Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 1, February 2026, pp. 680~692
ISSN: 2302-9285, DOI: 10.11591/eei.v15i1.7286

Harmonic path planning using Quarter-Sweep Boosted AOR iterative method

Sumiati Suparmin, Azali Saudi
Faculty of Computing and Informatics, Universiti Malaysia Sabah, Kota Kinabalu, Malaysia

Article Info

Article history:

Received Jul 30, 2023
Revised Jul 22, 2025
Accepted Dec 6, 2025

Keywords:

Half Sweep Boosted
Harmonic potential
Iterative methods
Laplace equation
Path planning
Quarter Sweep Boosted AOR

ABSTRACT

This paper presents the findings of a study that examined the effectiveness of the application of Quarter-Sweep Boosted AOR with the 9-Point Laplacian operator using the families of relaxation methods for computing the solutions of Laplace's equation to obtain the harmonic potentials+ This work is a continuation from the past study that applied the standard 5-Point Laplacian to solve path planning issue that a mobile robot faces when working in indoor environment. The robot can navigate from a given starting position to a goal position by following the safest path, ensuring it avoids any obstacles and minimizes the risk of collisions. By utilizing Laplace's equation and computing the distribution of potential values in the simulated environments, the robot can determine the safest path that avoids obstacles present in the environment. This method ensures that the robot moves along a path where the potential for collisions is minimized. The findings confirm that Quarter-Sweep Boosted AOR (QSBAOR) outperforms Half-Sweep Boosted AOR (HSBAOR) and Full-Sweep Boosted AOR (FSBAOR). QSBAOR and HSBAOR show 75% and 50% reduction respectively, compared to FSBAOR in terms of computational complexity.

Corresponding Author:

Azali Saudi
Faculty of Computing and Informatics, Universiti Malaysia Sabah
Kota Kinabalu, Sabah, Malaysia
Email: azali@ums.edu.my

1. INTRODUCTION

Mobile robots are widely used in various industrial fields where they are exposed to hazardous conditions such as space research, nuclear industry, and mining industry. In order to find a safe route in a dangerous environment, mobile robots are the most suitable and safe to use [1]. One of the difficult issues with moving robots is the problem of route planning [2]. Currently, research for mobile robot path planning is increasingly becoming a hot topic among researchers [3], [4].

A robot is an automated machine that can respond to the environment. To achieve such automated properties, it is necessary to use techniques from signal processing, control theory, and artificial intelligence [5]. This technique is accompanied by mechanics, detectors, and robot actuators. Therefore, designing a robot requires a deep understanding of its interface to the physical world. Among the key requirements for building a real automated robot is the ability to plan a route efficiently from the starting point to a specified destination point without colliding with objects or getting stuck in the path with obstacles it passes through.

Path planning is a vital component in robotics as it plays a crucial role in enabling robots to navigate from a designated starting point to a desired goal location. Especially in the ability to plan routes to allow robots to find a smooth path towards their destination. Algorithms for finding the safe path are important not only in robotics but also in network routing and video games. Route planning requires a map for the purpose

of allowing the robot to know its location in the environment to avoid getting stuck by any obstacles or walls while in motion.

Currently, the most common map is a grid map. In a grid map, the environment is divided into a number of uniformly sized squares such as 1×1 cm. Each square can represent an empty space or an obstacle object. The disadvantage of grid maps is that they require large memory and typically require a relatively long computation time. Other than that, another solution is to use topology maps. Topology maps encode the whole room as corners and edges to show the relationship between points. Each application may require a different solution that incorporates different types of maps. There is also a combination of discrete and continuous representations. For example, a road map for a GPS system is stored as a topology map that stores the GPS coordinates of each place point.

In general, path planning strategies for navigation are divided into two categories: local methods, which work in response to input sensors, and global methods, which involve the creation and execution of action plans [6]. The challenge of keeping the robot in a collision-free state is solved using local planning algorithms. These strategies are referred to as local since they only assess the robot's immediate environment when determining how it should react. Only immediate sensory data is dealt with by the local technique. As a result, it operates extremely quickly, allowing it to react quickly to changes in the environment. However, this speed comes at the expense of completeness. In general, a local method follows its specific functions greedily. Therefore, it may become stuck in local minima of the function and fail to reach its final destination [7], [8].

The global technique, on the other hand, solves the problem by creating a full representation of the environment. The environment model is a three-dimensional space with several obstacles of various shapes, as well as inner and outer borders. When global planners construct a plan, they take into account the entire environment, which demands a substantial amount of processing power [9]. Global path planning, in general, is computationally inflexible. The cost of computing the exact solution to a path planning problem grows exponentially as the environment grows larger [10]. The real world's dynamic nature is always in motion. Thus, the availability of time for a robot to make effective planning is greatly constrained in this dynamic setting. The researchers are challenged by the complexity of the computational demands of such planning challenges. To make matters worse, data and knowledge about the environment are gathered from noisy sensors, making them inaccurate and incomplete. As a result, in order to deal with incomplete and perhaps erroneous representations of the world, path planning algorithms must be reliable and efficient.

2. METHOD

The established a global method for path finding that used the harmonic potentials to construct a smooth and collision-free path [7]. Harmonic potentials are obtained by solving Laplace's equation, which is defined as (1):

$$\nabla^2 \emptyset = \sum_{i=1}^{n} \frac{\partial^2 \emptyset}{\partial x_i^2} = 0 \quad (1)$$

where the dimension is ∇n and the i-th Cartesian coordinate is x_i. Harmonic potentials have been established globally throughout the entire region. The harmonic solutions to Laplace's (1) are then used to find the path lines from the start point to the goal point. Obstacles are regarded as current sources, while the goal is regarded as a sink. Dirichlet boundary conditions are used in this case. The path to the goal point can be discovered by executing a path search on the harmonic potentials using the gradient descent method [11].

This study applies the above paradigm to solve pathfinding problems, employing the analogies of temperature and heat flow for the potential and path line, respectively, to characterize the solutions of Laplace's equation. Numerical methods are used to solve Laplace's equation and acquire the harmonic potential (temperature values) for each node. The obtained temperature values are then used in the path-finding process by descending from a start point (high temperature) to the goal point (lowest temperature).

The fundamental concept of the numerical methods is to represent the problem, i.e., Laplace's (1), in the form of a linear system.

$$Ax = b \quad (2)$$

where A is a coefficient matrix, x is a given vector, and b denotes the unknown vector to be determined. Although problem (2) can be solved using a direct method, the more efficient iterative methods are used to compute the solutions. This is because its application in path-finding problems often results in large linear systems with sparse coefficient matrices [12], [13]. Iterative methods are mathematical techniques that

produce a series of improving approximations [14]. These techniques are efficient in terms of memory storage and computation.

Most of the previous development approaches of the iterative methods are still oriented on the 5-point Laplacian operator. Recently, the 9-point Laplacian operator has been used successfully to solve several types of linear systems [15]–[17]. Traditionally, these iterative methods rely on the use of the full-sweep (FS) iteration that processes the computational nodes on the regular fine grid. The introduction of half-sweep (HS) on the rotated grid by Fauzi and Sulaiman [18] and quarter-sweep (QS) iterations that are operating on the coarse grid by Alibubin *et al.* [19], respectively, drastically improve the computational execution time. In addition, relaxation parameters were applied to further speed up the computation by utilizing SOR [20], accelerated over-relaxation (AOR) [21], and two-parameter over-relaxation (TOR) iterative methods [22].

Most of the existing solutions to the problem (1) are based on the 5-point iterative scheme. It is known that the iterative scheme based on 9-point had provided encouraging performance as reported in the previous works [16], [17], [23], [24]. Therefore, in this work, we applied the 9-point Laplacian and called these variants as a family of Boosted iterative schemes.

The approximation of the 2D Laplacian (1) based on the 9-point Laplacian is given as (3):

$$\nabla^2 f\ (x,y) = \frac{1}{6h^2}\big((4u(x-h,y) + 4u\ (x+h,y) + 4u\ (x,y-h) + 4u\ (x,y+h) + u\ (x-h,y-h)\ + u\ (x+h,y-h) + u(x-h,y+h) + u(x+h,y+h) - 20u(x,y)\big) \quad (3)$$

By rotating the x-y axis clockwise 45°, the rotated 9-point Laplacian approximation can be written as (4):

$$\nabla^2 f\ (x,y) = \frac{1}{12h^2}\big((4u(x-h,y-h) + 4u\ (x+h,y-h) + 4u\ (x-h,y+h) + 4u(x+h,y+h) + u\ (x-2h,y)\ + u\ (x+2h,y) + u(x,y-2h) + u(x,y+2h) - 20u(x,y)\big) \quad (4)$$

Moreover, by considering the points at grids size 2h, the 9-point approximation can be expressed as (5):

$$\nabla^2 f\ (x,y) = \frac{1}{24h^2}\big((4u(x-2h,y) + 4u\ (x+2h,y) + 4u\ (x,y-2h) + 4u\ (x,y+2h) + u\ (x-2h,y-2h)\ + u\ (x+2h,y-2h) + u(x-2h,y+2h) + u(x+2h,y+2h) - 20u(x,y)\big) \quad (5)$$

Figure 1(a) displays the computational molecule of the FS Boosted method, while Figures 1(b) and (c) show the computational molecules of the HS Boosted and QS Boosted methods, respectively. In all three figures, the central node has a coefficient of −20, as stated in (4) to (6). For the FS Boosted and QS Boosted methods, as shown in Figures 1(a) and (c), the four vertical and horizontal neighbouring nodes have coefficients of 4, whereas the four diagonal nodes have coefficients of 1. In contrast, for the rotated grid used in the HS Boosted method, shown in Figure 1(b), the left, right, top, and bottom nodes have coefficients of 1, while all diagonal nodes have coefficients of 4. The distance between the central node and its vertical and horizontal neighbours is h for the FS Boosted method and 2h for the QS Boosted method, as shown in Figures 1(a) and (c), respectively. For the HS Boosted method, shown in Figure 1(b), the distance between the central node and each of its four diagonal neighbours is $\sqrt{2}$h.

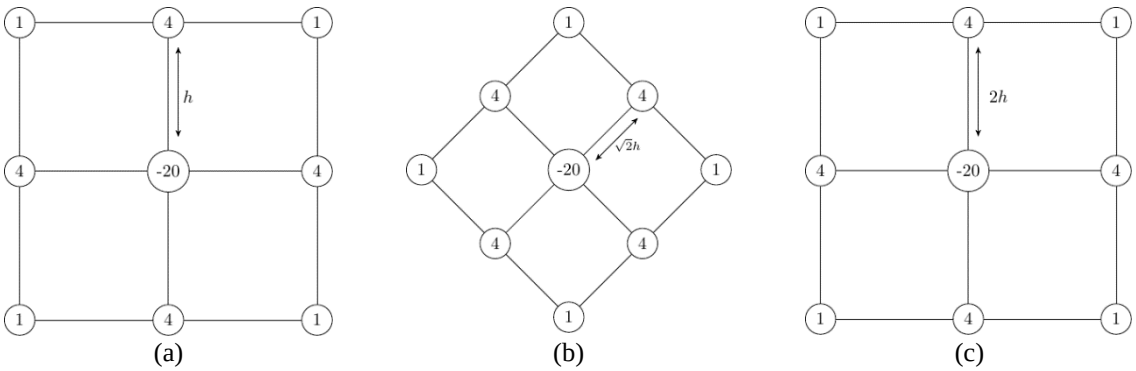

Figure 1. The 9-point Laplacian approximation computational molecules for; (a) FS, (b) HS, and (c) QS boosted operations

Figures 2(a) to (c) illustrate portions of the computational grids for the FS, HS, and QS Boosted methods, respectively. In all three figures, the central node has a coefficient of −20, as specified in (4) to (6). In the FS Boosted method, shown in Figure 2(a), the four vertical and horizontal neighbouring nodes have coefficients of 4, while the four diagonal nodes have coefficients of 1. In contrast, the HS Boosted method employs a rotated grid, as shown in Figure 2(b), in which the vertical and horizontal nodes have coefficients of 1 and all diagonal nodes have coefficients of 4. For the QS Boosted method, as shown in Figure 2(c), the coefficients of the central node and its neighbouring nodes are identical to those of the FS Boosted method, however, the distance between adjacent nodes is doubled (i.e., 2 units).

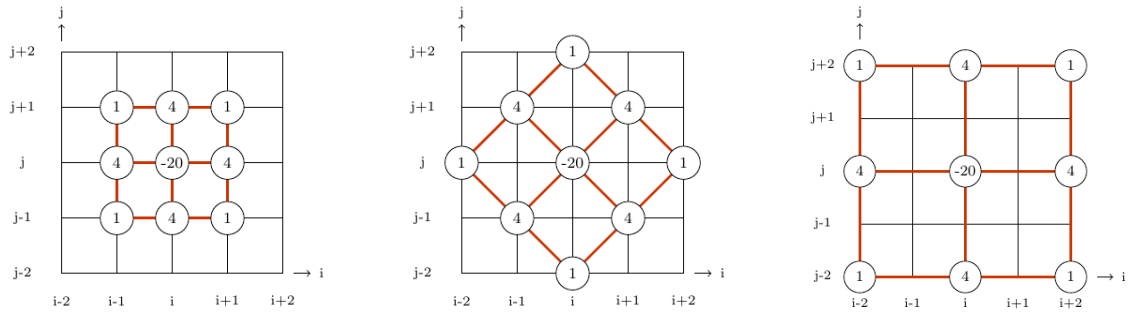

Figure 2. The 9-point Laplacian computational grids at (i; j) for; (a) FS, (b) HS, and (c) QS Boosted cases, respectively

The computational layer for the FS Boosted method is shown in Figure 3, where all nodes participate in the computation. Each node is evaluated using the 9-point Laplacian operator, as defined in (4). As highlighted in the figure, the central node has a coefficient of −20, the neighbouring vertical and horizontal nodes have coefficients of 4, and the diagonal neighbouring nodes have coefficients of 1.

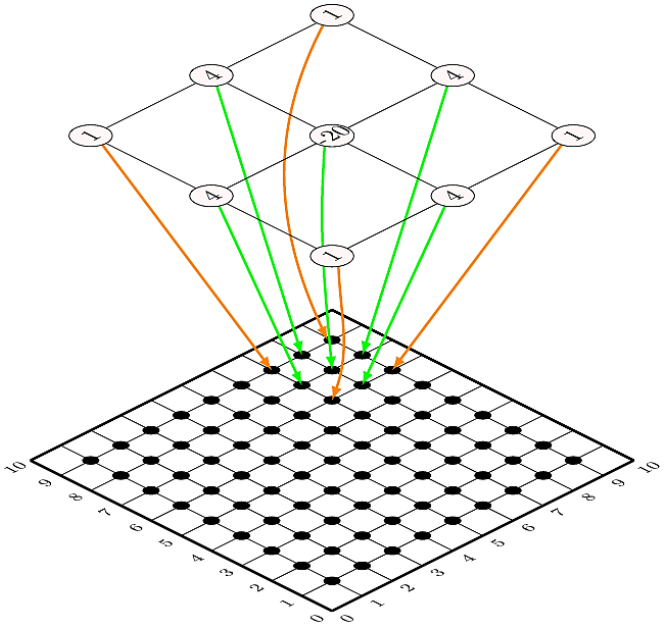

Figure 3. The boosted iterative methods with 9-point Laplacian based on FS iteration considers all nodes

The computational layer for the HS Boosted method is shown in Figure 4, where black nodes are evaluated using the 9-point Laplacian operator, as defined in (5). As illustrated in the figure, the HS Boosted method divides the grid nodes into black and white sets, as shown in the bottom layer. Initially, only the

black nodes are evaluated using a rotated grid, as depicted in the top layer, where a rotated 9-point Laplacian operator is applied. In this operator, the central node has a coefficient of −20, the neighbouring diagonal nodes have coefficients of 4, and the vertical and horizontal nodes have coefficients of 1. The computation of the black nodes proceeds iteratively until the termination criterion is satisfied. Upon convergence, the white nodes, shown in the middle layer, are subsequently computed using a direct method.

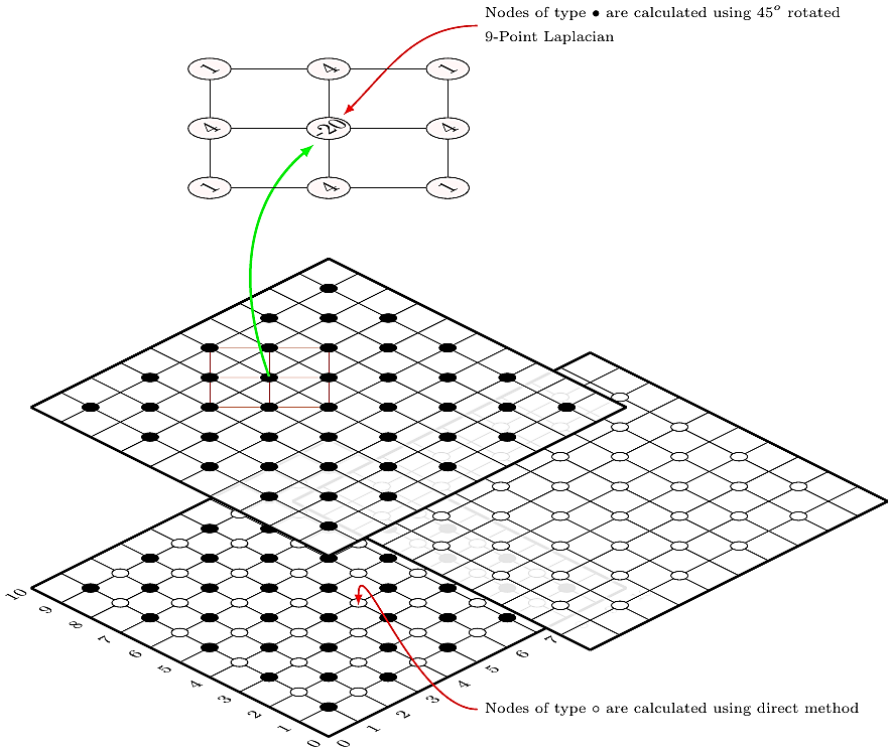

Figure 4. The boosted iterative methods with 9-point Laplacian based on HS iteration considers only half the total nodes

Figure 5 illustrates the computational layer of the QS Boosted method. As shown in the figure, the QS Boosted method partitions the grid nodes into black nodes, white square nodes, and white circle nodes. The black nodes, shown in the bottom layer, constitute one quarter of the total nodes and are evaluated using the 9-point Laplacian operator defined in (6). Initially, only the black nodes are computed, as depicted in the top layer, where a 9-point Laplacian operator with a node spacing of 2 units is applied. In this operator, the coefficients of the central node and its neighbouring nodes are identical to those used in the FS Boosted method. The computation of the black nodes proceeds iteratively until the termination criterion is satisfied. After convergence, the white square nodes are computed, followed by the white circle nodes, as illustrated in the second and third layers of the figure, using a direct method.

By using $u_{i,j}$ to approximate $f(x,y)$ and applying the 9-point approximations (2) to (4), the FS, HS, and QS. The approximation equations for problem (1) can be restated or expressed in a different form as (5) to (7):

$$4\left(u_{i-1,j}+u_{i+1,j}+u_{i,j-1}+u_{i,j+1}\right)+u_{i-1,j-1}+u_{i+1,j-1}+\ u_{i-1,j+1}+u_{i+1,j+1}-20u_{i,j}=0 \quad (5)$$

$$4\left(u_{i-1,j-1}+u_{i+1,j-1}+u_{i-1,j+1}+u_{i+1,j+1}\right)+u_{i-2,j}+u_{i+2,j}+\ u_{i,j-2}+u_{i,j+2}-20u_{i,j}=0 \quad (6)$$

$$4\left(u_{i-2,j}+u_{i+2,j}+u_{i,j-2}+u_{i,j+2}\right)+u_{i-2,j-2}+u_{i+2,j-2}+\ u_{i-2,j+2}+u_{i+2,j+2}-20u_{i,j}=0 \quad (7)$$

According to the finite difference (5) to (7), the iterative strategies for the FS, HS, and QS instances are defined as (8) to (10):

$$u_{i,j}^{(k+1)}=\frac{1}{5}\left(u_{i-1,j}^{(k+1)}+u_{i+1,j}^{(k)}+u_{i,j-1}^{(k+1)}+u_{i,j+1}^{(k)}\right)+\frac{1}{20}\left(u_{i-1,j-1}^{(k+1)}+u_{i+1,j-1}^{(k+1)}+u_{i-1,j+1}^{(k)}+u_{i+1,j+1}^{(k)}\right) \quad (8)$$

$$u_{i,j}^{(k+1)} = \frac{1}{5}\left(u_{i-1,j-1}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + \frac{1}{20}\left(u_{i-2,j}^{(k+1)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k+1)} + u_{i,j+2}^{(k)}\right) \quad (9)$$

$$u_{i,j}^{(k+1)} = \frac{1}{5}\left(u_{i-2,j}^{(k+1)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k+1)} + u_{i,j+2}^{(k)}\right) + \frac{1}{20}\left(u_{i-2,j-2}^{(k+1)} + u_{i+2,j-2}^{(k+1)} + u_{i-2,j+2}^{(k)} + u_{i+2,j+2}^{(k)}\right) \quad (10)$$

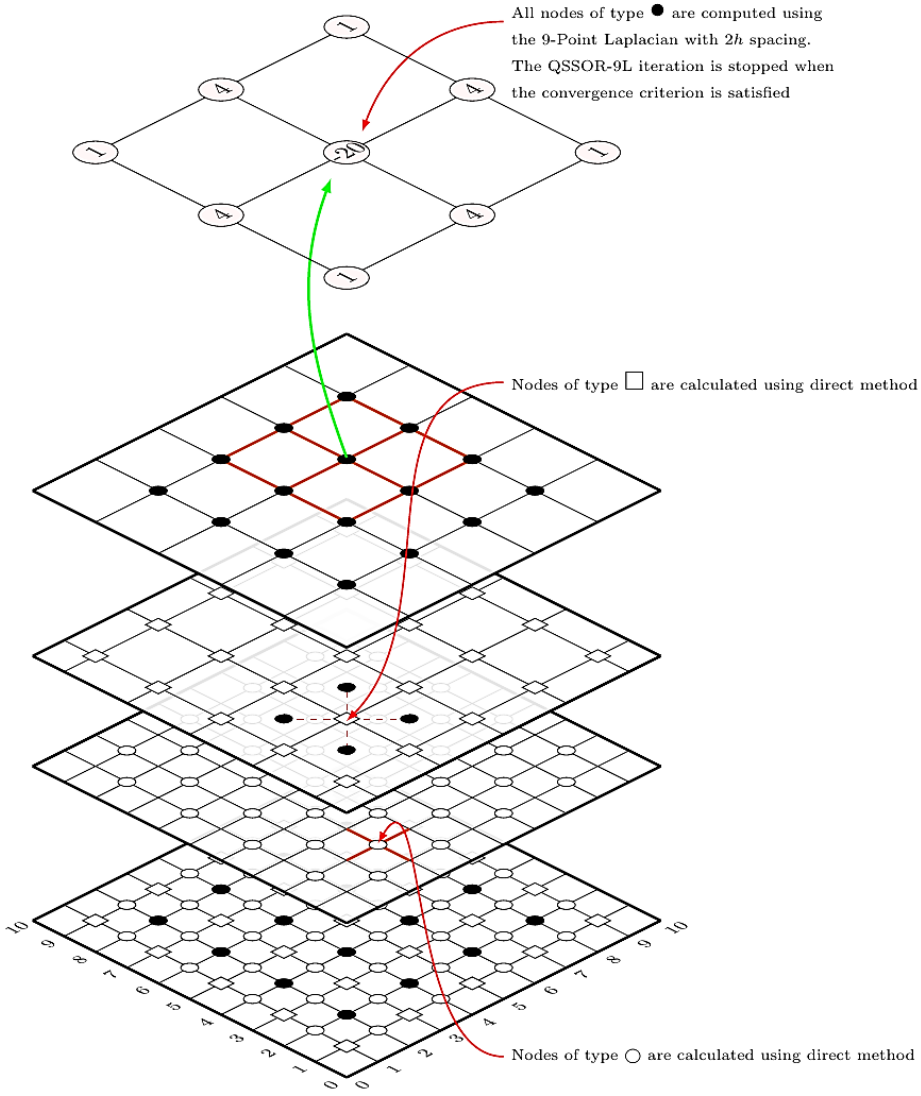

Figure 5. The Boosted iterative methods with 9-point Laplacian based on QS iteration considers only quarter of the total nodes

2.1. Boosted SOR methods with the 9-point Laplacian

By using a weighted parameter ω, the corresponding 9-point SOR iterative schemes for Full-Sweep Boosted SOR (FSBSOR), Half-Sweep Boosted SOR (HSBSOR) and Quarter-Sweep Boosted SOR (QSBSOR) are given as (11) to (13):

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-1,j}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k+1)} + u_{i,j+1}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-1,j-1}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}, \quad (11)$$

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-1-1,j}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-2,j}^{(k+1)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k+1)} + u_{i,j+2}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}, \quad (12)$$

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-2,j}^{(k+1)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k+1)} + u_{i,j+2}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-2,j-2}^{(k+1)} + u_{i+2,j-2}^{(k+1)} + u_{i-2,j+2}^{(k)} + u_{i+2,j+2}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}. \quad (13)$$

2.2. Boosted AOR methods with the 9-point Laplacian

The 9-point iterative schemes for the Full-Sweep Boosted AOR (FSBAOR), Half-Sweep Boosted AOR (HSBAOR), and Quarter-Sweep Boosted AOR (QSBAOR) cases are given as (14) to (16):

$$\mathrm{u}_{\mathrm{i,j}}^{(\mathrm{k+1})} = \frac{\omega}{5}\left(\mathrm{u}_{\mathrm{i-1,j}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i+1,j}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i,j-1}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i,j+1}}^{(\mathrm{k})}\right) + \frac{\mathrm{r}}{5}\left(\mathrm{u}_{\mathrm{i-1,j}}^{(\mathrm{k+1})} - \mathrm{u}_{\mathrm{i-1,j}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i,j-1}}^{(\mathrm{k+1})} - \mathrm{u}_{\mathrm{i,j-1}}^{(\mathrm{k})}\right) + \frac{\omega}{20}\left(\mathrm{u}_{\mathrm{i-1,j-1}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i+1,j-1}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i-1,j+1}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i+1,j+1}}^{(\mathrm{k})}\right) + \frac{\mathrm{r}}{20}\left(\mathrm{u}_{\mathrm{i-1,j-1}}^{(\mathrm{k+1})} - \mathrm{u}_{\mathrm{i-1,j-1}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i+1,j-1}}^{(\mathrm{k+1})} - \mathrm{u}_{\mathrm{i+1,j-1}}^{(\mathrm{k})}\right) + (1-\omega)\mathrm{u}_{\mathrm{i,j}}^{(\mathrm{k})}, \quad (14)$$

$$\mathrm{u}_{\mathrm{i,j}}^{(\mathrm{k+1})} = \frac{\omega}{5}\left(\mathrm{u}_{\mathrm{i-1,j-1}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i+1,j-1}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i-1,j+1}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i+1,j+1}}^{(\mathrm{k})}\right) + \frac{\mathrm{r}}{5}\left(\mathrm{u}_{\mathrm{i-1,j-1}}^{(\mathrm{k+1})} - \mathrm{u}_{\mathrm{i-1,j-1}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i+1,j-1}}^{(\mathrm{k+1})}\mathrm{u}_{\mathrm{i+1,j-1}}^{(\mathrm{k})}\right) + \frac{\omega}{20}\left(\mathrm{u}_{\mathrm{i-2,j}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i,j-2}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i+2,j}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i,j+2}}^{(\mathrm{k})}\right) + \frac{\mathrm{r}}{20}\left(\mathrm{u}_{\mathrm{i-2,j}}^{(\mathrm{k+1})} - \mathrm{u}_{\mathrm{i-2,j}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i,j-2}}^{(\mathrm{k+1})} - \mathrm{u}_{\mathrm{i,j-2}}^{(\mathrm{k})}\right) + (1-\omega)\mathrm{u}_{\mathrm{i,j}}^{(\mathrm{k})}, \quad (15)$$

$$\mathrm{u}_{\mathrm{i,j}}^{(\mathrm{k+1})} = \frac{\omega}{5}\left(\mathrm{u}_{\mathrm{i-2,j}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i+2,j}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i,j-2}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i,j+2}}^{(\mathrm{k})}\right) + \frac{\mathrm{r}}{5}\left(\mathrm{u}_{\mathrm{i-2,j}}^{(\mathrm{k+1})} - \mathrm{u}_{\mathrm{i-2,j}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i,j-2}}^{(\mathrm{k+1})} - \mathrm{u}_{\mathrm{i,j-2}}^{(\mathrm{k})}\right) + \frac{\omega}{20}\left(\mathrm{u}_{\mathrm{i-2,j-2}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i+2,j-2}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i-2,j+2}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i+2,j+2}}^{(\mathrm{k})}\right) + \frac{\mathrm{r}}{20}\left(\mathrm{u}_{\mathrm{i-2,j-2}}^{(\mathrm{k+1})} - \mathrm{u}_{\mathrm{i-2,j-2}}^{(\mathrm{k})} + \mathrm{u}_{\mathrm{i+2,j-2}}^{(\mathrm{k+1})} - \mathrm{u}_{\mathrm{i+2,j-2}}^{(\mathrm{k})}\right) + (1-\omega)\mathrm{u}_{\mathrm{i,j}}^{(\mathrm{k})} \quad (16)$$

3. RESULTS AND DISCUSSION

The simulation is run on the Robot 2D Simulator [25] on a Linux system with an Intel i5 processor running at 2.5 GHz and 8 GB of memory. The execution of the implementation of the QSBAOR approach based on (16) for solving 2-dimensional Laplace's problem as expressed in (1) is described in Algorithm 1. The start and goal points in the simulation are represented by red and green colored points, respectively. In the simulations, four different maps are employed, with five different sizes being tested. The experiment results for FS, HS, and QS Boosted methods, based on the 9-point Laplacian, are shown in this section. The iteration counts and CPU time for each algorithm are recorded as shown in Tables 1 and 2. Table 1 shows the results with optimal values of the relaxation and accelerated parameters are used throughout the experiments.

Algorithm 1. QSBAOR

i. $u \leftarrow$ Set configuration space (obstacles, destination)
ii. Set value of ω
iii. Divide the solution points into two types of points: black and white points.
iv. Compute all black points not including obstacles using equation (16)
v. iteration ← 0
vi. **repeat**

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-2,j}^{(k)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k)} + u_{i,j+2}^{(k)}\right) + \frac{r}{5}\left(u_{i-2,j}^{(k+1)} - u_{i-2,j}^{(k)} + u_{i,j-2}^{(k+1)} - u_{i,j-2}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-2,j-2}^{(k)} + u_{i+2,j-2}^{(k)} + u_{i-2,j+2}^{(k)} + u_{i+2,j+2}^{(k)}\right) + \frac{r}{20}\left(u_{i-2,j-2}^{(k+1)} - u_{i-2,j-2}^{(k)} + u_{i+2,j-2}^{(k+1)} - u_{i+2,j-2}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}$$

vii. iteration ← iteration +1
viii. **until** $\left\|u_{i,j}^{(k+1)} - u_{i,j}^{(k+1)}\right\| < \varepsilon$
ix. Compute all white points not including obstacles using rotated equation (12)
x. **do**

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-1-1,j}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-2,j}^{(k+1)} + u_{i+2,j}^{(k)} + u_{i,j-2}^{(k+1)} + u_{i,j+2}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}$$

end
Compute all square points not including obstacles using standard equation (11)
xi. **do**

$$u_{i,j}^{(k+1)} = \frac{\omega}{5}\left(u_{i-1,j}^{(k+1)} + u_{i+1,j}^{(k)} + u_{i,j-1}^{(k+1)} + u_{i,j+1}^{(k)}\right) + \frac{\omega}{20}\left(u_{i-1,j-1}^{(k+1)} + u_{i+1,j-1}^{(k+1)} + u_{i-1,j+1}^{(k)} + u_{i+1,j+1}^{(k)}\right) + (1-\omega)u_{i,j}^{(k)}$$

xii. **end**

Table 1. Performance of the methods in terms of number of iterations

Case	Method	N 300	600	900	1200	1500
Case 1	FSBSOR	2874	10450	23786	35469	62307
	FSBAOR	2317	8428	19327	28647	50524
	HSBSOR	1480	5381	12253	18229	32093
	HSBAOR	1179	4343	9899	14706	25994
	QSBSOR	757	2776	6375	9357	16575
	QSBAOR	187	2219	5145	8570	13379
Case 2	FSBSOR	2094	7956	19732	32360	45628
	FSBAOR	1721	6374	16021	26242	37267
	HSBSOR	1124	4175	9460	16674	23759
	HSBAOR	905	3353	7632	13502	19335
	QSBSOR	223	2082	4889	8558	12185
	QSBAOR	461	1591	3920	6859	10130
Case 3	FSBSOR	2548	9339	20264	23770	53719
	FSBAOR	2046	7523	16425	19306	43704
	HSBSOR	1302	4811	10459	12292	27751
	HSBAOR	1036	3870	8442	9975	22559
	QSBSOR	679	2495	5406	9364	14364
	QSBAOR	526	1998	4357	8150	11640
Case 4	FSBSOR	1674	6207	13529	23750	36996
	FSBAOR	1333	5007	10980	19321	30104
	HSBSOR	833	3193	6974	12278	19156
	HSBAOR	647	2569	5653	9952	15542
	QSBSOR	424	1654	3602	6359	9884
	QSBAOR	314	1319	2894	5125	8012

Table 2. Performance of the methods in terms of time of execution (in seconds)

Case	Method	N 300	600	900	1200	1500
Case 1	FSBSOR	1.602	23.318	123.358	326.310	932.092
	FSBAOR	1.497	22.092	120.699	307.958	893.052
	HSBSOR	0.533	8.251	44.013	116.616	328.061
	HSBAOR	0.493	7.420	39.259	103.669	291.926
	QSBSOR	0.219	3.329	17.712	46.912	137.155
	QSBAOR	0.187	2.908	16.589	45.593	120.228
Case 2	FSBSOR	1.422	22.837	150.992	345.879	756.108
	FSBAOR	1.361	20.787	153.955	322.320	713.486
	HSBSOR	0.516	8.019	37.637	119.921	266.421
	HSBAOR	0.437	6.585	33.287	105.910	235.898
	QSBSOR	0.223	3.117	14.704	46.390	103.631
	QSBAOR	0.199	2.578	12.829	40.690	93.254
Case 3	FSBSOR	2.084	29.294	138.506	308.722	1028.280
	FSBAOR	2.048	26.548	138.947	298.650	999.656
	HSBSOR	0.660	8.628	44.099	99.299	330.833
	HSBAOR	0.606	7.879	39.760	89.707	302.794
	QSBSOR	0.274	3.493	17.574	49.141	132.276
	QSBAOR	0.257	3.190	16.184	45.046	118.245
Case 4	FSBSOR	1.279	19.442	97.399	307.306	764.704
	FSBAOR	1.229	18.680	93.921	298.875	739.420
	HSBSOR	0.378	6.082	31.117	98.613	255.649
	HSBAOR	0.343	5.500	28.167	89.243	217.134
	QSBSOR	0.153	2.422	12.292	48.791	85.122
	QSBAOR	0.126	2.197	10.898	44.435	82.551

Figures 6 and 7 are graphs which represent the number of iterations and performance in time, respectively, shows the results of the suggested approaches based on Tables 1 and 2. Both figures indicate that the length of each execution increases with the number of iterations. It is clear from examination of both graphs that the QSBAOR outperforms the corresponding suggested approaches in terms of iteration count and time of performance. Tables 1 and 2 also make this concept quite obvious. As can be observed in the results table, the graphs for the quantity of iterations and performance time displayed the same trend. In comparison to other approaches, the QSBAOR iterative scheme clearly offers high efficiency according to performance time and number of iterations.

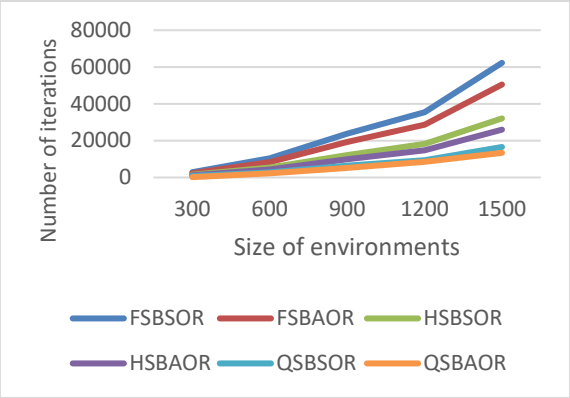

Figure 6. Number of iterations for the tested methods

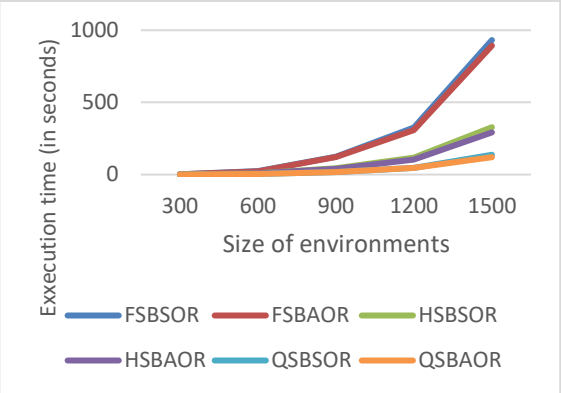

Figure 7. Execution time for the tested methods

The harmonic potentials obtained from the computation using the proposed approaches are forwarded to the path generation procedure that employs the gradient descent search (GDS) algorithm as described in Algorithm 2. The GDS is guided by the gradient of the harmonic potentials by moving from the start point with higher potential value to the next lower potential value until the lowest potential value in the environment that indicates the goal point is found. Figure 8 (in Appendix) shows the paths generated by the GDS algorithm using harmonic potentials computed by the proposed iterative methods on four 300×300 maps. In Case 1 (Figure 8(a)), smooth paths are obtained from three different start positions to the target locations. In Cases 2 and 3 (Figures 8(b) and (c)), large central obstacles do not prevent successful path generation from start to goal. Similarly, in Case 4 (Figure 8(d)), the presence of a narrow corridor does not hinder the generation of smooth and collision-free paths. Illustrations for other map sizes in Table 2 are omitted, as all proposed methods produce comparable path generation results.

Algorithm 2. GDS

```
i.      Setup start position as f [i, j] ← startposition [i, j].
ii.     Initializing found ← false and k ← 0.
iii.    Calculate
        Ucurrent ← U f [i, j]
        U g[i, j] ← MIN (U f [i-1, j], U f [i+1, j], U f [i, j-1], U f [i, j+1], U f [i-1,
        j-1], U f [i+1, j-1], U f [i-1, j+1], U f [i+, j+1])
        path [k] ← g[i, j]
        k ← k+1
iv.     If g[i, j] = goal position [i, j] then found ← true
v.      Else if Vg [i, j] < Vcurrent,, then f[i, j] ← g [i, j]
Step (ii) up to (iv) repeated until found or Vg [i, j] ≥ Vcurrent.  If not found then
path← 0
```

The number of arithmetic operations needed by the tested methods are shown in Tables 3 to 5, where $M=N^2-P$ represents the number of nodes calculated during the iteration, N^2 is the size of environment and P denotes the number of nodes occupied by obstacles. Based on these tables, the iterative techniques based on the HS Boosted (HSBSOR and HSBAOR methods) compute only half of the node points in a skewed manner during the iteration process. As a result, the computational complexity has been lowered by around 50%. Simulated results based on the QS Boosted (QSBSOR and QSBAOR methods) are provides much better performance. All of these iterative techniques assess only one of the four node points at a time during the iteration phase. As a result, the computational complexity has decreased by around 75%. In addition to the reduction in computational complexity, the performance improvement of the tested algorithms is further quantified in terms of the number of iterations and CPU time, as summarized in Table 6. Using the harmonic potentials derived through the above algorithms, the GDS had successfully constructed visually identical paths for all four different maps, similar to the results given in Figure 8.

Table 3. Arithmetic operations for SOR methods

Methods	ADD/SUB	MUL/DIV
FSBSOR	$8M$	$3M$
HSBSOR	$4M$	$\frac{3}{2}M$
QSBSOR	$2M$	$\frac{3}{4}M$

Table 4. Arithmetic operations for AOR methods

Methods	ADD/SUB	MUL/DIV
FSBAOR	$16M$	$5M$
HSBAOR	$8M$	$\frac{5}{2}M$
QSBAOR	$5M$	$\frac{5}{4}M$

Table 5. Number of additional arithmetic operations for the remaining white points

Methods	ADD/SUB	MUL/DIV
Half-Sweep cases	$4M$	$\frac{3}{2}M$
Quarter-Sweep cases	$6M$	$\frac{9}{4}M$

Table 6. Reduction percentages in terms of number of iterations and CPU time for the tested algorithms

Methods	Iteration	CPU time
FSBSOR vs FSBAOR	40.33	27.64
HSBSOR vs HSBAOR	26.48	24.77
QSBSOR vs QSBAOR	22.17	19.02

4. CONCLUSION

The intent of this research is to formulate and develop a way for combining iterative approaches and path finding algorithms to find a solution to the mobility robot's path problem. The concepts of HS and QS iteration are proposed in this work, as well as the use of computing the Laplacian harmonic potentials based on the 9-point Laplacian to solve the path planning problem by employing the family of relaxation iterative methods. The overall performances of the path planning algorithms are measured by using three criteria: successful generation of paths, number of iterations, and time performance.

Accordingly, the main contribution of this study, based on the summary of findings, is to introduce the application of QSBAOR which is Quarter-Sweep Boosted AOR in the 9-Point Laplacian operator using the families of relaxation methods for to determine the harmonic potentials by computing the solutions of Laplace's equation. The generated harmonic potentials were then employed by the GDS algorithm to guide the path finding method, resulting in a smooth path for the robot to traverse safely in a structured environment from any start point to the specified goal point.

FUNDING INFORMATION

The authors gratefully acknowledge the Ministry of Higher Education, Malaysia, for funding this work under Grant No. FRG0434-ICT-1/2016.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Sumiati Suparmin	✓			✓	✓	✓		✓	✓		✓			
Azali Saudi		✓	✓				✓			✓		✓	✓	✓

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The authors confirm that the data supporting the findings of this study are available within the article.

REFERENCES

[1] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar, and A. García-Cerezo, "Path Planning for Autonomous Mobile Robots: A Review," *Sensors*, vol. 21, no. 23, p. 7898, 2021, doi: 10.3390/s21237898.

[2] A. A. Panchpor, S. Shue, and J. M. Conrad, "A survey of methods for mobile robot localization and mapping in dynamic indoor environments," in *2018 Conference on Signal Processing And Communication Engineering Systems (SPACES)*, 2018, pp. 138–144, doi: 10.1109/spaces.2018.8316333.

[3] Y. Xiu-xia, C. Wei-yi, Z. Yi, F. Guo-wei, and Y. Xuan, "Mobile robot path planning in complex environment," in *2019 IEEE International Conference on Unmanned Systems (ICUS)*, 2019, pp. 426–431, doi: 10.1109/icus48101.2019.8996020.

[4] K. Shi, P. Wu, and M. Liu, "Research on path planning method of forging handling robot based on combined strategy," in *2019 IEEE International Conference on Unmanned Systems (ICUS)*, 2021, pp. 292–295, doi: 10.1109/icpeca51329.2021.9362595.

[5] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, Mar. 2019, doi: 10.1177/1729881419839596.

[6] S. M. LaValle, "Motion Planning," *Robotics & Automation Magazine, IEEE*, vol. 18, no. 1, pp. 79–89, 2011, doi: 10.1109/mra.2011.940276.

[7] C. I. Connolly, J. B. Burns, and R. Weiss, "Path planning using Laplace's equation," in *Proceedings, IEEE International Conference on Robotics and Automation*, 1990, pp. 2102–2106, doi: 10.1109/ROBOT.1990.126315.

[8] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," in *Proceedings 1987 IEEE International Conference on Robotics and Automation*, 1987, pp. 1152–1159, doi: 10.1109/robot.1987.1087982.

[9] J. S. Zelek and M. D. Levine, "Local-global concurrent path planning and execution," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 30, no. 6, pp. 865–870, 2000, doi: 10.1109/3468.895924.

[10] J. H. Reif, "Complexity of the mover's problem and generalizations," in *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, San Juan, PR, USA, 1979, pp. 421-427, doi: 10.1109/SFCS.1979.10.

[11] M. Karnik, B. Dasgupta, and V. Eswaran, "A comparative study of Dirichlet and Neumann conditions for path planning through harmonic functions," *Future Generation Computer Systems*, vol. 20. no. 3, pp. 441–452, 2004, doi: 10.1016/j.future.2003.07.008..

[12] S. M. La Valle, "Rapidly-Exploring Random Tree: A New Tool for Path Planning," Technical Report, Dept. of Comp. Sc., Iowa State University, Tech Report, 1998.

[13] S. Sasaki, "A practical computational technique for mobile robot navigation," in *Proceedings of the 1998 IEEE International Conference on Control Applications (Cat. No.98CH36104)*, Trieste, Italy, 1998, pp. 1323-1327 vol. 2, doi: 10.1109/CCA.1998.721675.

[14] D. Young, "Iterative methods for solving partial difference equations of elliptic type," *Transactions of the American Mathematical Society*, vol. 76, no. 1, pp. 92–111, 1954, doi: 10.1090/s0002-9947-1954-0059635-7.

[15] L. M. Kew and N. H. M. Ali, "New explicit group iterative methods in the solution of three dimensional hyperbolic telegraph equations," *Journal of Computational Physics*, vol. 294, pp. 382-404, 2015, doi: 10.1016/j.jcp.2015.03.052.

[16] W. K. Ling, A. A. Dahalan, and A. Saudi, "Autonomous path planning through application of rotated two-parameter overrelaxation 9-point Laplacian iteration technique," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 2, pp. 1116-1123, 2021, doi: 10.11591/ijeecs.v22.i2.pp1116-1123.

[17] J. Chen, D. S. Cheng, R. Jie, and X. Zhu, "A fourth-order 9-point finite difference method for the Helmholtz equation," *Journal of Physics: Conference Series*, vol. 1453, no. 1, p. 12044, 2020, doi: 10.1088/1742-6596/1453/1/012044.

[18] N. I. M. Fauzi and J. Sulaiman, "Half-Sweep Modified Successive Over Relaxation Method for Solving Second Order Two-Point Boundary Value Problems Using Cubic Spline," *International Journal of Contemporary Mathematical Sciences*, vol. 7, no. 32, pp. 1579–1589, 2012.

[19] M. U. Alibubin, A. Sunarto, and J. Sulaiman, "Quarter-sweep Nonlocal Discretization Scheme with QSSOR Iteration for Nonlinear Two-point Boundary Value Problems," *Journal of Physics: Conference Series*, vol. 710, p. 12023, Apr. 2016, doi: 10.1088/1742-6596/710/1/012023.

[20] M. K. M. Akhir, J. Sulaiman, M. Othman, Z. A. Majid, M. S. Muthuvalu, and E. Aruchunan, "Application of four-point MEGMSOR method for the solution of 2D Helmholtz equations," *International Journal of Mathematical Analysis*, vol. 9, no. 38, pp. 1847–1862, 2015, doi: 10.12988/ijma.2015.56170.

[21] J. H. Eng, A. Saudi, and J. Sulaiman, "Poisson image blending by 4-EDGAOR iteration via rotated five-point Laplacian operator," *Journal of Physics: Conference Series*, vol. 1123, p. 12033, Nov. 2018, doi: 10.1088/1742-6596/1123/1/012033.

[22] J. L. Santos, W. S. Yousif and M. M. Martins, "The explicit group TOR method," *Neural, Parallel & Scientific Computations*, vol. 20, no. 2012, pp. 459-474.

[23] N. M. Ali, "Design of a New Block Algorithm Stencil9-point in Boundary Value Problems (In Malay: Reka bentuk algoritma blok baru stensil 9-titik dalam masalah sempadan)," *MATEMATIKA*, vol. 18, pp. 45–62, 2002, doi: 10.11113/matematika.v18.n.495.

[24] S. T. Ling and N. H. M. Ali, "Fourth order modified explicit decoupled group scheme in the solution of Poisson equation," in *AIP Conference Proceedings*, 2016, vol. 1750, p. 30018, doi: 10.1063/1.4954554.

[25] A. Saudi and J. Sulaiman, "Path planning simulation using harmonic potential fields through four point-edgsor method via 9-point laplacian," *Jurnal Teknologi*, vol. 78, no. 8–2, Aug. 2016, doi: 10.11113/jt.v78.9537.

APPENDIX

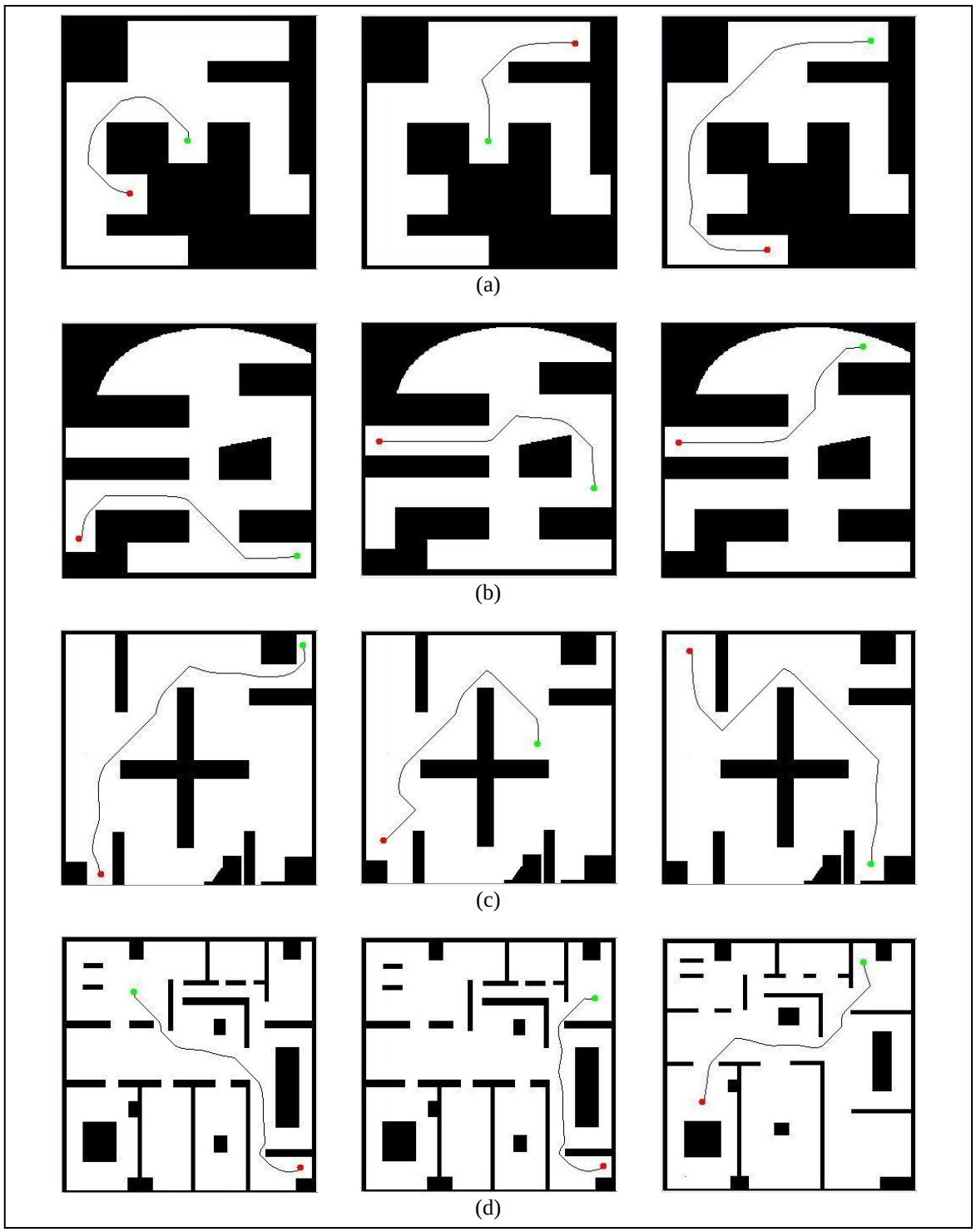

Figure 8. Path creation for various surroundings using various starting locations (green point) and target places (red point); (a) Case 1, (b) Case 2, (c) Case 3, and (d) Case 4

BIOGRAPHIES OF AUTHORS

Sumiati Binti Suparmin is currently student at the Faculty of Computing and Informatics, University Malaysia Sabah, Kota Kinabalu, Malaysia. She holds a Master degree in science with specialization Technology Computer Aided Design. Her research areas are robot path planning, heuristic methods, Laplacian equation, harmonic potential, and iterative methods. She can be contacted at email: cikksumi@gmail.com.

Azali Saudi is an Associate Professor at the Faculty of Computing and Informatics, Universiti Malaysia Sabah (UMS), Kota Kinabalu, Malaysia. His major areas of interest includes computational mathematics, artificial intelligence, and robot vision. He has a Ph.D. in Mathematics from UMS, a Master's degree in Artificial Intelligence from the University of Edinburgh, and a Bachelor of Science in Computer Science from Universiti Kebangsaan Malaysia. He has written and co-written over 100 articles for journals, book chapters, and conference proceedings papers. He leads research in intelligent robots, robot vision, deep learning, cloud computing, and computational mathematics. He can be contacted at email: azali@ums.edu.my.