Streamlined A* for Faster Robotic Inspections in Ports

Hartanto Wardana
Computer Engineering Dept.
Faculty of Electrical and Computer Engineering
Satya Wacana Christian University
Salatiga, Indonesia
hartanto.kusuma@uksw.edu

Atyanta Nika Rumaksari
Computer Engineering Dept.
Faculty of Electrical and Computer Engineering
Satya Wacana Christian University
Salatiga, Indonesia
atyanta.rumaksari@uksw.edu

Prischa Wilhelmina Picanussa
Computer Engineering Dept.
Faculty of Electrical and Computer Engineering
Satya Wacana Christian University
Salatiga, Indonesia
622020014@student.uksw.edu

Budihardja Murtianta
Electrical Engineering Dept.
Faculty of Electrical and Computer Engineering
Satya Wacana Christian University
Salatiga, Indonesia
budihardja.murtianta@uksw.edu

Adri Sooai
Department of Computer Science
Engineering Faculty
Widya Mandira Catholic University
Kupang, Indonesia
adrigabriel@unwira.ac.id

King Harold Recto
Department of Electronics Computer and Communications Engineering
Ateneo de Manila University
Manila, Philippines
krecto@ateneo.edu

Abstract—Research on automatic port inspections using robots has been carried out in the state-owned company Indonesia Port Corporation, Semarang Indonesia. However, increasing the efficiency of robotic inspections is critical because robots need to perform these tasks with much higher speeds than humans, while maintaining a high level of accuracy. The robot is equipped with sensors and computer vision technology to detect defects or problems that the human might miss. This aim is to increase overall inspection accuracy at a lower cost. In this research, introducing an optimized A* path planning algorithm that incorporating with the flood algorithm, node reductions process, and linear path planning optimization for an autonomous navigated port inspection robot. Our primary objective is to significantly increase the efficiency of the conventional A* algorithm in guiding robotic systems through complex paths. The proposed algorithm demonstrates exceptional efficiency in generating feasible paths, with success attributed to optimization steps that specifically target reducing node processing and enhancing route finding. The experimentation phase involves a comprehensive assessment of the algorithm using six key parameters: running time, number of nodes, number of turns, maximum turning angle, expansion nodes, and the total distances output. Through rigorous testing, the algorithm's performance is evaluated and compared against seven other current algorithms, namely A*, BestFirst, Dijkstra, BFS, DFS, Bidirectional A*, and Geometric A*. Results from the experiments reveal the algorithm's outstanding running time efficiency, surpassing all other algorithms tested. Notably, it exhibits a remarkable 6.5% improvement over the widely recognized Geometric A* algorithm.

Keywords— Optimization A* Algorithm, Metaheuristic algorithm, Autonomous Navigated Inspection Robot, Path Planning Efficiency.

I. INTRODUCTION

In the current era of Industry 4.0, the logistics industry extensively employs automation technology to enhance the efficiency of active operational costs, in the part of transportation, labor, and maintenance processes. Consequently, many companies are adopting Automated Guided Vehicles (AGV) to address various docking requirements such as equipment and worker transportation, detection of hazardous chemicals, scanning robots, identification of goods, and tracking targets [1]. As the challenges at ports increase resulting in adding complexity to AGV usage for handling goods transportation therefore it becomes crucial to enhance the path planning scenarios for robots. This improvement allows them to effectively manage footprint calculations comprehensively and navigation processes while avoiding obstacles.

In this research at the Port of Semarang, Indonesia, optimizing the Autonomous Path Planning Robot AGV algorithm for port inspections presents a very important challenge to

carry out the integration process between the existing Enterprise Resource Planning (ERP) System and the autonomous robot inspection operation process to obtain increased efficiency. The importance of path planning algorithms for autonomous robots in port inspection lies in their potential to revolutionize the efficiency and accuracy of the inspection process in the port environment. By implementing this it aims that suitable algorithms will increase the speed of inspections and ensure robots can maneuver through challenging terrain with precision and reliability. This research progress provides great hope for the port industry in Semarang by enabling more efficient inspection procedures and ultimately leading to increased safety, productivity and operational competitive advantage in port management.

Our development strategy is focusing first on the global design of AGV before conducting the local design process. In our global design, a fundamental static resistance is obtained along with the entire start-to-finish route. Therefore, our path planning goal is to construct a navigation map of an unknown environment. Subsequently, the robot's task is to efficiently determine a low-cost and collision-free path to move from the starting point to the destination [2], [3]. The integration of AGVs into the logistics framework exemplifies an integrated and holistic data-driven system approach, leveraging automation to optimize various operational processes that become a major challenge in the whole logistic supply chain. As the challenges in our port logistics evolve, this will make a continuous refinement in automatic robot path planning strategies essential for ensuring the efficient movement of AGVs when faced with the reality of complex environments.

When we compare it too other contemporary researches, in later ones has focused on initial environmental modeling, beginning with a layout process to form a map of a geographical representation of the environment. Algorithms commonly used for environmental modeling include Voronoi, visibility graphs, and grid structures. Voronoi diagrams are often employed as a solution for puzzle-like navigation paths. For example, in the case of fast exploration on random trees (RRT), there are challenges like trap-space problems in mazes and S-shaped corridors. This algorithm proves effective in enhancing the efficiency of robot exploration time [4].

However, mapping the location and position of the environment is crucial in scenarios such as catching fires for the safety purposes of victims and evacuation teams. Therefore, determining paths based on a visual model that closely resembles the original environment becomes important because robots will be more accurate in predicting path planning prior to the actions. Meanwhile, in predicting paths, the researchers developed methods from two-dimensional and three-dimensional visual mapping based on sensor arrays. This development has an impact on the dynamics of the algorithm's computational load [5]–[7]. However, in the current situation of the algorithms, the grid structure is becoming the most fundamental aspect of the algorithm that is designed to represent the virtual environment for robot trackers and path planners that match reality. Hart et al. introduced the grid algorithm for the first time in 1968 [8]. Subsequently, many other researchers have applied this idea to various fields. Essentially, this algorithm simplifies the complexity of environmental characteristics, allowing for a more efficient computational load when calculating costs.

Grid-based modeling is a simple and efficient approach to representing any environment because these array-based structures are easy to compute. Therefore, based on [9], this makes it easier to process and analyze data, especially in the context of complex environmental modeling such as port handling. This research chooses the grid-based modeling because it enables us to integrate with existing software and systems. Furthermore, this modeling can also provide a high level of accuracy in measuring or representing the environment, especially when high resolution is required. By performing this process, not only for limited scope in port situations but also useful in various applications such as environmental monitoring, area mapping, and navigation [10], [11]. Additionally, by using this grid-based model, it can update easily when there are changes

in the environment, like the addition or removal of elements. Therefore, this paper justify using this model because it more reliable in dealing with environmental variations and changing conditions.

When the locations are predicted, the grid method allows for the highly scale-based accurate mapping of environmental representations. This method will give the matching with the discrete nature of our system. This structure will divide the environment into small cells in a grid. Consequently, it provides our robots with an intricate understanding of the location and properties of each part of the environment. Therefore, the outcome includes the precise environmental target identification of obstacle locations, distances, and surface types. By providing it into real-world problems, this method allows positioning of the environmental representation in higher dimensions [12] which capability that not shared by other Voronoi and Visual methods.

Following the general definition of creating the environmental model, the AGV undertakes a trial search process between the start and finish points. However, in its development there are three main searching algorithms were generally is being employed: interpolation-based search, bionic search, and geometric search-based algorithms. Each algorithm has its unique functions and characteristics. Interpolation-based search algorithms are typically implemented using a smooth non-linear curve interpolation basis, such as B-spline curve interpolation [13], Dubins [14], and other interpolation curves that suit the shape of the road and its environment. The smoothness and continuity of the curve representation's shape are crucial features in determining the speed and acceleration of the AGV [15]. However, this algorithm exhibits weaknesses in representation if the shape of the road represented has a non-standard dynamic shape [16].

The bionic algorithm is different from the interpolation algorithm which uses one or several fixed references which are patterned as search directions, meanwhile, the bionic algorithm uses search based on random reference searches. The name bionics refers to biological evolutionary events that naturally occur, where these evolutionary events occur randomly at the gene level in living creatures [17]. As mentioned above, the general algorithms that are often used in automatic path planning are genetic algorithm (GA) [18], particle swarm optimization (PSO) [19], and ant colony (AC) [20]. However, based on the findings, this type of algorithm generally requires an expensive computational process in its implementation [21] and is often stuck in a local optimum [22], as a result, they do not always get the exact path planning when dealing with complex environments.

In contrast to bionic and interpolation-based algorithms, geometric algorithms are the most commonly used for path planning, such as Dijkstra's and A* algorithms. It is termed a geometric algorithm because it utilizes geometric calculations in representing the environment as grids. In comparison with the existing methods, A* is more efficient than Dijkstra, it is can be shown that based on the nature program, the robot doesn't need to traverse the entire grid to reach the goal point [23]. It calculates relative costs between grids to determine the movement position.

From the definition, A* was first introduced by Hart [8] as a heuristic search algorithm. It was designed to offer efficiency and minimal time consumption. Despite its efficiency, when dealing with complex and large environments, it can lead to cross-path and zig-zag routes, therefore will reduce the robot's efficiency. Nonetheless, researchers still favor it for its suitability in the application of simple grid environments [24], [25], and its out performance of the RRT algorithm in UAV shortest path planning scenarios [26].

There are several methods that generally overcome the inefficiency of the A star algorithm. The two general methods for enhancing its accuracy are interference in preprocessing [27] and postprocessing [28]. In preprocessing interference, even though computationally expensive, is more effective than postprocessing because it calculates obstacles relative to the robot's position. On the other hand, in postprocessing optimization enhances the efficiency of the output path planning from the standard A* algorithm.

Therefore, focusing on efficiency considerations involves a trade-off between the real cost function due to the environment and the choice of the shortest distance. For example, a path design [30] aids ships in navigating through waves and obstacles. Similarly, another approach [31] uses Delaunay triangulation in preprocessing. They chose it despite its computation intensity due to its ability to save robot changes in obstacle avoidance. Moreover, a standard package by [32] improves the A* algorithm for UAVs in a three-dimensional environment with sensor-calculated costs, demonstrating its applicability to dynamic obstacles.

In the postprocessing optimization phase A* ensures the robot's location relative to the target and other fleets. The evidence by Sang et al. [33] was successfully used to enhance the effectiveness of multiple robot formations while avoiding collisions. Another breakthrough application focuses on the final process is a guided A* trail planning. This method uses guiding decisions based on the robot's environment understanding [34], applied to valet parking. However, a dynamic environment necessitates both global and local path planning for capturing the whole environment and avoiding collisions during sudden changes [35]. Despite the hybrid algorithm's success, AGV suffers from unnecessary steps during the searching phase, and mechanisms to control planning efficiency have yet to be established despite some optimization research on the A* algorithm.

Our research challenges the claim of using hybrid preprocessing and postprocessing optimization algorithms performed on local processes only. Therefore, our research divides into two general-to-specific hybrid processes: the global process, also known as pre-processing, which recommends steps before entering the post-process, and the post-process, which completes the path selection. In other words, it uses the global process of preprocessing that serves to offer recommendations for steps before entering the post-process. This phase aims to find the most efficient distances based on the results. Our contributions can be summarized as follows:

1. The process is initiated by creating a grid of the observed environment using rasterization software like Photoshop™ and then adjusted to the robot's scale and mapped the given input. Then it applies three types of grid characteristics, such as robot start position, target goal, and obstacles.
2. The A* algorithm produces results for each step and makes the robot remember its path by using stack arrays. When the robot encounters a trapped area, the algorithm can retrace its steps until it reaches a new area.
3. For the optimization purposes that run in the local process, this research took the traversed path result given from point 2 is then rearranged using the flood algorithm.

Therefore based on this research, it presents a comprehensive solution for optimizing the A* path-planning algorithm in AGV for port inspection robots in a way for more efficient and adaptive in real-world applications.

II. RESEARCH METHODS

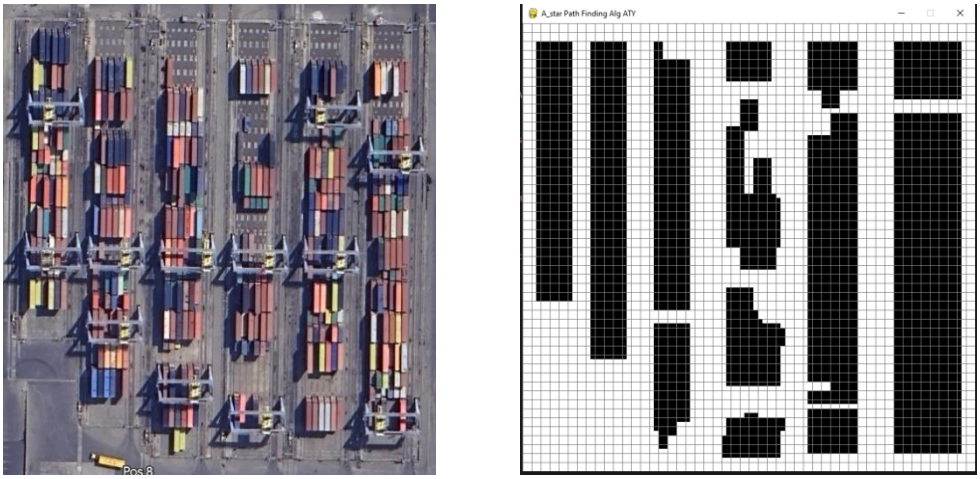
This research is based on a case study conducted at the port of Semarang, Indonesia. This study foundation comes from the work of Tang et al. [29] in modeling the port environment through the rasterization process and subsequent algorithm testing. The development of this path planning algorithm involves three essential components that each of which is elucidated. In the initial section, the process of building environmental models using rasterization is explained. This technique comprises several stages, starting from the acquisition of satellite images of port maps located in Semarang, Indonesia, to the segmentation of image dimensions into a grid map. The second stage of the A* algorithm takes shape by storing the results of each step in an array in order to memory

for the further process. This section delves into the algorithm's formation and implementation of locations virtual mapping. In the step of final process, our approach details the distance selection technique by utilizing the flood algorithm. We found that integration of this technique gives further enhance of overall performance and accuracy of the path planning algorithm. We apply this algorithm in the practical case study at the port of Semarang, Indonesia. It showcase the efficacy of the proposed methodology in real-world scenarios.

A. Modelling environment using rasterization

In Figure 1a, the satellite map of the container port post 8 north of Semarang, Indonesia, was obtained from the results of a Google Earth™ map capture, where these results will be the background scenario for testing and studying the trail planning algorithm created. The rasterization process is intended to simplify the presentation of curves and contours of complex container and road images. Before the process begins, it will divide the contours of the border walls between the container storage areas. So, it turns out that the container counter and boundary walls will be colored black and the road will be colored white (Figure 1b).

Since our aim is testing the path planning algorithm, the original details of the grid representation are not paid much attention, in accordance with reference [29]. Based on these reference results, the grid mapping results that are balanced in terms of performance and feasibility are in an area of 50 x 50 grids. Figure 1b is the output rasterization grids map from the satellite map. To store information into the grid, column and row numbering, it uses encoder relative to the grid positions. In this works, it uses special technique to deliver process cost inside the grids. Contrast to Tang et. al. which only concern of limited cost calculation codes, it determines selecting serial numbering, rows and columns for applying two-dimensional cost calculations. Furthermore, it has the advantage of supporting parallelization of cost calculations and ease of searching for specific positions on the map compared with the robot's real position in the position coordinate system.



(a) Image from satellite map (b) 50x50 map rasterization results.

Figure 1. Rasterization as a technique for simplifying three-dimensional simulation of ports from (a) input Semarang's port image from satellite map to (b) the 50x50 rasterization output.

B. Preprocessing: Grids, Representation and Positioning

In designing the algorithm, we must look at the important problem of why AGVs need automatic navigation to the shortest distance. This aims to ensure that the robot can provide high consumption efficiency for battery use and provide high working speed so that the inspection process can run quickly and avoid bottlenecks. As is known in real

company management, all business process movements have costs, so the longer the process takes, the more profits you get.

The considerations used as a benchmark are complex situations at the start and the furthest possible end location, so that we can predict the estimated cost of battery consumption in one trip to determine how long the robot will work. However, when the robot works for the first time to open a road, the complexity of the environmental situation makes the calculation of battery consumption cost efficiency is uncertain. Therefore, while we are operating the robot, we have to know the combination calculations that are possible for the robot to find an efficient path in the existing area. Based on the proposition, robot's distance d from ith obstacle can be defined as:

$$d_i(x, y) > r_A + r_0 \tag{1}$$

Where radius r_A is length from the robot to its circumference; while r_O is radius of from the obstacle. Ideally, robot needs to reduce the travel distance while avoiding obstacles. Therefore, total distance by robot is L_{path} relative from current position jth robot has passed: (x_j, y_j).

$$L_{path} = \sum_{j=1}^{n-1} \sqrt{\left(x_{j+1} - x_j\right)^2 + \left(y_{j+1} - y_i\right)^2} \tag{2}$$

Stability when robot is moving is expressing with turning angle θ must be less than the maximum angle needed to turn θ_{max} (its value will be determined by trial and error). Where n_{i-1}, n_i, and n_{i+1} are the adjacent nodes.

$$\theta(n_{i-1}, n_i, n_{i+1}) \leq \theta_{max} \tag{3}$$

C. Algorithm Design

By careful investigation and exploration, in algorithm design in Figure 2, we divide the algorithm processing series into three parts. The first part is designing a regular A* algorithm, then the results are fed to the Flood algorithm and then we carry out an optimization process by searching for nodes that make the robot change its direction of movement and then finally we carry out a linear reconstruction of the robot's path in the optimization algorithm section. In this first part of the process, we utilize the regular A* algorithm developed by [8], where this processing begins by looking for 8 neighbors who are not barriers and then put them in the open list. Next, the cost function equation $f = g + h$ is calculated for each neighbor. In this study we use the Manhattan distance for heuristic costs.

After A star finds the desired path, then in the second part of the process we employ the Flood algorithm to find the most efficient path selected from the numbering of increments carried out from the end to the start position. So, it can be said that, the shortest distance is searching for paths from the node with the smallest number closest to the current node until we find the final destination. The final result of the path obtained is stored in the optimization list variable. In the third process we carry out an iterative process to find which nodes have the same direction to be collected into one reference node. After all the reference nodes have been collected, then proceed with creating a new path using linear equations with the condition that the new path does not hit a barrier.

This third process will significantly improve the efficiency of path planning algorithm because it ensures the refinement of raw path planning from second process of flood algorithm into the most efficient output. In this research we employ linear equation because it will give higher efficiency and smoothness of robot's movement, minimizing arbitrary directional changes and potential errors.

The robustness of our algorithm lies in its ability to dynamically adapt and optimize paths in real condition and making it superior to traditional methods. In the testing process we through comprehensive assessment by comparing with other algorithms such as BestFirst, Dijkstra, BFS, DFS, Bidirectional A*, and Geometric A*. As result our proposed method consistently demonstrates improved performance metrics.

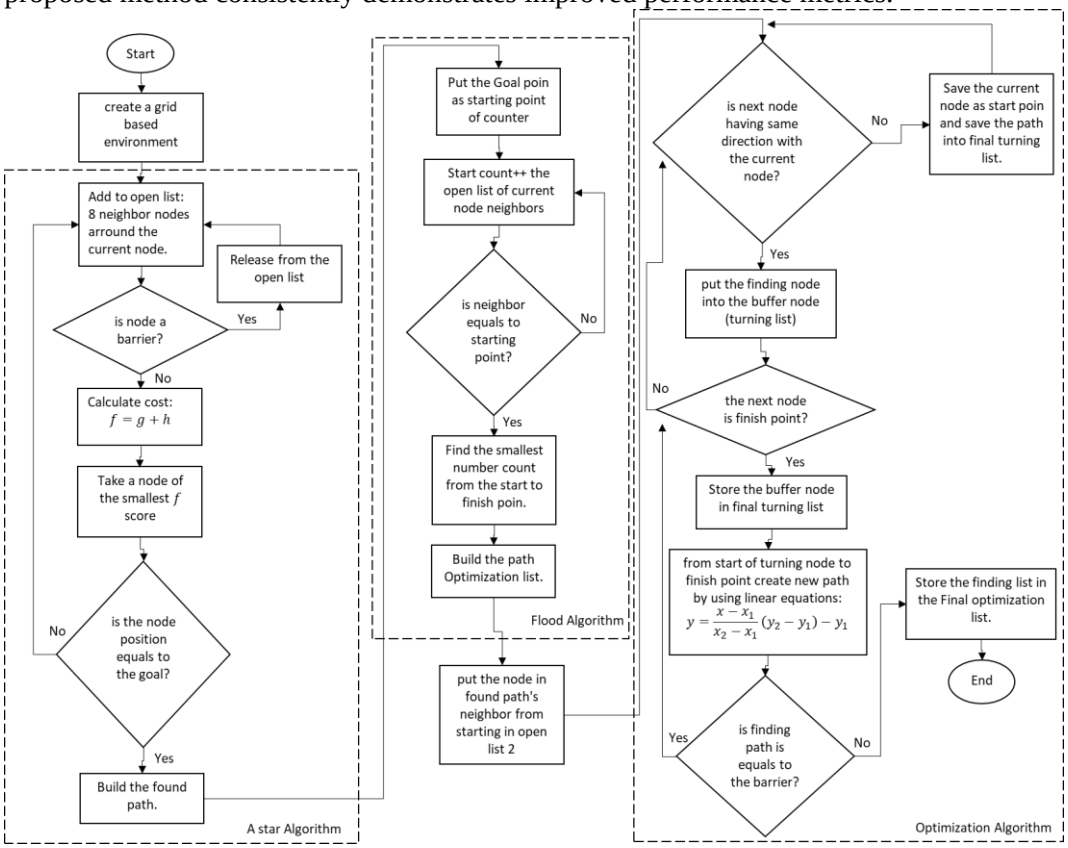


Figure 2. Proposed Algorithm.

III. RESULT AND DISCUSSION

A. Result

Creating grid maps with different specifications is essential for evaluating the effectiveness of the A* algorithm. The grids with contrast specifications represent the diversity of real-world scenarios where the algorithm is applied. By testing the algorithm on grids of various sizes, resolutions, or configurations, we assess its adaptability and robustness in different environments. However, the effectiveness of an algorithm can vary depending on the characteristics of the problem it is applied to. Moreover in employing the research, testing different grid specifications determine whether the A* algorithm generalizes well across a range of scenarios or only effective in specific conditions. Therefore, grid maps of different sizes and complexities help assess the scalability of the algorithm.

To determine the superiority of the algorithm based on its development, the severity of the algorithm's computational load is simulated on differences in grid map sizes. In real applications, the area of the box is analogous to the area of the robot's change range from the previous position to the next unit box. It also includes the spatial sensor range which determines what type of spatial information is in the sensor reception environment. Therefore, one box also includes the minimum movement that the robot can do. So, n squares are the units we use to represent the robot's movement.

Experiments were carried out in two grid resolution of 50x50 and 100x100. In this experiment, only two specifications are recommended. Meanwhile, the previous experiment used other specifications at a size of 20x20, but because the results did not change much with a size of 50x50, to save on the process, this experiment only used two sizes. Next, we set up the experiment according to the standard with the start point at $(0,0)$ and finish at (n_{max}, n_{max}). We orient these positions so that path planning is not affected when we change the grid resolution. Figure 3 shows the process.

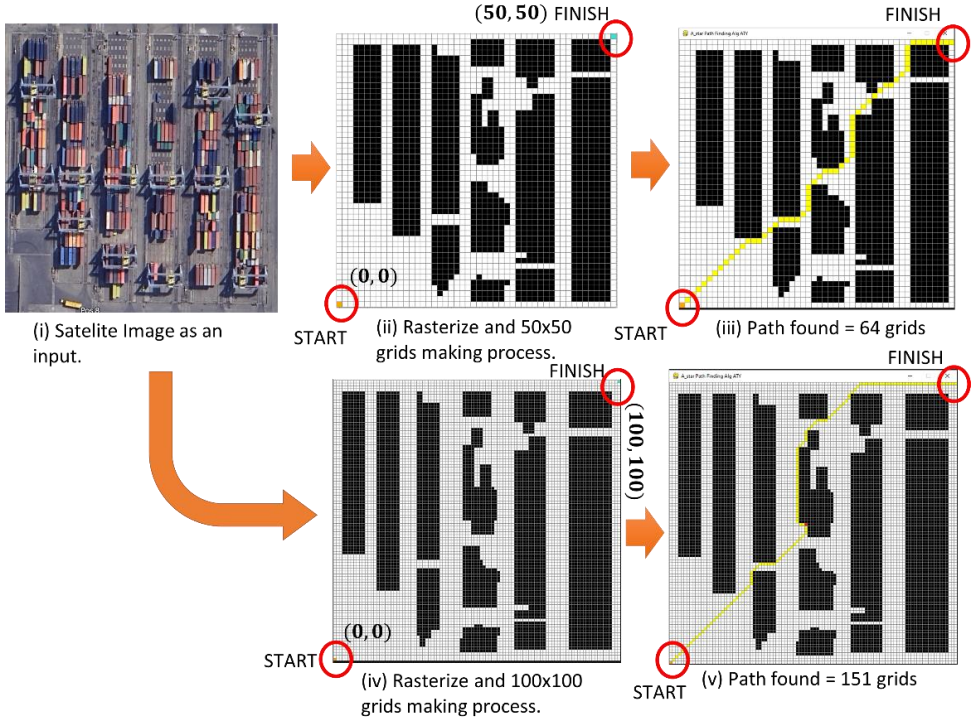


Figure 3. Proses comparison of a map between two scenarios with specification: 50x50 (ii) & (iii) grids and 100x100 (iv) & (v) grids.

Table 1. Comparison of experimental data in the two categories based on Semarang's port environment

Map-Size	Parameters	A*	AF	Our
50x50	PT (ms)	5.45	5.73	5.97
	RPT (s)	55.5	46.5	36.27
	NN (poin)	111	93	20
	TD (m)	122.1	102.3	87
100x100	APT (ms)	7.658	8.285	9.997
	RPT (s)	125.5	95.5	76.4
	NN (poin)	251	191	18
	TD (m)	276.1	210.1	201

The results from Table 1 obtains the most optimal results using the algorithm offered, using four test parameters such as algorithm processing time (PT in seconds), estimated processing time on the robot (RPT), number of nodes (NN) used by the robot, and total distance (TD in meters) traveled by robot. We compared three main scenarios to be able to show significant improvements internally in this case, namely traditional A*, A* with Flood Algorithm (AF) and our development results.

Furthermore, in Fig. 4, in order to test the reliability of this algorithm, we utilized tests from [29]. What is interesting about this test is the similarity of the themes and research models used. We use test specifications with type 6 map categories from [29] with a map

size of 100x100. We tested using six types of parameters, namely, running time, number of nodes, number of turns, max turning angle, expansion nodes, and total distances. We also compare with other seven algorithms namely A*, BFS, Dijkstra, BestFirst, DFS, and Geometric A*.

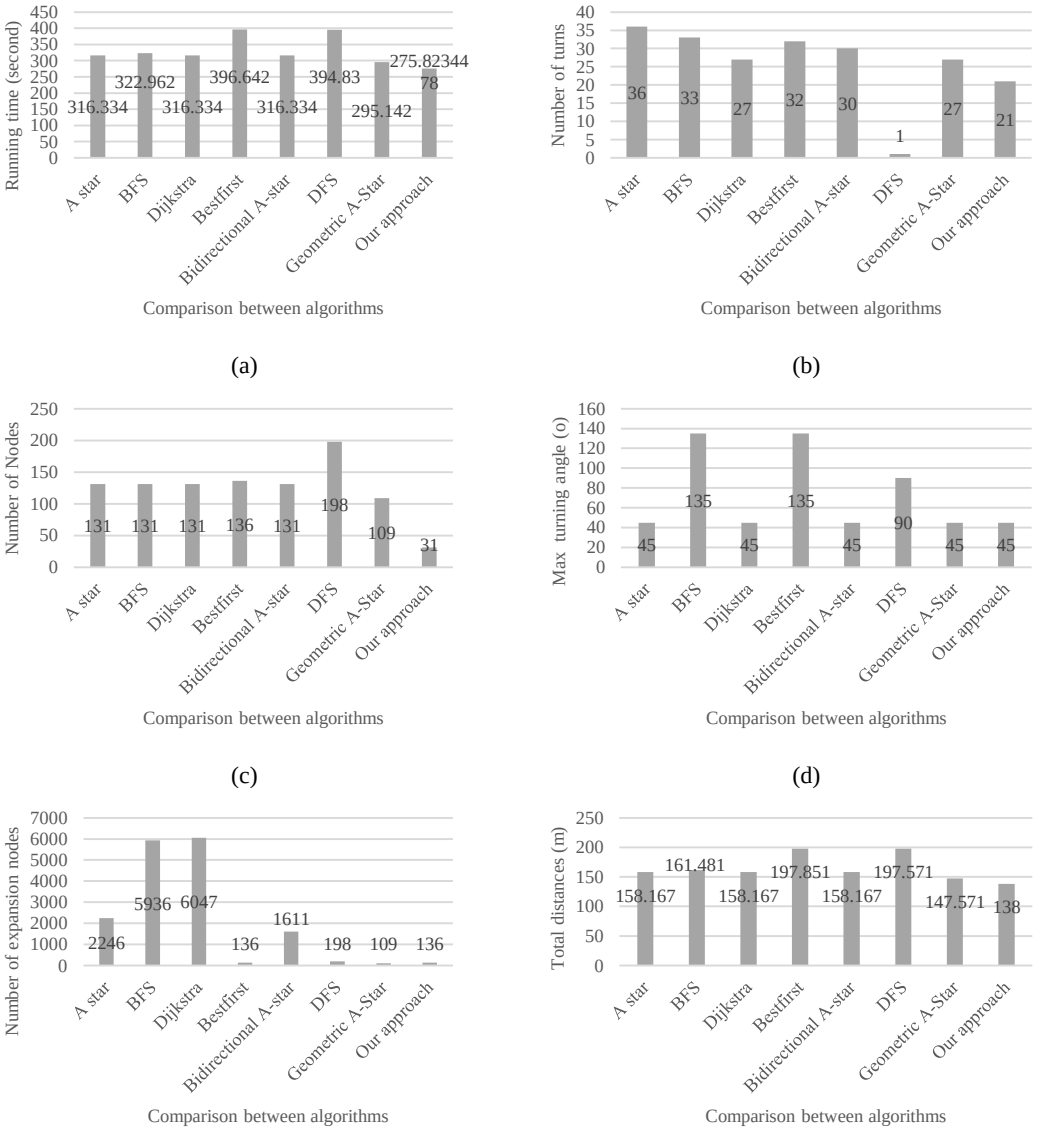


Figure 4. Comparison of our approach and other different algorithms based on parameters (a) Running time, (b) Number of nodes, (c) Number of turns, (d) Maximum of turning angle, (e) Number of expansion nodes, and (f) Total distances.

B. Discussion

The template is designed so that author affiliations are not repeated each time for multiple authors of the same affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization). This template was designed for two affiliations.

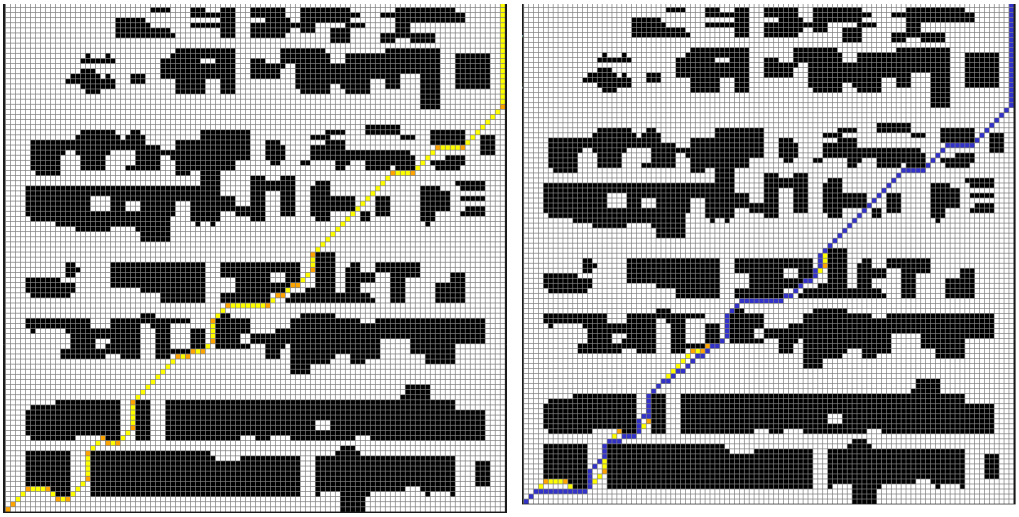
1) Simulation using Semarang's port scenario

In table 1, the results obtained for the processing time parameters of our developed algorithm get the longest time, namely 5.975 seconds for a map size of 50x50 and 9.97

seconds for a map size of 100x100, while the A* and flood algorithms and the regular A* algorithm get the time for a map size. 50x50 5.73 and 5.45 seconds and 8.28 and 7.65 seconds for map size 100x100. This is because in its processing our algorithm uses additional processing to linearize the resulting path from the processing of the A* based on the Flood algorithm. On average it takes us 20% longer than the A* based on the Flood algorithm. However, in terms of the estimated processing time parameters for robots in carrying out their tasks, our algorithm has the fastest time in all map size categories, namely in map size 50x50 our algorithm has a time of 36.27 seconds, meanwhile the A* algorithm with Flood and the regular A* algorithm has times of 46.5 and 55.5 seconds. In the map size 100x100 category, our algorithm has a time of 76.4 seconds and the A* algorithm with Flood and the regular A* algorithm have a time of 95.5 and 125.5 seconds. So, our algorithm has a time of 20% faster than the A* algorithm with Flood.

The aim of designing this algorithm is to be more optimal, we use a method of reducing the nodes passed so that the robot processing time can be faster. Therefore, a linearization process is used which is demonstrated by our algorithm's node reduction, namely in the map size category 50x50, which is 20 from 93 and 111 for the A* algorithm with Flood and the regular A* algorithm. Meanwhile, in a map size of 100x100, our algorithm has 18 nodes out of 191 and 251 for the A* algorithm with Flood and the regular A*. In other words, our algorithm has an average node reduction of 15.46% of the A* algorithm with Flood.

So, the final result we get from implementing this algorithm is a reduction in the distance the robot travels in carrying out its tasks. In the 50x50 map size category, the robot's travel results are 87 meters from 102.3 and 122.1 meters in the A* algorithm with regular Flood and A*. Furthermore, in the map size 100x100 category the robot's travel results were 201 meters from 210.1 and 276.1 meters in the A* algorithm with Flood and regular A*. Our algorithm has an average distance reduction of 9.64% from the A* algorithm with Flood.



(a) (b)

Figure 5. Final routes from our approach: (a) Reduce nodes (orange) from found path (yellow); (b) Optimization (blue) reconstructed from reduced nodes

2) Simulation using different port scenarios using different algorithms

In the Figure 4, it can be presented that the running time parameter (the time the robot completes its work) for our algorithm has the most efficient value, namely 275,823 seconds compared to other algorithms. When compared with the geometric algorithm

there is a difference of 6.5% better. In this table, the Bestfirst algorithm gets the most wasteful results, namely with a travel time of 396.64 seconds or 30.4% more inefficient than our algorithm. In this experiment, we also found that the results of the node reduction process caused the robot to increase processing time efficiency. Our experiments in table 2 show that our proposed algorithm produces 31 nodes from the total node expansion obtained, which is 136. From these results, even though our number of node expansions is 24.8% greater than the Geometric algorithm, we have succeeded in reducing the total number of node expansions by 77%. So based on this, with a total distance of 138 meters we obtain an optimum processing time of 275,823 seconds.

Figure 5.a shows that the robot has succeeded in getting the optimal working path (yellow). Meanwhile, results of the nodes found for reducing the work process are shown in orange. Meanwhile in figure 4.b. shows the new linear optimization path (blue) obtained from the linear equations on the reduction node path. We also found that the max turning angle of our approach is 45 degree which is the most efficient angle.

IV. CONCLUSION

In this research, we present a novel optimized A* path planning algorithm. Our objective is to improve the efficiency of the conventional A* algorithm, particularly in the context of pathfinding robots deployed in port environments. We shows the experiments success that conducted demonstrate that our proposed algorithm exhibits a higher efficiency rate in generating feasible paths. This algorithm gives evidence that the application is attributed to optimization steps, which also involve the reduction of node processing and the improvement of route finding through the use of a new route calculated from linear equations between adjacent reduction nodes. In general holistic applications, our approach evidently effectively shortens the path length compared to the ordinary A* algorithm.

The experimentation phase involves a comprehensive assessment of the algorithm using six key parameters: running time, number of nodes, number of turns, maximum turning angle, expansion nodes, and the total distances output. Through rigorous testing, the algorithm's performance is evaluated and compared against seven other algorithms, namely A*, BestFirst, Dijkstra, BFS, DFS, Bidirectional A*, and Geometric A*. Results from the experiments reveal the algorithm's outstanding running time efficiency, surpassing all other algorithms tested. Notably, it exhibits a remarkable 6.5% improvement over the widely recognized Geometric A* algorithm.

For future experiments, several key considerations should be addressed. Firstly, the real-world implementation of our approach in AGV (Automated Guided Vehicle) robots needs thorough investigation. We are aware that factors such as hardware design strategy and movement strategy play a crucial role in ensuring a high success rate. Additionally, in real-world applications, dynamic scene environments must be taken into account. Therefore, the robot should be equipped with sensor-actuator systems for motion analysis to avoid collisions.

ACKNOWLEDGEMENT

The authors would like to extend their sincere gratitude to the previous researchers whose work laid the foundation for this study, particularly Hoeko SW, Gloria Song Abimanyu, the R2C team, and the Faculty of Electrical Engineering and Computer Engineering at Satya Wacana Christian University.

DECLARATION

The contribution or credit of this works are equal portion for each author; This research was funded by SATYA WACANA CHRISTIAN UNIVERSITY, grant theme "HIBAH INTERNAL DOSEN UKSW- TA 2023/2024"; The authors declare no conflict of interest.

REFERENCES

[1] I. Rusinov, E. Besedina, and N. Shcherbinin, "Global Trends of the Cargo Handling Operations Automatization at Container Terminals," in International Scientific Siberian Transport Forum TransSiberia - 2021, A. Manakov and A. Edigarian, Eds., in Lecture Notes in Networks and Systems. Cham: Springer International Publishing, 2022, pp. 1492–1508. doi: 10.1007/978-3-030-96380-4_165.

[2] H. Zhang, W. Lin, and A. Chen, "Path Planning for the Mobile Robot: A Review," Symmetry, vol. 10, no. 10, Art. no. 10, Oct. 2018, doi: 10.3390/sym10100450.

[3] A. Ait Saadi, A. Soukane, Y. Meraihi, A. Benmessaoud Gabis, S. Mirjalili, and A. Ramdane-Cherif, "UAV Path Planning Using Optimization Approaches: A Survey," Arch Computat Methods Eng, vol. 29, no. 6, pp. 4233–4284, Oct. 2022, doi: 10.1007/s11831-022-09742-7.

[4] W. Chi, Z. Ding, J. Wang, G. Chen, and L. Sun, "A Generalized Voronoi Diagram-Based Efficient Heuristic Path Planning Method for RRTs in Mobile Robots," IEEE Transactions on Industrial Electronics, vol. 69, no. 5, pp. 4926–4937, May 2022, doi: 10.1109/TIE.2021.3078390.

[5] V. Agarwal, S. Tapaswi, and P. Chanak, "A Survey on Path Planning Techniques for Mobile Sink in IoT-Enabled Wireless Sensor Networks," Wireless Pers Commun, vol. 119, no. 1, pp. 211–238, Jul. 2021, doi: 10.1007/s11277-021-08204-w.

[6] H. Zhao et al., "Fire evacuation supported by centralized and decentralized visual guidance systems," Safety Science, vol. 145, p. 105451, Jan. 2022, doi: 10.1016/j.ssci.2021.105451.

[7] J.-S. Chou, M.-Y. Cheng, Y.-M. Hsieh, I.-T. Yang, and H.-T. Hsu, "Optimal path planning in real time for dynamic building fire rescue operations using wireless sensors and visual guidance," Automation in Construction, vol. 99, pp. 1–17, Mar. 2019, doi: 10.1016/j.autcon.2018.11.020.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.

[9] V. Raju and M. F. Selekwa, "On the Mapping Problem in SLAM Approaches for Autonomous Robot Navigation," presented at the ASME 2021 International Mechanical Engineering Congress and Exposition, American Society of Mechanical Engineers Digital Collection, Jan. 2022. doi: 10.1115/IMECE2021-70452.

[10] K. W. Chiang, G. J. Tsai, H. W. Chang, C. Joly, and N. EI-Sheimy, "Seamless navigation and mapping using an INS/GNSS/grid-based SLAM semi-tightly coupled integration scheme," Information Fusion, vol. 50, pp. 181–196, Oct. 2019, doi: 10.1016/j.inffus.2019.01.004.

[11] Md. Shafiqul Islam, Md. Tarequl Islam, and M. G. G. Faruque, "A Survey on LiDAR-Based SLAM Technique for an Autonomous Model Using Particle Filters," in Soft Computing for Security Applications, G. Ranganathan, X. Fernando, F. Shi, and Y. El Allioui, Eds., in Advances in Intelligent Systems and Computing. Singapore: Springer, 2022, pp. 227–240. doi: 10.1007/978-981-16-5301-8_17.

[12] P.-C. Song, J.-S. Pan, and S.-C. Chu, "A parallel compact cuckoo search algorithm for three-dimensional path planning," Applied Soft Computing, vol. 94, p. 106443, Sep. 2020, doi: 10.1016/j.asoc.2020.106443.

[13] S. A. Eshtehardian and S. Khodaygan, "A continuous RRT*-based path planning method for non-holonomic mobile robots using B-spline curves," J Ambient Intell Human Comput, vol. 14, no. 7, pp. 8693–8702, Jul. 2023, doi: 10.1007/s12652-021-03625-8.

[14] A.-D. Nguyen, N.-H. Tran, T.-T. Nguyen, A.-T. Nguyen, and T.-P. Tran, "A Hybrid Multi-waypoints Path Planning System for Robots with Minimum Turning Radius Constraint Using GA-B-Spline and Dubins Interpolation," in Proceedings of the International Conference on Advanced Mechanical Engineering, Automation, and Sustainable Development 2021 (AMAS2021), B. T. Long, H. S. Kim, K. Ishizaki, N. D. Toan, I. A. Parinov, and Y.-H. Kim, Eds., in Lecture Notes in Mechanical Engineering. Cham: Springer International Publishing, 2022, pp. 906–917. doi: 10.1007/978-3-030-99666-6_133.

[15] E. D. Lambert, "Optimization and Mathematical Modelling for Path Planning of Co-operative Intra-logistics Automated Vehicles," phd, University of Leeds, 2023. Accessed: Sep. 10, 2023. [Online]. Available: https://etheses.whiterose.ac.uk/32528/

[16] C. Zhou, B. Huang, and P. Fränti, "A review of motion planning algorithms for intelligent robots," J Intell Manuf, vol. 33, no. 2, pp. 387–424, Feb. 2022, doi: 10.1007/s10845-021-01867-z.

[17] S. P. Sahoo, B. Das, B. B. Pati, F. P. Garcia Marquez, and I. Segovia Ramirez, "Hybrid Path Planning Using a Bionic-Inspired Optimization Algorithm for Autonomous Underwater Vehicles," Journal of Marine Science and Engineering, vol. 11, no. 4, Art. no. 4, Apr. 2023, doi: 10.3390/jmse11040761.

[18] Y. Liang and L. Wang, "Applying genetic algorithm and ant colony optimization algorithm into marine investigation path planning model," Soft Comput, vol. 24, no. 11, pp. 8199–8210, Jun. 2020, doi: 10.1007/s00500-019-04414-4.

[19] T. Qiuyun, S. Hongyan, G. Hengwei, and W. Ping, "Improved Particle Swarm Optimization Algorithm for AGV Path Planning," IEEE Access, vol. 9, pp. 33522–33531, 2021, doi: 10.1109/ACCESS.2021.3061288.

[20] G. Yi, Z. Feng, T. Mei, P. Li, W. Jin, and S. Chen, "Multi-AGVs path planning based on improved ant colony algorithm," J Supercomput, vol. 75, no. 9, pp. 5898–5913, Sep. 2019, doi: 10.1007/s11227-019-02884-9.

[21] Y. Liu, S. Yan, Y. Zhao, C. Song, and F. Li, "Improved Dyna-Q: A Reinforcement Learning Method Focused via Heuristic Graph for AGV Path Planning in Dynamic Environments," Drones, vol. 6, no. 11, Art. no. 11, Nov. 2022, doi: 10.3390/drones6110365.

[22] A. Zhang, T. Qian, and X. Wu, "An improved ant colony algorithm for path planning of manipulator," in 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), Dec. 2020, pp. 1358–1362. doi: 10.1109/ITAIC49862.2020.9339081.

[23] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, "A Systematic Literature Review of A* Pathfinding," Procedia Computer Science, vol. 179, pp. 507–514, Jan. 2021, doi: 10.1016/j.procs.2021.01.034.

[24] Q. Wang et al., "Application of A star algorithm in amphibious hull cleaning robot," in 2022 IEEE International Conference on Mechatronics and Automation (ICMA), Aug. 2022, pp. 269–273. doi: 10.1109/ICMA54519.2022.9855911.

[25] A. Rumaksari et al., "Real world design and implementation of pathfinding sewer inspection robot using A* algorithm," Jurnal Mantik, vol. 7, no. 1, Art. no. 1, May 2023, doi: 10.35335/mantik.v7i1.3702.

[26] C. Zammit and E.-J. van Kampen, "Comparison Between A* and RRT Algorithms for 3D UAV Path Planning," Un. Sys., vol. 10, no. 02, pp. 129–146, Apr. 2022, doi: 10.1142/S2301385022500078.

[27] 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, Aug. 2018, doi: 10.1016/j.robot.2018.04.007.

[28] R. Song, Y. Liu, and R. Bucknall, "Smoothed A* algorithm for practical unmanned surface vehicle path planning," Applied Ocean Research, vol. 83, pp. 9–20, Feb. 2019, doi: 10.1016/j.apor.2018.12.001.

[29] G. Tang, C. Tang, C. Claramunt, X. Hu, and P. Zhou, "Geometric A* Algorithm: An Improved A* Algorithm for AGV Path Planning in a Port Environment," IEEE Access, vol. 9, pp. 59196–59210, 2021, doi: 10.1109/ACCESS.2021.3070054.

[30] C. Liu, Q. Mao, X. Chu, and S. Xie, "An Improved A* Algorithm Considering Water Current, Traffic Separation and Berthing for Vessel Path Planning," Applied Sciences, vol. 9, no. 6, Art. no. 6, Jan. 2019, doi: 10.3390/app9061057.

[31] Z. Liu, H. Liu, Z. Lu, and Q. Zeng, "A Dynamic Fusion Pathfinding Algorithm Using Delaunay Triangulation and Improved A* for Mobile Robots," IEEE Access, vol. 9, pp. 20602–20621, 2021, doi: 10.1109/ACCESS.2021.3055231.

[32] Z. Zhang, J. Wu, J. Dai, and C. He, "A Novel Real-Time Penetration Path Planning Algorithm for Stealth UAV in 3D Complex Dynamic Environment," IEEE Access, vol. 8, pp. 122757–122771, 2020, doi: 10.1109/ACCESS.2020.3007496.

[33] H. Sang, Y. You, X. Sun, Y. Zhou, and F. Liu, "The hybrid path planning algorithm based on improved A* and artificial potential field for unmanned surface vehicle formations," Ocean Engineering, vol. 223, p. 108709, Mar. 2021, doi: 10.1016/j.oceaneng.2021.108709.

[34] S. Sedighi, D.-V. Nguyen, and K.-D. Kuhnert, "Guided Hybrid A* Path Planning Algorithm for Valet Parking Applications," in 2019 5th International Conference on Control, Automation and Robotics (ICCAR), Apr. 2019, pp. 570–575. doi: 10.1109/ICCAR.2019.8813752.

[35] Z. Chen, Y. Zhang, Y. Zhang, Y. Nie, J. Tang, and S. Zhu, "A Hybrid Path Planning Algorithm for Unmanned Surface Vehicles in Complex Environment With Dynamic Obstacles," IEEE Access, vol. 7, pp. 126439–126449, 2019, doi: 10.1109/ACCESS.2019.2936689.