Inform : Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi
Vol.10 No.2 July 2025, P-ISSN : 2502-3470, E-ISSN : 2581-0367

Prototype of a Line Follower Smart Car Using the A Algorithm Approach for Shortest Path Finding*

Luthfie Budie Andika[1*], Armansyah[2], Mohammad Haziq Afiq bin Haji Mohammad[3]

[1,2]*Computer Science Department, Universitas Islam Negeri Sumatera Utara, Medan, Indonesia*

[3]*Pearson BTEC Level 5 Higher National Diploma in Computing, Kemuda Institute, Bandar Seri Begawan, Brunei Darussalam*

[1]luthfie0701212216@uinsu.ac.id (*)

[2]armansyah@uinsu.ac.id, [3]haziqafiq.m@gmail.com

Received: 2025-06-17; Accepted: 2025-08-10; Published: 2025-08-20

Abstract— This study aims to design and build a prototype of a smart car designed to operate independently in detecting and following paths. This prototype utilizes image processing technology, specifically OpenCV, to recognize predetermined paths, enabling the car to move efficiently without human intervention. As the brain of the system, an ESP32 microcontroller is used to regulate movement and decision-making based on the results of image processing. In addition, this study also implements the A* algorithm for finding the closest route, which allows smart cars to calculate distances and determine optimal paths by considering various intersections and alternative routes. Testing was conducted on a simulated environment consisting of 9 nodes. The A* algorithm successfully calculated the optimal route A-B-D-F-H-I, with a total travel distance of 118 cm and a total heuristic of 1,122.7710081. The results of this study show that the prototype demonstrated sTable and responsive behaviour in detecting and following the path without human assistance.

Keywords— Prototype Smart Car; Line Follower; A* Algorithm; Shortest Route Search; ESP32.

I. INTRODUCTION

The development of modern industry has led to the need for an automation system to assist humans in various ways, including the distribution system and transportation of goods [1]. One of them is a line follower robot, which can move along a specified path independently without human intervention. This type of robot is typically used in logistics, manufacturing, and warehousing sectors to increase productivity and streamline work processes [2]. This system utilizes Python programming through a webcam in real-time as a control mechanism, allowing the robot to navigate a path [3]. However, there is a problem when the path used is dynamic, so we must utilize artificial intelligence (AI) that allows the vehicle to work independently in setting its travel route. However, to develop an optimal automated steering system, a prototype robot is required that can control the vehicle according to the specified path [4].

To determine the path efficiently, the A* algorithm can be integrated due to its effectiveness in finding the optimal path. The A* algorithm, which forms the basis of Best First Search (BFS), works by summing up the movement from one node to another, as well as the approach from the node to the final destination. This allows the car to move quickly and efficiently, avoid obstacles, and find the best path [5].

By utilizing the ESP32 microcontroller as a control centre, this line follower robot can follow a path connected via Wi-Fi, allowing a webcam equipped with the OpenCV library to provide direct commands, enabling the robot to reach the destination point [6][7]. The working principle of this robot is based on colour detection, allowing it to follow the path accurately [8].

Previous research [9] discusses the application of algorithms to find the fastest typical culinary route in Palembang, Indonesia. Furthermore, research [10] has developed a moving line-following robot that utilizes a line sensor as an input reader. Where the results of the line follower robot are effective and can navigate a black track or path. According to the research [11], the robot can follow black or white paths and lift and stick objects based on their colour. The results of this study demonstrate that the robot can move goods along a stable path.

This research shares similarities with previous studies that apply the A* algorithm in robotics and automation for shortest path finding. However, the main difference lies in the focus and application; the previous research is more diverse, ranging from line-following robots to algorithm applications in gaming and healthcare, whereas this research is specifically designed to build an intelligent car prototype that integrates the OpenCV library with an ESP32 microcontroller for self-navigation. In addition, this research incorporates a more integrated aspect of system complexity as opposed to a single focus on a particular aspect in previous research.

The author hopes that this research will be carried out, which aims to design and build a smart car prototype equipped with a digital-based route search system and an ESP32 microcontroller. Through the development of this prototype, it is hoped that a system can be created that operates independently, utilizing the OpenCV library to read the entire map. Thus, it is expected that this smart car can not only move efficiently, but also optimize the journey by choosing the most effective path.

II. RESEARCH METHODOLOGY

DOI : https://doi.org/10.25139/inform.v10i2.10508

This research involves several stages in implementing A* algorithm for intelligent car prototypes. Involving data collection, data processing and analysis, model design and testing, and Final Evaluation, as shown in Fig.1.

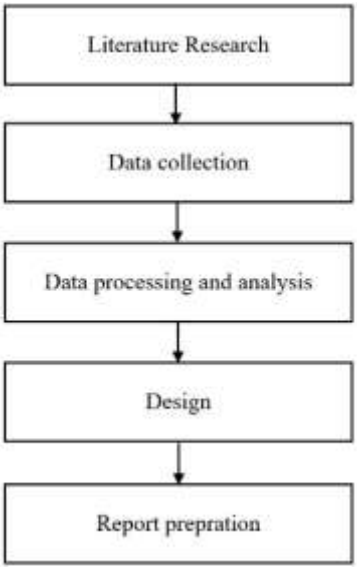


Fig. 1. Research Framework

In this study, the researcher used a private house as a research site. The private house was chosen because it provides a suitable room for route testing on the designed prototype. Additionally, this location is expected to facilitate researchers in conducting data collection and testing more effectively, resulting in more optimal research results that align with the research objectives. In the research literature, researchers explore research-related literacy by reading and studying journals to obtain data relevant to research problems, such as the closest path-finding algorithm, the A* algorithm, and robot line followers [12-15].

The data collection technique is the most important phase of the research. At this stage, researchers collect all the necessary data using various methods. When viewed from the data source, data collection can utilize both primary and secondary sources. Primary sources include data obtained directly from the design, while secondary sources include data obtained from previous studies, as relevant to the research under study [16]. In this study, the collected data will be thoroughly analyzed to determine the optimal path for the intelligent car prototype. The data is collected from a designed path with lines that follow a specific trajectory.

In this analysis, the A* algorithm will be used to find the fastest and most efficient route for the intelligent car. The analysis process involves evaluating the calculation of the cost of motion and distance estimation (heuristics) for each point passed, as well as processing data from simulation results to assess the effectiveness and accuracy of the A* algorithm in optimizing the path taken by the robot car. It is hoped that the results of this analysis will enhance our understanding of the algorithm's performance and reliability in determining the fastest route [17-20].

Before the model is tested, the design must be completed in three stages: designing prototypes and pathways to facilitate testing.

A. Prototype design

The A* algorithm is implemented in the Arduino microcontroller, where it calculates g(n) and h(n) for each node. Here is the 2D design of the prototype, which consists of an ESP32 microcontroller, a motor driver, and a DC motor [21][22]. The prototype design is illustrated in Fig. 2.

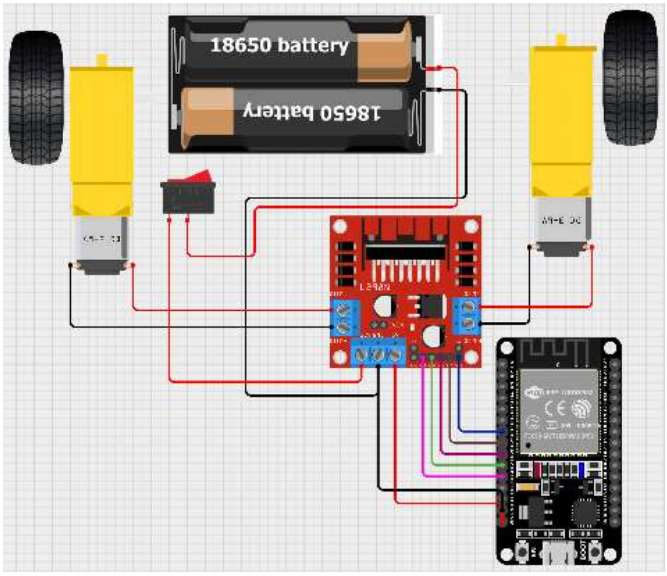


Fig.2. Prototype Circuit Design

The implementation process of the design was carried out using a programming language. In determining the fastest prototype route using the A* algorithm, the Arduino IDE is utilized as a robot tool, with the application written in the C programming language.

B. Path design

The prototype track consists of red duct tape attached to a base with a winding, track-shaped design [23]. The track is designed to enable the prototype to determine the optimal route, allowing it to reach its destination quickly and effectively. The shape of the path is illustrated in Fig.3.

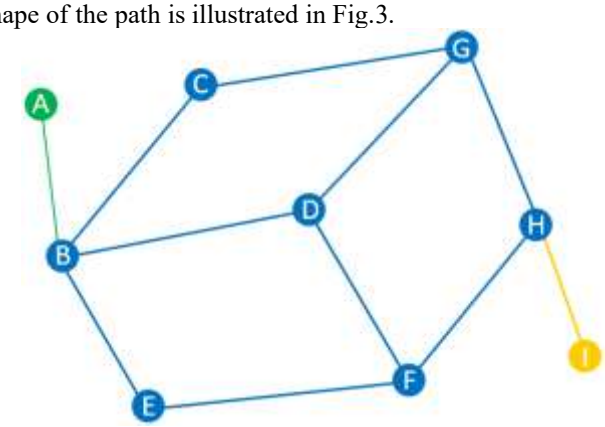


Fig. 3. Path Design

DOI : https://doi.org/10.25139/inform.v10i2.10508

C. System Design Algorithm

The A* algorithm serves to determine the shortest path from the starting position to the desired destination, while considering environmental conditions [24]. The A* algorithm is designed for a smart car prototype:

1. Insert the initial node into the open list, which stores all nodes that are candidates for exploration.
2. Find the node (n) with the lowest (f) value in the open list and make it the current node.
3. If not passable, then ignore

If it is already in the open list, then calculate the distance using Equation (1). Where, the $h(n)$ variable is the heuristic value, calculated using the Euclidean distance, which is derived from the Pythagorean theorem using Equation (2).

$$f(n) = g(n) + h(n) \quad (1)$$

$$h(n) = \sqrt{(x_{goal} - x_{current})^2 + (y_{goal} - y_{current})^2} \quad (2)$$

4. After evaluating a node, move it from the open list to the closed list, which stores nodes that have already been visited and will not be re-evaluated.
5. The search continues by selecting the next node with the lowest f f value in the open list.
6. If the destination node is found, move it to the closed list and stop the loop [24][25].

A system block diagram of a smart car prototype in determining the closest travel route with illustrations in Fig.4.

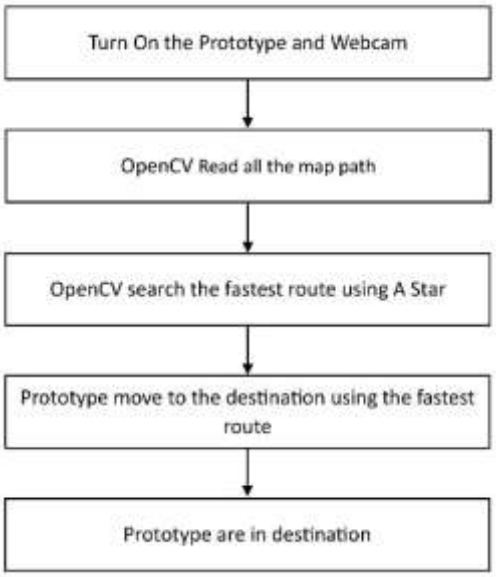


Fig.4. BFD Block

1) Prototype And Webcam Are Live: Turning on the prototype marks the initial stage, where the prototype is active and ready to operate.

2) OpenCV Library Reads the Map: The field is viewed through the webcam, where Python is used to access the webcam camera and detect the path.

3) A algorithm steps:* Determining heuristic values, calculating functions, choosing the minimum f (x) and moving cost to the closed list.

4) OpenCV Creates A Black Dot: Provide navigation directions to the destination.

5) OpenCV Orders The Prototype To Move: The black dot indicates that the prototype has reached its destination.

6) Stop Destination: The final stage of the navigation system on the prototype shows that the prototype has reached its destination.

D. Testing

At this stage, the prototype is tested using black box testing methods. Black box testing is a type of behavioural testing. Where the interior structure and logic of the software being tested are unknown to the tester. Testers are based on requirement specifications, and there is no need for code analysis. This test includes Path testing, condition testing, and loop testing. Black box testing ensures that the system functions properly in the context of the user and meets the expected specifications.

Data analysis is used to collect and process information to identify relevant data that can be utilized in this study. The A* algorithm is used to find the shortest route within the specified path. The A* algorithm combines Greedy Best-First Search (GBFS) and Dijkstra's algorithm to find the shortest path more quickly and efficiently. First, the calculation will be done manually with the $F(n)$ variable with the shortest *F(n)* variable to determine the next prototype route. Then, after obtaining the calculation results, it will be tested whether the calculated route is indeed the shortest. The shortest path is revealed after the A algorithm completes its process. In this algorithm, there is a concept of geographic cost, denoted as *G(n)*, which represents the total cost from the starting point to the current point, including obstacles such as dynamic route paths.

Then there is the concept of *F(n)*, or the total cost function, which is the sum of *G(n)* and *H(n)*. Where *H(n)* is the estimated cost from the current point to the destination. The prototype display is assembled using a box shaped like a drone, and on top of it, an ESP32 microcontroller is placed, which is used to connect the robot to OpenCV. The A* algorithm is modelled using Python within the OpenCV library. Then, when OpenCV detects the path as well as the starting point and destination point, it will provide a display of the fastest path.

III. Result and Discussion

A. Obstacle Representation

The previously collected data will be represented to solve the problem by converting it into a code format. In this prototype, the A* calculation process involves two main lists: the open list (nodes to be evaluated) and the closed list (nodes that have been visited).

The A* algorithm can predict the best path by evaluating each node based on a modified heuristic function. In the implementation, the path will be created using red duct tape. At

DOI : https://doi.org/10.25139/inform.v10i2.10508

each corner, blue origami will be placed. For the starting point, green origami will be used, and yellow origami will serve as a reference point. OpenCV will read the available paths and use the A* algorithm to determine the best route to the destination.

Experiments were conducted by creating a route path using red duct tape to simulate the A* algorithm on the prototype. In the simulation, the path followed is marked by black and red tape connecting the start and end routes. The results of this research will demonstrate the A* algorithm's ability to determine the most efficient and fastest route for the line follower car to reach the destination point. Therefore, researchers will conduct experiments to determine whether the A* algorithm can perform effectively in a dynamic prototype path. In Fig.5, there are 9 nodes with available distance values. To calculate the shortest distance, the prototype is used to drive through the path that must be taken from the green point (Start) to the yellow point (End).

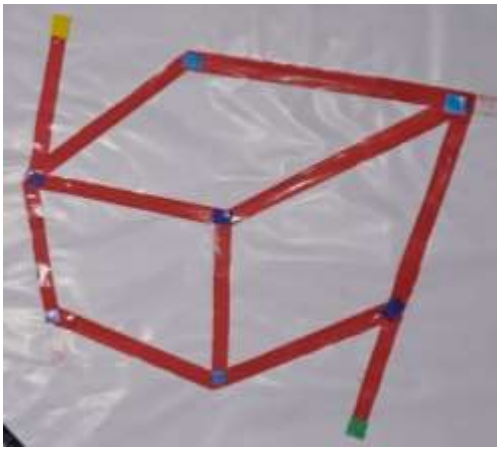


Fig. 5. Pathway Design

1) Axis: In mathematics, physics and engineering, an axis is a reference line used to determine the position of a point or object in a coordinate system. Axes are also often used to describe the relationship between various points in space. Here is an example of a path that has had axes added. The axis Fig.6 test is very useful for determining the direction, position, and destination along the path, which helps in the navigation and trajectory planning process.

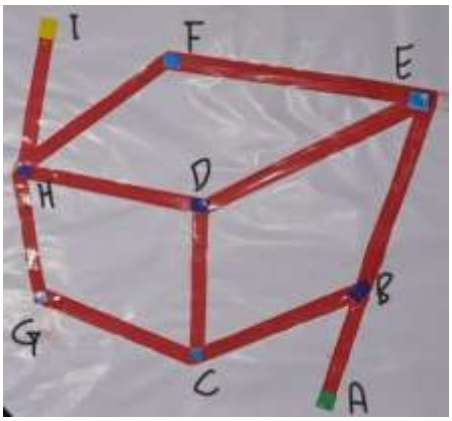


Fig. 6. Axis

2) Cost: The cost here represents the distance travelled from one axis to another, as shown in Table I. The cost on the path has been measured using a measuring device to produce an accurate cost from axis to axis. In this test, the cost is used to calculate the total distance travelled from axis A to axis I. Then, each total distance in Table I is summed up again to obtain the heuristic value.

TABLE I
COST BETWEEN AXES ONE TO THE OTHER

Path	Cost
A-B	21cm
B-C	30cm
B-D	30cm
B-E	20cm
C-G	41cm
D-G	38cm
D-F	25cm
E-F	27cm
G-H	31cm
F-H	26cm
H-I	16cm

B. Design and Implementation

At this stage, the design will be discussed and analyzed in the form of a prototype. The prototype is designed as an object to test the effectiveness of the A* algorithm.

1) Prototype design: The prototype of the line follower robot is designed to look like a drone in the rainbow six siege game, where the robot is shaped like a drone that has 2 tires and is not too big, as shown in Fig.7. The reason why the drone shape is taken is because the webcam does not capture a very wide real time video, so if the map is not too large then the robot should not be large because it can interfere with the robot in moving along the path. This prototype has a width and length of 12cm x 13cm and a height of 9cm, equipped with two DC motors on both sides. Below is a picture of the prototype. There are three different colours on the front, where each colour represents a part of the robot and helps it determine the turn while following the path. The ESP32 is positioned at the top of the robot, and the motor driver is located within the robot to simplify its design.

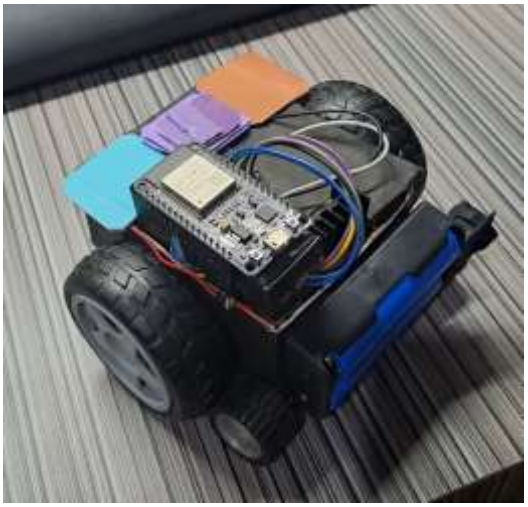


Fig. 7. Prototype Shape

DOI : https://doi.org/10.25139/inform.v10i2.10508

2) Motor Driver Design: This motor driver design is conducted to assess the suitability of the prototype's rotation direction control, which is controlled by the motor driver. The prototype speed setting is set on the usual motor driver in ENA and ENB. Table II is the data taken from the right and left DC motors. Table III presents the performance data of the left-side DC motor. Table IV presents a comparative analysis of the performance of the right and left DC motors when operated simultaneously under identical PWM settings.

TABLE II
RIGHT-HAND DC MOTOR DATA

Right Motor		
out1	**out2**	**Description**
1	0	CW
0	1	CCW

TABLE III
LEFT SIDE DC MOTOR DATA

Left Motor		
out3	**out4**	**Description**
1	0	CW
0	1	CCW

TABLE IV
DATA OF BOTH SIDES OF THE DC MOTOR

Right Motor		Left Motor		Description
out1	**out2**	**out3**	**out4**	
1	0	1	0	Forward
0	1	0	1	None
1	0	0	1	Left Turn
0	1	1	0	Turn right

3) A algorithm design:* The initial step in designing the line follower system begins with creating a track using red duct tape as the path. The path serves as a guide for the prototype to follow the route. The path is designed to form a dynamic line. The purpose of making this path is to facilitate route planning, distance calculation and testing the prototype's ability to determine routes from various turns. The path helps in designing navigation routes, identifying intersection points, and determining the optimal path to the destination using the A* algorithm. The flowchart in Fig.8 displays how the A* algorithm works on the prototype. The process begins with OpenCV reading the entire map, followed by the calculation of each function. If a path has the smallest heuristic value, it will be included in the closed list. After reaching the destination point, OpenCV will display the A* route and the black dot that moves to the destination point to navigate the prototype. After the prototype reaches its destination, it will stop.

4) Finding the heuristic value: Before finding the A* path, we must first find the heuristic value. The heuristic value is the estimated cost distance from a current point to the destination point. This value is not real but is estimated to help the algorithm make the most efficient decision in finding the best route. One of the most common methods to calculate the heuristic value is by using the Euclidean distance, which is derived from the Pythagorean theorem. The Pythagorean theorem is used to calculate the distance between two points on a flat plane (2 dimensions). The results of calculating the heuristic value using the Pythagorean theorem on the OpenCV visual in Fig.9.

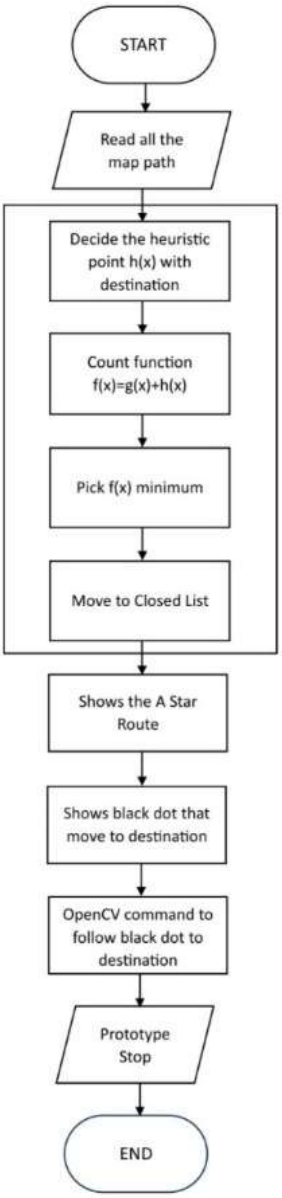
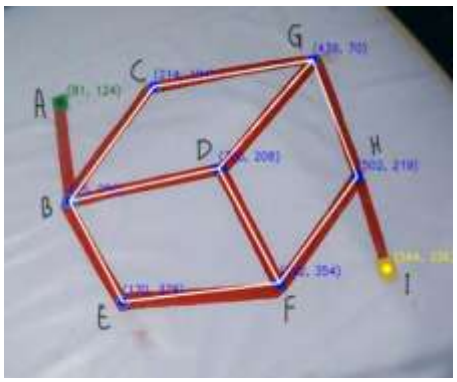


Fig.8. Flowchart of Algorithm A* on the Prototype



Description:
- is objective
- is the starting point
- is Corner Point

Fig.9. Pathway Design

DOI : https://doi.org/10.25139/inform.v10i2.10508

Before entering the summation, we must determine (*x,y*) at each point from Fig.9. These coordinates are listed in Table V.

TABLE V
COORDINATES EACH POINT

Point	Coordinates	
	X	Y
A	81	124
B	95	250
C	213	104
D	306	208
E	170	378
F	392	354
G	439	70
H	502	219
I	544	336

Once the coordinates of each point were obtained, the heuristic values were calculated using the Euclidean distance formula, as shown in Equation (2), which is derived from the Pythagorean theorem. The results of these calculations are summarized in Table VI.

TABLE VI
HEURISTIC RESULT OF EACH POINT

Point	h(n)
AI	509.2278468
BI	457.1618969
CI	404.2091043
DI	270.2369331
EI	376.3509001
FI	153.0620789
GI	285.973775
HI	124.3100961

5) Implementation of A Algorithm:* Fig.10. presents the visualized route, along with the heuristic values calculated for each point. As shown in Table VII, the total cost for each route segment is calculated using Equation (1), which is presented to illustrate the results of the A* Algorithm during the route computation process.

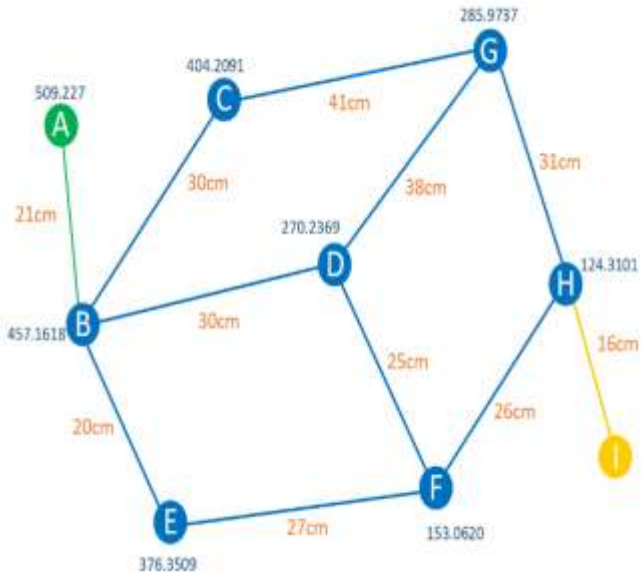


Fig. 10. Route in Visualization

TABLE VII
TOTAL COST EACH ROUTE

Route	COST
AB	478.1619
BC	434.2091043
BD	300.2369331
BE	396.3509001
CG	326. 973775
DG	320.973775
DF	178.0620789
EF	180. 0620789
GH	155. 3100961
FH	150.3100961
FE	402.3509001
HG	301.973775
HI	16

After each route was calculated, the following section explains the implementation of the A* Algorithm in finding the shortest path. The prototype starts from point A and has only one neighbour, point B. Therefore, point B will be added to the open list. Point B had the lowest cost at 478.1619 and will be included in the closed list Table. Point A and its neighbour B are also added to a rooted tree to track the path. Since the result has already been obtained, the Equation will not be reapplied in the following steps to avoid redundancy and streamline the explanation in Fig.11.

Fig. 11. Open List from Node A

After being at point B, point B has neighbours C, D, and E. Point D has the lowest cost, at 300.2369331. while other points have a cost that is much more expensive than point D, then point D will be included in the closed list, as shown in Fig.12

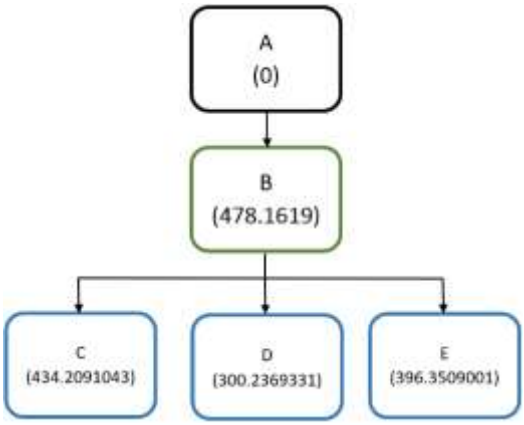


Fig.12. Open List from Node B

After reaching point D, it has neighbours G and F. Point F has the lowest cost, at 178.0620789. While point G has a cost

DOI : https://doi.org/10.25139/inform.v10i2.10508

that is more expensive than point G, it will be included in the closed list, as shown in Fig. 13.

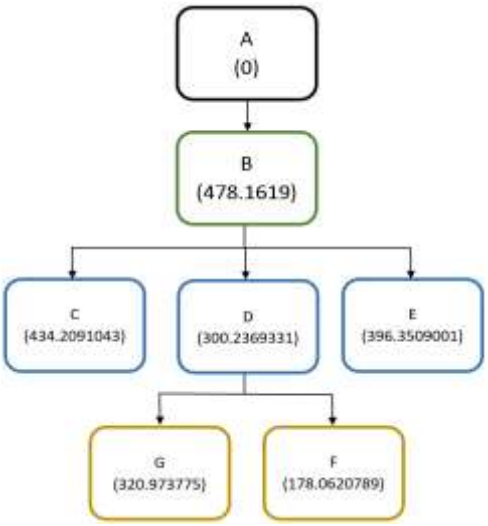


Fig.13. Open List From Node D

After being at point F, point F has neighbours H and E. However, point H has the lowest cost, at 150.3100961. Point F also has neighbour D, but since it has already been visited, it will not be reconsidered as a point of interest. Then, point H will be included in the closed list, as shown in Fig.14.

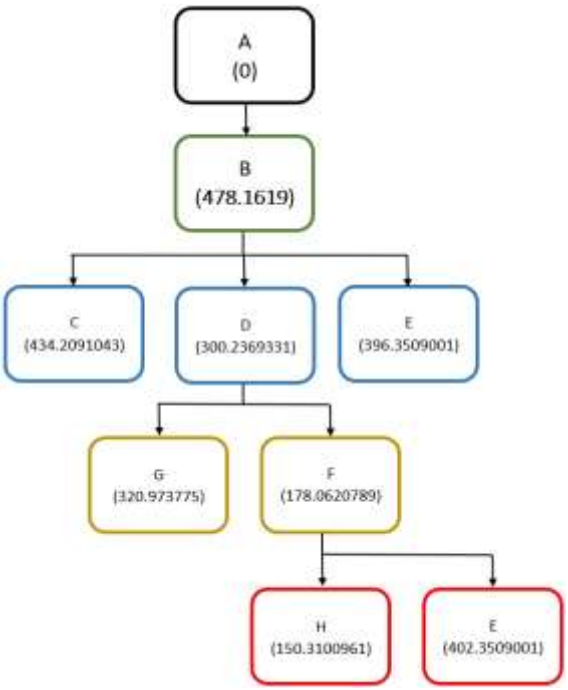


Fig. 14. Open List From Node F

After reaching point H, it has neighbours I and G. Point I has a cheaper cost of 16, so we have arrived at our destination. Also, point I can be included in the closed list as shown in Fig.15. Table VIII summarizes the nodes that have been explored and added to the CLOSED LIST during each iteration of the A* algorithm.

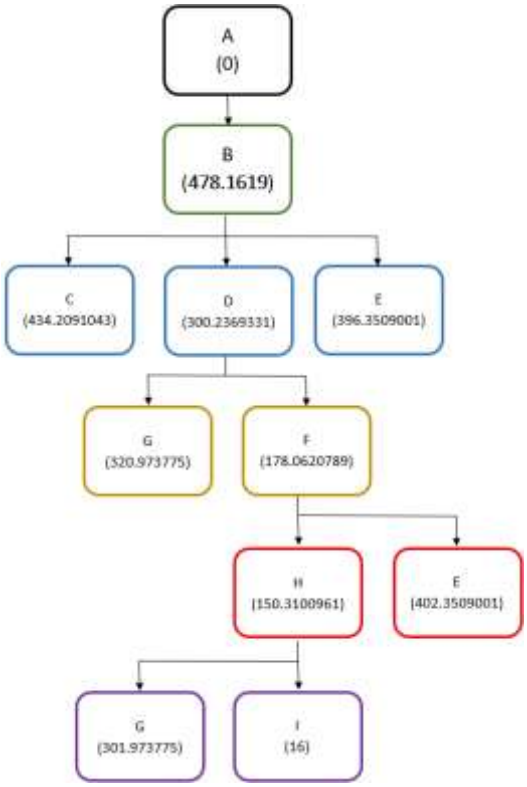


Fig. 15. Open List From Node H

TABLE VIII
COST OF EVERY CLOSED LIST

DESTINATION	COST	VIA
B	478.1619	A
D	300.2369331	B
F	178.0620789	D
H	150.3100961	F
I	16	H

After conducting the exploration process using open lists and closed lists, the results of the A* algorithm process are visualized in the final display using OpenCV, where the optimal path is depicted to show the best route, as shown in Fig. 16.

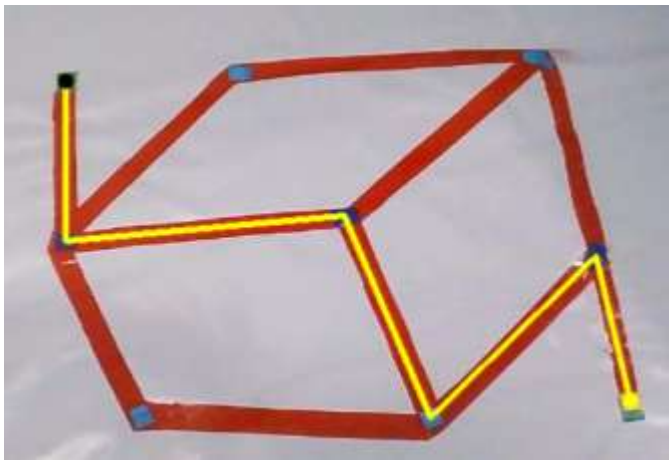


Fig. 16. A* Route View in OpenCV

DOI : https://doi.org/10.25139/inform.v10i2.10508

The test results show that OpenCV successfully found the fastest route through ABDFHI. This demonstrates that the A* algorithm effectively finds the fastest path through the specified map.

IV. CONCLUSION

The smart car prototype developed in this research is an early model of a smart vehicle designed to mimic the basic functions of a real car. Although on a miniature scale, the prototype has demonstrated the basic capabilities of an automated vehicle, such as navigation and decision-making, using OpenCV, which is displayed through a camera. In its implementation, the prototype utilizes the A* algorithm to determine the fastest route to the destination point. This algorithm enables the system to calculate the distance and select the most efficient route, considering intersections and alternative routes.

Several suggestions can be submitted for further development of the prototype. First, it is expected that OpenCV detects paths more effectively, thereby increasing the system's ability to recognize paths with greater accuracy. Second, it is expected that the robot's movement will be more optimal, thereby improving the system's ability to follow the path. With the implementation of these suggestions, it is expected that the smart car prototype can function more optimally and efficiently.

REFERENCES

[1] R. Purbasari, N. N. A. Jamil, and N. Kostini, "Digitalisasi Logistik Dalam Mendukung Kinerja E-Logistic Di Era Digital: A Literature Review," *Management, Business and Logistics (JOMBLO)*, vol. 01, no. 02, pp. 177–196, 2023.

[2] F. Agil, B. Putra, M. Arif, and R. Hakim, "Design a Smart Charity Box Based on Line Followers," *Indonesian Journal of Electrical Engineering and Electronics (IJEEE)*, vol. 1, no. 1, Dec. 2023.

[3] A. S. Priambodo, A. Nasuha, and O. A. Dhewa, "Integrated Implementation of Computer vision and PID Control for Dynamic Speed Control of Line follower Robot," *Telekontran : Jurnal Ilmiah Telekomunikasi, Kendali dan Elektronika Terapan*, vol. 12, no. 1, pp. 81–93, Jul. 2024, doi: 10.34010/telekontran.v12i1.13323.

[4] R. Y. Aritonang, Y. E. Widayanti, and R. Ganindha, "Urgensi pengaturan artificial intelligence kendaraan semi otonom dan otonom dalam aspek keselamatan dan keamanan berkendara berbasis kecerdasan buatan," *Jurnal Hukum Universitas Brawijaya*, vol. 1, no. 1, 2023, Accessed: Jun. 09, 2025.

[5] H. Liu, J. Cao, and Z. Wang, "Research on path planning of mobile robot in complex environment," *Discover Applied Sciences*, vol. 7, no. 4, Apr. 2025, doi: 10.1007/s42452-025-06713-y.

[6] B. H. Pratama, "Implementasi Line Tracking pada OpenCV Menggunakan Pyhton di Robot untuk Edukasi," vol. 11, no. 5, p. 5587, 2024.

[7] Y. M. R. da Silva *et al.*, "Computer Vision Based Path Following for Autonomous Unmanned Aerial Systems in Unburied Pipeline Onshore Inspection," *Drones*, vol. 6, no. 12, Dec. 2022, doi: 10.3390/drones6120410.

[8] K. Sugiarto, A. Giyantara, and R. D. Yulisya, "Object Tracking Dengan Menggunakan Color Filtering HSV Pada Robot World Cup," *Jurnal Bumigora Information Technology (BITe)*, vol. 4, no. 2, pp. 143–152, Dec. 2022, doi: 10.30812/bite.v4i2.2156.

[9] E. Sumantri and S. Hidayattullah, "Penerapan Algoritma A*Star Untuk Mencari Rute Terpendek Dari Kemayoran Ke Destinasi Monumen Nasional (MONAS)," *Jurnal Sains dan Teknologi*, vol. 5, no. 2, pp. 673–680, 2023, doi: 10.55338/saintek.v5i1.1432.

[10] B. Riki and P. N. Hardi, "Rancang Bangun Robot Pengikut Garis Berbasis Arduino Dan Menggunakan Sensor Inframerah," *Jurnal Teknologi Elektro Universitas Mercu Buana*, vol. 1, pp. 216–234, 2019.

[11] L. A. Siregar, Y. Ananda, A. M. Adha, and M. Iqbal, "Rancang Bangun Lengan Robot Berbasis Arduino Menggunakan Sistem Kontrol Sensor Giroskop," *Journal of Electrical and System Control Engineering*, vol. 8, pp. 201–206, Feb. 2025, doi: 10.31289/jesce.v6i2.12986.

[12] M. Ardiansyah Muktadir Gasba, L. Harlinda, and Irawati, "Implementasi Algoritma A* (A-Star) dalam Menentukan Jarak Terpendek Menuju Rumah Sakit Rujukan Covid-19," *Buletin Sistem Informasi dan Teknologi Islam (BUSITI)*, vol. 3, no. 3, pp. 203–212, 2022.

[13] Sudimanto and Kevin, "Perancangan Robot Pemindah Barang Berbasis Line Follower," 2020.

[14] B. T. D. Irianto, S. Andryana, and A. Gunaryati, "Penerapan Algoritma A-Star Dalam Mencari Jalur Tercepat dan Pergerakan NonPlayer Character Pada Game Petualangan Labirin Tech-Edu," *Jurnal Media Informatika Budidarma*, vol. 5, no. 3, Jul. 2021.

[15] Ridarmin, Fauzansyah, Elisawati, and P. Eko, "Prototype Robot Line Follower Arduino Uno Menggunakan 4 Sensor Tcrt5000," *Jurnal Informatika, Manajemen dan Komputer*, vol. 11, no. 2, 2019.

[16] L. D. S. Joseph, I. P. Saputro, and A. M. Adrian, "Game 'Finding Easter Eggs' Berbasis Augmented Reality Menggunakan Algoritma A-Star," *Cogito Smart Journal* |, vol. 7, no. 1, 2021.

[17] A. Bramato Wicaksono Putra, A. Aulia Rachman, A. Santoso, and Mulyanto, "Perbandingan Hasil Rute Terdekat Antar Rumah Sakit di Samarinda Menggunakan Algoritma A*(star) dan Floyd-Warshall," *Sistem Informasi dan Komputer*, vol. 09, pp. 59–68, 2019, doi: 10.32736/sisfokom.v9.i1.685.

[18] R. Kurniawan, M. Idris, and Armansyah, "Penerapan Algoritma A-Star Terhadap Enemy Pada Game Escape From Pirates Berbasis 2D Dengan Menggunakan Unity 3D," *Journal of Dinda Data Science, Information Technology, and Data Analytics*, vol. 4, no. 2, pp. 75–81, 2024.

[19] G. Yulianti, Benardi, N. Permana, and F. A. K. Wijayanti, "Transformasi Pendidikan Indonesia: Menerapkan Potensi Kecerdasan Buatan(AI)," *Journal Of Information Systems And Management (JISMA)*, vol. 2, no. 6, Dec. 2023, doi: https://doi.org/10.4444/jisma.v2i6.1076.

[20] P. A. Eka, A. A. Nur, and M. P. Ayunda, "Artificial Intelligence: Dampak Pergeseran Pemanfaatan Kecerdasan Manusia Dengan Kecerdasan Buatan Bagi Dunia," *Sindoro: Cendikia Pendidikan*, vol. 1, no. 10, pp. 31–40, 2023.

[21] S. P. Wayan, Sukarmi, and F. Sulistio, "Arrangements of Artificial Intelligence in Criminal Law in Indonesia," *International Journal of Multicultural and Multireligious Understanding (IJMMU)*, no. 1, pp. 296–303, Jan. 2022, doi: 10.18415/ijmmu.v9i1.3389.

[22] K. P. Sari, A. Masruri, and D. R. Rosalia, "Optimalisasi Temu Kembali Informasi Dengan Teknologi Kecerdasan Buatan di Perpustakaan," *JIPI (Jurnal Ilmu Perpustakaan dan Informasi)*, vol. 8, no. 2, p. 349, Nov. 2023, doi: 10.30829/jipi.v8i2.17775.

[23] L. Liu, B. Wang, and H. Xu, "Research on Path-Planning Algorithm Integrating Optimization A-Star Algorithm and Artificial Potential Field Method," *Electronics (Switzerland)*, vol. 11, no. 22, Nov. 2022, doi: 10.3390/electronics11223660.

[24] W. J. H. Brigido and J. M. Parente de Oliveira, "The line follower robot: a meta-analytic approach," *PeerJ Comput Sci*, vol. 11, 2025, doi: 10.7717/PEERJ-CS.2744.

[25] H. Wang, G. Li, J. Hou, L. Chen, and N. Hu, "A Path Planning Method for Underground Intelligent Vehicles Based on an Improved RRT* Algorithm," *Electronics (Switzerland)*, vol. 11, no. 3, Feb. 2022, doi: 10.3390/electronics11030294.

DOI : https://doi.org/10.25139/inform.v10i2.10508