# Optimized Maze-Solving Algorithm for Mobile Robots

C. Sánchez-López[1,*], C. Muñiz-Montero[2], J. Arellano-Hernández[1]
C. Hernández-Mejía[3] and D. Torres-Muñoz[4]

[1]Department of Electronics, Autonomous University of Tlaxcala, Apizaco, 90300 Mexico
[2]Electronics and Telecommunications Department,
Polytechnic University of Puebla, Puebla, 72640, Mexico
[3]Sub-Directorate of Postgraduate Studies and Research,
Higher Technological Institute of Misantla, Veracruz, 93821, Mexico
[4]Department of Computer Systems,
Higher Technological Institute of San Martin Texmelucan, Puebla 74120, Mexico
Email: csanchezl@uatx.mx (C.S.L.); carlos.muniz@uppuebla.edu.mx (C.M.M.);
jarellanoh@uatx.mx (J.A.H.); cmahernandez@gmail.com (C.H.M.);
deletsm@gmail.com (D.T.M.)
*Corresponding author

***Abstract*—This article addresses the development of an optimized algorithm for maze solving using image processing techniques and resistive grids. Since the diagonal square geometric cell is used as basic resistive cell, the proposed algorithm optimizes the process of finding the shortest obstacle-free route from initial node to the target node into an irregular maze plagued with irregular obstacles. Depending on the dimensions of the robot, a node matrix with the same size as the maze-image is generated and the distance between each node is equal to the largest dimension of the robot. The maze image can be scaled from 1m×1m to 40 m × 40 m. The equation system is improved by building it from the collision-free node matrix instead the full resistive grid. Nodal analysis formulates the admittance matrix and Kirchhoff's current law then determines the shortest path within the reduced resistive network, after removing nodes and branches absorbed by irregular obstacles. An undirected graph decision algorithm avoids singular matrices during formulation. Thus, the omnidirectional robot follows the collision-free shortest path from start to end node in the main sub-network. As a result, travel time, distance traveled and battery consumption of an omnidirectional mobile robot used as test vehicle are minimized. The proposed algorithm is validated using three case studies cases. Compared to the results obtained by directly using the square resistive cell, the experimental results demonstrate that the optimized algorithm works better and more efficiently, reducing the following for the largest maze: CPU-time=7.76 s, travel time=146.26 s, dis-tance=31.99 m and battery usage=2.42%.**

***Keywords*—circuit analysis, omnidirectional robot, resistive grid, maze-solving, micro-controllers**

## I. Introduction

Maze solving through robotic systems has attracted considerable attention in recent decades due to its wide range of autonomous and semi-autonomous applications involving human interaction [1], [2]. These include industrial automation, entertainment, intelligent traffic management, defense, medical assistance, logistics, domestic services, firefighting, agriculture and robotic rescue operations, among others [3], [4]. At its core, maze solving involves navigating from a predefined starting point to a target endpoint by following the most efficient, collision-free path within an environment filled with irregular obstacles. To address this challenge, numerous methodologies have been proposed in the literature, based on both graph theory [5]–[8] and non-graph approaches [9]–[12], many of which have been implemented in robotic systems with the goal of determining the shortest viable path through a maze. A comparative study presented in [3], [7] evaluated several graph- and non-graph-based algorithms, concluding that graph-based methods generally offer superior performance in terms of efficiency, reliability and computation time. For instance, [13] utilized cell decomposition and potential field techniques to develop global and local trajectory planning. However, a known limitation of this approach is the possibility of the robot becoming trapped in local minima due to the use of negative gradient fields. In [14], a weighted shortest-path algorithm for multi-agent environments was proposed. Here, the maze is divided into discrete cells and each time a robot enters a cell, the visit count is updated. Once a robot reaches the goal, this information is shared with the rest of the agents to guide their

Manuscript received February 8, 2026; revised March 22, 2026; accepted May 25, 2026; July 23, 2026.

doi: 10.18178/ijmerr.15.4.379-390

paths. The main drawback lies in the significant hardware requirements, even for relatively small mazes. Simpler strategies such as flood fill and wall-following algorithms were explored in [15], [16], proving effective in small-scale environments, though inadequate for larger and more complex mazes. A modified Breadth-First Search (BFS) approach was introduced in [17], where the maze is discretized and each robot movement is recorded in a list representing graph nodes. Nevertheless, this method becomes computationally expensive in large mazes, consuming substantial CPU resources, particularly when the maze division is performed arbitrarily. In [18], [19], the membrane pseudo-bacterial potential field method was used to develop a hybrid path planing algorithm for mobile robots. However, this method has some disadvantages such as the local minima problem and target inaccessibility with nearby obstacles. Furthermore, the real dimensions of the mobile robot were not considered during the computer simulations. Image-based solutions have also been proposed. In [20], a panoramic view of the maze was constructed from multiple photographs taken from different angles. However, distant areas of the maze remained unclear, leading to errors in path planning. The work in [21], [22] divided images into grids to facilitate path planning, though collisions occurred when grid nodes were placed too close to obstacles. In [23], shortest paths were derived from maze images using an n-block labeling technique, which proved sensitive to image noise, thereby affecting accuracy. Skeletonization methods were implemented in [24]–[26] to extract paths from binary maze images. However, these techniques depend on morphological pruning, which limits their effectiveness in large or complex mazes and leads to exponentially increasing memory requirements. In [27], [28], chaos $A^*$ and circulation heuristic search algorithms were proposed, yielding shorter paths with lower computational times compared to traditional $A^*$. However, these approaches were applied to planar manipulators and not to mobile robots. Alternative maze-solving techniques based on resistive [29]–[32] and memristive grids [33]–[36] have also been reported. These methods eliminate nodes and branches that intersect with obstacles, resulting in fragmented sub-networks within the resistive grid. This creates significant limitations, as the resulting systems of equations often involve singular matrices, making them difficult or impossible to solve. Furthermore, the complexity involved in solving large systems of equations leads to inefficient use of computational resources both during matrix generation and when limiting the shortest-path search to only a section of the network matrix. A critical observation across all existing maze-solving approaches, as detailed in [8], [32], is the assumption that mazes and their obstacles have regular geometries tailored for algorithmic convenience an assumption that diverges significantly from real-world environments. Moreover, most methods disregard the physical dimensions of the robot, which can result in path-planning failures when maze dimensions are scaled.

In this work, the maze-solving methodology for mobile robots based on resistive grids, previously reported in [32], is enhanced by incorporating diagonal movement into the navigation process. The proposed optimized approach leverages image processing techniques and Nodal Analysis (NA). Unlike the method in [32], where the full node matrix was used, the improved strategy generates an optimized, obstacle-free resistive grid directly from the reduced node matrix, significantly reducing computational demands. Once the resistive grid is established, the corresponding system of equations is formulated and node voltages are computed. Subsequently, by applying Kirchhoff's Current Law (KCL), the shortest collision-free path is determined. A key innovation of the proposed methodology is the explicit consideration of the mobile robots physical dimensions, which enables scalable path planning across maze sizes ranging from $1\text{m}\times1\text{m}$ to $40\text{m}\times40\text{m}$, without compromising accuracy or performance. This scalability, combined with enhanced computational efficiency and robustness, forms the central motivation for this research. The main contributions of this work are as follows

1. Efficient generation of the resistive grid from the reduced node matrix, resulting in lower computational overhead.
2. Incorporation of diagonal movement into the instruction transformation process, enabling full utilization of omnidirectional robot capabilities.
3. Improved performance metrics, including reduced CPU-time, travel time, distance traveled and battery consumption during maze navigation.

With regards to the article structure, Section II details the automatic generation and optimization of the resistive grid, incorporating both the robot dimensions and the maze scale. Section III presents the shortest path planning methodology based on NA. Section IV discusses numerical results from three case studies involving mazes of varying sizes. Section V provides the experimental validation

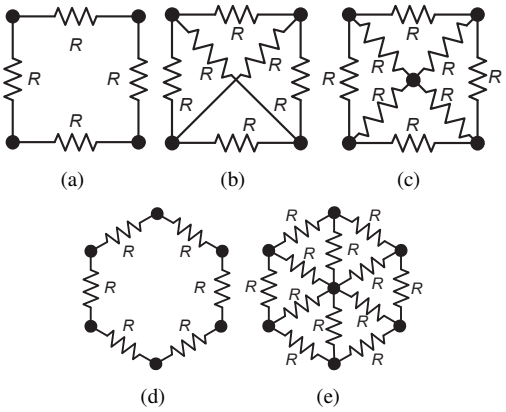


Fig. 1. Basic cells to build a resistive network: (a) Square, (b) Square-diagonal, (c) Square-center, (d) Hexagonal, (e) Hexagonal-center

of the proposed method. Section VI presents a discussion of the advantages and limitations of the developed methodology. Finally, Section VII concludes the paper with final remarks and future work perspectives.

## II. Improved Resistive Grid Generation

A resistive grid can be generated using a basic geometric cell, as those illustrated in Fig. 1. Of all of them, the square-diagonal cell is the one used here. In a first step, assuming the mobile robot has length $L$ [m] and width $W$ [m] and the maze dimensions are $(x_m, y_m)$ [m,m], a node matrix is then built as

$$A = \begin{bmatrix} 1 & r+1 & 2r+1 & \cdots & (c-1)r+1 \\ 2 & r+2 & 2r+2 & \cdots & (c-1)r+2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r & r+r & 2r+r & \cdots & (c-1)r+r \end{bmatrix} \quad (1)$$

where $r$=$ceil(y_m)/L$, $c$=$ceil(x_m)/L$, $r$ denotes the number of rows, $c$ denotes the number of columns and *ceil(.)* is a function that rounds the numeric value to the nearest and largest integer of the element. In a second step, a maze-image is processed following the procedure described in [8]. A camera with 17 Mp resolution and f/1.8 illumination aperture was used and each image was taken with natural light. Using bicubic interpolation, the image is resized to 500×500 pixels. Fig. 2(a) illustrates the binarized version with dimensions $(x_m, y_m)$=(3,3) and with irregular boundaries and obstacles depicted in black, where each pixel is represented by a logical **0** and each pixel of the white background is represented by a logical **1**. Further, the node matrix is generated and its graphical representation is shown in Fig. 2(b), where $L$=0.2 was used. Here, the top-left node is denoted as 1 and the bottom-right node as 225. To prevent collisions when the mobile robot passes near any

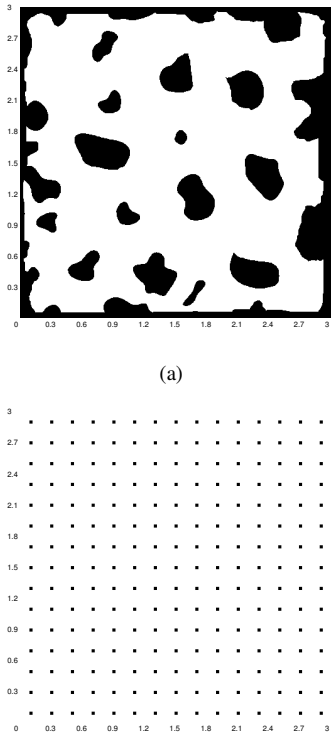


Fig. 2. (a) Binarized maze-image, (b) Graphic representation of the node matrix.

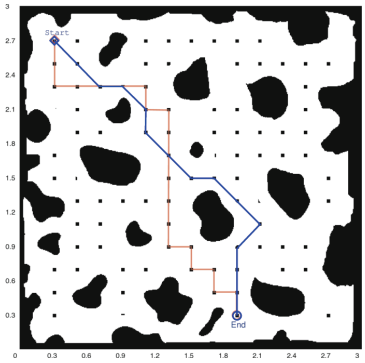


Fig. 3. 2D maze image of size 3m×3m containing collision-free nodes. The solid red line indicates the path computed via the methodology developed in [32]. The solid blue line shows the path found by applying the proposed methodology.

obstacle or maze boundary, each of these elements is expanded by a factor of

$$D = ceil\left(\frac{500L}{2min(x_m, y_m)}\right) \quad (2)$$

where *min(.)* returns the minimum value and *500* is the number of pixels of the scaled image. Consequently, the expanded maze-image and Fig. 2(b) are merged and as a result, all nodes overlapping with the expanded obstacles are absorbed and tagged as zero. The remaining nodes are collision-free.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 17 & 32 & 0 & 0 & 77 & 92 & 107 & 122 & 137 & 152 & 0 & 0 & 0 & 0 \\ 0 & 18 & 33 & 0 & 0 & 78 & 93 & 0 & 0 & 138 & 153 & 168 & 183 & 198 & 0 \\ 0 & 19 & 34 & 49 & 64 & 79 & 0 & 0 & 0 & 0 & 0 & 0 & 184 & 0 & 0 \\ 0 & 20 & 35 & 0 & 0 & 80 & 95 & 0 & 125 & 140 & 0 & 0 & 185 & 0 & 0 \\ 0 & 0 & 36 & 51 & 0 & 81 & 96 & 0 & 126 & 141 & 156 & 171 & 186 & 0 & 0 \\ 0 & 22 & 0 & 0 & 0 & 0 & 97 & 0 & 127 & 142 & 157 & 172 & 187 & 0 & 0 \\ 0 & 23 & 38 & 0 & 0 & 0 & 98 & 113 & 128 & 143 & 0 & 0 & 0 & 203 & 0 \\ 0 & 0 & 39 & 54 & 69 & 84 & 99 & 0 & 0 & 144 & 159 & 0 & 0 & 204 & 0 \\ 0 & 0 & 40 & 55 & 0 & 0 & 100 & 0 & 0 & 0 & 160 & 0 & 0 & 0 & 0 \\ 0 & 0 & 41 & 56 & 0 & 0 & 101 & 116 & 0 & 146 & 161 & 176 & 191 & 0 & 0 \\ 0 & 27 & 42 & 0 & 72 & 87 & 0 & 117 & 132 & 147 & 0 & 177 & 192 & 0 & 0 \\ 0 & 28 & 0 & 0 & 73 & 0 & 0 & 0 & 133 & 148 & 0 & 0 & 0 & 0 & 0 \\ 0 & 29 & 0 & 0 & 74 & 0 & 0 & 0 & 0 & 149 & 0 & 0 & 0 & 209 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3)$$

The outcome of this process is depicted in Fig. 3 (excluding the solid red and blue lines) and the reduced node matrix is given by Eq. (3). During the fusion of the expanded maze-image and the full node matrix, not only isolated nodes can be formed, such as node=209 in Eq. (3), but subsets of isolated nodes can also be generated. This negatively affects the formulation and solution of the subsequent system of equations by producing singular matrices. To mitigate this, the user selects a start point and an end point, which may or may not belong to a particular node subset. Thus, undirected graphs are built from Eq. (3) to identify the main subset and prevent singular matrices. Then, following the procedure in [8], the MATLAB instruction below is executed

$$B = nearest\,(graph\,(s, t), start, length(A)^2)$$

where *s* and *t* are vectors storing the start and target nodes, respectively, and both are used to build the undirected *graph(.)*; *nearest(.)* retrieves all nodes in the graph connected within a distance of $length(A)^2$ from start-node, and the nodes of the main subset are given by *B*. As a result, the node matrix is reduced once again. It should be noted that if the start node and the end node are not part of the same subset, it follows that the maze cannot be solved. In fact, this also indicates that there is not vertical, horizontal or diagonal connection among subset of nodes. At this point we highlight our first contribution, since the resistive grid will be built from the reduced node matrix of order *n=length(B)*, instead of the full node matrix given by Eq. (1) [32], saving computing resources. As a third step and following the square-diagonal cell of Fig. 1(b), between each pair of non-zero vertical nodes a resistor is connected as

$$net = [R_i \;\; A(r,c) \;\; A(r+1,c)] \quad (4)$$

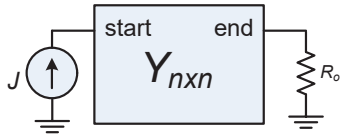


Fig. 4. Reduced resistive grid with *J* and $R_0$.

Next, between each pair of non-zero horizontal nodes a resistor is connected as

$$net = [R_i \;\; A(r,c) \;\; A(r,c+1)] \quad (5)$$

Subsequently, a resistor is connected between each pair of non-zero nodes in diagonal form as

$$net = [R_i \;\; A(r,c) \;\; A(r+1,c+1)] \quad (6)$$

and finally, a resistor is connected between each pair of non-zero nodes in an inverted diagonal fashion as

$$net = [R_i \;\; A(r,c+1) \;\; A(r+1,c)] \quad (7)$$

where *i*=1,2,3,$\cdots$. In each case, resistor are labeled in ascending order from top to bottom and left to right. Note that each node corresponds to a position coordinate of the robot and each branch represents a path to the next position. Besides, to ground the previously generated resistive grid, an output resistor $R_0$ is connected between the end-node and ground, and a current source *J* is connected between the start-node and ground, as illustrated in Fig. 4. The output resistor is also stored in the vector *net* and the current source is stored in the vector $J_n$. Observe that this procedure not only improves CPU-time and memory consumption in generating the reduced resistive grid compared to the method described in [32], where the full resistive grid is first generated and then, depending on the shape of obstacles, nodes and resistances are eliminated.

**Algorithm 1**. Path generation algorithm pseu-docode.

**Require:** start-node, Eq. (9) with reduced elements.

1: **while** start-node≠0 **do**
2: $p_1$ Stores all elements of Eq. (9) connected to start-node.
3: $p_2$ Find the element with the maximum current into $p_1$.
4: Path start-node.
5: start-node=Negative node of $p_2$.
6: **end while**

Therefore, valuable computing resources will be wasted generating resistive elements that will be removed when the expanded maze-image and node matrix are merged.

## III. Discovering the Shortest Path

Once the vector *net* is generated, nodal analysis is applied to build the system of equations

$$Y_{n\times n}V_n = J_n \tag{8}$$

where $Y_{n\times n}$ denotes the admittance matrix, $V_n$ the vector of nodal voltages and $J_n$ the vector of independent current sources. To fill $Y_{n\times n}$ and $J_n$, the stamp for resistors and independent current sources is applied, respectively. Of all solution methods reported in the literature [37]–[39], the LU factorization technique is used for solving Eq. (8). Thus, the node voltages are computed and stored in the vector $V_n$, considering a resistance value of 1Ω per resistor and *J*=1A. The branch currents and their corresponding node voltages are stored in the vector *net* as

$$net = [R_i \ + node \ - node \ + V_{n,R_i} \ - V_{n,R_i} \ i_{R_i}] \tag{9}$$

Examining each element of Eq. (9), the positive and negative voltage of some of them is the same and as consequence the branch current is zero. This indicates that the branch in question is a floating branch. Therefore, all elements with this characteristic are removed of the vector *net*. Moreover, if for any element of Eq. (9) $i_{R_i}$ is negative, then the positive and negative node along with their respective voltages are interchanged of position and thus, the current becomes positive. In this way, the vector *net* has only elements with non-zero positive current. Now let us examine the reduced vector *net* to determine the shortest path between the start and end nodes. Starting at the start node, all resistors connected to it are searched in Eq. (9) and grouped into a set. Within this set, the element with the highest current is selected and its negative node becomes the new start node. If multiple elements share the same current, the first one is chosen. This selection has no negative impact, as any branch leads to shortest path. The procedure is repeated until the zero node is found. Therefore, all previously visited nodes correspond to the collision-free shortest path and are stored in the vector *Path*. The Algorithm 1 shows the pseudocode for generating the *Path* vector previously described.

## IV. Numerical Tests

Three case studies validate the optimized collision-free path planning methodology. Notably for the numerical tests and consistent with the mobile robot dimensions, the maze sizes were chosen within the range $[\overbrace{1m}^{x_m^l} \times \overbrace{1m}^{y_m^l}, \overbrace{40m}^{x_m^u} \times \overbrace{40m}^{y_m^u}]$. The lower bound is determined by the physical dimensions of the robot used, while the upper bound is constrained by the maximum node matrix dimension achievable without loss of information integrity. This scaling range can be adjusted upward or downward based on the robot's dimensions. As a consequence, the proposed methodology can solve mazes of dimensions 100m×100m or larger, provided that the robot size is scaled proportionally to the maze size. The upper proportionality ratio is estimated as $p_r^u$=$\frac{max(x_m^u,y_m^u)}{L}$=$\frac{40m}{0.2m}$=200 and the lower proportionality ratio as $p_r^l$=$\frac{min(x_m^l,y_m^l)}{L}$=$\frac{1m}{0.2m}$=5. This implies that for a maze with $max(x_m^u, y_m^u)$=(100m,100m), the robot size must be at least *L*=$\frac{max(x_m^u,y_m^u)}{p_r^u}$=0.5m and the minimum maze size must satisfy $min(x_m^l, y_m^l)$=$Lp_r^l$=2.5m. Furthermore, if $min(x_m^l, y_m^l)$=(0.5m,0.5m), then *L*=$\frac{min(x_m^l,y_m^l)}{p_r^l}$=0.1m and $max(x_m^u, y_m^u)$=$Lp_r^u$=20m. On the other hand, it is not feasible to solve extremely large mazes when the robot size is too small, for the reasons previously discussed.

### A. Maze with size 3m×3m

The maze to be solved is depicted in Fig. 3 (excluding the solid red and blue lines). Selecting start-node=17 and end-node=149, the main node subset is obtained with *n*=89. Eq. (8) is then built, $V_n$ computed and the reduced vector *net* generated. Algorithm 1 yields the vector *Path* as

$$\begin{aligned} Path =&[17, 33, 49, 64, 80, 81, 97, 113, 128, \\ &144, 160, 146, 147, 148, 149] \end{aligned} \tag{10}$$

The solid blue line in Fig. 3 shows the shortest path found. This same maze-image was used in [8], [32]. Thus, whereas the solid red line in Fig. 3 shows the path found by applying the methodology

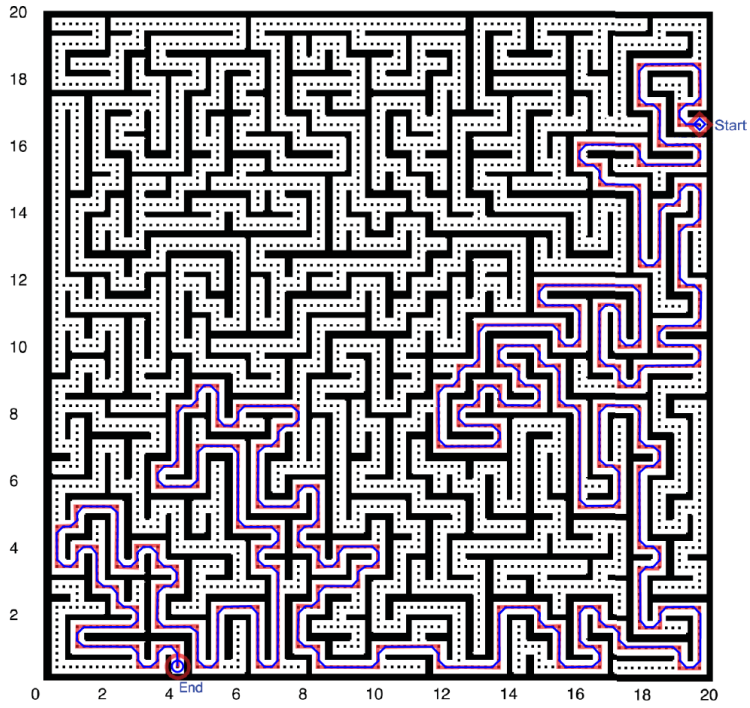


Fig. 5. Random 2D maze-image with collision-free nodes and size 20 m × 20 m.

developed in [32], whose *Path* vector length is 21, Eq. (10) has only 15 positions. This indicates a reduction not only in the travel time of the mobile robot to reach its destination, but also in energy consumption.

### B. Maze with size 20 m × 20 m

A random 2-D maze map can automatically be generated by executing the following Matlab instruction

*map* = *mapMaze* (*pW*,*wT*, '*MapSize*', [*xm ym*], '*MapResolution*', 1/*L* )

where *pW* is the passage width, *wT* is the wall thickness and *1/L* is the resolution. Using *pW*=2, *wT*=1, $x_m$=$y_m$=20 and *L*=0.2, the random 2-D maze map created is shown in Fig. 5 (excluding the dotted, solid red and blue lines). Next, the node matrix is generated (see Section II) and merged with the random 2-D maze map. Nodes overlapping maze walls are removed, yielding the maze shown in Fig. 5 (solid red and blue lines excluded). Because all hallways are connected, there are not isolated graphs, so the order of Eq. (8) is *n*=3266. With start-node=9818 end-node=2099, solving Eq. (8) and executing Algorithm 1 to the vector *net* given by Eq. (9), the vector *Path* is generated as

$$\begin{aligned} Path = & [9818, 9718, 9618, 9517, 9516, 9615, \\ & 9715, \cdots, 1798, 1797, 1896, 1996, \\ & 2097, 2098, 2099] \end{aligned} \quad (11)$$

As Eq. (11) has 747 positions, we cannot display them all. However, Eq. (11) shows the first and last seven positions. The solid blue line in Fig. 5 shows the shortest path found. On the other hand, the maze in Fig. 5 was also used in [32], where the Path vector length is 940 positions. In this case, the solid red line of Fig. 5 shows the Path already reported. Comparing the results of this work with those obtained in [8], [32], a reduction in the shortest path found, travel time and energy consumption can clearly be observed.

### C. Maze with size 40m×40m

For this last example, *pW*=3, *wT*=1 and $x_m$=$y_m$=40. A random 2D maze map is generated, as depicted in Fig. 6 (excluding the dotted, solid red and blue lines). The node matrix is then merged with the random map, resulting in Fig. 6 (solid red and blue lines excluded). The dotted lines indicate all valid paths and some hallways yield only a single dotted line. Since all hallways are fully connected, no isolated resistive sub-networks appear, so the order of Eq. (8) is *n*=18618. Selecting the start-node=39214, the end-node=4397, solving Eq. (8) and executing Algorithm 1 on Eq. (9) produces the vector *Path* as

$$\begin{aligned} Path = & [39214, 39213, 39212, 39211, 39210, \\ & 39209, \cdots, 3398, 3598, 3798, 3997, \\ & 4197, 4397] \end{aligned} \quad (12)$$

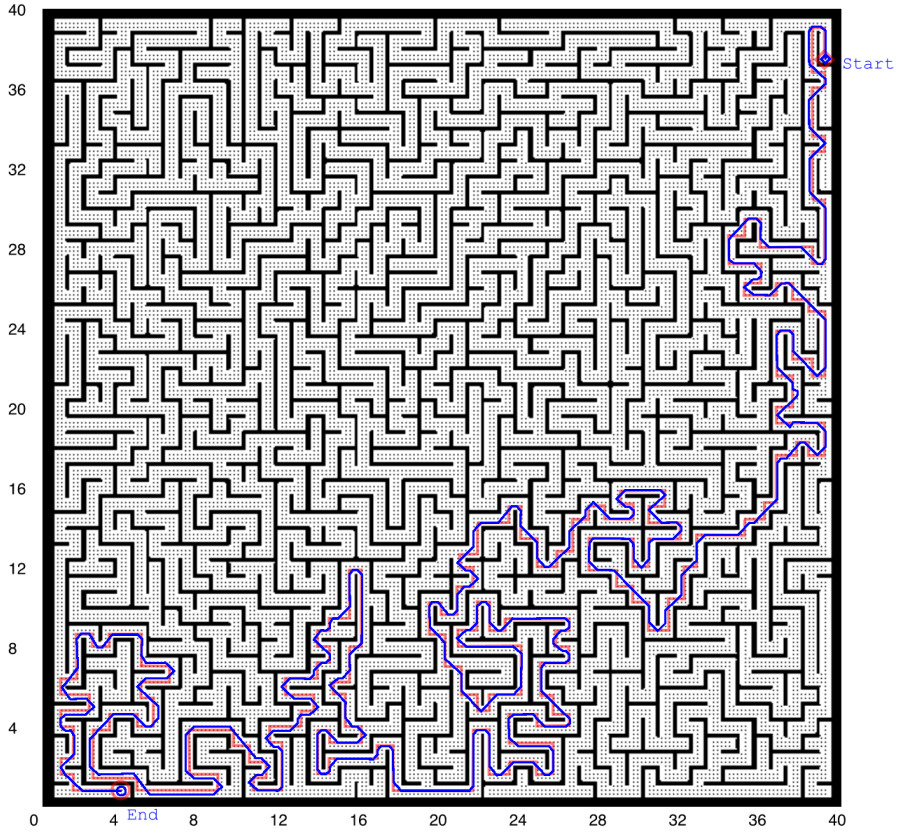


Fig. 6. Random 2D maze-image with collision-free nodes and size 40m×40m.

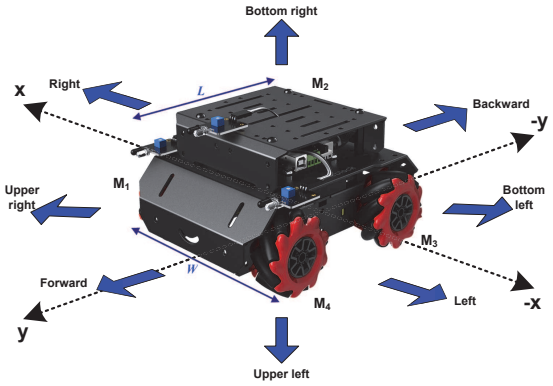


Fig. 7. MBot Mega mobile robot.

The total length of Eq. (12) is 1023 positions, which is smaller compared to the 1395 positions found in [32], and the solid blue line in Fig. 6 shows the shortest path found between the star- and end-nodes. Comparing the vector *Path* of this maze-map with those obtained in [32], a reduction is again observed in the shortest path found, travel time and energy consumption.

## V. Experimental Verification

Assuming the MBot Mega omnidirectional robot is on the Cartesian plane, as illustrated in Fig. 7, it can move horizontally (*x*-axis), vertically (*y*-axis) and diagonally (*x*-*y* axes). Therefore, eight directions of movement are possible: forward, backward, right, left, upper left, bottom right, upper right and bottom left [40]. The mobile robot has dimensions of *L*=0.2m in length and *W*=0.16m in width. After characterizing the robot [40], the velocity in the forward direction is set to $v_{fb}$=28m/min and the motors $M_1$ and $M_2$ are configured with velocity $v_{fb}$ and the motors $M_3$ and $M_4$ are configured with velocity $-v_{fb}$. For backward direction, the same velocity defined above is used but now $M_1$ and $M_2$ are configured with velocity $-v_{fb}$ and the motors $M_3$ and $M_4$ are configured with velocity $v_{fb}$. Additionally, the velocity in the right direction is set to $v_{rl}$=35m/min. Thus, the motors $M_1$ and $M_4$ are configured with velocity $-v_{rl}$, while each motor $M_2$ and $M_3$ is configured with velocity $v_{rl}$. For the left direction, both motors $M_1$ and $M_4$ are configured with velocity $v_{rl}$ and the motors $M_2$ and $M_3$ are configured with velocity $-v_{rl}$. Furthermore, the velocity in the diagonal

TABLE I
MECANUM WHEEL CONFIGURATION FOR EACH MOVEMENT ACCORDING TO FIG. 7

| | $\leftarrow$ | $\rightarrow$ | $\uparrow$ | $\downarrow$ | $\swarrow$ | $\nearrow$ | $\nwarrow$ | $\searrow$ |
|---|---|---|---|---|---|---|---|---|
| $M_1$ | $v_{fb}$ | $-v_{fb}$ | $-v_{rl}$ | $v_{rl}$ | $v_d$ | $-v_d$ | 0 | 0 |
| $M_2$ | $v_{fb}$ | $-v_{fb}$ | $v_{rl}$ | $-v_{rl}$ | 0 | 0 | $v_d$ | $-v_d$ |
| $M_3$ | $-v_{fb}$ | $v_{fb}$ | $v_{rl}$ | $-v_{rl}$ | $-v_d$ | $v_d$ | 0 | 0 |
| $M_4$ | $-v_{fb}$ | $v_{fb}$ | $-v_{rl}$ | $v_{rl}$ | 0 | 0 | $-v_d$ | $v_d$ |

direction is computed as

$$v_d = \sqrt{v_{fb}^2 + v_{rl}^2} = 44.82\text{m/min} \qquad (13)$$

Therefore, for upper left direction, $M_2$ and $M_4$ have zero velocity, whereas $M_1$ is configured with velocity $v_d$ and $M_3$ with velocity $-v_d$. For bottom right direction, the velocity of $M_1$ must be negative, for $M_3$ it must be positive and the motors $M_2$ and $M_4$ are off. Otherwise, for upper right direction, the motors $M_1$ and $M_3$ are off, whereas $M_2$ is configured with velocity $v_d$ and $M_4$ with velocity $-v_d$. Finally for bottom left direction, the motors $M_1$ and $M_3$ have zero velocity, the velocity of $M_2$ must be negative and for $M_4$ must be positive. Table I summarizes all the mecanum wheel configurations that should be used for each movement. All velocities specified in Table I remain active during a time interval defined as

$$t_{fb} = \frac{L}{|v_{fb}|} = \frac{0.2\text{m}}{28\text{m/min}} \approx 0.0071\text{min} \approx 0.428\text{s} \qquad (14)$$

$$t_{rl} = \frac{L}{|v_{rl}|} = \frac{0.2\text{m}}{35\text{m/min}} \approx 0.0057\text{min} \approx 0.342\text{s} \qquad (15)$$

$$t_d = \frac{\sqrt{2}L}{|v_d|} = \frac{\sqrt{2}0.2\text{m}}{44.82\text{m/min}} \approx 0.0063\text{min} \approx 0.378\text{s} \qquad (16)$$

Note that other active times and velocities can also be characterized to increase or decrease the movement velocity of the mobile robot. To control each movement of the MBot robot using the Arduino IDE platform, each entry of the vector *Path* must be converted into one of the following instructions: forward, backward, left, right, upper left, bottom right, upper right and bottom left, as decribed in Table I. To accomplish this task, Algorithm 1 described in [8] has been improved and now includes the four missing diagonal moves. At this point, we highlight our second contribution. Thus, the new instruction transformation procedure is described in the Algorithm 2. This procedure was programmed on the Arduino IDE platform

**Algorithm 2**. Instruction transformation algorithm pseudocode.

```
Require: vector Path
for i=1:length(Path)-1 do
    if Path(i)<Path(i+1) then
        if Path(i+1)-Path(i)=1 then
            Backward
        else if Path(i+1)-Path(i)=length(A) then
            Right
        else if Path(i+1)-Path(i)>length(A) then
            Bottom right
        else if Path(i+1)-Path(i)<length(A) then
            Upper right
        end if
    else
        if Path(i)-Path(i+1)=1 then
            Forward
        else if Path(i)-Path(i+1)=length(A) then
            Left
        else if Path(i)-Path(i+1)>length(A) then
            Bottom left
        else if Path(i)-Path(i+1)<length(A) then
            Upper left
        end if
    end if
end for
```

and from Eq. (10), the experimental evidence is depicted in Fig. 8, showing that the mobile robot follows the generated path and illustrated in Fig. 3 with solid blue line. Due to space and volume limitations on the complexity of the 2D maze construction, experimental evidences of the collision-free paths given by Eq. (11) and Eq. (12) are not shown. In any case, however, the vector *Path* should only be updated on reconfigurable hardware. Nevertheless, performance parameters were measured or predicted during the solution of each maze using the optimized methodology, as described in Table II. The CPU used has the following characteristics: Memory size of 4GB and microprocessor Intel core i5 at 1.6GHz, whereas the battery capacity of the MBot Mega robot is $I_b$=2600 mAh and voltage $V_b$=9 V [40]. It is important to stress that the current and voltage are linear up to approximately 230 minutes of continuous use. The travel time was predicted as

$$\text{Travel time} = t_{rl}M_h + t_{fb}M_v + t_dM_d \qquad (17)$$

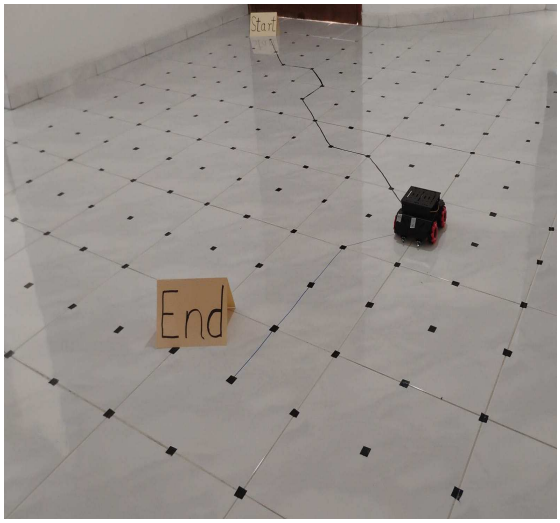


Fig. 8. Experimental validation of the found shortest path in Fig. 3 and denoted with solid blue line.

TABLE II
MEASURED AND PREDICTED PERFORMANCE PARAMETERS DURING THE MAZE SOLVING

| Shortest Path | CPU time (s) | Travel time (s) | Distance (m) | Battery usage (%) | Reference |
|---|---|---|---|---|---|
| Fig. 3 | 1.21 | 7.88 | 4 | 0.13 | [32] |
| | 0.83 | 5.43 | 3.46 | 0.1 | This work |
| Fig. 5 | 4.65 | 363.7 | 187.8 | 6.67 | [32] |
| | 2.23 | 287.91 | 165.68 | 5.4 | This work |
| Fig. 6 | 120.88 | 540.08 | 278.80 | 9.91 | [32] |
| | 113.12 | 393.82 | 246.81 | 7.49 | This work |

where $M_h$, $M_v$ and $M_d$ indicate the number of horizontal, vertical and diagonal movements in the vector *Path*, respectively. Further, the distance traveled is obtained as

$$\text{Distance} = L(M_h + M_v + \sqrt{2}M_d) \tag{18}$$

To predict the battery consumption, the energy consumption associated to each horizontal ($W_h$), vertical ($W_v$) and diagonal ($W_d$) movement is first predicted as

$$\begin{aligned} W_h &= \frac{(V_b - V_{bh})I_b L}{M} \\ W_v &= \frac{(V_b - V_{bv})I_b L}{M} \\ W_d &= \frac{(V_b - V_{bd})I_b \sqrt{2} L}{M} \end{aligned} \tag{19}$$

where $V_{bh}$, $V_{bv}$ and $V_{bd}$ are the final battery voltages measured after *M* horizontal, vertical or diagonal movements, respectively. Thus, $V_{bh}$ ≈8.91V, $V_{bv}$ ≈8.96V and $V_{bd}$ ≈8.95V were measured for *M*=20. Hence, the total energy consumption is approximated as

$$W_T = W_h M_h + W_v M_v + W_d M_d \tag{20}$$

and the battery usage at percentage is computed as

$$\text{Battery usage} = \frac{W_T}{V_b I_b} 100\% \tag{21}$$

Clearly Eq. (21) not only depends on $v_{fb}$, $v_{rl}$, $v_d$, $L$ and $M_{h,v,d}$, but also on the type of roughness of the surface where the robot moves, which is reflected in the measured voltages $V_{bh}$, $V_{bv}$ and $V_{bd}$. In this work, the floor used is moderately rough. Furthermore, Eq. (17), Eq. (18) and Eq. (21) were also used to predict the performance parameters of the methodology reported in [32], as written in Table II. Comparing each performance parameter, a better optimization of the robotic resources is observed to solve 2D mazes using the proposed methodology. At this point, we higlight our third contribution. Moreover, the proposed methodology is compared with those methodologies based on memristive grids, resistive grids and with those reported in [8], [32]. In these sense and according to Table III, most works do not take into account the dimensions of the mobile robot and for this reason, these methodologies can not solve mazes with different dimensions. Therefore, small and medium 2D mazes were used to test the methodologies reported in Table III, unlike the proposed methodology that can be scaled in the range mentioned before. But also, some maze solving methodologies are based on the use of the MNA method to build Eq. (8) and use commercial circuit simulators to compute currents and voltages of the resistive or memristive network. As a consequence, the computational complexity and cost increase [37], [38]. Another disadvantage found in the matrix-based methodologies described in Table III is that the generation of singular matrices is not avoided [39], unlike of [8], [32] and this work. Comparing [8] in terms of CPU time to solve the mazes, here it is slightly longer. This occurs due to the greater CPU time required to solve Eq. (8); specifically, higher-order admittance matrices demand increased CPU time. Comparing this same performance parameter with [32], the CPU time used is slightly shorter and this is because the order of Eq. (8) is smaller.

## VI. DISCUSSION

The optimized methodology presented in this paper, capable of solving mazes ranging from 1m×1m to 40m×40m with irregular shapes and obstacles, is relatively simple and well-suited for implementation on reconfigurable hardware. A key advantage of the proposed approach over those summarized in Table III is its incorporation of the

TABLE III
COMPARISON WITH OTHER MAZE-SOLVING METHODOLOGIES BASED ON MEMRISTIVE OR RESISTIVE GRIDS

| Reference | Basic algorithm | Maze size | Robot size | Scaling property | Experimental verification |
|---|---|---|---|---|---|
| [8] | Breadth-First Search | Very Large | Y | Y | Y |
| [30] | MNA | Medium | N | N | Y |
| [31] | MNA | Medium | N | N | N |
| [33] | KCL | Very Large | N | N | N |
| [34] | KCL | Very Large | N | N | N |
| [35] | MNA | Small | N | N | N |
| [36] | MNA | Small | N | N | N |
| [32] | NA | Very Large | Y | Y | Y |
| This work | NA | Very Large | Y | Y | Y |

Here Y means "Yes", N means "No", NA means "Nodal Analysis" and MNA means "Modified NA".

mobile robot physical dimensions during the automatic generation of the resistive grid. This ensures that the collision-free shortest path computed by applying NA and KCL is viable for the specific robot, and allows for scaling the 2D maze-image within the specified range, which itself is adjustable based on the robot size. An important aspect addressed in this work is the issue of isolated resistive subnetworks, which cause singular matrices during the formulation of the admittance matrix in Eq. (8). Experimental testing revealed a limitation related to wheel traction: if one or more wheels lose contact with the surface or skid, positional and orientation errors accumulate along the path, reducing final accuracy. This can be mitigated by ensuring the operating surface is moderately rough and develop a self-localization system within the environment. This work makes three main contributions

1) An improved admittance matrix formulation derived from a reduced resistive network.
2) An enhancement to Algorithm 2 to incorporate the eight possible robot movements illustrated in Fig. 7.
3) The measurement and prediction of performance parameters described in Table III. These parameters are influenced not only by surface roughness but also by the path nature. The shortest path may involve more directional changes (vertical, horizontal, diagonal) compared to a smoother, more optimal path, potentially leading to slightly higher travel time and energy consumption.

Furthermore, the methodology demonstrates greater computational efficiency in mazes densely populated with irregular obstacles. As shown in Table III, the reduced order of the admittance matrix decreases the CPU time required to find the shortest path. Consequently, this optimized navigation methodology is highly applicable in smart applications, agriculture, warehouse logistics, household management and so on.

Regarding limitations and issues that will be addressed in future work, the developed methodology suffers from the following.

1) Currently only static scenarios are supported. One solution to this drawback is to take images of the dynamic maze at predetermined time intervals. If there are no changes in the maze compared to the previous image, the robot continues its journey; if there is a change, the methodology is applied again to find the new path from the node where the robot stopped. Although this solution solves the problem, an increase in computational cost is anticipated.
2) The methodology can be applied to multiple robots, since the start and end nodes are defined for each one. The drawback is that when the paths are generated, several robots may use parts of the same path simultaneously, causing collision among them.
3) The maze-solving methodology developed is general. However, if a differential robot is used instead of an omnidirectional, the former behavioral model along with its physical limitations must be considered in the controller design to achieve smooth breaks, as described in [41].
4) To reduce position and orientation errors that accumulate along the robot's path, a self-localization system within the environment must be developed.

## VII. CONCLUSIONS

This paper has introduced an optimized methodology for mobile robot navigation through complex 2D mazes. As summarized in Table III, the proposed approach enables a robot to solve arbitrarily shaped mazes with irregular obstacles within a

scalable size range, distinguishing it from full-custom solutions prevalent in the literature. The collision-free path planning process is twofold. First, a safety margin around obstacles and maze boundaries is established using Eq. (2), which removes nodes from the navigable space. Second, a reduced-order resistive grid is generated using Eq. (4), Eq. (5), Eq. 6 and Eq. (7), explicitly accounting for both the maze dimensions and the physical size of the robot. To ensure numerical stability and prevent singular matrices, the method constructs undirected graphs from the node matrix and selectively retains the subnetwork containing the start and end nodes. The final collision-free shortest path is generated by applying Algorithm 1. A key practical advantage is the methodology compatibility with reconfigurable hardware. This means that once the control conditions of the mobile robot are established on any platform such as DSPs, FPGAs, FPAAs or microcontrollers, the vector *Path* only needs to be updated according to the new maze analyzed. These contributions collectively form a robust and efficient framework suitable for real-world applications, including those previously mentioned in smart environments, agriculture and logistics. Moreover, advantages, disadvantages, issues and limitations of the proposed maze-solving methodology were also described, and some research directions were provided to improve the methodology in future work.

## Conflict of interest

The authors declare no conflict of interest.

## Author Contributions

C.S.L. conducted the research, methodology and manuscript writing; C.M.M. was responsible for formal analysis and writing original draft preparation; J.A.H. and C.H.M. were responsible for experimentation and validation; D.T.M. contributed to supervision, conceptualization and software development; All authors have read and approved the final version.

## References


[1] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[2] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. 2nd ed. Cambridge, MA, USA: MIT Press, 2011.

[3] X. Liu and D. Gong, "A comparative study of a-star algorithms for search and rescue in perfect maze," in *2011 International Conference on Electric Information and Control Engineering*, 2011, pp. 24–27.

[4] X. Zang, S. Iqbal, Y. Zhu, X. Liu, and J. Zhao, "Applications of chaotic dynamics in robotics," *International Journal of Advanced Robotic Systems*, vol. 13, no. 2, p. 60, 2016.

[5] J. Gross, J. Yellen, and M. Anderson, *Graph theory and its applications*. Boca Raton USA: Taylor & Francis Group, 2019.

[6] N. Kumar and S. Kaur, "A review of various maze solving algorithms based on graph theory," *International Journal for Scientific Research and Development*, vol. 6, no. 12, 2019.

[7] A. Sadik, M. Dhali, H. Farid, T. Rashid, and A. Syeed, "A comprehensive and comparative study of maze-solving techniques by implementing graph theory," in *2010 International Conference on Artificial Intelligence and Computational Intelligence*, vol. 1, 2010, pp. 52–56.

[8] L. Avila-Sánchez, C. Sánchez-López, R. Ochoa-Montiel, F. Montalvo-Galicia, L. Sánchez-Gaspariano, C. Hernández-Mejía, and H. González-Hernández, "Maze solving mobile robot based on image processing and graph theory," *Technologies*, pp. 1–10, 2023.

[9] S. Alamri, H. Alamri, W. Alshehri, S. Alshehri, A. Alaklabi, and T. Alhmiedat, "An autonomous maze-solving robotic system based on an enhanced wall follower approach," *Machines*, vol. 11, no. 2, 2023.

[10] S. Alamri, S. Alshehri, W. Alshehri, H. Alamri, A. Alaklabi, and T. Alhmiedat, "Autonomous maze solving robotics: Algorithms and systems," *International Journal of Mechanical Engineering and Robotics Research*, vol. 10, no. 12, pp. 668–675, 2021.

[11] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Syst. Appl.*, vol. 227, no. C, jul 2023.

[12] R. Niemczyk and S. Zawiślak, "Review of maze solving algorithms for 2d maze and their visualisation," in *Engineer of the XXI Century*, S. Zawiślak and J. Rysiński, Eds. Cham: Springer International Publishing, 2020, pp. 239–252.

[13] H. Dei-Jeung, P. Jong-Hun, H. Uk-Youl, and K. Hakil, "Path planning and navigation for autonomous mobile robot," in *IEEE 2002 28th Annual Conference of the Industrial Electronics Society. IECON 02*, vol. 2, 2002, pp. 1538–1542.

[14] B. Rahnama, M. Özdemir, Y. Kiran, and A. Eli, "Design and implementation of a novel weighted shortest path algorithm for maze solving robots," in *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*, 2013, pp. 328–332.

[15] R. Covaci, G. Harja, and I. Nascu, "Autonomous maze solving robot," in *2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR)*, 2020, pp. 1–4.

[16] K. C. et al., "Shortest distance maze solving robot," in *2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS)*, 2020, pp. 283–286.

[17] Y. Pame, V. Kottawar, and Y. Mahajan, "A novel approach to maze solving algorithm," in *2023 International Conference on Emerging Smart Computing and Informatics (ESCI)*, 2023, pp. 1–6.

[18] U. Orozco-Rosas, K. Picos, and O. Montiel, "Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots," *IEEE Access*, vol. 7, pp. 156 787–156 803, 2019.

[19] U. Orozco-Rosas, O. Montiel, and R. Seplveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Applied Soft Computing*, vol. 77, pp. 236–251, 2019.

[20] B. Rahnama, A. Eli, and S. Metani, "An image processing approach to solve labyrinth discovery robotics problem," in *2012 IEEE 36th Annual Computer Software and Applications Conference Workshops*, 2012, pp. 631–636.

[21] H. Joshi and J. Shinde, "An image based path planning and motion planning for autonomous robot," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 4, pp. 4844–4847, 2014.

[22] M. Aqel, A. Issa, M. Khdair, M. ElHabbash, M. AbuBaker, and M. Massoud, "Intelligent maze solving robot based

on image processing and graph theory algorithms," in *2017 International Conference on Promising Electronic Technologies (ICPET)*, 2017, pp. 48–53.

[23] Y. Murata and Y. Mitani, "A fast and shorter path finding method for maze images by image processing techniques and graph theory," *Journal of Image and Graphics*, pp. 89–93, 2014.

[24] O. Kathe, V. Turkar, A. Jagtap, and G. Gidaye, "Maze solving robot using image processing," in *2015 IEEE Bombay Section Symposium (IBSS)*, 2015, pp. 1–5.

[25] A. Ambeskar, V. Turkar, A. Bondre, and H. Gosavi, "Path finding robot using image processing," in *2016 International Conference on Inventive Computation Technologies (ICICT)*, vol. 3, 2016, pp. 1–6.

[26] A. Ambeskar, A. Bondre, V. Turkar, and H. Gosavi, "Intuitive solution for robot maze problem using image processing," in *2019 IEEE Bombay Section Signature Conference (IBSSC)*, 2019, pp. 1–6.

[27] M. Muhammed, E. Flaieh, and A. Humaidi, "Embedded system design of path planning for planar manipulator based on chaos A* algorithm with known obstacle environment," *J. Eng. Sci. Technol*, vol. 17, no. 6, pp. 4047–4064, 2022.

[28] M. Muhammed, A. Humaidi, and E. Flaieh, "A comparison study and real-time implementation of path planning of two arm planar manipulator based on graph search algorithms in obstacle environment," *ICIC Express Letters*, vol. 17, no. 1, pp. 61–72, 2023.

[29] K. Althöfer, D. Fraser, and G. Bugmann, "Rapid path planning for robotic manipulators using an emulated resistive grid," *Electronics Letters*, vol. 31, pp. 1960–1961(1), October 1995.

[30] C. Hernández-Mejía, D. Torres-Muñoz, E. Inzunza-González, and C. Sánchez-López, "Exploring robotical implementation for planning of collision-free logistical paths using rgppm algorithm," *Iran J Sci Technol Trans Electr Eng*, vol. 47, pp. 221–231, 2023.

[31] C. Hernández-Mejía, H. Vázquez-Leal, A. Sánchez-González, and A. Corona-Avelizapa, "A novel and reduced cpu time modeling and simulation methodology for path planning based on resistive grids," *Arabian Journal for Science and Engineering*, vol. 44, pp. 2321–2333, 2019.

[32] C. Sánchez-López, "Maze solving mobile robot based on image processing and standard nodal analysis," *Franklin Open*, vol. 12, p. 100357, 2025.

[33] P. Dopazo, C. de Benito, O. Camps, S. G. Stavrinides, and R. Picos, "Gerard: General rapid resolution of digital mazes using a memristor emulator," *Physics*, vol. 4, no. 1, pp. 1–11, 2022.

[34] Y. Pershin and M. D. Ventra, "Solving mazes with memristors: A massively parallel approach," *Phys. Rev. E*, vol. 84, p. 046703, Oct 2011.

[35] A. Sarmiento-Reyes and Y. Rodríguez-Velásquez, "Maze-solving with a memristive grid of charge-controlled memristors," in *2018 IEEE 9th Latin American Symposium on Circuits & Systems (LASCAS)*, 2018, pp. 1–4.

[36] Z. Ye, S. Wu, and T. Prodromakis, "Computing shortest paths in 2d and 3d memristive networks," in *Handbook of Memristor Networks*, 2013.

[37] C. Sánchez-López, "Pathological equivalents of fully-differential active devices for symbolic nodal analysis," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 3, pp. 603–615, 2013.

[38] C. Sánchez-López, B. Cante-Michcol, F. Morales-López, and M. Carrasco-Aguilar, "Pathological equivalents of cms and vms with multi-outputs," *Analog Integr Circ Sig Process*, vol. 75, no. 1, pp. 75–83, 2013. [Online]. Available: https://doi.org/10.1007/s10470-012-0003-9

[39] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*. New York: Springer New York, 1983.

[40] M. MBot-Mega, "Available online: https://www.makeblock.com/pages/mbot-mega-smart-remote-control-robot (accessed on 27 september 2025)."

[41] C. Sánchez-López, R. Ochoa-Montiel, and F. Montalvo-Galicia, "A novel collision-free navigation method for autonomous chaotic mobile robots," *Chaos, Solitons & Fractals*, vol. 186, p. 115303, 2024.