Optimizing Route Planning for Autonomous Electric Vehicles Using the D-Star Lite Algorithm

Bhakti Yudho Suprapto, Suci Dwijayanti, Desi Windisari, Gatot Aria Pratama
Department of Electrical Engineering, Universitas Sriwijaya, Inderalaya, South of Sumatera, Indonesia

Abstract*—Every vehicle, including autonomous vehicles, requires a route to navigate its journey. Route planning is a critical aspect of autonomous vehicle operations, as these vehicles rely on guided paths or sequential steps to move effectively. Ensuring that the route is optimal is a key consideration. This study tests the D-Star Lite algorithm to determine the most efficient route. In simulation tests, the D-Star Lite algorithm was compared with the A-Star algorithm. The results showed that D-Star Lite outperformed A-Star, achieving an average distance reduction of 124 meters. Real-time testing involved finding a route from node 36 to node 0, resulting in a total distance of 803 meters. Additional tests focused on route replanning in real-time scenarios. For instance, the initial route passing through nodes 36 ➝37➝38➝39➝40➝41➝42➝43➝44➝45➝0 was adjusted to an alternative route: 36➝37➝38➝46➝26➝11➝2➝4➝1➝0. Based on the results, the D-Star Lite algorithm proves effective in identifying the best route for autonomous electric vehicles while also enabling real-time route replanning.

Keywords*—*Autonomous vehicle; D-Star Lite; path planning; realtime; replanning route; optimal route

I. Introduction

Recent advancements in computing and communication technologies have significantly contributed to the development of autonomous vehicles. The emergence and evolution of these vehicles are the results of research in fields such as wireless communication technology, navigation, sensor technology, ad hoc networking, data acquisition and distribution, and data analysis [1] [2]. In addition to their ability to navigate autonomously to destinations, other critical factors must be considered in autonomous vehicles, such as the time required to reach the destination [3]. To plan movements efficiently, it is equally important to consider the routes the vehicle will follow [4].

Route planning is a critical aspect of robotics. Autonomous robots and vehicles require guidance paths or next steps to navigate effectively [5]. Defining a destination coordinate as a target ensures that the robot or autonomous vehicle can reach that destination via a predetermined route. This route must be the fastest to optimize efficiency and effectiveness while avoiding unnecessary steps [6]. This aspect is crucial to ensuring the efficiency and accuracy of the vehicle's movement [7]. Therefore, using a path planning method that works in dynamic environments is essential to handle obstacles that may change or be unpredictable [8]. Several methods can be used to determine the route, such as the Dijkstra algorithm, A-star, and D-Star Lite [9].

In research on the Dijkstra algorithm, it is explained that this algorithm is used for route planning in smart cars by implementing both the Dijkstra algorithm and the dynamic window approach [10]. This method was successfully applied to a self-developed smart car to avoid obstacles and reach a predetermined position. The study involved both simulation experiments and real-world testing, demonstrating the effectiveness and reliability of the Dijkstra algorithm and the implemented system.

In research on the A-star (A*) algorithm, improvements were made to the A-star-based path planning algorithm implemented in autonomous vehicles [11]. These improvements cover various aspects, such as the use of evaluation standards to measure performance, incorporating human guidance or global path planning to develop heuristic functions, leveraging key points around obstacles for more effective avoidance, and applying the variable-step-based A-star algorithm to reduce computation time [11].

However, both of these algorithms have their drawbacks. The Dijkstra algorithm is categorized as a greedy algorithm that can optimally find the shortest path solution [12]–[14], but it requires a longer search time. On the other hand, the A-star algorithm is a best-first search algorithm that can find the shortest path more quickly but does not always produce an optimal solution [15]. Nevertheless, A-star has an advantage over Dijkstra in its calculations. The A-star algorithm utilizes a heuristic distance [16] added to the straight-line path, resulting in a more efficient route. A-star is well-suited for situations where it is important to find a path quickly and efficiently in various environments [17]. Both the Dijkstra and A-star algorithms are only effective in solving the path search process in static environments (where conditions do not change) [18]. However, for an autonomous vehicle to adapt to unknown and potentially changing road conditions, a path planning algorithm that can be implemented in dynamic (changing) environments is needed. Therefore, the D-Star Lite algorithm will be developed.

The D-star Lite algorithm can address the efficiency issues of other algorithms when used in dynamic environments [18]. In previous research [10] and [11] , path planning algorithms have been implemented using autonomous vehicles, some of which involved simulations. However, none of the studies using Dijkstra or A-star algorithms have been able to perform real-time replanning when obstacles change. In the study [19] that discusses the D-star Lite algorithm, the goal was to design a modified D-star Lite algorithm for global path planning in UAV-based (unmanned aerial vehicle) and mobile robots in large-scale disaster areas. This algorithm aims to address the challenges of dynamic environments that are only partially known by providing shorter paths and faster execution times, ultimately improving the performance and efficiency of rescue robots in such situations. Although in study [19] explores the use of the D-Star Lite algorithm for path planning in dynamic environments with UAVs and mobile robots, no study has implemented the D-star Lite algorithm for path planning in autonomous vehicles. Therefore, this study uses the D-Star Lite algorithm to determine the fastest route for autonomous electric vehicles.

The contributions of this study are as follows: implementing the D-Star Lite algorithm to determine the fastest route for the autonomous vehicle, with real-time testing performed using a path on the Universitas Sriwijaya campus, which represents road conditions in Indonesia. Additionally, the study compares the D-Star Lite algorithm with other well-known path planning algorithms.

The paper is organized as follows: Section II explains path planning, the method is presented in Section III, Section IV discusses the results and findings, and finally, the paper is concluded in Section V.

II. Path Planning

Path planning is a technique used to determine the best route for an autonomous electric vehicle to move from its current position to the desired destination while avoiding obstacles along the way [9]. Based on the environment in which it is applied, path planning can be performed in either static or dynamic environments.

In a static environment, obstacles have fixed positions and do not change location. In contrast, in a dynamic environment, obstacles may be partially known or entirely unknown, and their positions can change over time.

There are two types of path planning: global and local path planning.

1) Global path planning: Global path planning involves determining the route from the starting point to the destination within a larger environment. This requires extensive mapping and information about the robot's initial position and target destination. The focus is on finding the optimal route to reach the goal without considering the detailed surroundings near the robot. The global path planning process typically takes more time, as it involves analyzing the entire environment to identify the best route over a larger distance.

2) Local path planning: Local path planning focuses on determining the route around the robot's current position. Its primary objective is to avoid nearby obstacles and ensure the robot reaches its destination safely and efficiently. Local path planning is faster to execute because it focuses on a smaller area surrounding the robot.

Both approaches are essential for enabling autonomous vehicles to navigate complex environments effectively, combining broad-route optimization with immediate obstacle avoidance to ensure safety and efficiency.

B. Lifelong Planning A-star Algorithm (LPA*)

The Lifelong Planning A-Star (LPA*) algorithm is an enhancement of the A-Star algorithm. LPA* is an incremental version of A-Star, enabling it to adapt to changing environments by utilizing two key values: g(s), which represents the cost accumulated so far to move from the current node to the start node (the formula for calculating g(s) is provided in Eq. (1) [20]) and rhs(s), which represents the best-known cost to reach a node from the start node (its formula is provided in Eq. (2) [20]).

By leveraging these two values, the LPA* algorithm efficiently recalculates paths as the environment changes, making it well-suited for dynamic scenarios.

$$g(s) = g(s') + d(s',s) \quad (1)$$

$$rh(s) = min_{s' \in neighbor(s)}((g(s') + d(s',s)) \quad (2)$$

where $g(s)$ is the cost to move from the start node to the current node, s is current node, s' is the predecessor node, and $d(s,s))$ is the cost of moving from the predecessor node to the current node.

If g(s) = rhs(s), the node can be considered consistent**.** However, if the calculated node is inconsistent, it indicates a possible error in the calculation process.

In the LPA* algorithm, a priority queue is used to store nodes that are known and need to be evaluated or updated. Each node in the priority queue is assigned a key value, which determines the priority of the node. Nodes with the smallest key value are evaluated and updated first.

The function used to determine the key value of each node is provided in Eq. (3). This mechanism ensures that the algorithm efficiently processes nodes in the correct order, maintaining accuracy and minimizing computational overhead.

$$k(s) = \min(g(s), rhs(s)) + h(s) \quad (3)$$

where s is current node, *g(s)* is g-value of the current node, *rhs(s)* is rhs-value of current node and *h(s)* is heuristic value of the current node.

C. D-star Lite (D* Lite) algorithm

The D-Star Lite algorithm, first developed by Sven Koenig and Maxim Likhachev in 2002, is a path planning algorithm capable of optimally finding a route between a start point and a goal point in environments that are known, partially known, or dynamic.

This algorithm operates on a data structure consisting of interconnected nodes. A node leading to the current position is called a predecessor node, while a node that will be traversed next is referred to as a successor node.

D-Star Lite is based on the Lifelong Planning A-Star (LPA*) algorithm, an incremental version of A-Star that adapts to changes in the map graph. However, unlike traditional approaches, D-Star Lite performs route planning starting from the goal node (finish) and works toward the start node. In this context, the *g(s)* value represents the estimated cost from the current node to the goal node.

This reverse planning approach allows D-Star Lite to efficiently handle replanning when changes occur in the environment. The algorithm achieves this by maintaining an estimated cost for each traversed node, representing the distance to the goal node. This capability makes D-Star Lite particularly well-suited for dynamic and unpredictable environments.

D* Lite uses distance as a fundamental component because it is a path planning algorithm designed to find the shortest or least costly path between a start point and a goal. Here's why distance plays such a central role [7] [21] [22] [23]:

1) Core purposes path planning: The primary objective of D* Lite is to navigate an autonomous vehicle efficiently from a start point to a goal while avoiding obstacles. Distance or cost is the metric used to evaluate the optimality of the path. This ensures that the agent follows the shortest or least costly route, saving time, energy, or other resources.

2) Adaptation to dynamic information: In dynamic and partially known environments, the map can change due to new obstacles or updated information. D* Lite re-evaluates the distance (or cost) between nodes when changes occur, allowing the algorithm to efficiently update the path without recalculating everything from scratch. This incremental approach relies on comparing distances to ensure the agent can still reach the goal optimally.

3) Grid representation and node expansion: D* Lite often uses a grid or graph-based representation of the environment where nodes represent possible positions, and edges represent paths between these positions. The algorithm assigns a cost to each edge, typically based on physical distance or other factors like terrain difficulty. Calculating the shortest path through these nodes inherently involves summing distances or costs.

4) Real-world relevance: Distance is a straightforward and intuitive metric that directly translates to practical scenarios. Whether it's minimizing travel time, energy consumption, or fuel usage, distance serves as a universal measure of efficiency. For example, in rescue operations, D* Lite's reliance on distance ensures that the robot can reach victims or resources quickly.

D. Euclidean Distance

The Euclidean distance is a technique used to measure the distance between two points by considering the straight-line distance between them, not the angles. In Euclidean distance measurement, the calculation is conducted within a single plane and involves applying the Pythagorean theorem.

This method is commonly used to compute the distance between nodes and to determine heuristic values in the D-Star Lite algorithm. It achieves this by utilizing longitude and latitude values obtained from GPS sensors.

The formula for Euclidean distance is provided in the equation below, offering a straightforward way to calculate the straight-line distance between two points in a given space.

$$h = \sqrt{(x_{destination} - x_{start})^2 + (y_{destination} - y_{start})^2} \quad (4)$$

With x is the heuristic distance value, $x_{destination}$ is the longitude value of the target position, x_{start} is the longitude value of the starting position, $y_{destination}$ is the latitude value of the target position, and y_{start} is the latitude value of the starting position.

Eq. (4) above can be used to calculate the distance between two coordinate points, which will be applied in the D-star Lite algorithm. To obtain the distance in kilometers, Eq. (4) must be multiplied by the Earth's degree value, approximately 111.319888.

III. Method

A. Design System

In this study, the system design is presented in the form of a flowchart, as shown in Fig. 1 below. The flowchart illustrates the stages involved in determining the optimal route for an autonomous electric vehicle, as well as the steps taken to replan the route if obstacles are encountered.

In Fig. 1, the process begins with reading GPS data via ROS, followed by inputting the target node. The D-Star Lite algorithm determines the optimal route by identifying the direction of the next node based on the previous heading. The autonomous vehicle then starts moving toward the next node.

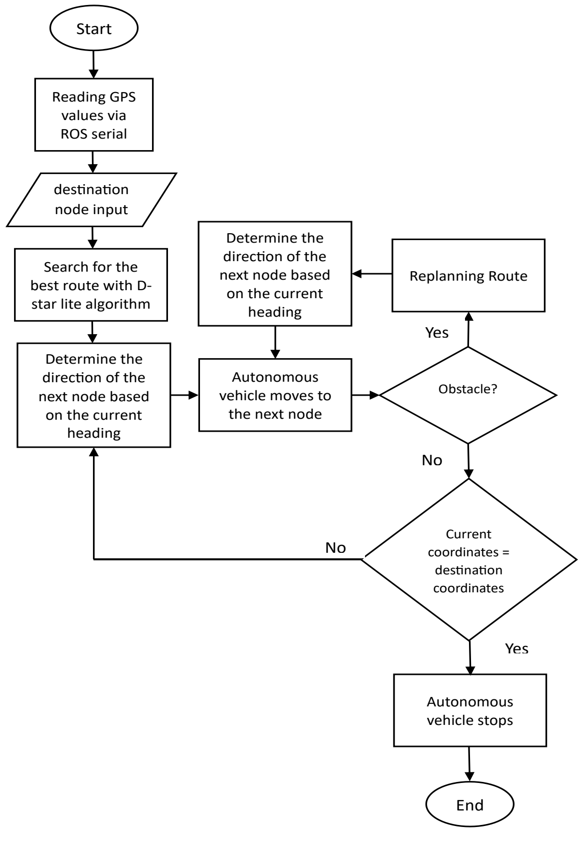

Start
Reading GPS values via ROS serial
destination node input
Search for the best route with D-star lite algorithm
Determine the direction of the next node based on the current heading
Autonomous vehicle moves to the next node
Obstacle?
Yes
Replanning Route
Determine the direction of the next node based on the current heading
No
Current coordinates = destination coordinates
No
Yes
Autonomous vehicle stops
End

Fig. 1. Flowchart of path planning system design.

If obstacles are encountered along the route, the D-Star Lite algorithm will replan and determine a new direction for the next node. The autonomous vehicle will continue its movement. If no obstacles appear along the path, the system will check the vehicle's current position. If the current coordinates match the target coordinates, the autonomous vehicle will stop, indicating that it has reached the desired destination. The route search process using the D-Star Lite algorithm must be able to replan if obstacles are detected while the autonomous vehicle is moving toward the target point. The flow diagram for the designed software can be seen in Fig. 2.

In Fig. 2, it can be seen that the algorithm initially reads the coordinate values from the GPS system, which are transmitted via ROS serial communication. After obtaining the initial coordinates, the current coordinates are determined. The next step is to define the destination or target node. The D-Star Lite algorithm calculates the global path from the current node to the target node. The target node result is then sent to the controller via ROS serial communication.

The camera sensor provides image data that is sent to a computer for identification processing, which then sends input to the controller. If the camera detects an obstacle, the D-Star Lite algorithm adjusts the route and performs replanning, which is transmitted via ROS serial communication. However, if no obstacle is detected, the movement continues until the target node is reached.

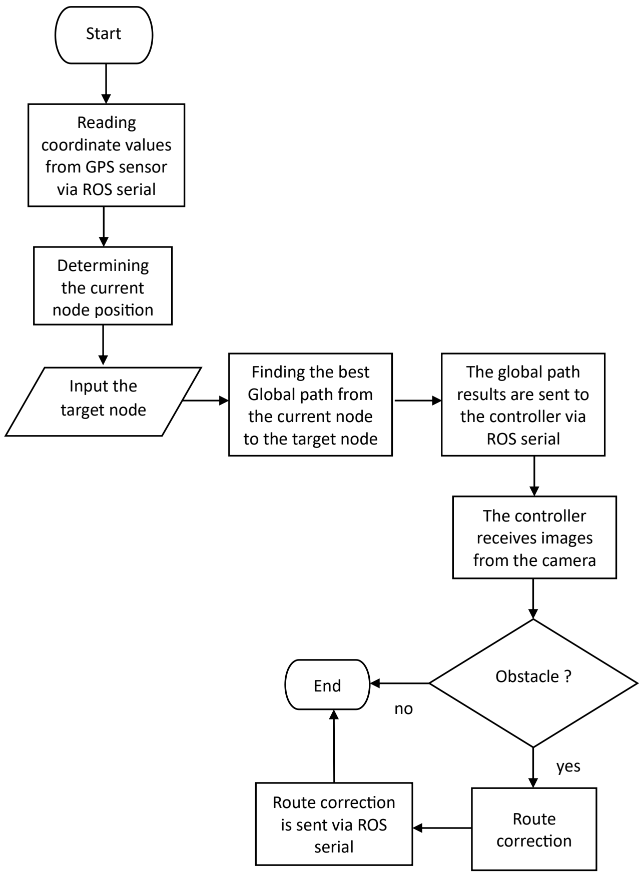

Start
Reading coordinate values from GPS sensor via ROS serial
Determining the current node position
Input the target node
Finding the best Global path from the current node to the target node
The global path results are sent to the controller via ROS serial
The controller receives images from the camera
Obstacle ?
no
yes
End
Route correction
Route correction is sent via ROS serial

Fig. 2. The flowchart of design software.

B. Route Data

At this stage, longitude and latitude coordinate data are collected directly at each point designated as a node. A total of 47 longitude and latitude data points were obtained during this process, which will be used for testing purposes in both simulations and real-time scenarios. The nodes are labeled with numbers from 0 to 46, as shown in Table I.

TABLE I. NODE POINTS ON THE CAMPUS OF UNIVERSITAS SRIWIJAYA INDRALAYA

Node	Longitude	Latitude	Description
0	-3.21738259	104.64643749	Engineering Faculty
1	-3.21667979	104.64656550	North of the Faculty of Engineering T-junction
2	-3.21545079	104.64774530	The Faculty of Medicine Intersection
3	-3.21548959	104.64955100	The Southern T-Junction of the Rectorate
4	-3.21667550	104.64773890	The Eastern Intersection of the Library
5	-3.21666990	104.64954630	The Western Intersection of the Library
6	-3.21667029	104.65052800	Faculty of Social and Political Sciences Intersection
7	-3.21737769	104.65052250	South of Faculty of Social and Political Sciences Intersection
8	-3.21668260	104.65088070	T-Junction of Faculty of Social and Political Sciences
9	-3.21735050	104.65089470	South of the FISIP T-Junction
10	-3.21399860	104.65086760	The Northern Intersection of the Faculty of Law
11	-3.21385989	104.64773350	Auditorium intersection
12	-3.21820369	104.65055840	South intersection of Faculty of Economics
13	-3.21911079	104.65051810	Intersection of Faculty of Computer Science
14	-3.21950100	104.64873060	West intersection of Faculty of Agriculture
15	-3.21804735	104.64875234	Intersection behind the library
16	-3.21564319	104.64739540	Faculty of Medicine
17	-3.21670539	104.64872703	Library
18	-3.21391629	104.64873580	Landmark UNSRI
19	-3.21644240	104.65090500	Faculty of Social and Political Sciences
20	-3.21536880	104.65088300	Faculty of Law
21	-3.21795930	104.65054590	Faculty of Economics
22	-3.21949390	104.64932110	Faculty of Teacher Training and Education
23	-3.21855209	104.64639790	Faculty of Mathematics and Natural Sciences
24	-3.21950639	104.64806430	Faculty of Agriculture
25	-3.21911473	104.65089650	Faculty of Computer Science
26	-3.21397368	104.64540105	Faculty of Public Health
27	-3.21735675	104.64956288	South of node 5
28	-3.21805961	104.64956063	South of node 27
29	-3.21903399	104.64654099	South of Faculty of Mathematics and Natural Sciences
30	-3.21945477	104.64699562	South of node 29

31	-3.21843234	104.64683263	South of the canteen intersection
32	-3.21803595	104.64686478	Intersection of canteen
33	-3.21940914	104.65020614	West of Faculty of Teacher Training and Education
34	-3.21579602	104.65035667	North of node 6
35	-3.21394241	104.64953128	Rectorate
36	-3.21736794	104.64538617	Department of Electrical Engineering
37	-3.21730889	104.64475116	East intersection of Electrical Engineering Department
38	-3.21735422	104.64370186	T-junction of Faculty of Engineering
39	-3.21894491	104.64379309	T-junction of south node 38
40	-3.21884979	104.64427315	West of node 39
41	-3.21838310	104.64493115	West of node 40
42	-3.21834656	104.64501846	Behind of Department of Mechanical Engineering
43	-3.21793954	104.64534572	Behind of Department of Electrical Engineering
44	-3.21791539	104.64564074	East of node 45
45	-3.21821540	104.64647486	T-junction of Faculty of Mathematics and Natural Sciences
46	-3.21396715	104.64410885	T-junction of Faculty of Public Health

The mapping of these 46 nodes is shown in Fig. 3.

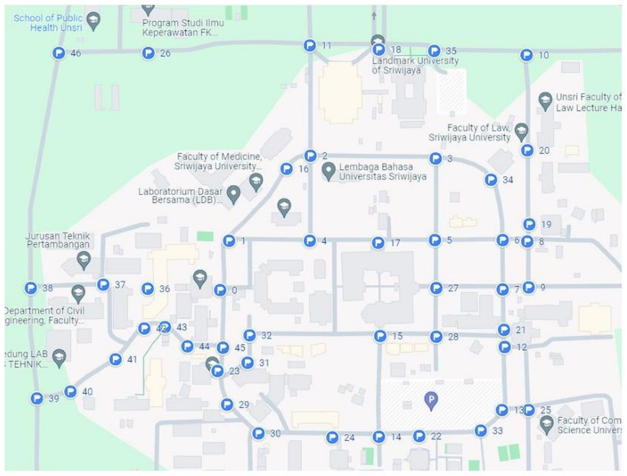

School of Public Health Unsri
Program Studi Ilmu Keperawatan FK
Landmark University of Sriwijaya
Unsri Faculty of Law Lecture Ha
Faculty of Law, Sriwijaya University
Faculty of Medicine, Sriwijaya University
Lembaga Bahasa Universitas Sriwijaya
Laboratorium Dasar Bersama (LDB)
Jurusan Teknik Pertambangan
Department of Civil Engineering, Faculty
Gedung LAB TEHNIK
Faculty of Computer Science Univer

Fig. 3. The mapping routes on the Universitas Sriwijaya Indralaya campus.

In this study, the selected location is the road around the Inderalaya campus of Sriwijaya University because the roads in this area have challenging characteristics, such as the absence of road markings, road barriers, and the surface condition of the roads which is not very smooth. The roads around the Inderalaya campus reflect those in rural areas of South Sumatra Province in general. In terms of traffic density, it is not as congested as rural roads in Sumatra, but it is already quite busy due to the many students who use the roads by riding motorcycles, driving cars, or taking buses.

IV. Results and Discussions

A. Path Planning Testing Through Simulation

In this testing, the path planning system is evaluated using the D-Star Lite algorithm to determine whether the developed system functions properly. A comparison will also be made between the route search results using the D-Star Lite algorithm and the A-Star algorithm from previous research. This experiment involves finding the best route across 10 different routes. In the first trial, the route search was tested from the Faculty of Engineering to the Faculty of Law. The results of this test are presented in Table II, and the traversed route is shown in Fig. 4.

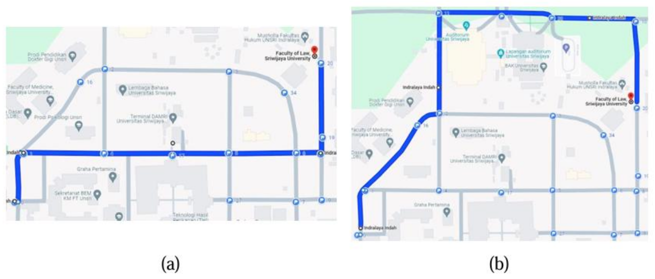


Fig. 4. Route from the faculty of engineering to the faculty of law: (a) D-Star Lite method, (b) A-Star method.

TABLE II. Testing the Route from the Faculty of Engineering to the Faculty of Law

Method	*Nodes skipped*	Total euclidean distance (m)	Distance based on google maps (m)
D-Star lite	0➔1➔4➔17➔5➔6➔8➔19➔ 20	704,9	702
A-Star	0➔1➔16➔2➔11➔18➔35➔10➔20	948,9	948
Distance difference (m)		244	246

In the second trial, the route search was tested from the Faculty of Engineering to the Faculty of Economics. The results of this test are presented in Table III, and the traversed route is shown in Fig. 5.

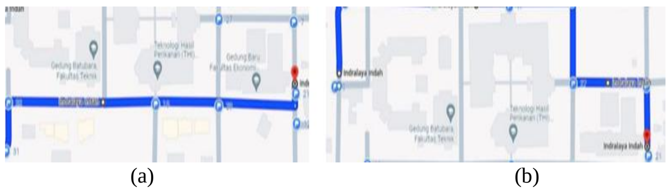
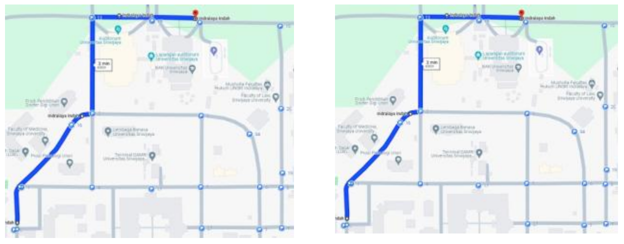


Fig. 5. Route from the Faculty of Engineering to the Faculty of Economics: (a) D-Star Lite Method, (b) A-Star Method

TABLE III. Testing the Route from the Faculty of Engineering to the Faculty of Economics

Method	Nodes skipped	Total euclidean distance (m)	Distance based on google maps (m)
D-Star lite	0➔45➔23➔31➔32➔15➔28➔21	633,6	633
A-Star	0➔1➔4➔17➔5➔27➔7➔21	658,1	658,3
Distance difference (m)		24,5	25,3

In the third trial, the route search was tested from the Faculty of Engineering to the Rectorate. The results of this test are presented in Table IV, and the traversed route is shown in Fig. 6.



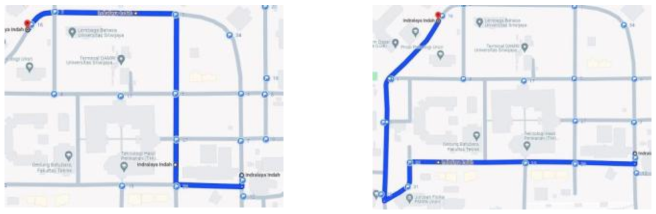
(a) (b)

Fig. 6. Route from the faculty of engineering to the rectorate: (a) D-Star lite method, (b) A-Star method.

TABLE IV. TESTING THE ROUTE FROM THE FACULTY OF ENGINEERING TO THE FACULTY OF ECONOMICS

Method	Nodes skipped	Total euclidean distance (m)	Distance based on google maps (m)
D-Star lite	0➔1➔16➔ 2➔11➔18➔35	648	650,6
A-Star	0➔1➔16➔2➔1 1➔18➔35	648	650,6
Distance difference (m)		0	0

In the fourth trial, the route search was tested from the Faculty of Economics to the Faculty of Medicine. The results of this test are presented in Table V, and the traversed route can be seen in Fig. 7.



(a) (b)

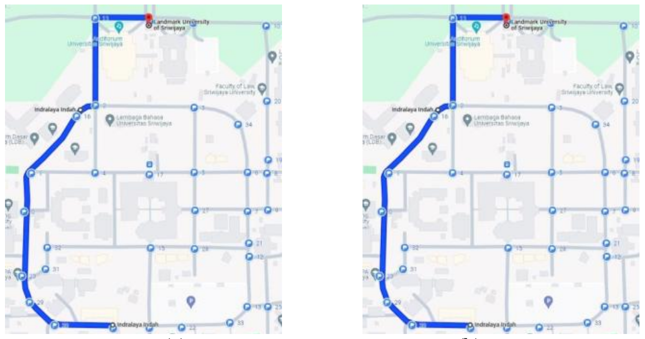
Fig. 7. Route from the faculty of economics to the faculty of medicine: (a) D-Star lite method, (b) A-Star method.

TABLE V. TESTING THE ROUTE FROM THE FACULTY OF ECONOMICS TO THE FACULTY OF MEDICINE

Method	Nodes skipped	Total euclidean distance (m)	Distance based on google maps (m)
D-Star lite	21➔28➔27➔5➔3➔2 ➔16	640,6	640,2
A-Star	21➔28➔15➔32➔31 ➔ 23➔45➔0➔1➔16	860,6	858,6
Distance difference (m)		220	218,4

In the fifth trial, the route search was tested from the Faculty of Agriculture to the landmark. The results of this test are presented in Table VI, and the traversed route is shown in Fig. 8.

From the five trials conducted, the D-Star Lite algorithm shows a larger error in comparison to Google Maps readings than the A-Star algorithm. However, when comparing the routes taken and the best route searches, the D-Star Lite algorithm outperforms the A-Star algorithm. This is evident in the first, second, and fourth trials, with the largest difference being 244 meters in the second trial. This occurs because the A-Star algorithm prioritizes only the nodes leading directly to the destination as the best route, whereas the D-Star Lite algorithm evaluates each node in the dataset to determine the shortest path to the destination. Consequently, the D-Star Lite algorithm sometimes finds a more optimal route than the A-Star algorithm. Therefore, the D-Star Lite algorithm is a viable method for finding the best route.



(a) (b)

Fig. 8. Route from the faculty of agriculture to the landmark: (a) D-Star lite method, (b) A-Star method.

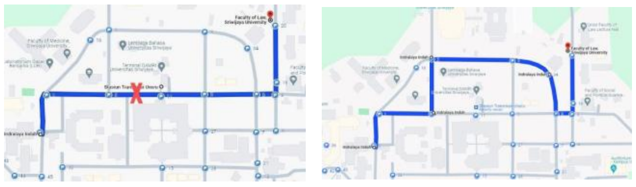
TABLE VI. TESTING THE ROUTE FROM THE FACULTY OF ECONOMICS TO THE FACULTY OF MEDICINE

Method	Nodes skipped	Total Euclidean distance (m)	Distance based on google maps (m)
D-Star lite	24➔30➔29➔23➔45➔0 ➔1➔16➔2➔11➔18	933,3	934,2
A-Star	24➔30➔29➔23➔45➔0 ➔1➔16➔2➔11➔18	933,3	934,2
Distance difference (m)		0	0

B. *Route Replanning Testing via Simulation*

In this simulation test, the replanning system using the D-Star Lite algorithm was tested to determine whether it could successfully perform route replanning when an obstacle appeared on the route. This experiment included five tests to evaluate whether the D-Star Lite algorithm's replanning system could be used in real-time conditions.

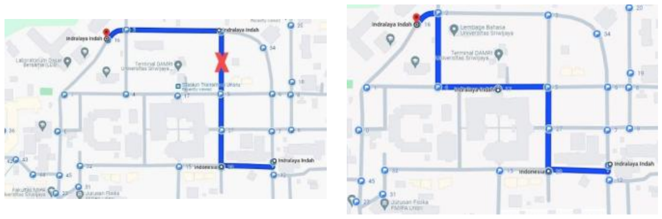
In the first trial, a route search was conducted from the Faculty of Engineering to the Faculty of Law. The best route identified passed through nodes0 ➔ 1 ➔ 4 ➔ 17 ➔ 5 ➔ 6 ➔ 8 ➔ 19 ➔ 20. After establishing the route, node 17 was designated as an obstacle or closed, prompting the D-Star Lite algorithm's replanning system to search for the best alternative route avoiding the closed node. The resulting route passed through nodes 0 ➔ 1 ➔ 4 ➔ 2 ➔ 3 ➔ 34 ➔ 6 ➔ 8 ➔ 19 ➔ 20, as shown in Fig. 9.



(a) (b)

Fig. 9. Replanning route from the faculty of engineering to the faculty of law (a) Before replanning (b) After replanning.

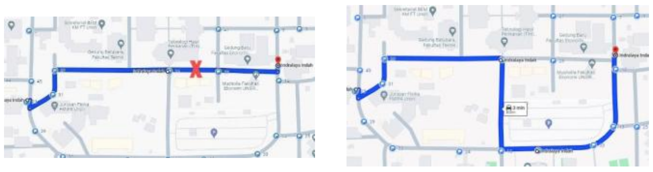
In the second trial, a route search was conducted from the Faculty of Economics to the Faculty of Medicine. The best route identified passed through nodes 21 ➔ 28 ➔ 27 ➔ 5 ➔ 3 ➔ 2 ➔ 16. After establishing the route, node 3 was designated as an obstacle or closed, prompting the D-Star Lite algorithm's replanning system to search for the best alternative route, avoiding the closed node. The resulting route passed through nodes 21 ➔ 28 ➔ 27 ➔ 5 ➔ 17 ➔ 4 ➔ 2 ➔ 16, as shown in Fig. 10.



(a) (b)

Fig. 10. Replanning route from the Faculty of Economics to the Faculty of Medicine (a) Before replanning (b) After replanning.

In the third trial, a route search was conducted from the Faculty of Mathematics and Natural Sciences to the Faculty of Economics. The best route identified passed through nodes 23 ➔ 31 ➔ 32 ➔ 15 ➔ 28 ➔ 21. After establishing the route, node 28 was designated as an obstacle or closed, prompting the D-Star Lite algorithm's replanning system to search for the best alternative route, avoiding the closed node. The resulting route passed through nodes 23 ➔ 31 ➔ 32 ➔ 15 ➔ 14 ➔ 22 ➔ 33 ➔ 13 ➔ 12 ➔ 21, as shown in Fig. 11.

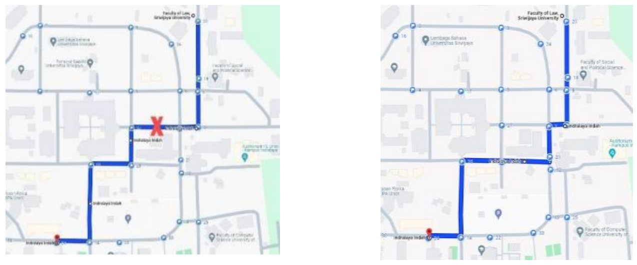


(a) (b)

Fig. 11. Replanning route from the faculty of mathematics and natural sciences to the faculty of economics (a) Before replanning (b) After replanning.

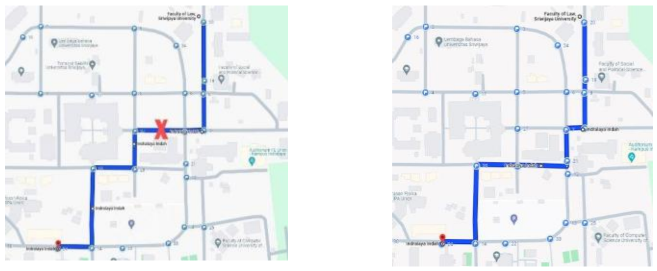
In the fourth trial, a route search was conducted from the Faculty of Law to the Faculty of Agriculture. The best route identified passed through nodes 20 ➔ 19 ➔ 8 ➔ 9 ➔ 7 ➔ 27 ➔ 28 ➔ 15 ➔ 14 ➔ 24. After the route was established, node 27 was designated as an obstacle or closed, prompting the D-Star Lite algorithm's replanning system to search for the best alternative route, avoiding the closed node 27. The resulting route passed through nodes 20 ➔ 19 ➔ 8 ➔ 9 ➔ 7 ➔ 21 ➔ 28 ➔ 15 ➔ 14 ➔ 24, as shown in Fig. 12.

In the fifth trial, a route search was conducted from the Faculty of Public Health to the Faculty of Law. The best route identified passed through nodes 26 ➔ 11 ➔ 18 ➔ 35 ➔ 10 ➔ 20. After establishing the route, node 18 was designated as an obstacle or closed, prompting the D-Star Lite algorithm's replanning system to search for the best alternative route, avoiding the closed node. The resulting route passed through nodes 26 ➔ 11 ➔ 2 ➔ 3 ➔ 34 ➔ 6 ➔ 8 ➔ 19 ➔ 20, as shown in Fig. 13.



(a) (b)

Fig. 12. Replanning route from the faculty of law to the faculty of agriculture (a) Before replanning (b) After replanning.



(a) (b)

Fig. 13. Replanning route from the faculty of public health to the faculty of law (a) Before replanning (b) After replanning.

From the five route replanning trials conducted, it is evident that the route replanning system using the D-Star Lite algorithm successfully performs the route replanning process. Therefore, when an obstacle or blockage occurs, it generates a new optimal route to follow. Consequently, the D-Star Lite algorithm is suitable for real-time route replanning system testing.

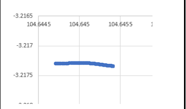
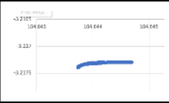
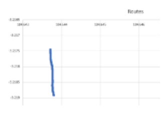
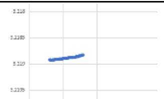
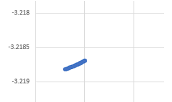
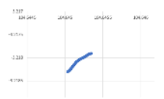
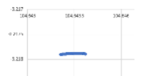
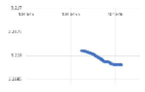
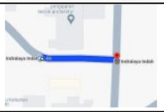
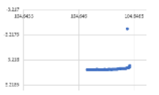
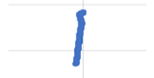
C. *Real-time Path Planning Testing*

Next, this section discusses the path planning system testing under real-time conditions. In this test, an autonomous electric vehicle is used, with its position monitored in real-time via GPS. The objective is to evaluate the path planning system, designed with the D-Star Lite algorithm, to guide the autonomous electric vehicle towards its destination by following the optimal route.

In this test, the autonomous electric vehicle will move from its starting position, the Digital Control Laboratory in the Electrical Engineering Department (node 36), to its destination, the Faculty of Engineering Dean's office building (node 0). The best route will then be determined from the starting position to the destination. The optimal route found passes through nodes 36 ➔ 37 ➔ 38 ➔ 39 ➔ 40 ➔ 41 ➔ 42 ➔ 43 ➔ 44 ➔ 45 ➔ 0. For the autonomous electric vehicle to reach the destination, it must pass through 10 node stages. The path taken is shown in Table VII.

In the real-time path planning tests conducted with an electric vehicle, as shown in Table VII, the autonomous electric vehicle successfully reached the target position by following the optimal route determined by the D-Star Lite algorithm. This demonstrates that the D-Star Lite algorithm is an effective method for finding the best route for autonomous electric vehicles.

TABLE VII. REAL-TIME PATH PLANNING TESTING

Node stages	Total distance (m)	Google maps distance (m)	Route based on google maps	Route taken electric vehicle
36 ➔ 37	70,8	70,9		
37 ➔ 38	116,6	116,5		
38 ➔ 39	177,1	181,1		
39 ➔ 40	54,3	54,5		
40 ➔ 41	89,6	89,1		
41 ➔ 42	67	67,8		
42 ➔ 43	32,8	32,9		
43 ➔ 44	46,6	48		
44 ➔ 45	55,6	56,8		
45 ➔ 0	92,6	92,4		

D. Real-Time Route Replanning Test

In this real-time route replanning experiment, a direct test will be conducted using an autonomous electric vehicle to determine whether the route replanning system of the D-Star Lite algorithm can effectively adjust the route when encountering obstacles in real-time conditions.

In this test, the autonomous electric vehicle is programmed to move from its starting point at the Digital Control Laboratory to the Faculty of Engineering Dean's office. The planned route passes through the following nodes: 36 ➔ 37 ➔ 38 ➔ 39 ➔ 40 ➔ 41 ➔ 42 ➔ 43 ➔ 44 ➔ 45 ➔ 0.

However, when the autonomous electric vehicle reaches node 38 and detects an obstacle blocking the path to node 39, the system identifies this path as impassable. The road closure toward node 39 is illustrated in Fig. 14.

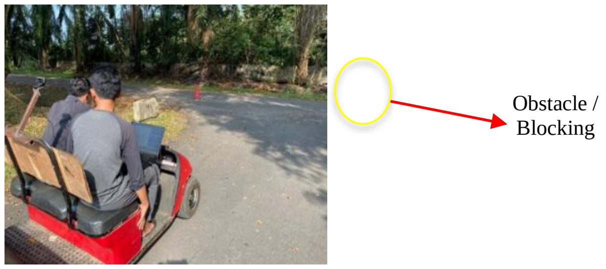
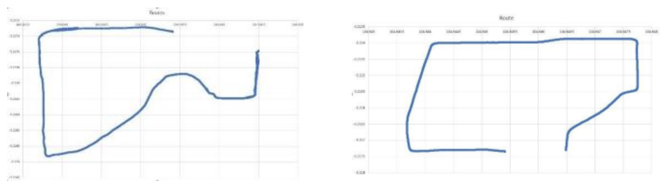


Fig. 14. Road closure condition toward node 39.

Fig. 14 shows the visual closure of the road to node 39. This road closure occurs when the autonomous electric vehicle detects an obstacle blocking the path. The D-Star Lite algorithm handles this condition by dynamically recalculating an alternative route in real-time to ensure the vehicle can continue its journey toward the destination.

Once the road closure is detected, the replanning system in the D-Star Lite algorithm is activated to recalculate and adjust the route, ensuring that the autonomous electric vehicle can still reach its predetermined destination. After the replanning process, the new route is as follows: 36 ➔ 37 ➔ 38 ➔ 46 ➔ 26 ➔ 11 ➔ 2 ➔ 4 ➔ 1 ➔ 0. A comparison between the original route (before replanning) and the new route (after replanning) is shown in Fig. 15.



(a) (b)

Fig. 15. Route replanning from the control and robotic laboratory to the faculty of engineering (a) Before replanning, (b) After replanning.

The real-time route replanning test demonstrated that the designed system can dynamically adjust the route in real-time whenever obstacles are encountered during the autonomous electric vehicle's journey toward its destination.

D-Star Lite uses distance as a core metric because it aligns with the algorithm's goal of finding optimal paths while efficiently adapting to dynamic environments. Distance serves as a universal measure of cost that simplifies computations, ensures practical relevance, and facilitates heuristic optimization.

If we compare the D-Star Lite algorithm with Dijkstra, the core characteristics are as follows: Dijkstra's algorithm is one of the earliest graph-based approaches for finding the shortest path between nodes. It is deterministic and guarantees an optimal solution by systematically exploring all possible paths in a static and fully known environment. The algorithm's primary strength lies in its simplicity and optimality for static graphs. On the other hand, D-Star Lite is a dynamic and incremental path planning algorithm designed for environments that are partially known or subject to change. It builds on the principles of Dijkstra's algorithm but introduces significant enhancements to handle real-time updates efficiently. By focusing only on affected nodes when the environment changes, D-Star Lite reduces the computational overhead typically associated with path recalculations in dynamic scenarios. For the performance: Dijkstra's algorithm guarantees optimal paths in static settings but suffers from high computational complexity in large graphs due to its exhaustive exploration. This limitation becomes apparent when applied to vast areas or dense graphs, as the algorithm must evaluate all possible nodes and edges systematically [7]. D-Star Lite, however, is optimized for efficiency in dynamic and partially known environments. It updates only the necessary parts of the graph when changes occur, significantly reducing computational demands. Techniques like auto-clustering further enhance its performance by segmenting large maps, as demonstrated in Heo et al. (2022), where the Auto-Splitting D-Star Lite method reduced unnecessary node expansions [23].

Dijkstra and D-Star Lite algorithms cater to distinct path planning requirements. Dijkstra's algorithm is ideal for static, structured environments where optimality and simplicity are paramount. D* Lite, on the other hand, is tailored for dynamic and partially known environments, offering computational efficiency and adaptability. The choice between these algorithms depends on the specific use case, environmental constraints, and computational resources. Future advancements, such as hybrid approaches or machine learning integration, may further enhance their capabilities, bridging the gap between static and dynamic path planning needs.

V. Conclusions

After conducting five trials to compare the D-Star Lite algorithm with the A-Star algorithm, it was concluded that D-Star Lite demonstrates more optimal route-finding capabilities than A-Star. The average difference in route distance between the two algorithms was 97.7 meters, with D-Star Lite consistently providing shorter routes. Additionally, D-Star Lite's ability to calculate the distance to the target at each node enables it to perform route replanning effectively when encountering obstacles.

In the conducted tests, the D-Star Lite algorithm proved capable of finding the shortest route in real-time, covering a distance of 803 meters from the starting point at the Digital Control Laboratory to the Faculty of Engineering. Furthermore, the D-Star Lite algorithm successfully performed route replanning. Initially, the route was: 36 ➔ 37 ➔ 38 ➔ 39 ➔ 40 ➔ 41 ➔ 42 ➔ 43 ➔ 44 ➔ 45 ➔ 0. After replanning due to an obstacle, the route was adjusted to: 36 ➔ 37 ➔ 38 ➔ 46 ➔ 26 ➔ 11 ➔ 2 ➔ 4 ➔ 1 ➔ 0. This study has shown the effectiveness of using the D-Star Lite algorithm in real-time applications for autonomous vehicles, even with paths containing obstacles. However, it is limited to simple obstacles. Thus, further studies are needed to improve the algorithm's handling of different types of obstacles along the vehicle's path.

Acknowledgment

The research/publication of this article was funded by DIPA of Public Service Agency of Universitas Sriwijaya 2024. No SP DIPA 023.17.2.677515/2024, On November 24, 2023. In accordance with the Rector's Decree Number: 0013/UN9/LP2M.PT/2024, On May 20, 2024.

References

[1] R. Hussain and S. Zeadally, "Autonomous Cars: Research Results, Issues, and Future Challenges," IEEE Commun. Surv. Tutorials, vol. 21, no. 2, pp. 1275–1313, 2019, doi: 10.1109/COMST.2018.2869360.

[2] J. Wang, Y. Yan, K. Zhang, Y. Chen, M. Cao, and G. Yin, "Path planning on large curvature roads using driver-vehicle-road system based on the kinematic vehicle model," IEEE Trans. Veh. Technol., vol. 71, no. 1, pp. 311–325, 2021.

[3] C. Jung, D. Lee, B. Kim, and D. H. Shim, "Lane level path planning for urban autonomous driving using vector map," in 2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), 2020, pp. 1–4.

[4] J. Yu, J. Hou, and G. Chen, "Improved Safety-First A-Star Algorithm for Autonomous Vehicles," in 2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM), Dec. 2020, pp. 706–710, doi: 10.1109/ICARM49381.2020.9195318.

[5] J. Chen et al., "Path Planning for Autonomous Vehicle Based on a Two - Layered Planning Model in Complex Environment," J. Adv. Transp., vol. 2020, no. 1, p. 6649867, 2020.

[6] A. H. Ahmad, O. Zahwe, A. Nasser, and B. Clement, "Path Planning Algorithms For Unmanned Aerial Vehicle: Classification, Performance, and Implementation," in 2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), 2023, pp. 1–6.

[7] S. Sundarraj, R. V. K. Reddy, M. B. Basam, G. H. Lokesh, F. Flammini, and R. Natarajan, "Route planning for an autonomous robotic vehicle employing a weight-controlled particle swarm-optimized Dijkstra algorithm," IEEE Access, vol. 11, pp. 92433–92442, 2023.

[8] S. Kadry, G. Alferov, and V. Fedorov, "D-Star Algorithm Modification.," Int. J. Online Biomed. Eng., vol. 16, no. 8, 2020.

[9] M. Aizat, A. Azmin, and W. Rahiman, "A survey on navigation approaches for automated guided vehicle robots in dynamic surrounding," IEEE Access, vol. 11, pp. 33934–33955, 2023.

[10] L. S. Liu et al., "Path Planning for Smart Car Based on Dijkstra Algorithm and Dynamic Window Approach," Wirel. Commun. Mob. Comput., vol. 2021, 2021, doi: 10.1155/2021/8881684.

[11] S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei, "An improved A-Star based path planning algorithm for autonomous land vehicles," Int. J. Adv. Robot. Syst., vol. 17, no. 5, pp. 1–13, 2020, doi: 10.1177/1729881420962263.

[12] X. Li, "Path planning of intelligent mobile robot based on Dijkstra algorithm," in Journal of Physics: Conference Series, 2021, vol. 2083, no. 4, p. 42034.

[13] S. W. G. Abusalim, R. Ibrahim, M. Zainuri Saringat, S. Jamel, and J. Abdul Wahab, "Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization," IOP Conf. Ser. Mater. Sci. Eng., vol. 917, no. 1, 2020, doi: 10.1088/1757-899X/917/1/012077.

[14] R. Chen, "Dijkstra' s Shortest Path Algorithm and Its Application on Bus Routing," Proc. 2022 Int. Conf. Urban Plan. Reg. Econ. 2022）, vol. 654, no. Upre, pp. 321 - 325, 2022, doi: 10.2991/aebmr.k.220502.058.

[15] A. Candra, M. A. Budiman, and K. Hartanto, "Dijkstra's and A-Star in Finding the Shortest Path: A Tutorial," 2020 Int. Conf. Data Sci. Artif. Intell. Bus. Anal. DATABIA 2020 - Proc., pp. 28–32, 2020, doi: 10.1109/DATABIA50434.2020.9190342.

[16] Y. Yan, “Research on the A Star Algorithm for Finding Shortest Path,” Highlights Sci. Eng. Technol., vol. 46, pp. 154–161, 2023.

[17] M. R. Wayahdi, S. H. N. Ginting, and D. Syahputra, “Greedy, A-Star, and Dijkstra’s algorithms in finding shortest path,” Int. J. Adv. Data Inf. Syst., vol. 2, no. 1, pp. 45–52, 2021.

[18] K. Xie, J. Qiang, and H. Yang, “Research and optimization of d-start lite algorithm in track planning,” IEEE Access, vol. 8, pp. 161920–161928, 2020.

[19] S. nyeong Heo, J. Chen, Y. chi Liao, and H. hyol Lee, “Auto-splitting D* lite path planning for large disaster area,” Intell. Serv. Robot., vol. 15, no. 3, pp. 289–306, 2022.

[20] S. Koenig and M. Likhachev, “Incremental A*,” Adv. Neural Inf. Process. Syst., 2002.

[21] P. Paliwal, “A survey of a-star algorithm family for motion planning of autonomous vehicles,” in 2023 IEEE International Students’ Conference on Electrical, Electronics and Computer Science (SCEECS), 2023, pp. 1–6.

[22] R. Chen, J. Hu, and W. Xu, “An RRT-Dijkstra-based path planning strategy for autonomous vehicles,” Appl. Sci., vol. 12, no. 23, p. 11982, 2022.

[23] S. Heo, J. Chen, Y. Liao, and H. Lee, “Auto-splitting D* lite path planning for large disaster area,” Intell. Serv. Robot., vol. 15, no. 3, pp. 289–306, 2022.