of multi-UAV. *IEEE Transactions on Intelligent Transportation Systems*, *23*(12), 24008-24019.

[10] Liu, F., He, H., Li, Z., Guan, Z. H., & Wang, H. O. (2020, July). Improved potential field method path planning based on genetic algorithm. In *2020 39th Chinese Control Conference (CCC)* (pp. 3725-3729). IEEE.

[11] Jin, Q., Tang, C., & Cai, W. (2021). Research on dynamic path planning based on the fusion algorithm of improved ant colony optimization and rolling window method. *IEEE access*, *10*, 28322-28332.

[12] YU, N., WANG, C., MO, F., & CAI, J. (2017). Dynamic environment path planning based on Q-learning algorithm and genetic algorithm. *Journal of Beijing University of technology*, *43*(7), 1009-1016.

[13] Han, C., & Li, B. (2023, December). Mobile robot path planning based on improved A* algorithm. In *2023 IEEE 11th joint international information technology and artificial intelligence conference (ITAIC)* (Vol. 11, pp. 672-676). IEEE.

[14] Liu, L. S., Lin, J. F., Yao, J. X., He, D. W., Zheng, J. S., Huang, J., & Shi, P. (2021). Path planning for smart car based on Dijkstra algorithm and dynamic window approach. *Wireless Communications and Mobile Computing*, *2021*(1), 8881684.

[15] Paulino, L., Hannum, C., Varde, A. S., & Conti, C. J. (2021, August). Search methods in motion planning for mobile robots. In *Proceedings of SAI Intelligent Systems Conference* (pp. 802-822). Cham: Springer International Publishing.

[16] Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM journal on computing*, *1*(2), 146-160.

[17] Peake, I., La Delfa, J., Bejarano, R., & Blech, J. O. (2021, March). Simulation components in Gazebo. In *2021 22nd IEEE International Conference on Industrial Technology (ICIT)* (Vol. 1, pp. 1169-1175). IEEE.

[18] Zhou, X., Yao, C., Wen, H., Wang, Y., Zhou, S., He, W., & Liang, J. (2017). East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 5551-5560).

[19] Fıçıcı, T., Alimovski, E., & Erdemir, G. (2025, June). Performance Comparison of OCR Techniques for Real-Time Signboard Recognition of a Mobile Robot in Simulation. In *2025 60th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST)* (pp. 1-5). IEEE.

[20] Difflib, "Difflib Library Documentation," https://docs.python.org/3/library/difflib.html, accessed June 25, 2025.

[21] Sa, X., Huaiyu, W., & Zhihuan, C. (2020, November). Research of mobile robot path planning based on improved A* algorithm. In *2020 Chinese Automation Congress (CAC)* (pp. 7619-7623). IEEE.