Application of Odometry and Dijkstra Algorithm as Navigation and Shortest Path Determination System of Warehouse Mobile Robot

Achmad Ubaidillah [1*], Hanifudin Sukri [2]
[1, 2] Department of Electrical Engineering, Universitas Trunojoyo Madura, Jawa Timur, Indonesia
Email: [1] ubaidillah.ms@trunojoyo.ac.id, [2] hanifudinsukri@trunojoyo.ac.id
*Corresponding Author

Abstract—**One of the technologies in the industrial world that utilizes robots is the delivery of goods in warehouses, especially in the goods distribution process. This is very useful, especially in terms of resource efficiency and reducing human error. The existing system in this process usually uses the line follower concept on the robot's path with a camera sensor to determine the destination location. If the line and destination are not detected by the sensor or camera, the robot's navigation system will experience an error. it can happen if the sensor is dirty or the track is faded. The aim of this research is to develop a robot navigation system for efficient goods delivery in warehouses by integrating odometry and Dijkstra's algorithm for path planning. Holonomic robot is a robot that moves freely without changing direction to produce motion with high mobility. Dijkstra's algorithm is added to the holonomic robot to obtain the fastest trajectory. by calculating the distance of the node that has not been passed from the initial position, if in the calculation the algorithm finds a shorter distance it will be stored as a new route replacing the previously recorded route. the distance traversed by the djikstra algorithm is 780 mm while a distance of 1100 mm obtains the other routes. The time for using the Djikstra method is proven to be 5.3 seconds faster than the track without the Djikstra method with the same speed. Uneven track terrain can result in a shift in the robot's position so that it can affect the travel data. The conclusion is that odometry and Dijkstra's algorithm as a planning system and finding the shortest path are very efficient for warehouse robots to deliver goods than ordinary line followers without Dijkstra, both in terms of distance and travel time.**

Keywords—Warehouse Robot; Navigation; Shortest Path; Dijkstra; Odometry.

I. INTRODUCTION

The development of robotics provides conveniences for humans [1][2][3][4]. Robot is a technological product that combines hardware and software with a propulsion program used in a particular job [5][6][7][8][9]. Many activities that are too heavy for humans can be done easily with the help of robots, such as activities that require large amounts of energy, high costs, fast time and detailed accuracy [10][11]. Warehouse robots for delivery of goods have become an important part of the logistics and delivery of goods in the industry. This system improves operating efficiency and reduces cycle times. To maximize the benefits of this system, it is necessary to determine the fastest path for the robot to deliver goods.

The benefits of robots in industrial activities have been widely used such as delivery of goods, production, packaging, picking, dropping, inventory, unloading, manufacturing processes, controllers and others in accelerating and increasing the quality and quantity of production, like in [12][13][14] that develop in industrial robot especially in manufacturing. The application of delivery robots has existed in several developed countries [15][16]. The navigation system usually uses a line follower [17]. However, the system is highly dependent on the sensitivity to recognition of the color of the line traversed. If there is a problem in the recognition system, the robot's navigation system will also have a problem [18][19]. This must be overcome by the system within the robot itself [20][21]. So the application of odometry is needed as an alternative method that can answer some of the limitations of line follower system [22][23][24].

The odometry system was chosen in this study because several studies have proven that odometry has high accuracy in measuring the distance, direction and position of the robot without dependence on light and line search [25]. Odometry is used to estimate position coordinates relative to the initial position [26][27]. The odometry system requires a rotary encoder sensor to detect the number of wheel rotations [28][29]. Area mapping using the odometry method is to estimate changes in the robot's position over time in a Cartesian diagram [30][31]. The result is data on the coordinates and direction of the robot [32][33][34]. Odometry uses actuator movement data to estimate coordinate changes [35][36]. The robot position coordinates include three parameters, namely the diameter of the free wheel, the number of encoder resolutions and the number of rotary encoder pulses [37][38][39]. This data is used as a parameter for robot navigation combined with a holonomic system [40][41]. Holonomic is a type of mobile omnidirectional wheel [42][43][44]. It can move without changing direction due to omni wheel kinematics [45][46]. It can move forward, backward, slide sideways and rotate in a fixed position so that it is more effective in maneuvering [47][48]. Therefore, this study hypothesizes that holonomic can simplify and accelerate mobile robots to maneuver in all directions and follow the application of Dijkstra's algorithm for the effectiveness and speed of distributing goods in warehouses [49][50].

After the path has been mapped, the next step is finding the shortest path in each route on the path map [51]. Searching for the shortest route becomes very important in path planning because it is related to delay [52]. The smaller the delay, the more reliable a system [53]. One of the methods used for this is Dijkstra's algorithm [54]. Dijkstra's algorithm was chosen in this study because it is simple to implement and sufficiently detailed considering the load of each path [55]. Dijkstra is a shortest route search method that is used in various fields [56], such as in telecommunications [57], regional maps [58], transportation [59], energy [60] and others.

Some previous research on robots is [61] which developed a mobile robot navigation system using visual odometry based on ceiling vision. Research [62] developed the design of an omnidirectional wheeled mobile robot. At the same time, research [63] developed a control system for trajectory tracking on an omnidirectional wheeled mobile robot. But they don't consider the shortest path problem. Another research is [64] which proposes the application of Dijkstra's Algorithm in determining the fastest trajectory of a wheeled soccer robot. But it doesn’t discuss about path planning method. This research develops the application of Dijkstra's Algorithm and odometry as a navigation system for the fastest path in the warehouse robot delivering goods. This study applies the movement of the holonomic drive system and uses omnidirectional wheels [65].

This research has an essential objective in combining the odometry system method and Dijkstra's algorithm to improve the efficiency of warehouse robot operations. The odometry system is used as a path planning method, where encoder or sensor-based odometry allows the robot to determine its position and orientation in a warehouse environment. With this knowledge, the robot can plan an efficient path and avoid obstacles.

In addition, this study also utilizes Dijkstra's algorithm to find the shortest path to maximize the efficiency of robot movement. In this context, Dijkstra's algorithm is used to speed up the process of finding the shortest path from the point of origin to the point of destination in a warehouse whose structure and condition can change.

Therefore, this research contributes to the development of warehouse robot technology by combining the odometry approach and Dijkstra's algorithm. Combining these two methods can increase the robot's operational efficiency, speed up the delivery time of goods, and reduce the potential for damage or errors that may be caused by ineffective navigation. This research also demonstrates the potential use of this technology in the warehouse and other logistics contexts, where efficient and accurate path planning is essential.

II. Theory

A. Omnidirectional Wheel and Odometry

Omni wheels are a special wheel design that has many wheels on the main wheel [66][67]. There are two types of wheels, namely large core wheels and small wheels on a large core wheel that is perpendicular to the core wheel axis [68][69]. In contrast to normal non-holonomic robots, holonomic omni-directional robots can move in all directions without changing the direction of the wheels [70][71]. The omni robot can move forward, backward, slide and rotate in a fixed position, so that the robot is able to maneuver more agile in tight corners [72][73][74]. Omni directional wheels are a type of holonomic mobile robot [75][76][77]. The kinematics of an omni wheel allows position changes between global coordinates and internal configurations [78][79][80].

Odometry presents changes in data over time [81][82]. Odometry estimates position coordinates relative to the position of the actuator movement data to estimate changes in position coordinates to initial conditions [83][84]. In the wheeled robot odometry system, the sensor used is a rotary encoder to detect the number of wheel rotations [85][86]. Identification of errors is determined from the integration of velocity measurements against time in the position estimation process [87][88].

The robot's relative position can be estimated using the calculation of the number of pulses generated by the rotary encoder sensor for each unit of measure which is then converted into millimeters. The formula for the number of pulses for each freewheel movement is equations (1) and equation (2). The differential movement system is carried out by two wheels, namely the right and left wheels. The number of $pulses_{per\_mm}$ for the right wheel is the $right\ enc$ and the left wheel is the $left\ enc$. The distance between the two wheels is $wheelbase$ with distance traveled ($Distance$) and orientation angle (θ), as shown by equations (3), (4), and (5).

$$K_{wheel} = 2\pi r \tag{1}$$

$$Pulse_{per\_mm} = \frac{resolution\_enc}{K_{wheel}} \tag{2}$$

$$Distance = \frac{(left\ enc - right\ enc)}{2} \tag{3}$$

$$\theta = \frac{(left\ enc - right\ enc)}{wheel\ base} \tag{4}$$

$$Heading = \theta \frac{180}{\pi} \tag{5}$$

Heading θ is the angle in radians. Equation (5) shows that the heading value will be negative if the robot rotates counterclockwise and is positive if the robot rotates clockwise. If the distance and θ are known, then the X and Y coordinates are obtained using the trigonometry equation. Look at Fig. 1.

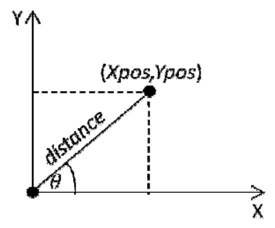


Fig. 1. Calculation of robot distance on odometry system

From the illustration of Fig. 1, the robot coordinates are obtained with equation (6) and (7). After that, equations (8), (9), (10), and (11) can be obtained.

$$X_{pos} = distance\ x\ \sin\theta \tag{6}$$

$$Y_{pos} = distance\ x\ \cos\theta \tag{7}$$

$$x = X_{target} - X_{pos} \tag{8}$$

$$y = Y_{target} - Y_{pos} \tag{9}$$

$$TargetDistance = \sqrt{(x^2 + y^2)} \tag{10}$$

The direction error value of the robot towards the destination point is calculated by the Pythagorean theorem. The current position and distance to the destination point are calculated using equation (6) dan (7). Heading error can be calculated based on the heading of the robot. β is the target bearing, namely the angle between the robot's current position and the destination points. Meanwhile, parallel lines are auxiliary lines that are parallel to the X and $Y\ axes$. The value of β is obtained by the equation (11).

$$\beta = \tan^{-1}\frac{(Y_{target} - Y_{pos})}{(X_{target} - X_{pos})} \tag{11}$$

B. *Dijkstra's Algorithm*

Dijkstra's algorithm is applied to find the shortest path on a directed graph [89][90]. This algorithm can also be used for undirected graphs [91][92]. It looks for the shortest path in a number of steps [93][94]. It uses the greedy principle [95][96]. The greedy principle in Dijkstra's algorithm always chooses the position with the smallest weight and includes it in the solution set [97][98]. It performs calculations against all possibilities to find the smallest weight from each node to node [99][100]. The mechanism of Dijkstra's algorithm can be explained in the following steps:

- Determine the starting point, then weight the distance from the first node to the nearest node one by one. Dijkstra's algorithm will develop the search for the smallest value step by step.
- Give a distance weight for each point, then set a value of 0 at the initial node and an infinite value for other nodes.
- Consider the untraversed neighboring nodes and calculate their distance from the departure point. If this distance is smaller than the previous distance, delete the old data, save and recalculate the distance data with the new distance.
- Mark the node that has been passed as a “passed node”. Passed nodes will never be checked again. The distance that is stored is the last distance and the most minimal weight.
- Set “Node not passed” with the smallest distance from the departure node as the next “Departure Node” and repeat the steps

III. METHODOLOGY

The following Fig. 2 is a block diagram of the warehouse delivery robot system developed in this study.

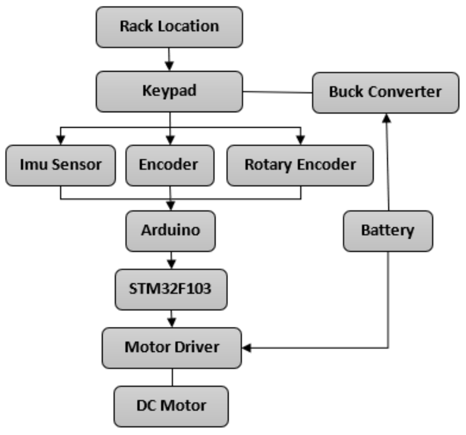


Fig. 2. Block diagram of the robotic system

The Arduino microcontroller voltage source comes from a 12V battery which after the buck converter the voltage drops to 5V which is then supplied to the Arduino microcontroller and STM32f103 [101]. The motor driver voltage source comes from a 12 V battery to obtain maximum motor rotation speed without going through a buck converter. MPU6050 sensor and DC motor internal encoder (RPM) require 5V voltage. Fig. 3 is the flowchart of the system.

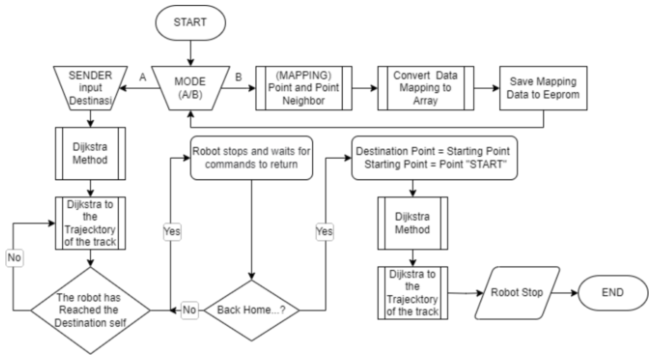

START
SENDER input Destinasi
A
MODE (A/B)
B
(MAPPING) Point and Point Neighbor
Convert Data Mapping to Array
Save Mapping Data to Eeprom
Dijkstra Method
Robot stops and waits for commands to return
Destination Point = Starting Point Starting Point = Point "START"
Dijkstra to the Trajecktory of the track
Yes
Yes
No
Dijkstra Method
The robot has Reached the Destination self
No
Back Home...?
Dijkstra to the Trajecktory of the track
Robot Stop
END

Fig. 3. Flowchart of the system

System stages include robot operation, trajectory mapping, data processing and movement or action of the robot [102]. The first operation on the robot is carried out by the user, namely turning on the power button. Then the user chooses robot mode for mapping or mode for delivery (sender). The mapping area must be formed first so that it can then be traversed [103][104].

Mapping is done by running the robot manually on its trajectory until it returns to its starting point. With a rotary encoder sensor that is processed using the odometry method, the position of the robot's movement can be recorded [105][106]. If the Robot Movement finds an intersection, the user must press the node button. The robot path is in the form of a graph consisting of “vertex” and “node”.

Mapping result data is stored in the microcontroller's internal EEPROM [107]. EEPROM was chosen as a storage medium because it has many benefits, including reliable design and performance and easy erasing and programming without disturbing the board. Then the user can carry out the sending process by pressing the keypad button according to the destination node. The robot will send to the destination node according to the previously mapped location. If it reaches the destination rack, the robot will stop. After placing the goods on the appropriate rack, the user presses a button on the keypad to order the robot to return to the starting point.

Fig. 4 is the flowchart of robot speed control and shows the flowchart of 3-wheel speed regulation. The system begins by determining the kp, ki, kd values on the PID then the encoder sensor reads the motor speed. The result is an RPM value obtained through the calculation of treeomniwheel kinematics. This value is then processed by the PID and used as a PWM value to move the robot according to its destination. If the robot has not reached its destination, the process will continuously loop back to the initial process until the robot reaches its destination. If it reaches the destination, then the process is complete.

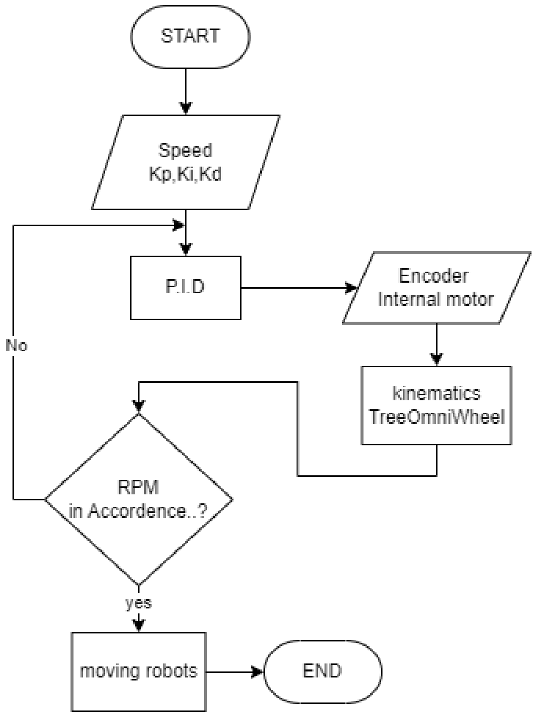


Fig. 4. Flowchart of robot speed control

Fig. 5 is the flowchart of node mapping system. Retrieval of node data is carried out by the user by moving the robot manually. The first step determines the starting position or the position when the robot is in standby mode, and the motor is off. The robot must be reset first by pressing the reset button provided. Then the user must determine all the node points on the track, including the stops on each rack. The second step is to map the nodes by delivering the robot to each node by pressing the '#' key. The '#' key is pressed to save the coordinate value made at each robot stop. The last step is to press the '*' button. This is done if all nodes have been mapped, and all node data will be stored in EEPROM to proceed to the following process. In the process of using Dijkstra's algorithm, neighbors must first be identified at each node. The identification of neighboring nodes is made by entering each weight via the keypad.

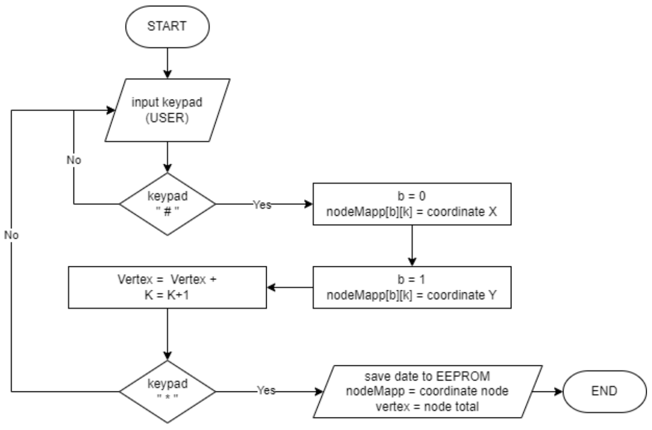


Fig. 5. Flowchart of node mapping

Fig. 6 shows the process of weighting neighboring nodes. The process of assigning weights to neighboring nodes must be carried out according to the path that has been planned and converted to a data graph stored in a matrix variable, with as many dimensions as the nodes that have been stored. Initially, all element values will be assigned a temporary value, namely ∞. In the conversion process, the node elements that have neighbors are updated with the value of the elements while the other nodes still have a value of ∞. The updated value is the value of the distance between the nodes and their neighboring nodes.

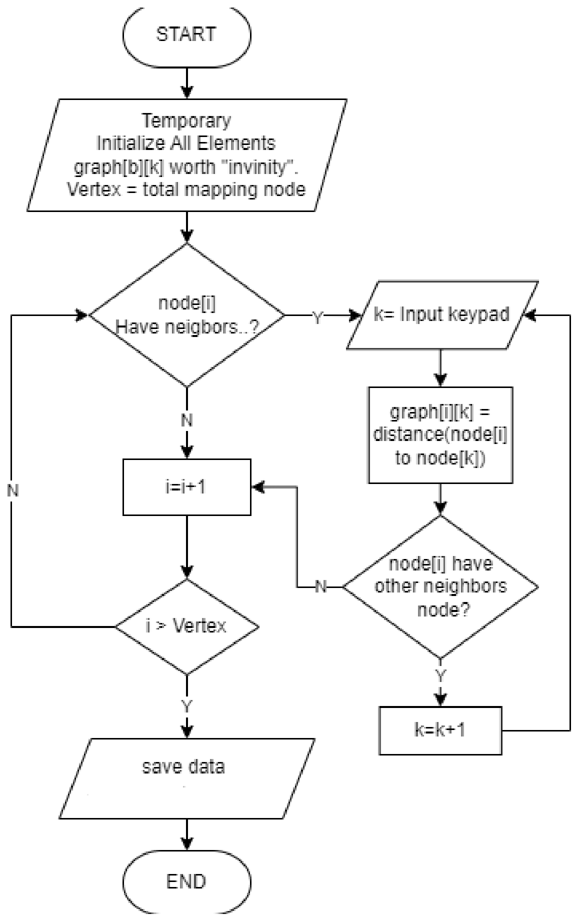


Fig. 6. Flowchart of weighting at neighbor node

Fig. 7 is the flowchart of Dijkstra Application. Fig. 7 is a flowchart of applying Dijkstra's algorithm to find the shortest route from several paths. The Dijkstra process compares all the path weights to be selected by:

- Give the value of the weight of the distance from one node to another then determine the departure node with a value of 0 and ∞ at the other nodes.
- If node A to B has a distance value of 5 and from node B to point C has a value of 2. So, the distance to node C through node B is 5 + 2 = 7. If this distance is smaller than the previously recorded distance, delete the old data and resave the new distance value.
- Mark the nodes that have been skipped as 'skipped nodes'. Missed nodes will never be checked again and the distance stored is the last distance from the departure node and the minimum distance value.
- Determine the node that has not been passed with the smallest distance value from the departure node as the next departure node and continue by repeating step 3.

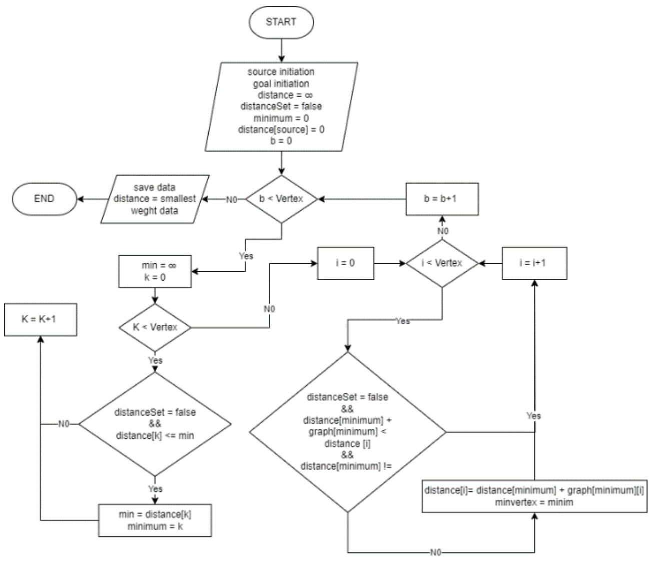


Fig. 7. Flowchart of Dijkstra Application

Fig. 8 is the flowchart of trajectory. Fig. 8 is the process for obtaining trajectory data from Dijkstra's algorithm. The main parameter used is the value of the “target” variable which stores the value of the destination node. The weight of the path is then searched for and stored in the “Trajectory” variable until the “destination” variable value is the same as the initial or source node value. Trajectory data is taken from Trajectory data $[b]$ or lines in the graph.

The process begins with the initialization of the source, destination and variable i which is then in the decision stage if $i > vertex$ then the process is complete. If $i < \; ertex$ then the trajectory $[b]$ is the same as the data trajectory $[i - x]$. Then check again if the destination is not the same as the source, then $i = i + 1$ and the process returns to the beginning. If the destination is the same as the source, the process will continue to determine whether $x < i$. If yes, it will be counted until the last stage, then the process is complete and the data is saved.

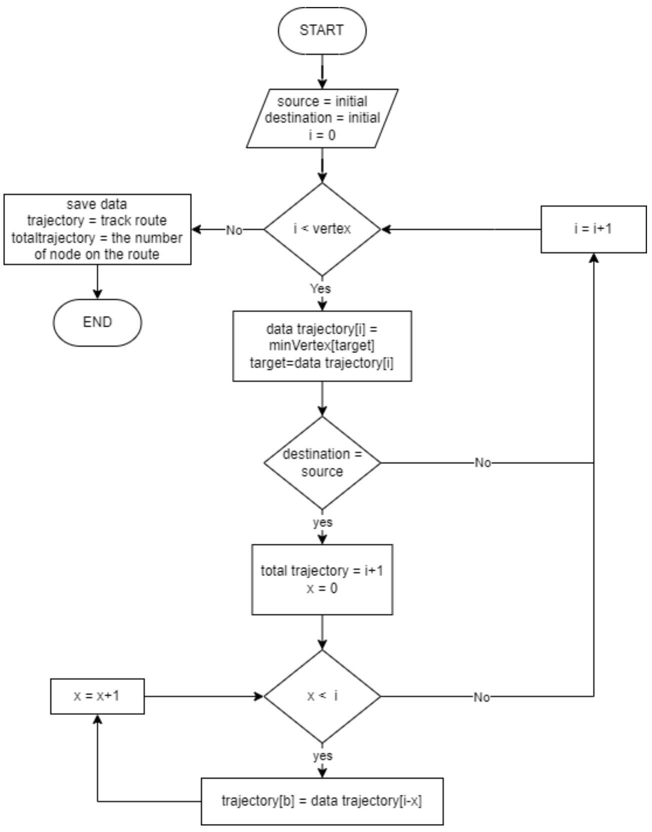


Fig. 8. Flowchart of Trajectory

The hardware requirements for realizing the wheeled robot planning in this study are:

- STM32F103C8T6
- Motor DC
- Battery managemen System 12 V
- Driver L298N
- 3 Omni Wheel
- Motor dc JGA 25 100 RPM
- Buck Converter
- Acrylic 3mm
- Sensor of Rotary Encoder
- Sensor of IMU
- Sensor of encoder RPM internal motor

IV. Result and Discussion

As in research [64], which applied Dijkstra's algorithm to a wheeled mobile robot in a soccer robot game, this research applies and combines the Odometry method as a path mapper with Dijkstra's algorithm as a finder of the shortest path applied to warehouse robots. This research examines the movement of warehouse robots using the Djikstra algorithm and odometry as a mapping navigation system. The robot has dimensions of 30 cm in diameter and 14 cm in height. It uses a keypad as the tool's operating buttons and a 20x4 LCD as an interface. Fig. 9, Fig. 10, Fig. 11 and Fig. 12 are the view of robot.

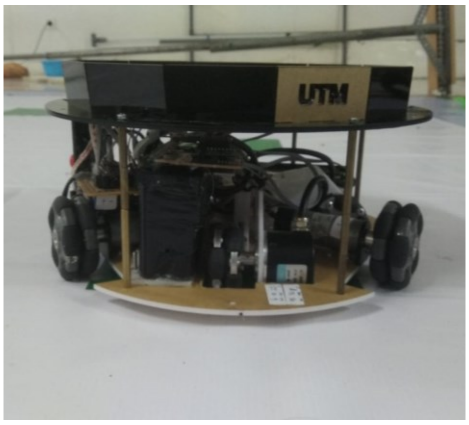


Fig. 9. Front view of the robot

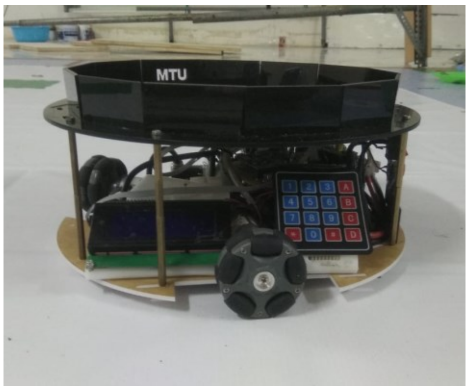


Fig. 10. Rear view of the robot

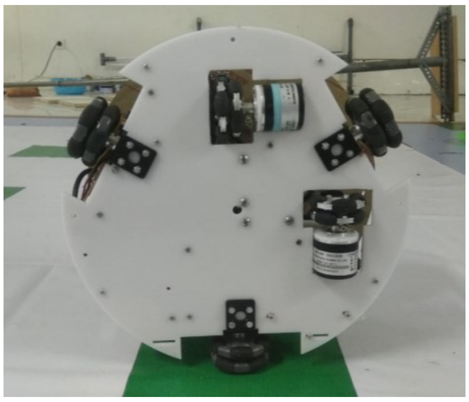


Fig. 11. Robot view from below

Fig. 12. Robot view from above

Testing the odometry method aims to determine the actual position of the robot, bearing angle (direction), and the distance from the current position of the robot to a predetermined destination point. The odometry method is used to measure changes in the position of a moving object, such as a warehouse robot in this case, which is based on data analysis from sensors installed on the robot.

Odometry, in the context of robotics, usually involves the use of sensors such as wheel encoders, which measure the rotation of the wheels and therefore the movement of the robot. Using this data, in combination with knowledge of wheel size and wheel-to-movement ratio, the odometry method can estimate the robot's position.

In this test, the odometry method refers to certain mathematical equations related to the movement and orientation of the robot. For example, a change in position can be calculated by integrating the speed of the robot against time, and a change in orientation can be calculated by considering the number of revolutions of the wheels and the width of the distance between the wheels.

At this stage, testing is done manually to verify the accuracy of the odometry method. This involves performing calculations based on mathematical equations and comparing the results with the actual data from the robot's sensors. By carrying out tests like this, it can be ensured that the odometry method works effectively and reliably for use in warehouse robot navigation systems.

The accuracy of the odometry method is very important because it is the basis for the robot to plan routes and navigate its environment. If the estimated position or orientation of the robot is inaccurate, it may cause the robot to crash into obstacles or fail to reach its destination. Therefore, it is important to carry out thorough testing and verification of this method.

Calculation Parameters Known:

$DW = 48\,mm$

$pulse_x = 1500$

$pulse_y = 100$

$resolution\ x = 600$

$resolution\ y = 100$

$x2 = 300$

$y2 = -300$

Solution:

$$KW = DW \times \pi = 48 \times 3.14 = 150.79\,mm$$

$$x1 = \frac{pulse_x}{resolution\ x} KW = \frac{1500}{600} \times 150.79 = 376.97\,mm$$

$$y1 = \frac{pulse_y}{resolution\ y} KW = \frac{100}{100} \times 150.7 = 150.79\,mm$$

$$\begin{aligned} Distance &= \sqrt{(x2 - x1)^2 - (y2 - y1)^2} \\ &= \sqrt{(300 - 376.97)^2 + (-300 - 150.79)^2} \end{aligned}$$

$$= \sqrt{5{,}924.38 + 203{,}211.62}$$

$$= 457.31\ mm$$

$$Bearing \quad = \tan^{-1}\left(\frac{y2 - y1}{x2 - x1}\right)\frac{180}{\pi}$$

$$= \tan^{-1}\left(\frac{-300 - 150.79}{300 - 376.97}\right)\frac{180}{\pi}$$

$$= \tan^{-1}(5.87)\frac{180}{\pi}$$

$$= 80.33°$$

The first test is three omni directions. This test aims to determine and compare the speed of the three wheels used, namely $v1$, $v2$, and $v3$ which is then multiplied by a certain multiplier, in this case the multiplier factor is 100 rpm. The following is an example of a calculation involving bearing parameters calculated using the odometry method.

$$bearing = 80.33°$$

$$maxRPM = 100$$

$$sudut = 120$$

$$\omega = 0$$

$$x = maxRPM \times \cos(bearing) = 100 \times \cos(80.33)$$

$$x = 16$$

$$y = maxRPM \times \sin(bearing) = 100 \times \sin(80.33)$$

$$y = 99$$

$$v_1 = (x \times \cos(sudut)) - (y \times \sin(sudut)) + \omega$$

$$v_1 = (16 \times \cos(120)) - (99 \times \sin(120)) + 0 = -77.14$$

$$v_2 = (x \times \cos(sudut)) + (y \times \sin(sudut)) + \omega$$

$$v_2 = (16 \times \cos(120)) + (-99 \times \sin(120)) + 0 = 93.14$$

$$v_3 = -(x + \omega) = -(16 + 0) = -16$$

TABLE I. TESTING OF THREE OMNI-DIRECTION

No.	Degrees (°)	v_1	v_2	v_3
1	0	9	-10	1
2	45	11	2	-9
3	90	6	6	-12
4	180	9	-10	1
5	225	-10	2	9
6	270	-6	-6	12
7	315	2	-11	8

Table I is a three omni-direction calculation based on angles. v is the speed of the robot obtained from the encoder sensor on each motor.

The next result is trajectory mapping. The purpose of mapping is to find out the coordinates of each node and the distance between nodes. Then the process of determining the shortest path is done by Dijkstra's algorithm.

In the mapping process, the selection of the position of the node point is determined by the user by pressing the (#) button. The coordinate value of the node will be read and stored in the nodeMapp variable. The nodeMapp variable is a matrix variable with dimensions of 2x50, where row 0 is used to store X values and row 1 to store Y values. The "nodeCount" variable will store the total value of the many nodes stored. If the user presses (∗), the mapping will stop and continue to the next step. Fig. 13 shows the track scheme used and mapped.

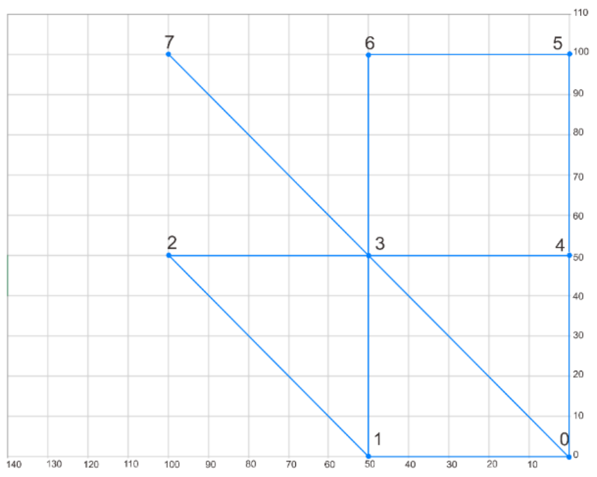


Fig. 13. Track mapping

TABLE II. MAPPING RESULT

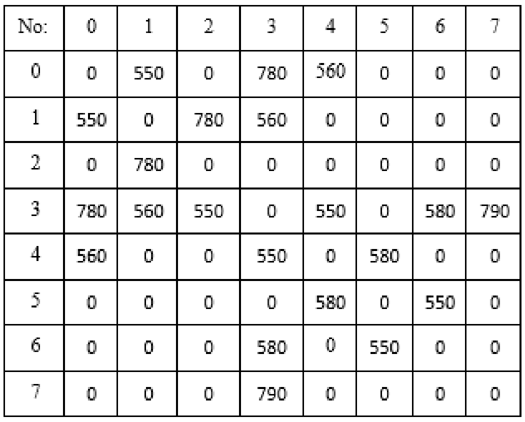
Node	X	Y
0	0	0
1	0	300
2	-300	300
3	-600	300
4	-900	300
5	-900	900
6	-600	900
7	-300	900
8	0	900

Table II is the coordinate position value at each node. The data is coordinate values in millimeters resulting from direct mapping. The data will then be converted to the path graph and determine whether the node has neighbors.

The main variables that will be used as data containers are "neighbourn" and "graphMapp". The "neighbourn" variable is used to store neighboring node data. The procedure for entering neighboring nodes is done by pressing the number on the keypad available on the robot. if the (∗) button is pressed, it will switch to the next node to be entered by its neighboring nodes. After all the neighboring nodes have been entered, the program will continue to convert the data into a path graph.

The "graphMapp" variable is the result of calculating the distance from the selected node to its neighboring nodes by applying the odometry method, the results of which will be used as weights for calculations in Dijkstra. if the selected node has no neighbors, then the value will be left 0 and not changed. Only nodes that have neighbors will have a value greater than zero as shown in Fig. 14. Fig. 14 is a path scheme obtained from the results of the "graphMapp" variable data.

Following is an experiment from the initial node = 0 and the destination node = 7. Then there are 3 path routes that can be traversed to node 7. Then the shortest path is produced, namely (0-3-7). Table III shows the results of comparing distances and times from node 0 to node 7 using Dijkstra's algorithm without Dijkstra.



No:	0	1	2	3	4	5	6	7
0	0	550	0	780	560	0	0	0
1	550	0	780	560	0	0	0	0
2	0	780	0	0	0	0	0	0
3	780	560	550	0	550	0	580	790
4	560	0	0	550	0	580	0	0
5	0	0	0	0	580	0	550	0
6	0	0	0	580	0	550	0	0
7	0	0	0	790	0	0	0	0

Fig. 14. Path graph weighting

TABLE III. RESULTS OF COMPARISON OF ROUTE DISTANCE DATA WITH DIJKSTRA AND WITHOUT DIJKSTRA

Route of Dijkstra				**Other Route**			
Track (0-3-7)				**Track (0-4-3-7)**			
From	**To**	**Time (s)**	**Distance (mm)**	**From**	**To**	**Time (s)**	**Distance (mm)**
0	3	9.9	780	0	4	6.9	560
3	7	8.6	790	4	3	6.8	550
				3	7	10	790
Sum:		**18.5**	**1570**	**Sum:**		**23.7**	**1900**

The test results show that using Dijkstra's algorithm, the robot is able to cover a distance of 780 mm. However, if using other routes (for example, routes designed manually or using a different path planning algorithm), the distance traveled by the robot is 1100 mm. That is, the use of Dijkstra's algorithm shows a higher effectiveness of 29.09%, seen from the decrease in the distance traveled.

In addition, Dijkstra's algorithm is also effective in designing the shortest path from node 0 to node 7. Based on the data, using Dijkstra's algorithm, the robot can reach its destination by traveling a distance of 1570 mm and a time of 18.5 seconds. Without using Dijkstra's algorithm, the distance covered is 1900 mm with a time of 23.7 seconds. This means that with Dijkstra's algorithm, we are able to get an effectiveness of 17.37% in terms of distance and time traveled.

This comparison shows how Dijkstra's algorithm can significantly improve the effectiveness and efficiency of warehouse robot operations. In this context, effectiveness is measured in terms of decreasing the distance traveled and decreasing the time required to reach the goal. Therefore, these results confirm the superiority of Dijkstra's algorithm in planning warehouse robot paths compared to other methods.

V. CONCLUSION

Based on this research, it can be concluded that the combination of the odometric system method and Dijkstra's algorithm can be applied to warehouse robots for goods delivery and is proven to be more effective and efficient both in distance and time compared to systems without Dijkstra. The results showed that the implementation of Dijkstra's algorithm resulted in the fastest running, which was 17% shorter distance and 33% faster time than the line follower system without Dijkstra's application.

The limitation of this research is that testing and application of the odometry system and Dijkstra's algorithm have not been carried out on a large scale or prototype. In the existing literature, similar studies have been carried out on a larger scale, involving different types of barriers and other environmental variables. However, this research is more focused on implementing and incorporating the odometry system and Dijkstra's algorithm.

The practical implication of this research is that this technology, if widely applied, can generate significant cost efficiencies in warehouse operations. This efficiency can be measured in terms of absorption of shooting range and travel time, which has a direct impact on reducing operational costs and increasing productivity.

For future research, it is recommended to carry out the test in an authentic warehouse considering the mass of goods and the speed of delivery. In addition, consideration of the GPS for large areas and other shortest displacements also needs to be considered.

In closing, this study shows great potential in using the odometry system and Dijkstra's algorithm in the context of warehouse robots. Despite the limitations in the research scale, the results make an essential contribution in this field and pave the way for further research that can optimize warehouse operations with this technology.

REFERENCES

[1] Y. Jia, B. Zhang, M. Li, B. King, and A. Meghdari, "Human-Robot Interaction," *J. Robot.*, vol. 2018, 2018, doi: 10.1155/2018/3879547.

[2] M. Hamaya, T. Matsubara, T. Teramae, T. Noda, and J. Morimoto, "Design of physical user–robot interactions for model identification of soft actuators on exoskeleton robots," *Int. J. Rob. Res.*, vol. 40, no. 1, pp. 397–410, 2021, doi: 10.1177/0278364919853618.

[3] S. Rossi, M. Staffa, and A. Tamburro, "Correction to: Socially Assistive Robot for Providing Recommendations: Comparing a Humanoid Robot with a Mobile Application," *Int. J. Soc. Robot.*, vol. 11, no. 1, p. 207, 2019, doi: 10.1007/s12369-018-0489-0.

[4] J. F. Hoorn, "Theory of Robot Communication: II. Befriending a Robot over Time," *Int. J. Humanoid Robot.*, vol. 17, no. 6, pp. 1–25, 2020, doi: 10.1142/S0219843620500279.

[5] H. Choi *et al.*, "Intuitive Bilateral Teleoperation of a Cable-driven Parallel Robot Controlled by a Cable-driven Parallel Robot," *Int. J. Control. Autom. Syst.*, vol. 18, no. 7, pp. 1792–1805, 2020, doi: 10.1007/s12555-019-0549-8.

[6] D. Ji, T. H. Kang, S. Shim, and J. Hong, "Analysis of twist deformation in wire-driven continuum surgical robot," *International Journal of Control, Automation and Systems*, vol. 18, no. 1, pp. 10-20, 2020 doi: 10.1007/s12555-018-0400-7.

[7] S. Shaju, T. George, J. K. Francis, M. Joseph, and M. J. Thomas, "Conceptual design and simulation study of an autonomous indoor medical waste collection robot," *IAES Int. J. Robot. Autom.*, vol. 12, no. 1, p. 29, 2023, doi: 10.11591/ijra.v12i1.pp29-40.

[8] W. J. Jang, J. G. Kim, S. H. Lee, and D. H. Kim, "Mechanism design for walking typed solar panel-cleaning robot using triple driving lines," *IAES Int. J. Robot. Autom.*, vol. 12, no. 1, p. 1, 2023, doi: 10.11591/ijra.v12i1.pp1-19.

[9] R. Khalesi, H. Nejat Pishkenari, and G. Vossoughi, "Independent control of multiple magnetic microrobots: design, dynamic modelling, and control," *J. Micro-Bio Robot.*, vol. 16, no. 2, pp. 215–224, 2020, doi: 10.1007/s12213-020-00136-1.

[10] A. Amin, X. Wang, A. Alroichdi, and A. Ibrahim, "Designing and

Manufacturing a Robot for Dry-Cleaning PV Solar Panels," *International Journal of Energy Research.*, vol. 2023, 2023, doi:10.1155/2023/231554.

[11] Y. Sakata and T. Suzuki, "Coverage Motion Planning Based on 3D Model's Curved Shape for Home Cleaning Robot," *J. Robot. Mechatronics*, vol. 35, no. 1, pp. 30–42, 2023, doi: 10.20965/jrm.2023.p0030.

[12] J. Yan and H. Cheng, "Designing and Manufacturing of Industrial Robots with Dual- Angle Sensors Taking into Account Vibration Signal Fusion," *Journal of Robotics.*, vol. 2023, 2023, doi: 10.1155/2023/1855226.

[13] X. Xie, "Industrial Robot Assembly Line Design Using Machine Vision," *J. Robot.*, vol. 2023, 2023, doi: 10.1155/2023/4409033.

[14] F. Ore, B. Vemula, L. Hanson, M. Wiktorsson, and B. Fagerström, "Simulation methodology for performance and safety evaluation of human–industrial robot collaboration workstation design," *Int. J. Intell. Robot. Appl.*, vol. 3, no. 3, pp. 269–282, 2019, doi: 10.1007/s41315-019-00097-0.

[15] J. Hažík, M. Dekan, P. Beňo, and F. Duchoň, "Fleet Management System for an Industry Environment," *Journal of Robotics and Control (JRC)*, vol. 3, no. 6, pp. 779–789, 2022, doi: 10.18196/jrc.v3i6.16298.

[16] A. A. N. Kumaar, S. Kochuvila, and S. R. Nagaraja, "A Scalable Tree Based Path Planning for A service Robot," *Journal of Automation, Mobile Robotics and Intelligent Systems*, vol. 16, pp. 31–45, 2022, doi: 10.14313/JAMRIS/1.

[17] A. Latif, H. A. Widodo, R. Rahim, and K. Kunal, "Implementation of line follower robot based microcontroller atmega32a," *Journal of Robotics and Control (JRC)*, vol. 1, no. 3, pp. 70–74, 2020, doi: 10.18196/jrc.1316.

[18] Y. Zhao, Y. Zhang, and J. Lee, "Lyapunov and Sliding Mode Based Leader-follower Formation Control for Multiple Mobile Robots with an Augmented Distance-angle Strategy," *Int. J. Control. Autom. Syst.*, vol. 17, pp. 1314-1321, 2019, doi: 10.1007/s12555-018-0194-7.

[19] S. M. Swadi, A. K. Kadhim, and G. M. Ali, "Design of Path Planning Controller of Autonomous Wheeled Mobile Robot Based on Triple Pendulum Behaviour," *Int. J. Mech. Eng. Robot. Res.*, vol. 12, no. 1, pp. 23–31, 2023, doi: 10.18178/ijmerr.12.1.23-31.

[20] Y. Wang, K. Gong, Y. Duan, B. He, and H. Ma, "Dynamic Modelling and Continuous Trajectory Tracking Control of Space Robots Based on Lie Group SE (3)," *International Journal of Aerospace Engineering*, vol. 2023, 2023, doi: 10.1155/2023/7435217.

[21] I. Kostavelis, E. Boukas, L. Nalpantidis, and A. Gasteratos, "Stereo-based visual odometry for autonomous robot navigation," *International Journal of Advanced Robotic Systems*, vol. 13, no. 1, p. 21, 2016, doi: 10.5772/62099.

[22] H. Wu, Y. Gao, and S. Li, "Odometry Estimation Utilizing 6-DOF Force Sensors and IMU for Legged Robot," *2020 Chinese Automation Congress (CAC)*, pp. 6901-6905, 2020, doi: 10.1109/CAC51589.2020.9326974.

[23] S. Takeda and T. Umetani, "Initial Localization of Mobile Robot Based on Expansion Resetting Without Manual Pose Adjustment in Robot Challenge Experiment," *Journal of Robotics and Mechatronics*, vol. 35, no. 2, pp. 380–386, 2023, doi: 10.20965/jrm.2023.p0380.

[24] Y. Gao and L. Zhao, "Coarse TRVO: A Robust Visual Odometry with Detector-Free Local Feature," *J. Adv. Comput. Intell. Intell. Informatics*, vol. 26, no. 5, pp. 731–739, 2022, doi: 10.20965/jaciii.2022.p0731.

[25] Q. Wang, J. Zhang, Y. Liu, and X. Zhang, "High-Precision and Fast LiDAR Odometry and Mapping Algorithm," *J. Adv. Comput. Intell. Intell. Informatics*, vol. 26, no. 2, pp. 206–216, 2022, doi: 10.20965/jaciii.2022.p0206.

[26] G. Xie, Q. Zong, X. Zhang, and B. Tian, "Loosely-coupled lidar-inertial odometry and mapping in real time," *Int. J. Intell. Robot. Appl.*, vol. 5, no. 2, pp. 119–129, 2021, doi: 10.1007/s41315-021-00164-5.

[27] F. Spiess, J. Friesslich, T. Kaupp, S. Kounev, and N. Strobel, "Survey and Experimental Comparison of RGB-D Indoor Robot Navigation Methods Supported by ROS and Their Expansion via Fusion with Wheel Odometry and IMU Data," *Int. J. Mech. Eng. Robot. Res.*, vol. 9, no. 12, pp. 1532–1540, 2020, doi: 10.18178/IJMERR.9.12.1532-1540.

[28] D. U. Rijalusalam and I. Iswanto, "Implementation kinematics modeling and odometry of four omni wheel mobile robot on the trajectory planning and motion control based microcontroller," *Journal of Robotics and Control (JRC)*, vol. 2, no. 5, pp. 448–455, 2021, doi: 10.18196/jrc.25121.

[29] A. Kostusiak and P. Skrzypczyński, "On the Efficiency of Population-Based Optimization in Finding Best Parameters for RGB-D Visual Odometry," *J. Autom. Mob. Robot. Intell. Syst.*, vol. 13, no. 2, pp. 5–14, 2019, doi: 10.14313/JAMRIS/2-2019/13.

[30] N. I. Giannoccaro, T. Nishida, A. Lay-Ekuakille, R. Velazquez, and P. Visconti, "Processing of LiDAR and IMU data for target detection and odometry of a mobile robot," *Journal of Automation, Mobile Robotics and Intelligent Systems*, vol. 16, no. 1, pp. 3-13, 2022, doi: 10.14313/JAMRIS/1-2022/1.

[31] X. Fu *et al.*, "Self-supervised learning of LiDAR odometry based on spherical projection," *Int. J. Adv. Robot. Syst.*, vol. 19, no. 1, pp. 1–13, 2022, doi: 10.1177/17298806221078669.

[32] Z. Zhao, Y. Zhang, L. Long, Z. Lu, and J. Shi, "Efficient and adaptive lidar–visual–inertial odometry for agricultural unmanned ground vehicle," *Int. J. Adv. Robot. Syst.*, vol. 19, no. 2, pp. 1–15, 2022, doi: 10.1177/17298806221094925.

[33] A. N. Albab, E. Rahmawati, M. Yantidewi, I. Sucahyo, Dzulkiflih, and R. R. Firmansyah, "Control Position of Mobile Robot Based on Odometry Method and PID Controller," *J. Phys. Conf. Ser.*, vol. 1491, no. 1, p. 012039, 2020, doi: 10.1088/1742-6596/1491/1/012039.

[34] M. Taufiqqurohman and N. F. Sari, "Odometry Method and Rotary Encoder for Wheeled Soccer Robot," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 407, no. 1, 2018, doi: 10.1088/1757-899X/407/1/012103.

[35] S. Saeedvand, H. S. Aghdasi, and J. Baltes, "Novel lightweight odometric learning method for humanoid robot localization," *Mechatronics*, vol. 55, pp. 38–53, 2018, doi: 10.1016/j.mechatronics.2018.08.007.

[36] G. Gong, S. Zeng, J. Gao, Q. Zhang, and X. Wang, "Discovery and Discrimination of Bridge Engineering Safety Issues by BIM Virtual Scene Combined with Robotic Mapping," *Journal of Robotics*, vol. 2023, 2023, doi: 10.1155/2023/3028505.

[37] B. Sebastian and P. Ben-Tzvi, "Support vector machine based real-time terrain estimation for tracked robots," *Mechatronics*, vol. 62, p. 102260, 2019, doi: 10.1016/j.mechatronics.2019.102260.

[38] A. Deo, A. Gupta, H. Khemani, and R. R. Das, "Path tracking mobile robot using steppers," *E3S Web Conf.*, vol. 87, p. 01028, 2019, doi: 10.1051/e3sconf/20198701028.

[39] T. Zhang, M. Zhang, and Y. Zou, "Time-optimal and Smooth Trajectory Planning for Robot Manipulators," *Int. J. Control. Autom. Syst.*, vol. 19, no. 1, pp. 521–531, 2021, doi: 10.1007/s12555-019-0703-3.

[40] J. Ahmed Abdulsaheb and D. Jasim Kadhim, "Real-Time SLAM Mobile Robot and Navigation Based on Cloud-Based Implementation," *Journal of Robotics*, vol. 2023, 2023, doi: 10.1155/2023/9967236.

[41] D. H. T. Kim *et al.*, "Adaptive Control for Uncertain Model of Omni-directional Mobile Robot Based on Radial Basis Function Neural Network," *Int. J. Control. Autom. Syst.*, vol. 19, no. 4, pp. 1715–1727, 2021, doi: 10.1007/s12555-019-1004-6.

[42] N. Zijie, L. Qiang, C. Yonjie, and S. Zhijun, "Fuzzy Control Strategy for Course Correction of Omnidirectional Mobile Robot," *Int. J. Control. Autom. Syst.*, vol. 17, no. 9, pp. 2354–2364, 2019, doi: 10.1007/s12555-018-0633-5.

[43] Y. Ueno, I. Ikemura, T. Tanaka, and Y. Matsuo, "Development of a Front-Wheel-Steering-Drive Dual-Wheel Caster Drive Mechanism for Omni-Directional Wheelchairs with High Step Climbing Performance," *J. Robot. Mechatronics*, vol. 34, no. 6, pp. 1431–1440, 2022, doi: 10.20965/jrm.2022.p1431.

[44] R. T. Yunardi, D. Arifianto, F. Bachtiar, J. I. Prananingrum, and U. Airlangga, "Holonomic Implementation of Three Wheels Omnidirectional Mobile Robot using DC Motors," *Journal of Robotics and Control (JRC)*, vol. 2, no. 2, 2021, doi: 10.18196/jrc.2254.

[45] B. Wu, D. Qin, Y. Chen, T. Q. Cao, and M. Wu, "Structure design of an omni-directional wheeled handling robot," *J. Phys. Conf. Ser.*, vol. 1885, no. 5, 2021, doi: 10.1088/1742-6596/1885/5/052013.

[46] X. Yang, "The invention relates to a small logistics handling trolley based on omni-directional wheel movement," *J. Phys. Conf. Ser.*, vol. 2246, no. 1, 2022, doi: 10.1088/1742-6596/2246/1/012005.

[47] L. Song, H. Ju, W. Li, C. Sun, and B. Yuan, “Design and Research of Omni-directional Moving AGV,” *J. Phys. Conf. Ser.*, vol. 1575, no. 1, 2020, doi: 10.1088/1742-6596/1575/1/012095.

[48] Y. Wang, H. Zhu, Y. Yu, and B. Hu, “The Path Planning and Location Method of Inspection Robot in a Large Storage Tank Bottom,” *Computational Intelligence and Neuroscience*, vol. 2023, 2023, doi: 10.1155/2023/3029545.

[49] Z. Pan, D. Wang, H. Deng, and K. Li, “A Virtual Spring Method for the Multi-robot Path Planning and Formation Control,” *Int. J. Control. Autom. Syst.*, vol. 17, no. 5, pp. 1272–1282, 2019, doi: 10.1007/s12555-018-0690-9.

[50] C. Ye, D. Zhao, S. Yu, C. Jiang, and P. Li, “Stability Improvement of Mobile Robot with Mutative Driving Axial Distance Omni-Directional Wheels,” *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pp. 325-330, 2018, doi: 10.1109/RCAR.2018.8621658.

[51] M. Fronita, R. Gernowo, and V. Gunawan, “Comparison of Genetic Algorithm and Hill Climbing for Shortest Path Optimization Mapping,” *E3S Web Conf.*, vol. 31, pp. 1–5, 2018, doi: 10.1051/e3sconf/20183111017.

[52] W. Liao, X. Wei, J. Lai, and H. Sun, “Numerical Method with High Real-time Property Based on Shortest Path Algorithm for Optimal Control,” *Int. J. Control. Autom. Syst.*, vol. 19, no. 6, pp. 2038–2046, 2021, doi: 10.1007/s12555-020-0196-0.

[53] Y. Liu and Y. Jiang, “Robotic Path Planning Based on a Triangular Mesh Map,” *Int. J. Control. Autom. Syst.*, vol. 18, no. 10, pp. 2658–2666, 2020, doi: 10.1007/s12555-019-0396-z.

[54] X. Li, “Path planning of intelligent mobile robot based on Dijkstra algorithm,” *J. Phys. Conf. Ser.*, vol. 2083, no. 4, 2021, doi: 10.1088/1742-6596/2083/4/042034.

[55] I. G. S. Rahayuda and N. P. L. Santiari, “Dijkstra and Bidirectional Dijkstra on Determining Evacuation Routes,” *J. Phys. Conf. Ser.*, vol. 1803, no. 1, 2021, doi: 10.1088/1742-6596/1803/1/012018.

[56] D. Verma, D. Messon, M. Rastogi, and A. Singh, “Comparative Study Of Various Approaches Of Dijkstra Algorithm,” *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, pp. 328-336, 2021, doi: 10.1109/ICCCIS51004.2021.9397200.

[57] K. Wei, Y. Gao, W. Zhang, and S. Lin, “A Modified Dijkstra’s Algorithm for Solving the Problem of Finding the Maximum Load Path,” *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, pp. 10-13, 2019, doi: 10.1109/INFOCT.2019.8711024.

[58] G. Deepa, M. Angamuthu, K. Rajakumar, and K. Venkatesan, “Dijkstra Algorithm Application: Shortest Distance between Buildings,” *International Journal of Engineering and Technology*, vol. 7, no. 4.10, pp. 974-976, 2018, doi: 10.14419/ijet.v7i4.10.26638.

[59] I. E. Salem, M. M. Mijwil, A. W. Abdulqader, and M. M. Ismaeel, “Flight-schedule using Dijkstra’s algorithm with comparison of routes findings,” *Int. J. Electr. Comput. Eng.*, vol. 12, no. 2, pp. 1675–1682, 2022, doi: 10.11591/ijece.v12i2.pp1675-1682.

[60] M. Lotfi *et al*., “A Dijkstra-Inspired Algorithm for Optimized Real-Time Tasking with Minimal Energy Consumption,” *2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*, pp. 1-6, 2020, doi: 10.1109/EEEIC/ICPSEurope49358.2020.9160688.

[61] Q. Lin, X. Liu, and Z. Zhang, “Mobile Robot Self-LocalizationUsing Visual Odometry Based on Ceiling Vision,” *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1435-1439, 2019, doi: 10.1109/SSCI44817.2019.9003092.

[62] T. T. Pham, M. T. Le, and C. N. Nguyen, “Omnidirectional mobile robot trajectory tracking control with diversity of inputs,” *International Journal of Mechanical Engineering and Robotics Research*, *10*(11), 639-644, 2021, doi: 10.18178/ijmerr.10.11.639-644.

[63] S. Morales, J. Magallanes, C. Delgado, and R. Canahuire, “LQR Trajectory Tracking Control of an Omnidirectional Wheeled Mobile Robot,” *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*, pp. 1-5, 2018, doi: 10.1109/CCRA.2018.8588146.

[64] A. Ubaidillah, A. F. Ibadillah, I. Turmudzi, and A. Rachmad, “Representation of Soccer Robotics in The Fastest Trajectory Tracking,” *IEEE 8th Information Technology International Seminar (ITIS)*, pp. 90–95, 2022, doi: 10.1109/ITIS57155.2022.10010172.

[65] K. N. Hitesh, J. M. Kumar Reddy, K. T. Ilayarajaa, R. M. Joany, and V. Vijayakumar, “IOT Based Omni Directional Robot Control by Using ARM-Series,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 590, no. 1, 2019, doi: 10.1088/1757-899X/590/1/012054.

[66] S. Fadlo, N. Rabbah, and A. Ait Elmahjoub, “Energy estimation based on path tracking for a differential drive wheeled mobile robot,” *E3S Web Conf.*, vol. 229, pp. 1–5, 2021, doi: 10.1051/e3sconf/202122901029.

[67] A. J. Clark, K. A. Cissell, J. M. Moore, and X. Liu, “Evolving Controllers for a Transformable Wheel Mobile Robot,” *Complexity*, vol. 2018, pp. 1-12, 2018, doi: 10.1155/2018/7692042.

[68] W. Ao, L. Zhang, H. Zhang, Z. Li, and G. Huang, “Structure Design and Event-Triggered Control of a Modular Omnidirectional Mobile Chassis of Life Support Robotics,” *Fractal and Fractional*, vol. 7, no. 2, p. 121, 2023, doi: 10.3390/fractalfract7020121.

[69] B. He, S. Wang, and Y. Liu,”Underactuated robotics: a review,” *International Journal of Advanced Robotic Systems*, vol. 16, no. 4, pp. 1-29, 2019, doi: 10.1177/1729881419862164.

[70] G. Yi, J. Mao, Y. Wang, S. Guo, and Z. Miao, “Adaptive Tracking Control of Nonholonomic Mobile Manipulators Using Recurrent Neural Networks,” *Int. J. Control. Autom. Syst.*, vol. 16, no. 3, pp. 1390–1403, 2018, doi: 10.1007/s12555-017-0309-6.

[71] Z. Hong, W. Du, and H. Wang, “Design and Implementation of Path Planning for Wheel-Track Hybrid Mobile Robot,” *Mob. Inf. Syst.*, vol. 2022, 2022, doi: 10.1155/2022/6418706.

[72] B. Xu and C. Sem-Lin, “Motion Trajectory Error of Robotic Arm Based on Neural Network Algorithm,” *J. Control Sci. Eng.*, vol. 2023, 2023, doi: 10.1155/2023/3958434.

[73] N. A. Abd Rahman, K. S. M. Sahari, N. A. Hamid, and Y. C. Hou, “A coverage path planning approach for autonomous radiation mapping with a mobile robot,” *International Journal of Advanced Robotic Systems*, vol. 19, no. 4, 2022, doi: 10.1177/17298806221116483.

[74] A. M. El-Dalatony, T. Attia, H. Ragheb, and A. M. Sharaf, “Cascaded PID Trajectory Tracking Control for Quadruped Robotic Leg,” *Int. J. Mech. Eng. Robot. Res.*, vol. 12, no. 1, pp. 40–47, 2023, doi: 10.18178/ijmerr.12.1.40-47.

[75] J. Santos, A. Conceição, T. Santos, and H. Araújo, “Remote control of an omnidirectional mobile robot with time-varying delay and noise attenuation,” *Mechatronics*, vol. 52, pp. 7–21, 2018, doi: 10.1016/j.mechatronics.2018.04.003.

[76] A. Saenz, V. Santibañez, E. Bugarin, A. Dzul, H. Ríos, and J. Villalobos-Chin, “Velocity Control of an Omnidirectional Wheeled Mobile Robot Using Computed Voltage Control with Visual Feedback: Experimental Results,” *Int. J. Control. Autom. Syst.*, vol. 19, no. 2, pp. 1089–1102, 2021, doi: 10.1007/s12555-019-1057-6.

[77] F. Umam, M. Fuad, I. Suwarno, A. Ma'arif, and W. Caesarendra, “Obstacle Avoidance Based on Stereo Vision Navigation System for Omni-directional Robot,” *Journal of Robotics and Control (JRC)*, vol. 4, no. 2, pp. 227-242, 2023, doi: 10.18196/jrc.v4i2.17977.

[78] P. L. Wu, J. J. Li, and J. S. Shaw, “Development of an Omnidirectional AGV by Applying ORB-SLAM for Navigation Under ROS Framework,” *J. Autom. Mob. Robot. Intell. Syst.*, vol. 16, no. 1, pp. 14–20, 2022, doi: 10.14313/JAMRIS/1-2022/2.

[79] Q. Ran, S. Yao, X. Chen, and G. Bi, “Trajectory Tracking of Swing-Arm Type Omnidirectional Mobile Robot,” *Math. Probl. Eng.*, vol. 2022, 2022, doi: 10.1155/2022/3297789.

[80] Y. Zhao, “Dynamic Path Planning Analysis of Warehouse Handling Robot,” *J. Sensors*, vol. 2022, pp. 1–7, 2022, doi: 10.1155/2022/4434971.

[81] G. Ziwei and L. Rong, “2D Range Flow-based Odometry fusing LiDAR and IMU,” *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2761-2765, 2019, doi: 10.1109/ROBIO49542.2019.8961747.

[82] S. A. S. Mohamed, M. H. Haghbayan, T. Westerlund, J. Heikkonen, H. Tenhunen, and J. Plosila, “A Survey on Odometry for Autonomous Navigation Systems,” *IEEE Access*, vol. 7, pp. 97466–97486, 2019, doi: 10.1109/ACCESS.2019.2929133.

[83] G. Liu, J. Guan, H. Liu, C. Wang, and X. L. Wang, “Multirobot Collaborative Navigation Algorithms Based on Odometer/Vision

Information Fusion," *Math. Probl. Eng.*, vol. 2020, pp. 1-16, 2020, doi: 10.1155/2020/5819409.

[84] X. Zhao, H. Min, Z. Xu, X. Wu, X. Li, and P. Sun, "Image antiblurring and statistic filter of feature space displacement: Application to visual odometry for outdoor ground vehicle," *J. Sensors*, vol. 2018, 2018, doi: 10.1155/2018/2987819.

[85] S. Maldonado-Bascón, R. J. López-Sastre, F. J. Acevedo-Rodríguez, and P. Gil-Jiménez, "On-board correction of systematic odometry errors in differential robots," *J. Sensors*, vol. 2019, 2019, doi: 10.1155/2019/8269256.

[86] J. Zhu, Y. Tang, X. Shao and Y. Xie, "Multisensor Fusion Using Fuzzy Inference System for a Visual-IMU-Wheel Odometry," in *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1-16, 2021, doi: 10.1109/TIM.2021.3051999.

[87] Z. Huai and G. Huang, "Robocentric visual–inertial odometry," *The International Journal of Robotics Research*, vol. 41, no. 7, pp. 667-689, 2022, doi: 10.1177/0278364919853361.

[88] Y. Teekaraman, I. Kirpichnikova, H. Manoharan, R. Kuppusamy, and A. Radhakrishnan, "Uncovering Resilient Actions of Robotic Technology with Data Interpretation Trajectories Using Knowledge Representation Procedures," *Security and Communication Networks*, vol. 2023, 2023, doi: 10.1155/2023/7419259.

[89] C. Gu, A. Feng, G. Wang, and X. Liu, "Robot Path Planning of Improved Adaptive Ant Colony System Algorithm Based on Dijkstra," *J. Robot.*, vol. 2022, 2022, doi: 10.1155/2022/9229155.

[90] Y. Sun, M. Fang, and Y. Su, "AGV Path Planning based on Improved Dijkstra Algorithm," *J. Phys. Conf. Ser.*, vol. 1746, no. 1, 2021, doi: 10.1088/1742-6596/1746/1/012052.

[91] L. S. Liu *et al.*, "Path Planning for Smart Car Based on Dijkstra Algorithm and Dynamic Window Approach," *Wirel. Commun. Mob. Comput.*, vol. 2021, pp. 1-12, 2021, doi: 10.1155/2021/8881684.

[92] Q. Liu, H. Xu, L. Wang, J. Chen, Y. Li, and L. Xu, "Application of Dijkstra Algorithm in Path Planning for Geomagnetic Navigation," *2020 IEEE 11th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, pp. 1-4, 2020, doi: 10.1109/SAM48682.2020.9104382.

[93] Z. Halim, A. Khan, M. Sulaiman, S. Anwar, and M. Nawaz, "On finding optimum commuting path in a road network: A computational approach for smart city traveling," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 2, pp. 1–28, 2022, doi: 10.1002/ett.3786.

[94] B. Zhang and D. J. Hu, "Retraction Note: Research on the construction and simulation of PO-Dijkstra algorithm model in parallel network of multicore platform," *Eurasip J. Wirel. Commun. Netw.*, vol. 2022, no. 121, pp. 1-14, 2022, doi: 10.1186/s13638-022-02201-8.

[95] W. Hadikurniawati, E. Winarno, A. Hernawan, and D. Abdullah, "Retracted: Optimization of ISP Service Maintenance Router Using Dijkstra and Flyod-Warshall Algorithm," *J. Phys. Conf. Ser.*, vol. 1114, no. 1, 2018, doi: 10.1088/1742-6596/1114/1/012101.

[96] A. Alyasin, E. I. Abbas, and S. D. Hasan, "An Efficient Optimal Path Finding for Mobile Robot Based on Dijkstra Method," *2019 4th Scientific International Conference Najaf (SICN)*, pp. 11-14, 2019, doi: 10.1109/SICN47020.2019.9019345.

[97] T. Irfan, R. Hakimi, A. C. Risdianto, and E. Mulyana, "ONOS Intent Path Forwarding using Dijkstra Algorithm," *2019 International Conference on Electrical Engineering and Informatics (ICEEI)*, pp. 549-554, 2019, doi: 10.1109/ICEEI47359.2019.8988853.

[98] Z. Ullah, H. Bashir, R. Anjum, S. A. Alqahtani, S. Al-Hadhrami, and A. Ghaffar, "Analysis of the Shortest Path in Spherical Fuzzy Networks Using the Novel Dijkstra Algorithm," *Math. Probl. Eng.*, vol. 2021, pp. 1-15, 2021, doi: 10.1155/2021/7946936.

[99] L. Wenzheng, L. Junjun, and Y. Shunli, "An Improved Dijkstra's Algorithm for Shortest Path Planning on 2D Grid Maps," *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, pp. 438-441, 2019, doi: 10.1109/ICEIEC.2019.8784487.

[100] H. Li, P. Tong, and X. Zhang, "Method for Determining the Location of Highway Passenger Transportation Hubs Using POI Data and the Dijkstra Algorithm in Large City," *Math. Probl. Eng.*, vol. 2022, 2022, doi: 10.1155/2022/2189598.

[101] C. Zhou, Z. Chen, X. Lv, D. Gao, and M. Zhao, "Design of intelligent sorting trash dustbin based on STM32," *E3S Web Conf.*, vol. 198, pp. 1–4, 2020, doi: 10.1051/e3sconf/202019804032.

[102] X. He, Y. Kuang, N. Song, and F. Liu, "Intelligent Navigation of Indoor Robot Based on Improved DDPG Algorithm," Mathematical Problems in Engineering., vol. 2023, 2023, doi: 10.1155/2023/6544029.

[103] N. Matsui *et al.*, "Local and Global Path Planning for Autonomous Mobile Robots Using Hierarchized Maps," *J. Robot. Mechatronics*, vol. 34, no. 1, pp. 86–100, 2022, doi: 10.20965/jrm.2022.p0086.

[104] B. Tan, "Soccer-assisted training robot based on image recognition omnidirectional movement," *Wirel. Commun. Mob. Comput.*, vol. 2021, pp. 1-10, 2021, doi: 10.1155/2021/5532210.

[105] M. Ouyang, Z. Cao, P. Guan, Z. Li, C. Zhou, and J. Yu, "Visual-gyroscope-wheel odometry with ground plane constraint for indoor robots in dynamic environment," *IEEE Sensors Letters*, vol. 5, no. 3, pp. 1-4, 2021, doi: 10.1109/LSENS.2021.3057088.

[106] J. Li *et al.*, "A Lightweight Stereo Visual Odometry System for Navigation of Autonomous Vehicles in Low-Light Conditions," *Wirel. Commun. Mob. Comput.*, vol. 2022, 2022, doi: 10.1155/2022/5249449.

[107] G. Schatzberger, F. P. Leisenberger, P. Sarson, and A. Wiesner, "High efficient low cost EEPROM screening method in combination with an area optimized byte replacement strategy which enables high reliability EEPROMs," *2018 IEEE 36th VLSI Test Symposium (VTS)*, pp. 1-6, 2018, doi: 10.1109/VTS.2018.8368631.