A Boundary Node Method for Path Planning of Mobile Robots

R. A. Saeed[a], Diego Reforgiato Recupero[a], Paolo Remagnino[b]

[a]*Department of Mathematics and Computer Science, University of Cagliari, Cagliari, Italy*
[b]*The Robot Vision Team, Kingston University, London, UK*

Abstract

In this paper we propose a new method for solving the path planning problem in a static environment to find an optimal collision-free path between starting and goal points. First, the grid model of the robot's working environment is constructed, and then the potential value of the grid cells is calculated based on the new proposed potential function. This function is used to guide the robot to move toward the desired goal, it has the lowest value at the goal position and the value is increased as the robot moves further away. Second, we developed an efficient method, called the Boundary Node Method, to find the initial feasible path. In this method, the robot is simulated by a nine-node quadrilateral element, where the centroid node represents the robot's position. The robot moves in the working environment toward the goal with eight-boundary nodes based on the potential value of the boundary nodes. The initial feasible path is generated from a sequence of waypoints that the robot has to traverse as it moves toward the goal point without colliding with any obstacles. However, the proposed method can generate the path safely and efficiently, but the path is not optimal in terms of the total path length. Therefore, in order to construct an optimal or near-optimal collision-free path, an additional method, called the Path Enhancement Method, is developed. Finally, the cubic spline interpolation is adopted to generate a continuous smooth path that connects the starting point to the goal point. The proposed method has been tested in several working environments with different degrees of com-

Email addresses: `raza.saeed@unica.it` (R. A. Saeed), `diego.reforgiato@unica.it` (Diego Reforgiato Recupero), `p.remagnino@kingston.ac.uk` (Paolo Remagnino)

plexities. The results demonstrated that the proposed method is able to generate near-optimal collision-free path efficiently. Moreover, we compared the performance of the proposed methods with the other path planning methods in terms of path length and computational time. The results revealed that the proposed method can solve the robot path planning problem more efficiently. Finally, in order to verify the performance of the developed method for generating a collision-free path, experimental studies were carried out on the real robot.

Keywords: Robot Path Planning, Path Optimization, Simulation Model, Autonomous Mobile Robot, Potential Function, Boundary Node Method, Path Enhancement Method

---

1. Introduction

The aim of path planning is to find a collision-free path for a mobile robot to move from a starting point to a goal point in a given working environment based on certain optimization criteria, such as, the walking distance, the walking time, the energy consumption, and so on [1, 2, 3]. It is expected that the robot reaches the final destination point safely through the shortest walking path within the minimum computational time. Path planning has been widely applied in many robotic applications to perform various tasks that humans could not accomplish in several domain such as nuclear facilities [4], for space exploration [5], for rescue mission, landmines and enemies in war field [6]. In addition, path planning approaches are useful for repeatable tasks in static environments where optimality is essential (e.g. industrial applications) [7]. These factors make the path planning an interesting and challenging subject for researchers [6].

The path planning problem started around the sixties, but the interest in the path planning area for mobile robot grew after the work of authors in [8] after which many methodologies have been proposed [2, 9]. The existing methods are mainly categorized into classical and heuristic path planning [6, 2]. The classical methods include cell decomposition, potential field method, subgoal network and road map [10]. They involve finding a set of defined steps to search for a path starting from an initial position to a goal position. In classical methods only deterministic actions are considered [11, 10].

Table 1: Abbreviations used in this study

$Notation$	Description
BNM	Boundary Node Method
PEM	Path Enhancement Method
IFP	Initial Feasible Path
C	Workspace
C_{obs}	Space occupied by obstacles
C_{free}	Free Space, $C_{free} = C – C_{obs}$
C_s	Start Point, $C_s(x_s, y_s) \in C_{free}$
C_g	Goal Point, $C_g(x_g, y_g) \in C_{free}$
C_{BGC}	Boundary Grid Cells, $C_{BGC} \subset C_{obs}$
C_r	Robot Position, $C_r(x_r, y_r) \in C_{free}$
$p_1(t)$	Current Location, $p_1(t) = [x_1(t)\,;\, y_1(t)]$
$p_2(t)$	Updated Location, $p_2(t) = [x_2(t)\,;\, y_2(t)]$
s_y	Variation Potential Value between $p(4)$ and $p(6)$
s_x	Variation Potential Value between $p(2)$ and $p(8)$
p_{best}	Best Node Position
d	Distance between the centre and the edge of the obstacle, $d = 0.5\,unit$
e	Motion Directions, $e(u), (u = 1...8)$
w	Set of waypoints

However, it has been found that the classical methods have some disadvantages such as the high computational cost, trapping into local minima, and high time complexity in high dimensions [6, 9, 1]. As classical search methods fail to find exact solutions, many heuristic methods have been proposed, i.e. Genetic Algorithm (GA) [12, 7], Particle Swarm Optimization (PSO) [6], Artificial Neural Networks ($ANNs$), Ant Colony Optimization (ACO) [9, 13], and Fuzzy Logic (FL) [9]. Surveys works in [1, 13] showed that the heuristic path planning methods are computationally more efficient in terms of path distance, obstacle avoidance, and elapsed time [9]. Heuristic methods attempt to find a good solution to the path planning problem in a short amount of time, but these methods are not guaranteed to provide an optimal solution [11, 10]. The combinatorial path planning methods in continuous space can solve many path planning problem and construct optimal solution efficiently [10, 14, 15, 16, 17].

Many of the existing methods for robot path planning are able to find a path for the robot, but in most of the cases, the quality of the generated path is not accurate enough or their efficiency is not sufficient [18]. Researchers have always been seeking for a better solution to improve the performance of the existing path planning methods. A list of goals that researchers of several earlier works have pursued is the following: improve the accuracy [19, 5, 20], improve the efficiency [21, 18], increase safety [4, 5, 22], increase the capability [23], reduce the processing time [24, 25], overcome the non-reachable goal problem [26], pass through narrow passages [27], overcoming the local-trap problems, and improve the quality of planned paths [18]. However, several important gaps and limitations still need to be addressed, as outlined in the following:

1. In several works, the computational time is still too high because the process of a large number of unnecessary points. Moreover, the search for an optimal path might not succeed [2].
2. In many previous studies, the considered environments are relatively too simple and unusual for testing the efficiency of the proposed method [28, 29]. Obviously, the path planning problem in a complex environment can be very difficult and it is still a challenging issue.
3. As the range of a robot's application is expanding over time, the complexity

of the path planning problem and working environment are increasing as well. For this reason, it becomes much more difficult to find an optimal path within a reasonable amount of time [2].

4. In computational complexity theory, path planning is classified as a non - deterministic polynomial time problem NP-complete, and the required computational time grows exponentially as the complexity of the path-planning problem increases [9].

5. There are many methods that use random operation to produce a set of solutions for each independent run. Then, in order to find the optimal, all of these different solutions are selected, combined and replaced. This process requires a lot of computational time, therefore, reducing the variation of the final solution is important. [2].

Based on the limitations and research gaps, as previously explained above, we investigate a novel off-line path planning method for a mobile robot in a two-dimensional ($2D$) static environment. In the developed method, that we called the Boundary Node Method (BNM), the robot is simulated by a nine-node quadrilateral element, where the centroid node represents the robot's location and it moves with eight-boundary nodes in the working environment. The robot is exploring the environment with the help of the node's potential value at each location, where the potential value is calculated based on the proposed potential function. In this method, we have considered only 8-generated grid points that are overlapping with the eight-boundary node, rather than considering all the generated points which lead to less computational time. Moreover, the proposed method is capable of generating an efficient path for a mobile robot safely and quickly and it can also overcome the local minima problem. We also developed an additional method, that we called the Path Enhancement Method (PEM), to construct an optimal path by reducing the number of waypoints (w) and path length.

The term BNM has already been used in one of the meshless boundary integral equation methods that combine the Moving Least squares (MLS) interpolation with the Boundary Integral Equations ($BIEs$) to solve boundary value problems in potential theory and engineering.

To evaluate the contribution of the proposed approach, a comparison study has been conducted between the proposed method and the other path planning methods, namely, PSO, GA, $A-Star$, and Artificial Potential Field (APF). The comparison results are presented and discussed in subsection 5.3. The PSO algorithm is widely used in path planning problems [19, 29, 23] as it is fast and simple [29], easy to implement [30], and a powerful means [14] to solve mobile robot navigation problems [23]. Moreover, $A-Star$ algorithm is an effective and direct method to search paths [22], which was used for many path planning applications [31], and it is mainly employed on an environmental grid [20]. In addition, GA is known as a robust optimization method among the existing approaches for robot motion planning problem [18]. By taking advantage of its strong optimization ability, the GA has been widely used in previous study to generate an optimal path [32]. The potential field method is a fast [31, 29, 13], simple [31, 26], easy to implement [31] method, and it has good results for path planning [13]. The disadvantages of artificial potential field method are related to the local minima problem that it can incur [29, 31, 26].

Therefore, the main contributions of this paper can be summarized in the following points:

1. The proposed method, BNM, is capable of finding the initial feasible path (IFP) for a mobile robot without colliding with any obstacles even if the complexity of the environment is increased.
2. The proposed method uses an optimization technique based on the lowest potential value to accelerate the robot to find the path safely and quickly in reasonable time.
3. An additional method, PEM, is developed to find an optimal or near optimal path from the IFP by reducing the number of waypoints and the overall path length.
4. The proposed method does not work through random operations and there is no uncertainty in generating points, which leads to finding the final solution for the problem without variation in solution.
5. The proposed method generates a safe path for a mobile robot to navigate in a

complicated environment within a relatively short computational time.

6. The concept involved in the proposed method is simple and can be applied in a grid environment efficiently.
7. The computational time required to solve path planning problem by using BNM does not increase significantly with the increase of the environment's complexity.
8. The comparison between the developed approach and other path planning methods reveals that the BNM can solve the path planning problem effectively and efficiently in terms of path computational times and path length.

The remainder of this paper is structured as follows: Section 2 includes background work within the domain of mobile robot path planning. Section 3 introduces the path planning problem in a static and completely known environment. In Section 4, the details of the novel method and potential function are described with several illustrative examples. In Section 5, the application results of the developed method is presented and discussed for several working environments with different complexity. It also presents and discusses the comparison results between BNM and the other path planning method. The experimental study is presented in Section 6. Final conclusions and prospective future research are provided in Section 7.

2. Related Work

The path planning problem has attracted many researchers' attention due to the uncertainties, complexities and real-time nature of the problem [28], and it has been a very active area of research over the last few decades. In the literature, the problem of path planning for mobile robots has been widely discussed and various solutions and approaches have been proposed to solve it. For example, the authors in [2] proposed a new methodology to solve the path planning problem in two steps. First, they generate the IFP based on the surrounding point-set (SPS), which refers to a set of points that surround the obstacles. Then, they applied the path improvement algorithm to get the optimal path by using the outcome of the first step. As stated in [2], this method has a low-level of randomness that reduces the variation of solutions, and also this method is able to generate points in narrow or small spaces in the map. Another method is the

Bacterial Potential Field (BPF), developed by [15] to compute an optimal path for a mobile robot in a real-world scenario with static and dynamic obstacles. As reported in [15], the path planning with the BPF allows a robot to navigate in an autonomous way without being trapped in local minima.

Furthermore, there are a number of researchers who combined algorithms to improve path planning performance. For example, the authors in [14] presented a hybrid meta-heuristic $GA - PSO$ algorithm for mobile robot navigation to find an optimal path between starting and ending point in a grid environment. The proposed algorithm avoids time complexity and premature convergence in conventional GA and PSO algorithms. The hybrid $GA - PSO$ is used to generate the IFP, then a cubic B-spline technique is applied to construct a near-optimal collision-free path. To reduce the complexity of robot path planning, authors in [33] proposed a hierarchical path planning method by integrating fuzzy theory and genetic algorithm. To solve path planning problem, researchers in [34] suggested another method named $SACOdm$ Based on Simple Ant Colony Optimization Meta-Heuristic ($SACO - MH$). One of the main contributions of $SACOdm$ is the inclusion of memory capabilities to the artificial ants to prevent stagnation. Another contribution of this method is the use of the fuzzy cost function to evaluate the best path. An additional methodology has been proposed in [16] by integrating the Artificial Bee Colony (ABC) algorithm with the evolutionary programming algorithm. In this method ABC algorithm has been studied and applied to generate a feasible path, then the feasible path is enhanced by using an evolutionary programming algorithm.

Additionally, many researchers built on top of existing methods to improve their performance and to overcome their limitations. In fact, authors in [22] proposed an improved version of $A - Star$ algorithm to overcome inherent drawbacks of the original $A - Star$. One of the main improvements of the proposed method is that the local path is planned before the next search in the current node's neighbourhood. $A - Star$ algorithm calculates heuristic function's value at each node on the work area and checks adjacent nodes in order to find the optimal solution with zero probability of collision. However, its time complexity is too high. To overcome this problem, [25, 24] introduced a number of improvements to the $A - Star$ algorithm to reduce the computational time and to increase the overall performance. Another improvement is the minimization of the resulting path length by reducing the number of local paths. With the aim of reducing the chances of collisions between robot and obstacles, researchers in [20] presented a new approach of path planning technique, where they assumed that the virtual obstacle's size increased approximately $(2n + 1)$ times the size of the cell in the workspace. The capabilities of GA for solving the path planning problem for mobile robots in static and dynamic environments have been investigated by several researchers, who have extended the method. For example, the authors in [12] proposed a new fitness function for GAs whereas authors in [21] proposed Knowledge-based GA and authors in [18] proposed an adaptive GA. Furthermore, [35] presents the new variant of GA using the binary codes through matrix. Calculation of artificial potential values is another solution to obtain a collision-free path. This method was first used [36] for a collision-free robot motion planning problem. This method is based on attraction and repulsive values which are considered two fields produced by the target point and obstacles, and the robot is considered as a moving object in these fields. The robot moves toward the target based on the negative slope of the potential function. The problem with this approach is that the robot can get stuck in local minima of the potential field [18]. Consequently, various techniques have been proposed to avoid the minima, i.e. authors in [37] tried to solve the problem by using harmonic potential functions around obstacles. In order to solve the problem of non-reachable goals with obstacles nearby ($GNRON$) in potential field method, a new repulsive potential function is proposed by [26]. In order to overcome the local minima and heavy computational time for robotic path planning, probabilistic sampling-based algorithms such as the rapidly-exploring random tree (RRT), and the probabilistic roadmap (PRM) algorithms are introduced due to their remarkable practical performance and strong theoretical properties [38, 39]. Such algorithms work by computing multiple distributed random points in the free workspace and connect them to construct a tree or graph, after that a search method is used to find a path [22]. In RRT, the most important factor that affects the overall efficiency of path planning is how to select a tree to extend or connect. In the literature, the Rapidly-exploring Random Trees (RRT) algorithm has been widely used. [40] proposed a novel learning-based multi-$RRTs$ ($LM - RRT$)

approach for robot path planning in complex environments with narrow passages. As stated in [40], this approach can guarantee the efficiency of global path planning and enhance the local space exploration ability of each tree. The investigation of the shortest path with the minimum time required for the global path planning is carried out in [23] by using Modified PSO. Authors in [41] has made a comparison between between PSO and Q-learning, a Reinforcement Learning-type algorithm. For the single robot case, they showed that the final performance obtained with Q-learning approach is very similar to the one obtained with PSO. Some optimal path planning algorithms are presented in [27] for navigating mobile robot among obstacles and weighted regions. These algorithms can search an optimal path and also intelligently rotate the robot configuration to pass through narrow passages.

Researchers provided great effort for real application to solve path planning problem for mobile robots, and they proposed many new methods. For example, the authors in [7] presented preliminary results of the application of two-Kinect cameras system on a two wheeled indoor mobile robot for off-line optimal path planning and execution. To solve path planning problem for rovers, authors in [5] presented a new algorithmic improvement. In this study, they proposed $OUM - BD$ over the Ordered Upwind Method (OUM) to include a bi-directional search. They stated that the proposed method $OUM - BD$ is faster than the existing OUM. Authors in [6] proposed a multi-objective path planning algorithm based on improved PSO for robot navigation where the robot often involves various danger sources, such as a fire in a rescue mission, landmines and enemies in the war field. For emergency evacuation simulation, authors in [19] proposed a new path planning approach. In this approach, the Extended Social Force Model ($ESFM$) is combined with the improved ABC algorithm to improve the efficiency of crowd evacuation. Another approach called Grid-Based Random Tree Star ($GB - RRT$) has been developed by [4] to provide minimum dose path for occupational workers in nuclear facilities in complex environments. The probabilistic roadmap (PRM) method has been applied by [42] to optimize the walking path, and to reduce the radiation exposure of the staff in a radioactive environment of nuclear facilities. A research study revealed that in the radioactive environment of nuclear facilities, the proposed method has a good effect on path-planning, and it can make a route in a very short time.

3. Problem formulation

In this section, we state the path planning problem, that is moving the robot from a starting position and tracking it through all the intermediate waypoints until it reaches the goal position in a two-dimensional environment with static obstacles. The robot does not have to collide any obstacles and must optimize the path from starting position to the goal position.

Let us consider a $2D$ workspace $C = R^2$ for a mobile robot, the region of space occupied by obstacles is denoted by C_{obs}, and the obstacle-free region is represented by $C_{free} = C - C_{obs}$. The continuous workspace is divided into square grid cells. The grid cells have integer coordinates in the form $C(x, y) \in C$, with $1 \leq x \leq n$, and $1 \leq y \leq m$. A given cell can either correspond to a navigable area $C(x, y) \in C_{free}$ or to a space occupied by obstacles $C(x, y) \in C_{obs}$. Each grid cell $C(x, y)$ in C_{free} has a potential value $E(x, y) \in E$, which is calculated according to the potential function. The boundary grid cells of the workspace are also considered as obstacles. Grid cells are represented by $C_{BGC} \subset C_{obs}$. The robot position in the workspace is denoted by $C_r(x_r, y_r) \in C_{free}$, and the starting point $C_s(x_s, y_s) \in C_{free}$ and the goal point $C_g(x_g, y_g) \in C_{free}$. We assume that all the information related to the workspace is known in advance, as well as the obstacles which are assumed to be fixed, meaning that they do not change while the robot moves toward the target. For such a reason, the proposed method is known as off-line path planning, and generates the entire path to the goal before the motion begins.

In the proposed method, the robot is simulated by a nine-node quadrilateral element $p(q), (q = 1...9)$. The centroid node $p(5)$ is considered as the robot's location and the other nodes $(p(1 \rightarrow 4) \& p(6 \rightarrow 9)$ represent the eight-boundary nodes which are distributed uniformly around the robot's location, as shown in Figure $1a$. The robot moves forward and changes its direction based on the potential values and features of boundary nodes. The potential value $E(q), (q = 1...9)$ for the robot and boundary nodes are equivalent to the potential value of the corresponding generated points in the workspace. All the visited waypoints w, starting from C_s and ending at C_g, represent the obtained initial feasible path IFP. Figure 1 (b) shows the motion directions, and Figure 1 (c) shows the exploration location in the workspace.

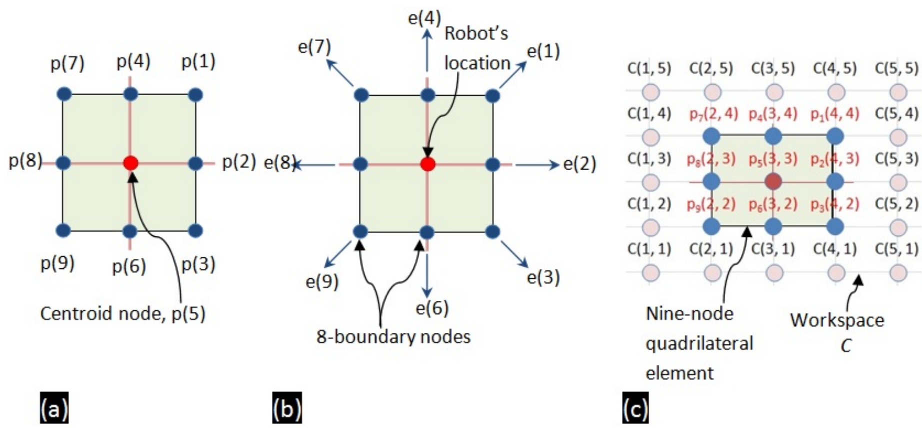

Figure 1: A nine-node quadrilateral element (a) along with its motion directions (b) and exploration location in the workspace (c).

4. Proposed method

This section describes the proposed method used in the presented study to find the optimal or near optimal collision-free path. The proposed method consists of four main steps:

1. Construct a $2D$ grid model of the robot's working environment, and then calculate the potential value of the grid cells based on the new proposed potential function. This function has the lowest potential value at C_g and the potential value is increasing as the robot moves further away.
2. Develop an efficient method, BNM, to generate the IFP for a mobile robot.
3. Develop an additional method, PEM, to construct an optimal or near optimal path from IFP, (as the obtained IFP is not optimal path in terms of the total path length).
4. Generate a continuous smooth path that connects the starting point to the goal point by using cubic spline method.

Figure 2 shows an overview of the four steps above mentioned. In the next subsections we will detail each of the four steps.

4.1. Modeling of the workspace

In the proposed method, all the grid cells of the given workspace meet the following equation:

$$C = \sum_{x=1}^{n} \sum_{y=1}^{m} C(x, y) \tag{1}$$

where n and m represent, respectively, the width and height of the workspace, and $C(x, y)$ represents the grid cells in the workspace. After constructing the workspace model, the potential value of each grid cell is calculated based on the new proposed potential function, as explained in the following paragraphs.

4.1.1. Potential Function PF

This section presents a new proposed potential function to calculate the potential value of grid cells in the workspace C. The procedure of calculating the potential value $E(k)$, with ($1 \leq k \leq N$) and N is the number of grid cells ($N = n \times m$) based on the proposed potential function is illustrated in Algorithm 1. Two examples of the new proposed potential function are shown in Figures $3a$ and $3b$. In these figures, the cell's colour represents the potential value, i.e. the blue cell corresponds to cells with the lowest potential value whereas the yellow cell corresponds to cells with the highest potential value. As shown in the figures, the shape of the potential function is conic and the global minimum of the total potential is located at the goal position. Because the lowest potential value of the goal point, it attracts the robot.

In Algorithm 1, the computed $E(k)$ represents the potential value of each grid cell $C(h, k)$, with ($h = 1...2$), and ($k = 1...N$) in the workspace C. The minimum potential value is formulated at the goal point $C_g(x_g, y_g)$. The distance between the start point $C_s(x_s, y_s)$ and the goal point $C_g(x_g, y_g)$ is represented by D, where the slope of a straight line D is denoted by m. The distance between the goal point $C_g(x_g, y_g)$ and surrounding point $C(h, k)$ in the workspace is represent by $d_p(1, k)$.

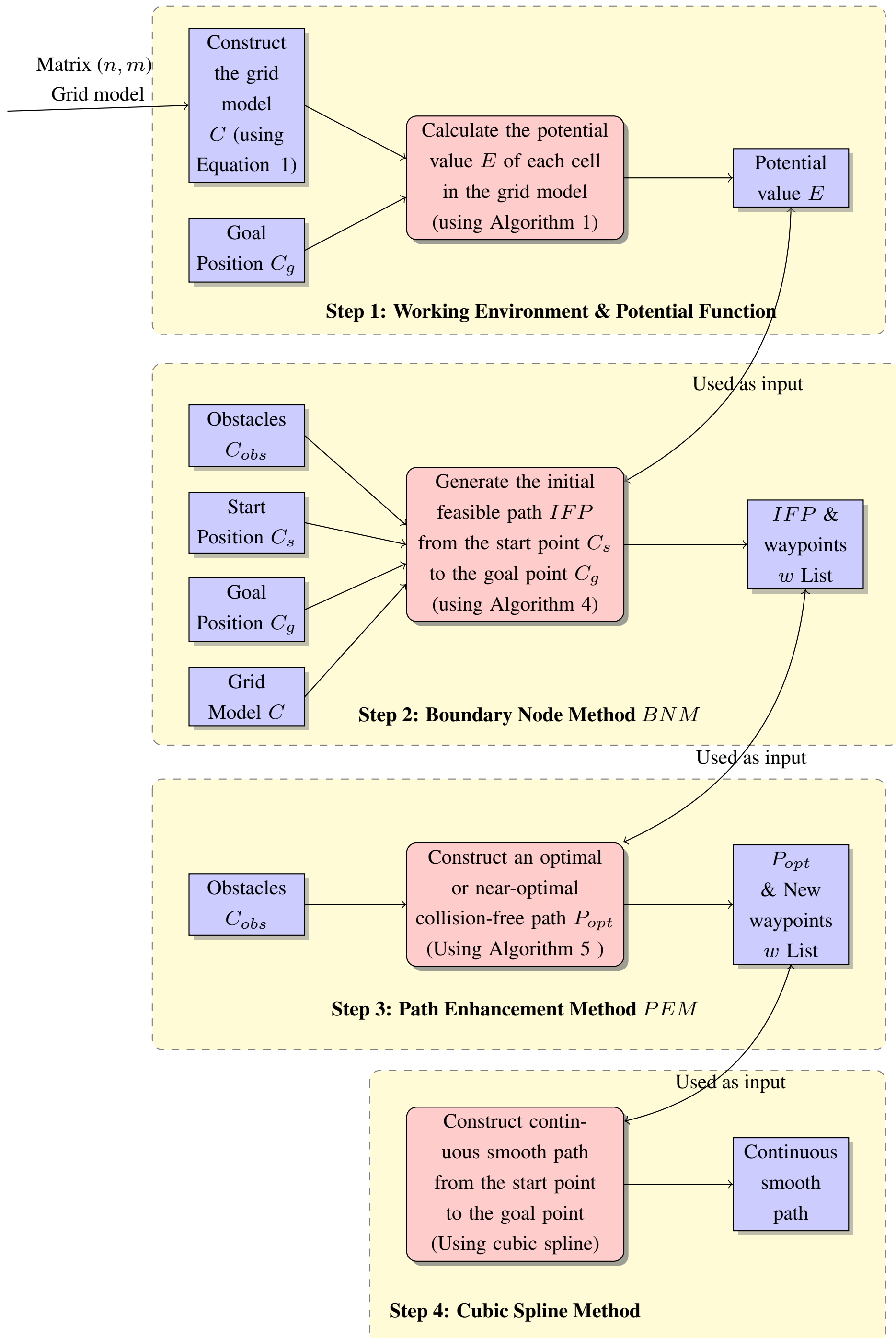

Figure 2: Flow diagram of the proposed method.

Algorithm 1 The calculation of potential value of grid cells in the workspace.

Inputs:

C_g and $C(h,k)$, $(h=1...2)$, and $(k=1...N)$

Initialize:

$E(k) \leftarrow 0, (k=1...N)$

$D = sqrt((x_s - x_g)^2 + (y_s - y_g)^2)$

$m = ((y_s - y_g)/(x_s - x_g))$

$c = (y_s - m * x_s)$

$ll = sqrt(m^2 + b^2), \quad (b=-1)$

for k = 1 to N **do**

$\quad d_p(1,k) = sqrt((C(1,k) - x_g)^2 + (C(2,k) - y_g)^2)$

$\quad L(1,k) = m \times C(1,k) + b \times C(2,k) + c$

$\quad d_l(1,k) = |L(1,k)|/ll$

$\quad E(k) = sqrt(d_l(1,k)^2 - d_p(1,k)^2)$

end for

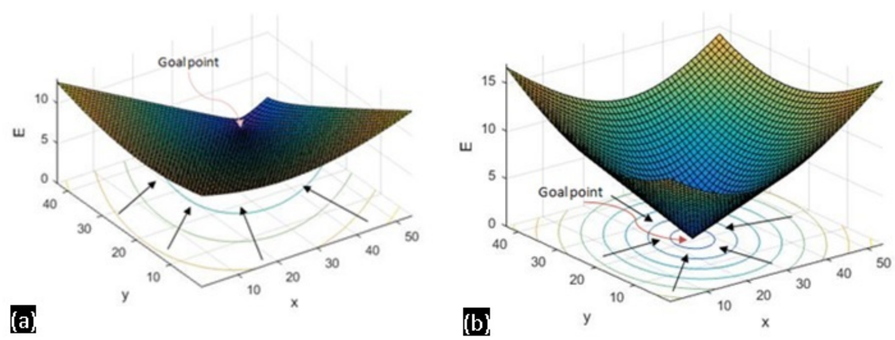

Figure 3: The potential value of grid cells in the workspace in $3D$ view with contour plot. The size of the workspace is 50×50, and the C_g is located at $a)$ $(40, 45)$ and $b)$ $(25, 25)$.

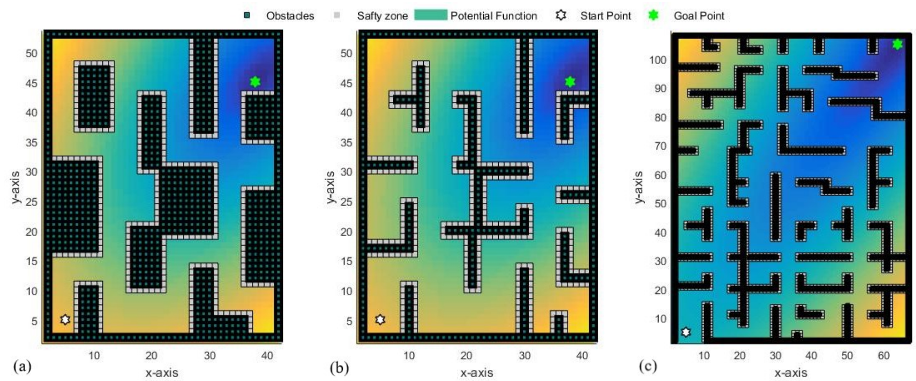

Figure 4: $2D$ model of the robot's workspaces.

4.1.2. Obstacles Representation

After constructing the workspace model for a mobile robot, a number of static obstacles are distributed at different locations in the workspace. To reduce the complexity of the proposed method, we assume that the obstacles form a set of square cells (1×1 $unit$). The centre of the obstacle's cells are denoted by a matrix $C_{obs}(h, l)$, with ($h = 1, 2$), and ($l = 1...O$), where O represents the number of obstacles. The distance d between the centre and the edge of the obstacle is constant, $d = 0.5$ $unit$. As the robot might move very close to the obstacle, they should keep a certain margin for safety. In this study, to avoid the possibility of overlapping the paths traced by the robot with obstacle boundary, we have created a safety zone around the obstacles.

An example of three different workspace scenarios with different obstacle layouts are shown in Figure 4. The characteristics of these three scenarios are illustrated in Table 2. The workspaces shown in Figure 4 are divided into square grid cells, where each cell is considered as either an obstacle C_{obs} or a non-obstacle C_{free}. The potential value of the grid cells in the C_{free} is calculated based on the proposed potential function, as illustrated in Algorithm 1. The grid cells of the workspace use a different colour to differentiate between C_{free} and C_{obs}, where the black cells represent C_{obs}, and the coloured cells represent the potential value in the C_{free}. The safety zone is represented by a number of gray square grid cells of the same size (1×1) $square\ unit$ around the obstacles.

Table 2: Characteristics of three different workspace scenarios of example.

Workspace $No.$	$C_s(x,y)$	$C_g(x,y)$	Workspace [cells]	Obstacles [cells]
1	(5,5)	(38,45)	2226	770
2	(5,5)	(38,45)	2226	345
3	(5,5)	(65,105)	7303	904

For the first (see Figure 4a) and second (see Figure 4b) designed scenario, the obstacles represent about 34.6% and 15.5% of the workspace, respectively. In the third scenario (see Figure 4c), a more complex environment with a higher number of obstacles of different size is considered, and here the obstacles represent about 12.4% of the workspace. After constructing the workspaces with obstacles and calculating the potential value of the grid cells, the robot's path needs to be determined.

4.2. *Proposed method BNM*

The BNM method consists of three steps:

1. Simulate the robot,
2. Exploration process, and
3. Obstacle avoidance

4.2.1. *Simulate the robot*

In the simulated model, the nodes are denoted by $p(q), (q = 1...9)$, and their locations are formulated by Equation 2. At iteration t, the current location of nodes denoted by $p_1(t)$. The x, y coordinates of the nodes' location represent by two vectors $x_1(t) = (x_{11}, x_{12}, ..., x_{19})$ and $y_1(t) = (y_{11}, y_{12}, ..., y_{19})$, respectively. Therefore, the current location of nodes $p_1(t)$ is formed by vertically concatenating $x_1(t)$ and $y_1(t)$, $p_1(t) = [x_1(t);\ y_1(t)]$.

$$p(q) = \begin{cases} x, y & q=5 \\ (x+v_x, y), (x, y+v_y), (xv_x, y), (x, yv_y) & q = 2, 4, 6, \text{ and } 8 \\ (x+v_x, y+v_y), (xv_x, y+v_y), (xv_x, yv_y), (x+v_x, yv_y) & q = 1, 3, 7, \text{ and } 9 \end{cases} \tag{2}$$

where x and y represent the coordinate of the robot location p_r. Moreover, v_x and v_y represent the horizontal and vertical distances between p_r and boundary nodes, $v_x = v_y = 1\ unit$. The boundary nodes p_1 can only move in eight-possible directions $e(u), (u = 1...8)$ (see Figure 1b), which we will explain in the next subsection.

4.2.2. Exploration process

In each iteration t, the current location of the robot and boundary nodes move in one particular direction. The new updated location of nodes $p_2(t)$ are calculate according to the following equations:

$$x_2(t) = x_1(t) + \Delta x \tag{3}$$

$$y_2(t) = y_1(t) + \Delta y \tag{4}$$

$$p_2(t) = [x_2(t);\ y_2(t)] \tag{5}$$

where $x_2(t)$ and $y_2(t)$ represent the coordinate of the new updated nodes' location.

The values of Δx and Δy are computed by using Algorithm 2. This algorithm is used to find the new updated location $p_2(t)$ of the current location of nodes $p_1(t)$. In this algorithm, the value g_x and g_y represent the distance between the current location of the robot $p_r(x_r, y_r)$ and the goal point C_g in x and y directions, respectively. The variables s_x and s_y represent the variation of the potential value between $p(2)\&p(8)$ and between $p(4)\&p(6)$, respectively, and the value of s_x and s_y are calculated by using Equation 6 and 7.

$$s_x(t) = E(p_1(1,8), p_1(2,8)) - E(p_1(1,2), p_1(2,2)) \tag{6}$$

$$s_y(t) = E(p_1(1,6), p_1(2,6)) - E(p_1(1,4), p_1(2,4)) \tag{7}$$

Δx and Δy have the same sign as the variation of the potential value (both positive or both negative). The coefficients α and β are constant, and these two coefficients will influence the convergence behaviour. The distance between $p_r(t)$ and C_g is decreasing step by step until the robot reaches the global minimum at the goal position.

The proposed method uses an optimization technique based on the lowest potential value to accelerate the robot to find the path and yield to fast convergence. Among all

Algorithm 2 Compute the values of Δx and Δy

Inputs:

$C_g, p_r(t)$

$E(q), (q = 1...9) \leftarrow E$

$s_x, s_y \leftarrow$ Equation 6 and 7

$g_x = x_r(t) - x_g$

$g_y = y_r(t) - y_g$

if s_x ¡ 0 **then**

compute $g_x = -1 * g_x$

end if

if s_y ¡ 0 **then**

compute $g_y = -1 * g_y$

end if

if $g_x = 0$ **then**

compute $\Delta x = 0$ and $\Delta y = \beta * g_y$

else if $g_y = 0$ **then**

compute $\Delta x = \alpha * g_x$ and $\Delta y = 0$

else

compute $\Delta x = \alpha * g_x$ and $\Delta y = \beta * g_y$

end if

boundary nodes, the node with the lowest potential value is chosen as the best position and denoted by p_{best}. At each iteration t, the robot update its position to the best position p_{best}. The boundary nodes, their position and potential value, guide the robot to move toward the goal location and help the robot to avoid obstacles, which we will discuss in the next section.

4.2.3. Obstacle Avoidance

In the workspace that contains no obstacles, the robot will reach the goal point along a straight line from any starting point. As obstacles exist, the robot interfere with obstacles when the distance between the robot and the obstacles is less than the distance d. Therefore, the robot and boundary nodes require to avoid obstacles and to change their moving direction by selecting a new position in the C_{free}.

To explain the obstacle avoidance, consider an example shown in Figure 5. The boundary nodes $p(1 \rightarrow 4)$ and $p(6 \rightarrow 9)$ are generated around the robot position $p(5)$ by using Equation 2. As shown in the Figure 5a, the red object represents the robot, and the blue objects represent the boundary nodes. At iteration t, the robot and boundary nodes are changing their positions from the current position (see Figure 5a) to the new updated position (see Figure 5b) by using Equations 3, 4, and 5. As a result, the nodes $p(7)$, $p(8)$, and $p(9)$ interfere with the obstacles (see Figure 5b). Therefore, the robot needs to investigate the workspace to find next position without colliding obstacles. In this case, the robot will move in y-direction either to upward or to downward direction. The motion direction depends on the value of s_y (see Figure 5c). The robot moves backward when $s_y(t)$ is negative, and it moves forward when $s_y(t)$ is positive.

Furthermore, in order to demonstrate how the robot avoids the obstacles and changes its motion direction with the help of boundary nodes, consider an example shown in Figure 6. As illustrated in Figure 6a and 6b, in the iterations $t = 1 - 4$, the robot starts to move from C_s and it moves forward toward the goal point from $p_1(t)$ to $p_2(t)$ using Equations 3, 4, and 5. At each iteration, all obstacles in the working environment are examined for possible collisions with the direct path from $p_1(t)$ to $p_2(t)$. As the robot moves toward the goal point C_g in the iteration $t = 5 - 6$, nodes $p(1)$, $p(2)$, and $p(3)$ interfere with obstacles (see Figure 6c). This implies that the robot can only move in

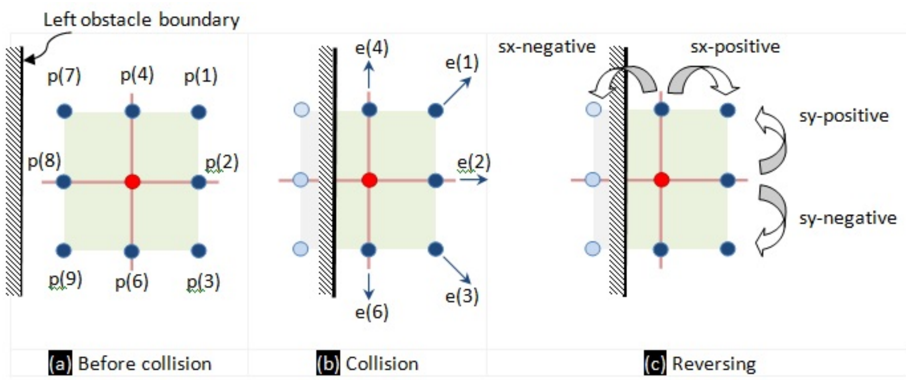

Figure 5: Obstacle avoidance in a static environment using BNM.

the y-direction, either upward (when s_y is positive) or downward (when s_y is negative). The next position of the robot must be in upward direction, because the value of s_y is positive ($E(6) > E(4)$). The same procedure is repeated for the iteration $t = 7 - 10$ by shifting the robot upward until the robot passes the block of obstacles, as shown in Figures 6d and 6e. For the iterations $t = 11 - 16$, the BNM method directs the robot to move forward (see Figures 6f and 6g) until the robot reaches its final destination point at the C_g (see Figure 6h).

Suppose that the long horizontal set of obstacles block the robot path as demonstrate in Figure 7a. As the robot moves toward the goal point, nodes $p(1)$, $p(4)$, and $p(7)$ interfere with obstacles. Therefore, the robot needs to change its motion direction along the x-direction to avoid the obstacles. The motion direction depends on the value of s_x. The robot moves to the right when $s_x(t)$ is positive, and it moves to the left when $s_x(t)$ is negative. In this case, the nodes at $p(8)$ and $p(2)$ have the same level of potential value $E(8) = E(2)$. This implies that the variation of the potential value between $p(2)$ and $p(8)$ is equal to zero $s_x = 0$. To solve this problem, the robot moves along both direction (see Figure 7b). As shown in the figure, nodes $p(7)$, $p(8)$, and $p(9)$ move one step to the left and nodes $p(1)$, $p(2)$, and $p(3)$ move one step to the right at the same time. Two temporary sets, which can be described as a ”waiting list”, of visited grid cells on the left and right-side are stored. As the simulated robot reaches the end of obstacles in left-side earlier (see Figure 7c), then the BNM method chooses

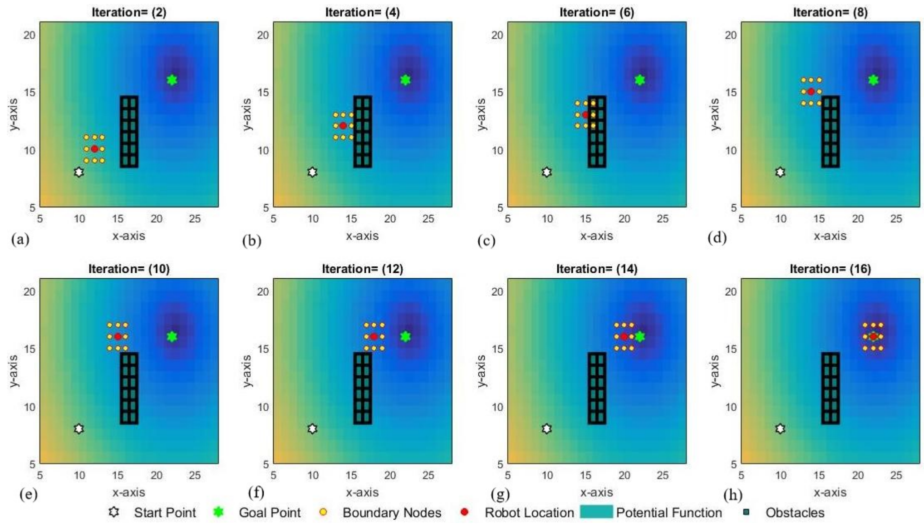

Figure 6: Demonstration of robot exploration in a two dimensional environment using BNM.

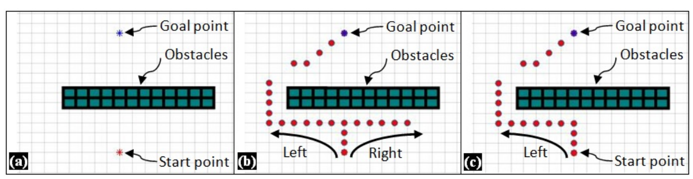

Figure 7: Workspace contains long horizontal set of obstacles that block the path of the robot.

the stored set of the left-side and disregards the stored points of the right-side.

In order to solve the local minima problem, we introduced an algorithm (see Algorithm 3). This algorithm executes a sequence of steps that pulls the robot out of a local minimum. In order to illustrate the steps required by the robot to come out of a local minimum Algorithm 3 is used, for instance the workspace with a U-shaped obstacle as shown in Figure 8. As shown in Figure 8a, the robot starts to move at position $(3, 15)$ toward the goal point at $(23, 3)$. Similarly, in Figure 8b, the robot moves from $(23, 15)$ to $(3, 3)$. The robot uses two different modes while moving in the simulated environment, namely the "normal mode" and the "local minimum recovery mode". In the normal mode, for iteration $(t)_{t=1,..,6}$, as the robot moves from the point $p_1(t)$ toward the point $p_2(t)$, the line between $p_1(t)$ and $p_2(t)$ does not intersect with the obstacles (see $step\,(1)$ Figure 8). In order to check the feasibility of the path represented by each line segment between corresponding points in $p_1(t)$ and $p_2(t)$, we create a new row matrix ($chk_{(q),(q=1...9)}$). The value of each element of the row matrix is equal to "0" or "1". At iteration t, for the $1^{st}\,element\,(q=1)$, if the line between the first node of the simulated model in $p_1(t)$ and $p_2(t)$ intersects obstacles, then the value $chk_{(q)}=1$, otherwise $chk_{(q)}=0$. The same procedure is then repeated for the $2^{nd}\,element\,(q=2)$, $3^{rd}\,element\,(q=3)$ until the last $element\;(q=9)$. In the normal mode, the values of $chk_{(q),(q=1...9)}$ are equal to "0", and the robot travels with the help of Algorithm 4. In the recovery mode, the robot switches to Algorithm 3, as shown in $steps\,(2\rightarrow 9)$ in Figure 8. The proposed method gives the highest priority to the obstacle avoidance processes and the lowest priority to the potential value. In $t=7$, as the robot moves forward from $p_1(t)$ to $p_2(t)$, the line segment connecting corresponding nodes $(3,6,\,and\,9)$ intersects the obstacles (see Figure 8). In this case the values of $chk_{(3)}$,$chk_{(6)}$, and $chk_{(9)}$ are equal to "1", then the robot moves to the right (see Figure 8a) or to the left (see Figure 8b) with the help of the Algorithm 3. Once the robot comes out from a local minimum, it can move smoothly again by using Algorithm 4 (see $step\,(10)$). In the $step(10)$ the BNM method gives the highest priority to the potential value until the robot reaches the goal point. As it can be seen in Figure 8, the robot is not blocked by the U- shaped obstacles, it always finds the path (if it exists) to reach the final destination point.

The proposed potential function is similar to the attractive potential field in the sense that both guide the robot to move toward the desired goal location, but differ in calculating the potential value E (see Algorithm 1), where the potential value $E_{(1,k)}$ is calculated by using Equation 8.

$$E_{(1,k)}=f(C_g,C_{(h,k)}) \tag{8}$$

As illustrated in the figure, in the $step\,(1)$ the robot starts to move toward the goal until it collides with obstacles. When the simulated robot detects a collision, the position of the interfered points in the boundary nodes is computed by using Equation 9.

Algorithm 3 local minima problem

1: Inputs: $C_{obs}, s_x, s_y, p_1(t), p_2(t)$

2: Check line segments between $p_{1(q),(q=1...9)}(t)$ and $p_{2(q),(q=1...9)}(t)$ for feasibility

3: If line between $p_{1(q)}(t)$ and $p_{2(q)}(t)$ interfered C_{obs} then $chk_{(q)} = 1$ otherwise $chk_{(q)} = 0$

4: Construct matrix $chk_{(q),(q=1...9)}$

5: **while** $sum(chk) > 0$ **do**

6:     **if** $chk_{(1)}, chk_{(2)}$, and $chk_{(3)} = 1$ **then**

7:         $p_{2x}(t)$=$p_{2x}(t)$-c_1, $c_1\, is\, constant$

8:         if $s_y > 0$ then $p_{2y}(t)$=$p_{2y}(t)$+c_2 otherwise $p_{2y}(t)$=$p_{2y}(t)$-c_2

9:         repeat $steps\, 2, 3,\, and\, 4$

10:         update $p_{1y}(t) \leftarrow p_{2y}(t)$

11:         store p_5 in a way-points w list

12:     **end if**

13:     **if** $chk_{(1)}, chk_{(4)}$, and $chk_{(7)} = 1$ **then**

14:         $p_{2y}(t)$=$p_{2y}(t)$-c_2, $c_2\, is\, constant$

15:         if $s_x > 0$ then $p_{2x}(t)$=$p_{2x}(t)$+c_1 otherwise $p_{2x}(t)$=$p_{2x}(t)$-c_1

16:         repeat $steps\, 2, 3,\, and\, 4$

17:         update $p_{1x}(t) \leftarrow p_{2x}(t)$

18:         store p_5 in a way-points w list

19:     **end if**

20:     **if** $chk_{(7)}, chk_{(8)}$, and $chk_{(9)} = 1$ **then**

21:         $p_{2x}(t)$=$p_{2x}(t)$+c_1

22:         if $s_y > 0$ then $p_{2y}(t)$=$p_{2y}(t)$+c_2 otherwise $p_{2y}(t)$=$p_{2y}(t)$-c_2

23:         repeat $steps\, 2, 3,\, and\, 4$

24:         update $p_{1y}(t) \leftarrow p_{2y}(t)$

25:         store p_5 in a way-points w list

26:     **end if**

27:     **if** $chk_{(3)}, chk_{(6)}$, and $chk_{(9)} = 1$ **then**

28:         $p_{2y}(t)$=$p_{2y}(t)$+c_2

29:         if $s_x > 0$ then $p_{2x}(t)$=$p_{2x}(t)$+c_1 otherwise $p_{2x}(t)$=$p_{2x}(t)$-c_1

30:         repeat $steps\, 2, 3,\, and\, 4$

31:         update $p_{1x}(t) \leftarrow p_{2x}(t)$

32:         store p_5 in a way-points w list

33:     **end if**

34: **end while**

35: return $p_1(t), p_2(t), w$

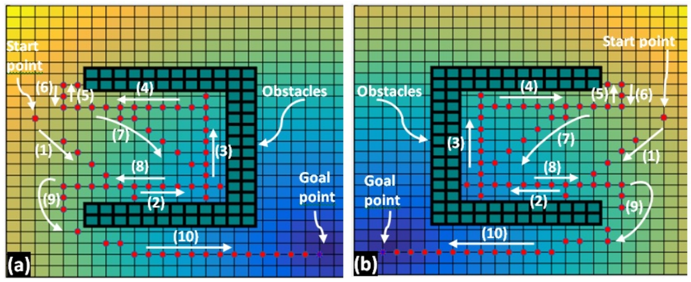

Figure 8: Simulation results of local minima problem solution using the proposed algorithm for a simple environment with U-shape obstacle.

$$p_{(t,h)} = [x_2(t); y_2(t)], p_{(t,h)} \in R^2 : p_{(t,h)} = (p_r \cap C_{obs}) \tag{9}$$

The new updated location of nodes $p_2(t)$ is calculated according to Equations 10 and 11, to avoid obstacles, as follow:

$$x_2(t) = x_1(t) + f(E_{(1,k)}, p_{(t,h)}, p_1, p_2, C_{obs}) \tag{10}$$

$$y_2(t) = y_1(t) + f(E_{(1,k)}, p_{(t,h)}, p_1, p_2, C_{obs}) \tag{11}$$

For the $step\,(2)$ to $step\,(9)$, the proposed method gives the highest priority to the obstacle avoidance processes and the lowest priority to the potential value. Afterwards, in the $step(10)$ the BNM method gives the highest priority to the potential value until the robot reaches the goal point. As it can be seen in Figure 8 the robot is not blocked in the U- shape obstacles, it always finds the path (if it exists) to reach the final destination point.

In this study, the BNM method is used to find IFP for a mobile robot to move from C_s to C_g in the workspace without colliding with any obstacles. The IFP is generated from a set of waypoints w that the robot visits before reaching the final destination point. For better clarity, the waypoints are connected into a continuous path. The line segment that connects two waypoints in sequence is represented by $P_{l,l+1}$, and the length of all line segments that connect all waypoints sequentially to each other is representing the length of IFP. A complete path IFP is formed by concatenation of all inter-line segments $P_{l,l+1}, 1 \leq l \leq w-1$ as follows: $IFP = [P_{1,2}, P_{2,3} \ldots, P_{w-1,w}]$. The main steps to find IFP for a mobile robot by using the proposed method is summarized in Algorithm 4.

According to Algorithm 4, the robot starts to move at the point $C_s(x_s, y_s)$ toward the goal point $C_g(x_g, y_g)$. The current nodes' location $p_1(t)$ of all nodes $p(q), (q = 1...9)$ at iteration t is formulated by Equation 2, where the x and y coordinate of the robot location p_r at the first iteration coincide with the x_s and y_s of the start point $C_s(x_s, y_s)$. The node with the lowest potential value among all boundary nodes is chosen as the best position and it is denoted by p_{best}, where the potential value $E(q), (q = 1...9)$ of nodes is computed by using Algorithm 1. For iteration t, the new updated location of nodes $p_2(t) = [x_2(t); y_2(t)]$ is calculated by Equations 3, 4 and 5. The variation of the potential value s_x and s_y is calculated by using the Equations 6 and 7. Afterwards the line segments between $p_1(t)$ and $p_2(t)$ check for feasibility. So, if collision is not found, then a new set of $E(q), (q = 1...9)$ and p_{best} need to be calculated, as previously explained. Subsequent, the current location $p_1(t)$ updates to the new location $p_2(t)$, and the robot $p_r(t)$ updates its position to the best position p_{best}. The proposed method stores the robot's location $p_r(t)$ in a waypoints w list. On the other hand, if the line segments between $p_1(t)$ and $p_2(t)$ collides with obstacles, another updated location $p_2(t)$ needs to be found, as previously explained in Section 4.2.3. This procedure will continue untill the mobile robot reaches the final destination point at $C_g(x_g, y_g)$ or the maximum number of iterations is reached.

Time complexity is the computational complexity that estimates the run-time of an algorithm. In the developed method, the computational time to find a set of waypoints w of the IFP can be calculated by summing the time needs for each line from 8 to 18 in Algorithm 4. The time complexity of the developed method BNM can be analysed as following: when the size of simulated model is $q (q = 9)$, the number of iterations is M, the problem size is N ($N = n \times m$), and the number of iterations needed by the robot to pass the block of obstacles is M_1.

1. In step 2, the time complexity of computing $p_1(q), (q = 1...9)$ is $T_1 = O(q)$.
2. In step 3, the time complexity of computing $E(k), (k = 1...N)$ is $T_2 = O(N)$.

Algorithm 4 BNM

Inputs:

C_s, C_g, C_{obs}, and $C(x, y), (x = 1...n, y = 1...m)$, maximum iteration number M

Initialize:

$p_1 \leftarrow Equation\, 2,\ x = x_s,\ y = y_s$

$E(k), (k = 1...N) \leftarrow$ Algorithm 1

$E(q), (q = 1...9) \leftarrow E$

$p_{best} \leftarrow$ minimum $E(q)$

$s_x, s_y \leftarrow$ Equation 6 and 7

while $(x_r \neq x_g \quad or \quad y_r \neq y_g$ within a $M)$ **do**

  $p_2(t) \leftarrow$ Equation 3, 4, and 5

  $s_x, s_y \leftarrow$ Equation 6 and 7

  Check the line segment between $p_1(t)$ and $p_2(t)$ for feasibility

  if $p(t)$ interfered with C_{obs} **then**

    $p_2(t) \leftarrow Obsticle\ Avoidance$

  end if

  $E(q), (q = 1...9) \leftarrow E$

  $p_{best} \leftarrow$ minimum $E(q)$

  $p_1(t) \leftarrow p_2(t)$

  $p_r(t) \leftarrow p_{best}$

  $p_r(t)$ in a way-points w list

end while

$IFP \leftarrow$ way-points w list

$P_{opt} \leftarrow$ Algorithm 5

$U \leftarrow$ Equation 12

End

3. In steps 4-6, the time complexity of calculating $E(q)$, p_{best}, s_x, and s_y, is $T_3 = O(q)$.
4. In steps 4-6, 8-9, the time complexity of calculating $p_2(t)$ is $T_4 = O(M * q)$.
5. In step 10, the time complexity of collision checking the line segment between $p_1(t)$ and $p_2(t)$ for feasibility, can be done by $T_5 = O(M * N * q)$.
6. In steps 11-13, in case the line between $p_1(t)$ and $p_2(t)$ collides with obstacles, the computing time spent in these is considerable longer, so the time complexity of these steps is $T_6 = O(M * N * M_1 * q)$.
7. In steps 14-18, the time complexity of determining the new set of $E(q), (q = 1...9)$ and p_{best}, together with updating p_1 and p_r, and store p_r in a way-points w list is $T_7 = O(q)$.

The total time complexity of the developed method is: $T = T_1 + T_2 + T_3 = T_4 + T_5 + T_6 + T_7$ $T = O(q) + O(N) + O(q) + O(M*q) + O(M*q*N) + O(M*N*M_1*q) + O(q)$ $= O(N * M * M_1)$

The obtained IFP for a mobile robot is a safe path, however, it is not a shortest path between C_s and C_g. In order to reduce the overall path length, a new method called path enhancement method PEM is developed, as we will explain in the following subsection.

4.3. Path Enhancement Method PEM

This section introduces the PEM method to generate the shortest path (see Figure 9b) from IFP (see Figure 9a). The PEM method is used to reduce the number of waypoints of the IFP between C_s and C_g obtaining an optimal or close-to-optimal path. As shown in Figure 9, the waypoints of the IFP are represented by red circle objects, and the obtained shortest path is represented by a thick red line. In order to explain the basic idea of PEM, consider an example shown in Figure 10.

In this example, the robot starts to move from the starting point and passes through all the intermediate waypoints until it reaches the goal point. As illustrated in Figure 10b, the IFP consists of 14 waypoints w, and they are connected by line-segments. In the figure, a line segment U has two end points, u_1 and u_2. For the first line segments

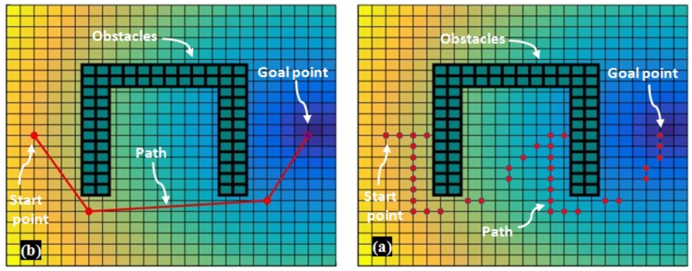

Figure 9: Example of path planning for a mobile robot. (a) The obtained solution of IFP by using BNM, where the sequence of the red circle objects is represents the IFP. (b) The shortest path found by using PEM, where the solid red line represents the shortest path.

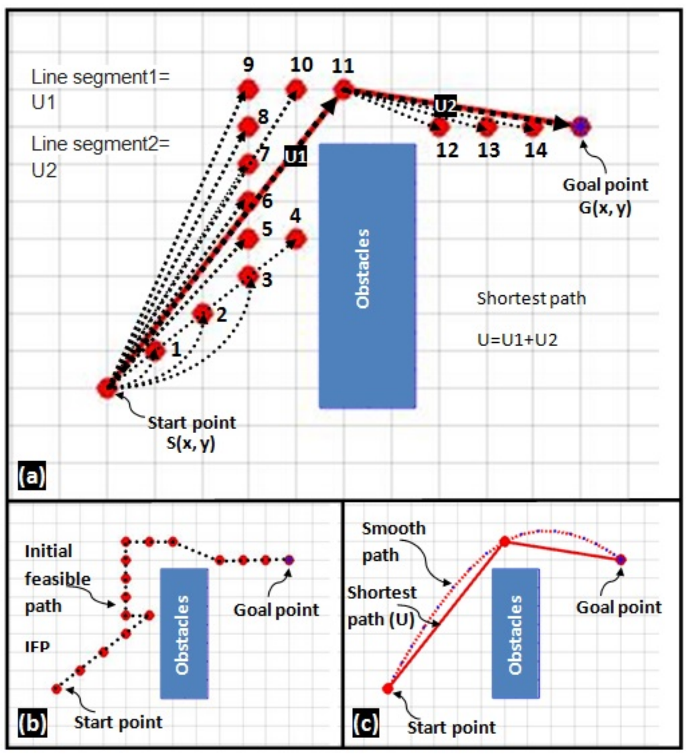

Figure 10: Construction of the shortest-path from 14 waypoints in the 2D workspace, where the waypoints are marked with the red circle objects. (a) PEM is used to find the shortest path between start and goal point. (b) IFP is generated by using BNM. (c) The shortest line-segment path (U) found by using PEM and the smooth path constructed by using spline method.

U_1, the starting position of u_1 coincides at the C_s. In order to determine the starting position of u_2, the PEM method connects u_1 with $w(j), (j = 1...J)$, $J = 14$ iteratively. First, u_1 is connected with the first waypoint $w(1)$, then the line between these two points is checked for feasibility. If a collision is not found, then u_1 is connected to $w(2)$. Afterward the line between u_1 and $w(2)$ is checked for feasibility. If the line does not collide with any obstacles, then u_1 is connected to $w(3)$, and this procedure continues in the same way until $j = 12$. When $j = 12$, u_1 is connected to $w(12)$; in this case the line between these two points collides with obstacles, as shown in Figure $10a$. Therefore, u_2 of the first line segment is placed in $w(11)$. For the second line segment U_2, the left-hand end u_1 is coincides at u_2 of the first line segment. In order to find u_2 of the second line segment, the PEM connect u_1 with $w(j), (j = 12...14)$, iteratively. Therefore, u_1 is connected with $w(12)$, $w(13)$, and $w(14)$ one after another, and the lines between u_1 and these points are check for feasibility. As shown in Figure $10a$, these line segments did not collide with obstacles. Therefore, u_2 of the second line segment is placed in C_g. The total length of the shortest path U is calculated by summing the length of all the line segments $U(i)$ in the path between C_s and C_g, as follows:

$$U = \sum_{i=1}^{I} (sqrt(u_{1x}(i) - u_{2x}(i))^2 + (u_{1y}(i) - u_{2y}(i))^2) \tag{12}$$

where I represents the number of the line segment, which is equal to 2 in this example. $u_{1x}(i)$, $u_{2x}(i)$, $u_{1y}(i)$, $u_{2y}(i)$ represent the coordinates of the line segment $U(i)$. The general procedure of PEM is illustrated in Algorithm 5.

4.4. Path smoothing using interpolation technique

The path we obtained so far may contain sharp turns. This goes against many real-world applications where smooth paths are preferred [43]. Moreover, the robot may not be able to make a sharp turn due to its momentum [10, 14]. Finally, the cubic spline interpolation is adopted to generate a continuous smooth path that connects the starting point to the goal point. The spline method is one of the most efficient curve

Algorithm 5 PEM method

Inputs:

C_{obs}, and $w(j), (j = 1...J)$

$j \leftarrow 1$

while $j \leq J$ **do**

  $u_2 = w(j)$

  check the line U_i for feasibility between u_1 and u_2

  if $U_{(i)}$ collide with C_{obs} **then**

    store $u_1, u_2 = w(j-1)$

    $u_1 \leftarrow w(j-1)$

  end if

  $j \leftarrow j+1$

end while

insert C_s and C_g to the beginning and to the end of the new waypoints list.

interpolating methods which has many applications in robotics, signal processing, and computer graphics [18, 14].

From the Figure 10c, consider the generated shortest path by using PEM. The path consists of two line segments U_1 and U_2 between C_s and C_g in the form of X and Y vectors, where X=[$x_1 \quad x_2 \quad x_3$] and Y=[$y_1 \quad y_2 \quad y_3$]. We use the cubic spline interpolation to calculate the spline for three waypoints ($w = 3$). Therefore, a new vector t of about 200 points is generated between the starting point at (x_1, y_1) and the goal point at (x_3, y_3). Vectors of interpolated values x_{sp} and y_{sp} are calculated based on equations x_{sp}=$Spline(t_n, x, t)$ and y_{sp}=$Spline(t_n, y, t)$, where $t_n = [1 \quad 2 \quad 3]$. As illustrated in the Figure 10c, the constructed path passes smoothly through the waypoints thus eliminating the sharp turn.

5. Simulations

In this study, the proposed methods are implemented in $MATLAB$ and run on a laptop with Intel(R) core(TM) i5-2450M CPU 2.5GHz 6GB RAM. The performances of the developed method have been tested on many different workspace scenarios with different obstacle layouts. In all the tested scenarios, the workspace size and the number of obstacles scattered in the workspace have been varied. Additionally, the starting C_s and the goal C_g points have been positioned in different locations in the free space C_{free}. An example of three workspace scenarios are shown in Figure 4. The proposed method is examined to find an optimal or near optimal path from C_s to C_g. The simulation results of BNM and PEM are presented in Sections 5.1 and 5.2, respectively. Additionally, the performance of the proposed method is compared with the other path planning methods in Section 5.3. Then, the proposed method is applied to the multi-robot path-planning problem, and the results presented in Section 5.4.

5.1. Simulation results of BNM

This section presents the results of the BNM method for generating the IFP between C_s and C_g for all the workspace scenarios shown in Figure 4. The achieved result of the IFP is represented by a set of waypoints $w(j), (j = 1 \to J)$. Each new position of waypoint $w(j + 1)$ is allocated after the current waypoint position $w(j)$ on the IFP, where J represents the time in which the robot is reaching the goal point.

The simulation results for all the tested scenarios are presented in Figure 11, and the summary of the obtained results is provided in Table 3. From the figure, it is observed that the obtained IFP allows the robot to move from C_s to C_g without colliding with any obstacle in the workspace. The waypoints of the path are represented by red circle objects and for better clarity, these waypoints are connected into a continuous path. As it can be seen from the results, the BNM method is able to overcome the local minima problem. From Table 3, we can clearly see that the developed method provides the collision-free path for the robot in short time, in particular for the high complex environment shown in Figure 11c. As presented in the third scenario, the total computing time to find a IFP is less than 1.1 $second$.

The results show that the BNM method has been well applied to generate the IFP for a mobile robot, and also this method has achieved good results in terms of safety and short computational time. However, the generated path is not optimal in terms of the total path length. In order to reduce the overall path length, a new method called PEM is developed as explained in Section 4.3, and the results are presented in the

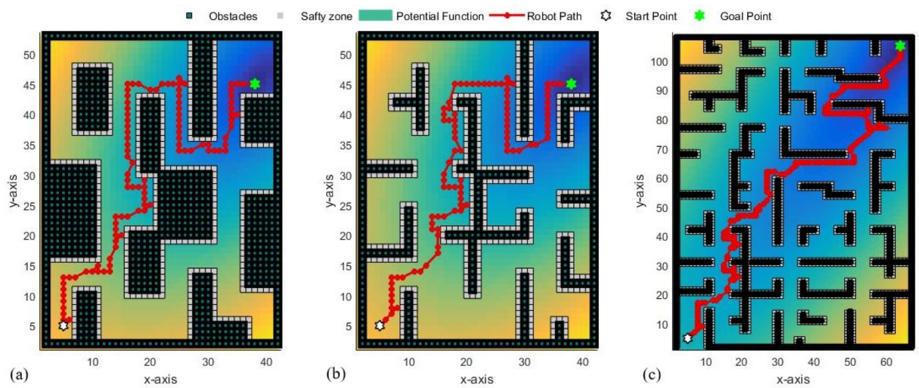

Figure 11: The simulation results to generate IFP for all three workspace scenarios using BNM.

Table 3: The total computational time and path length of the IFP and final path by using BNM and PEM

Workspace	Total Computational Time [s]		Total Path Length [unit]	
$No.$	IFP	Final Path	IFP	Final Path
1	0.955601	1.043110	112.9662	83.4301
2	0.896196	1.004691	110.1285	81.0895
3	1.100025	1.138494	201.3783	146.3850

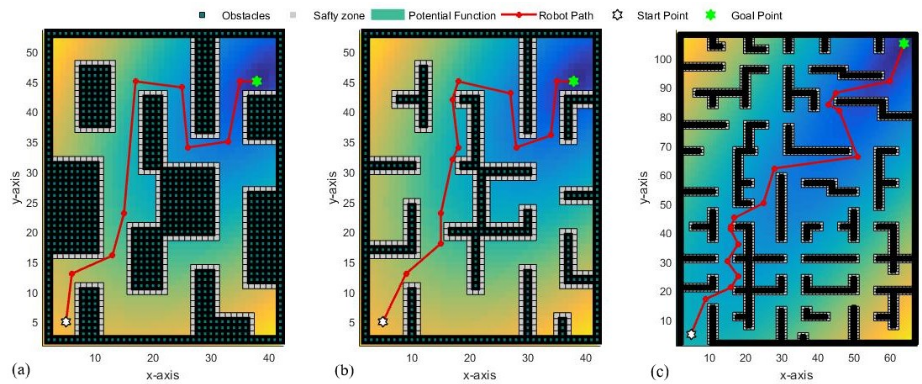

Figure 12: The simulation results to generate an optimal or near-optimal path for all three workspace scenarios using PEM.

following subsection.

5.2. Simulation results of PEM

This section presents the obtained results of the PEM method to find optimal or near-optimal path for the three workspace scenarios. The best-obtained results are presented in Figure 12, and the results of computational time and path length for all the scenarios are provided in Table 3. As shown in the Figure 12, the PEM method can find the collision-free path that covers the least number of waypoints, where the solid red lines represent the best solution found so far. Additionally, Table 3 revealed that the total path length for all the three designed workspaces is significantly reduced, and the percentage of enhancement of the path length for all the three scenarios are 26.2%, 26.4% and 27.3%, respectively.

Obviously, the geometrical complexity of the workspace is the main factor affecting the computational time. However, the results show that the computational time required to obtain the IFP and the final path by using $BNM\&PEM$ is not increased significantly with the growing complexity of the workspace. For example, the size of the workspace is increased 3.2, times and the number of obstacles are increased 2.6 times from the second (see Figure 12b) to the third scenario (see Figure 12c), accordingly the total computational time to find the IFP and the final path is increased only by about 1.5 and 1.4 times, respectively (see Table 3). On the other hand, in the second

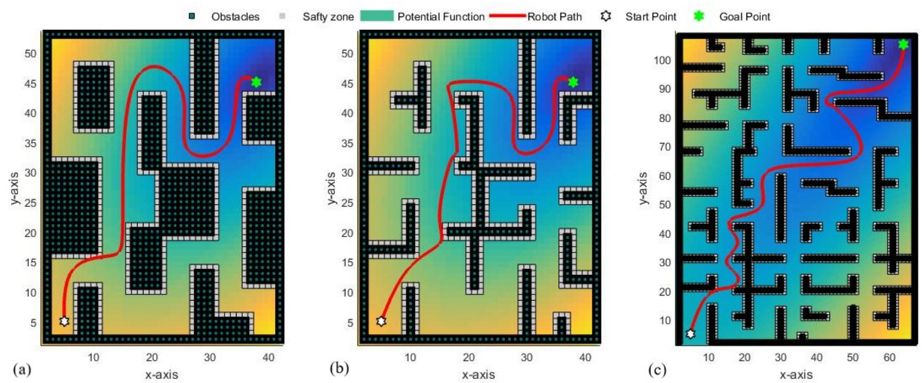

Figure 13: The simulation results to generate a smooth path by using spline method for all three workspace scenarios.

(see Figure 12b) and first (see Figure 12a) scenario, the required computational time to find the IFP and the final path is increased by 6.6% and 3.8% respectively, as the number of obstacles increased by 123.2% for the same workspace. This is because, for each iteration t during the search process, all obstacles in the workspace are examined for possible collisions with the direct path from the current $p_1(t)$ to updated $p_2(t)$ nodes' location.

In the simulations results presented in Figure 12, we observe that the proposed method generates a path consisting of straight lines between waypoints with sharp turns. In real applications when the robot follows a path in the workspace, it may not be able to make a sharp turn and also it is not the safest path for the robot. In order to improve the path with respect to the robot dynamics, the proposed method applied $MATLAB$ cubic spline to construct a continuous smooth path that connects the starting point to the end point for all three designed workspaces, and the results are presented in Figure 13.

The results demonstrate that the spline method can be used to generate a continuous smooth path to eliminate sharp turns. On the other hand, the cumulative length of the smooth paths shown in Figure 13 are longer than the cumulative length of the line-segment path presented in Figure 12, and the length of the paths are increased by 7%, 4.4%, and 4.5% for all three scenario, respectively.

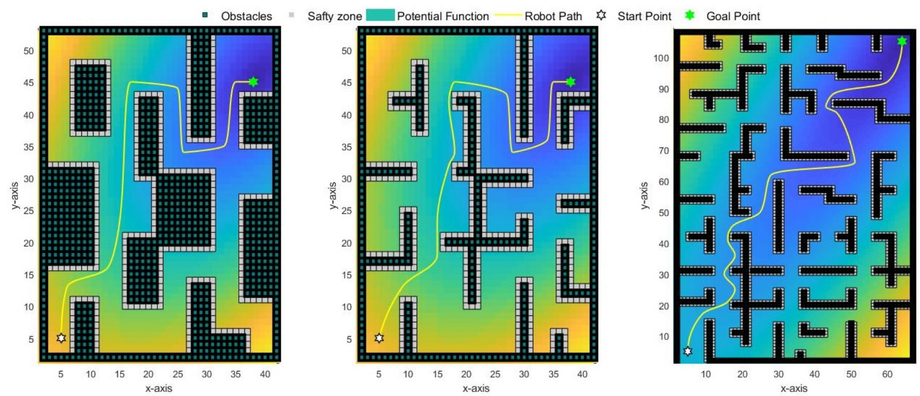

Figure 14: The simulation results to generate a smooth path by using $PCHIP$ for all three workspace scenarios.

The aim of the cubic spline method is to generate a smooth path for an initial feasible path that connects the starting point to the goal point. However, in some cases, the constructed smooth path can bring the robot close to the safety zone around the obstacles or the robot collides with the safety zone, which is undesirable in practice. To avoid the possibility of overlapping the paths traced by the robot with safety zone, additional waypoints can be inserted between the original waypoints until no safety zone or obstacles were found along the resulting path, as explained [44]. Alternatively, Piecewise Cubic Hermite Interpolating Polynomial ($PCHIP$) can be used to construct a continuous smooth path, as illustrated in [7, 45]. The $PCHIP$ is like cubic spline interpolation, but $PCHIP$ interpolation ensures a shape-preserving interpolation and avoiding the overshoots and oscillations that could arise from spline interpolation. The generated path from the X and Y vectors of the waypoints w is a zigzag line; we generate a new vector x_i of about 1000 points from start point to goal point. $y_i = PCHIP(X, Y, x_i)$ returns a vector of interpolated values y_i containing elements corresponding to the elements of x_i. Resulting x_i versus y_i give the smoothed path. The obtained results of smoothed paths by using $PCHIP$ is shown in Figure 14. The length of the paths are increased by 4.4%, 2.4%, and 2.6% for all three scenario, respectively. We can see the difference between the interpolation results produced by $PCHIP$ and cubic spline in Figures 13 and 14.

In the proposed approach, the grid-based method is used to create a workspace environment. In this method, the workspace environment is divided into a number of small square grid cells of the same size (1×1 $unit$). Each grid cell can either correspond to a navigable area or to a space occupied by obstacles. Different obstacle shapes can be generated, such as circular or non-convex obstacles, by approximating the shape of the obstacles and dividing it into square grid cells. The completeness of the obstacles' shape depends on the resolution of the grid environment. Figures $15a$ and e show two examples of different workspace scenarios. In these scenarios, the workspace consists of (50×50) grid cells, and the number of the obstacles in the workspace is 312 and 316 grid cells, respectively. The starting C_s and goal C_g points are positioned in the free space C_{free} at (5,5) and (45,45), respectively. The proposed method is examined to find an optimal or near optimal path from C_s to C_g. The simulation result of the BNM method for generating the IFP between C_s and C_g is presented in Figures $15b$ and f. From the figures, it is observed that the obtained IFP can successfully drive the robot toward the goal while avoiding obstacles in the highly complex environment. The robot location is represented by red circles object at each iteration. The obtained results of the PEM method to find optimal or near-optimalpath are presented in Figures $15c$ and g. As shown in the figures, the PEM method can find the short path, where the solid red lines between C_s and C_g represent the best solution found so far. Additionally, the generated path from the PEM is smoothed by using the cubic spline method and the result are presented in Figures $15d$ and h.

The proposed method can easily be extended to include altitude as a third coordinate to solve the path planning problems in three-dimensional ($3D$) workspace. The method was implemented with several $3D$ scenarios and the results were found to be satisfactory. An example of the workspace scenario is presented in Figure 16. The workspace is discretized into uniform cubic grid cells ($1 \times 1 \times 1$ $unit$), and the generated path is a sequence of cubic cells in a $3D$ grid model.

5.3. Comparison results

This section presents the performance evaluation of the $BNM\&PEM$ method in comparison with PSO, $A - Star$, and APF. Therefore, a simple example of a

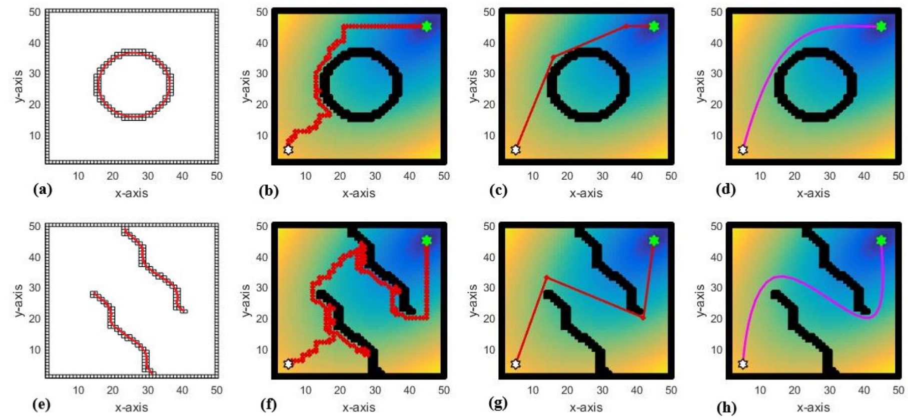

Figure 15: Examples of grid cells with obstacles (a and e) simulation result of BNM (b and f), PEM (c and g) and cubic spline method (d and h)

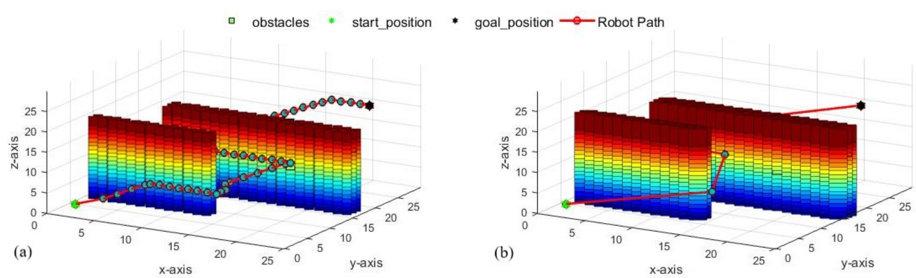

Figure 16: The simulation result of the BNM (a) and PEM (b) to solve the path planning problems in three-dimensional ($3D$) workspace.

Table 4: A summary of the obtained results of the computational time and path length by using BNM, PSO, $A - Star$, and APF.

$Method$	Total Computational Time [s]	Total Path Length [unit]
BNM	0.82	53.49
PSO	1.51	57.70
$A - Star$	2.57	57.11
APF	0.66	61.00

$2D$ workspace is designed as shown in Figures 17. The size of the workspace is set to 43×68, where the space occupied by obstacles C_{obs} consists of 1078 grid cells and the obstacle-free space C_{free} consists of 1846 grid cells. After constructing the workspace with obstacles, all four methods namely $BNM\&PEM$, PSO, $A - Star$, and APF are used to find the shortest path between C_s at $(8, 10)$ and C_g at $(32, 56)$. The obtained results of $BNM\&PEM$, PSO, $A - Star$, and APF are shown in Figures 17a, 17b, 17c, and 17d, respectively. A summary of the obtained results of the computational time and path length is provided in Table 4. By comparing the results presented in Table 4, it can be seen that the proposed method is able to find the shortest path within less than one second, and it requires less than 55% and 32% of the computational time to find shortest path by using PSO and $A - Star$, respectively. In terms of the total path length, the shortest path achieved by $BNM\&PEM$ is about 7.2% and 6.3% shorter than the path length generated by PSO and $A - Star$, respectively. In this workspace, the computational time required to find the shortest path by using APF is lower by 20% compare to $BNM\&PEM$. In contrast, the shortest path achieved by $BNM\&PEM$ is 12% shorter than the path length generated by APF.

The PEM method can also be used to optimize the paths obtained by using PSO, $A - Star$, and APF as shown in Figures 17b, 17c, and 17d, respectively. The best-obtained results are presented in Figures 18b, 18c, and 18d, respectively. As shown in the figures, the PEM method can find the collision-free path that covers the least number of waypoints, where the solid red lines represent the best solution found so far. The results revealed that the length of the paths obtained from PSO, $A - Star$, and

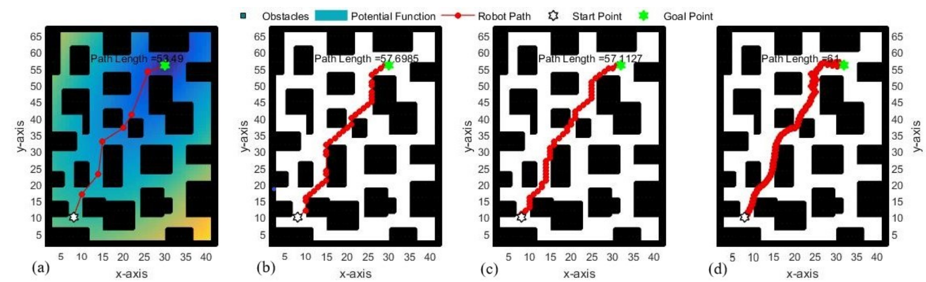

Figure 17: The simulation results of the $BNM\&PEM$ (a), PSO (b), $A-Star$ (c), and APF (d) to solve the path planning problem in two-dimensional $(2D)$ workspace.

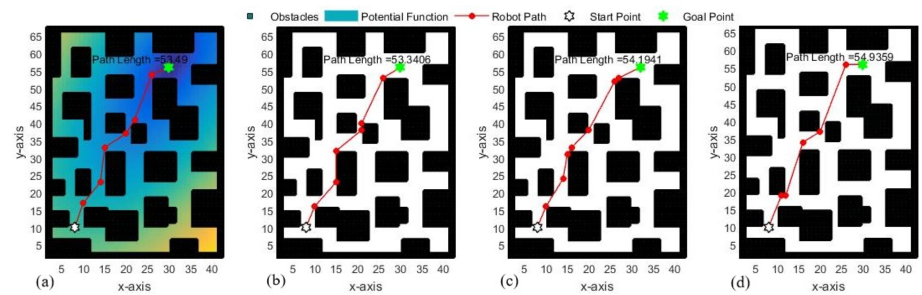

Figure 18: Simulation results for generating an optimal or near-optimal path for BNM (a), PSO (b), $A-Star$ (c), and APF (d) using PEM method.

APF reduced by 7.6%, 5.1% and 9.9%, respectively. Furthermore, the cubic spline interpolation is used to generate a continuous smooth path that connects the starting point to the end point bu using BNM, PSO, $A-Star$, and APF, and the results are presented in Figures 19a, 19b, 19c, and 19d, respectively.

In order to make an extra comparison and to demonstrate the ability of BNM for solving robot path planning problem in the workspaces that have previously been used in [32, 46, 18, 47], a $2D$ workspace is created as shown in Figure 20. The size of the workspace is set to 67×67, where the space occupied by obstacles C_{obs} consists of 1520 grid cells and the obstacle-free space C_{free} consists of 2969 grid cells. After constructing the workspace with obstacles, the proposed method is used to generate a IFP (Figure 20a), shortest path (see Figure 20b), and smooth path (see Figure 20c) from the C_s at $(64, 4)$ to the C_g at $(4, 64)$. The obtained computational results of the

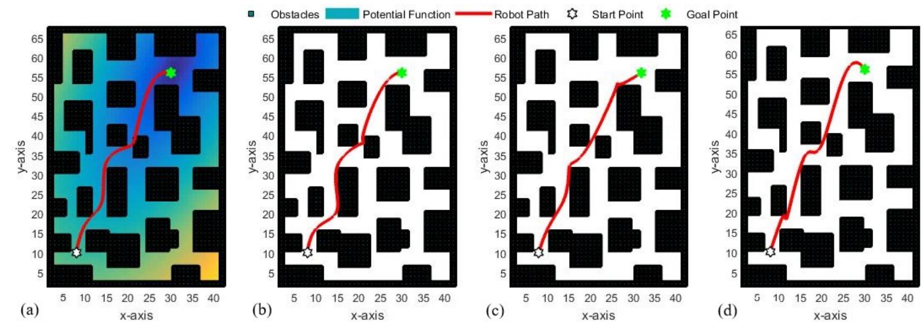

Figure 19: Simulation results for generating an optimal or near-optimal path for BNM (a), PSO (b), $A - Star$ (c), and APF (d) using cubic spline method.

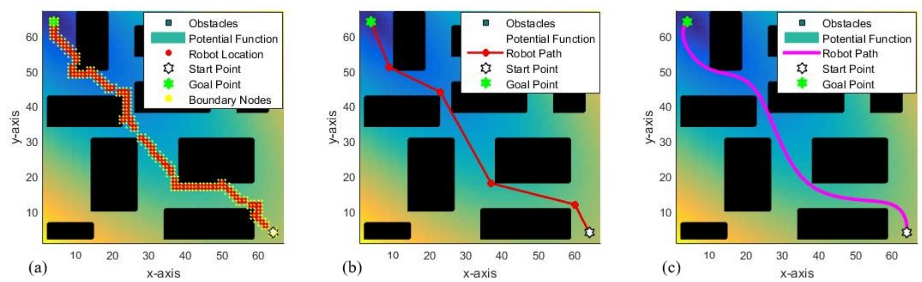

Figure 20: The simulation result of the BNM (a), PEM (b), and cubic spline (c) method for solving robot path planning problem in the workspace that is previously have been used in [32, 46, 18, 47].

BNM and an improved GA is provided in Table 5. By comparing the obtained results of BNM with an improved GA in the previous studies (see Table 5), it is observed that the computational time of the proposed method is remarkably reduced.

The comparison results demonstrate the effectiveness and efficiency of the proposed method for solving robot path planning problem. For the comparison test, a simple workspace scenario has been selected because the required time to find optimal or near-optimal path grows exponentially as the complexity of the path-planning problem increases (see Section 1, point 4) even in some circumstance the path planning methods cannot find a feasible path, whereas the proposed method BNM solve these problems.

In order to validate the proposed method $BNM\&PEM$, and compare its performance with the $A - Star$, PSO, and GA, a $2D$ environment of the static robot's

Table 5: The total computational time required to find shortest path using BNM and improved GA.

$Method$	Total Computational Time [s]
Improved GA Ref [46]	1.03
Improved GA Ref [18]	4.07
Improved GA Ref [32]	1.68
Improved GA Ref [47]	0.85
BNM	0.964

workspace is created. The size of the workspace is set to 60×60, and the space occupied by obstacles C_{obs} consists of 136 grid cells. Thereafter, all methods are implemented simultaneously to find a feasible path for 1000 independent runs. At each time in this test, the starting point C_s, the goal point C_g, and obstacles are placed randomly in the working environment, each random placement of the obstacles led to different workspace layout. Two measures of evaluation are used for comparison among path planning methods: the length of the obtained feasible path as well as the execution time of the method. The mean and the standard deviation (Std) of the computational time and the path length are calculated and presented in Table 6. The results shown in the table reveal that the proposed method achieved the best solution within a reasonable computational time. Moreover, the mean value of the computational time to find a feasible path is decreased significantly compared with other path planning methods. In comparison with PSO and GA, the proposed method showed noticeable improvement in terms of the path length. Additionally, the mean value of the path length obtained by the proposed method is smaller than that obtained with PSO and GA by $\%11.73$ and $\%7.3$, respectively. However, the mean value of the path length generated by $BNM\&PEM$ is slightly larger than that obtained with $A - Star$ by $\%2.21$. The PSO method had the least variance of the computational time, and GA better than the other method in terms of variance of the path length. The comparative study shows that heuristic algorithms did not yield optimal results, and the results agree with [48]. The graphical representation of the simulation results of all methods is illustrated in Figure 21.

Table 6: Mean and standard deviation of the computational time and path length for 1000 independent runs to find feasible path using proposed method with PSO, GA, and $A - Star$,

Methods	Computational Time, CT [s]		Path Length, PL [unit]	
$Methods$	$Mean_{CT}$	Std_{CT}	$Mean_{PL}$	Std_{PL}
PEM	0.0142	0.0072	30.7710	15.9340
$A - Star$	0.0489	0.0640	30.0907	14.6124
PSO	0.0217	0.0052	34.3788	15.2495
GA	0.1188	0.2122	33.0144	11.1463

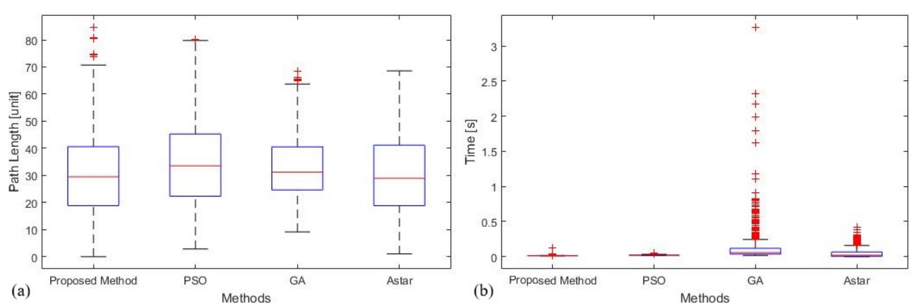

Figure 21: Performance of final evaluations for 1000 independent runs to find feasible path using proposed method with PSO, GA, and $A - Star$, the obtained results for the path length data presented in (a) and the computational time data presented in (b).

5.4. Path planning of a Multi-robot system

In this section, the implementation of the proposed method for collision avoidance in multi-robot systems is presented. We conducted different simulations with different multi-robot system parameters, i.e. the number of robots, the initial and goal positions for each robot, and the positions of static obstacles. Figures 22 shows an example of the simulation results for a multi-robot system, in which there are 4 robots, 4 different start C_s and goal C_g positions corresponding to each robot, and 304 static obstacles. The problem formulation is to determine the path of each robot in the simulated environment by avoiding the collision with static obstacles and other moving robots in the system. Each robot moves from a starting position C_s, through all the intermediate waypoints w until it reaches the goal position C_g. Each robot uses the BNM to find IFP to move from C_s to C_g in the workspace without colliding with any obstacle (see Figures 22a and d). IFP is generated from a set of waypoints w that the robot visits before reaching the final destination point. Then for each robot, the PEM method is used to reduce the number of waypoints of the IFP between C_s and C_g to obtain an optimal or close-to-optimal path (see Figures 22b and e). Finally, the cubic spline interpolation is applied to construct a continuous smooth path that connects the starting position to the goal position (see Figures 22c and f). The simulation results show that all robots reached to their final destination positions successfully without any collision with either static obstacles or other robots.

6. Experimental results

In this section, a real robot is employed to test the performance of the developed method $BNM\&PEM$ and illustrate how the robot can navigate along a collision-free path. An e-puck robot, shown in Figure 23a, is used for the experimental test, and the experimental set-up shown in Figure 23b. First, the developed method is used to generate a collision-free path to direct the robot to move among the static obstacles from the starting point C_s toward the goal point C_g as shown in Figure 24a. As illustarted in the figure, the waypoints w are represented by red circle objects, and the obtained shortest path is represented by a thick red dashed-line. The obtained shortest path by

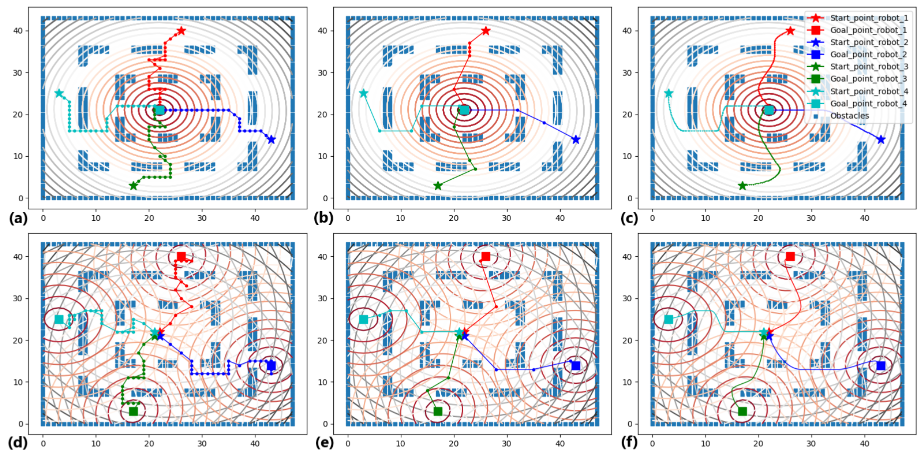

Figure 22: The simulation results for the multi-robot path planning problem.

$BNM\&PEM$ consists of the waypoints, $w(j), (j = 1...J)$, $J = 5$, whose x, y coordinates are known with respect to the simulated environment. Based on the generated data of the obtained path, the e-puck robot motion data is determined. Next, the e-puck robot is connected to the computer via Bluetooth and the generated motion data are transmitted to the robot via a toolbox $ePic(v2.1.2)$, where $ePic(v2.1.2)$ is used to control e-puck in $MATLAB$. Let w_{1x} and w_{1y} be the centroid coordinates of the first waypoint w_1 of the generated path, and w_{2x} and w_{2y} those of the centroid of the second waypoint w_2. Then, the orientation of the robot is calculated in $MATLAB$ by using $atan2(w_{2y}$-w_{1y}, w_{2x}-$w_{1x})$. Subsequently, in order to move the e-puck robot towards the second waypoint w_2, the angle of the w_2 with respect to the robot is calculated. Thereafter, the e-puck robot starts to move from w_1 to w_2 and so on until it reaches the goal point. Figures 24($b \rightarrow f$) show the robot's positions at different locations in the robot's working environment during the experimental test. The test results demonstrate that the proposed method is able to generate the shortest path to direct the e-puck robot to final destination point.

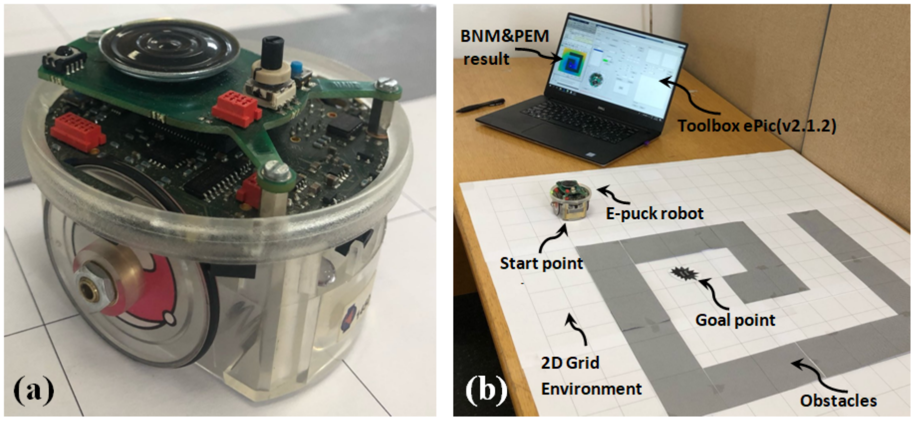

Figure 23: The robot used in the experiment (a) and the experimental set-up (b)

7. Conclusions and Future Works

In this paper a novel off-line path planning method called Boundary Node Method is developed for solving the path planning problem of a mobile robot in a two-dimensional working environment. The developed method is used to find collision-free path for a mobile robot through a sequence of way-points that the robot has to traverse from the starting point to the goal point without colliding with any obstacles. The concept involved in the developed method is simple and can be applied in a grid environment efficiently. Additionally, this method does not work through random operations and there is no uncertainty in generating points, which leads to finding the final solution for the problem without variation in solution. Moreover, this method uses an optimization technique based on the lowest potential value to accelerate the robot to find the path safely and quickly in reasonable time. The simulation results show that the Boundary Node Method can successfully find an initial feasible path, and generates a safe path for a mobile robot to navigate in a complicated environment within a relatively short time. And also the computational time required to find shortest path does not increase significantly with the increase of the environment's complexity. Furthermore, the results have verified that the boundary node method solves the local minima problem effectively. An additional method that we called Path Enhancement Method, has been applied on top to build an optimal or close-to-optimal collision-free path by

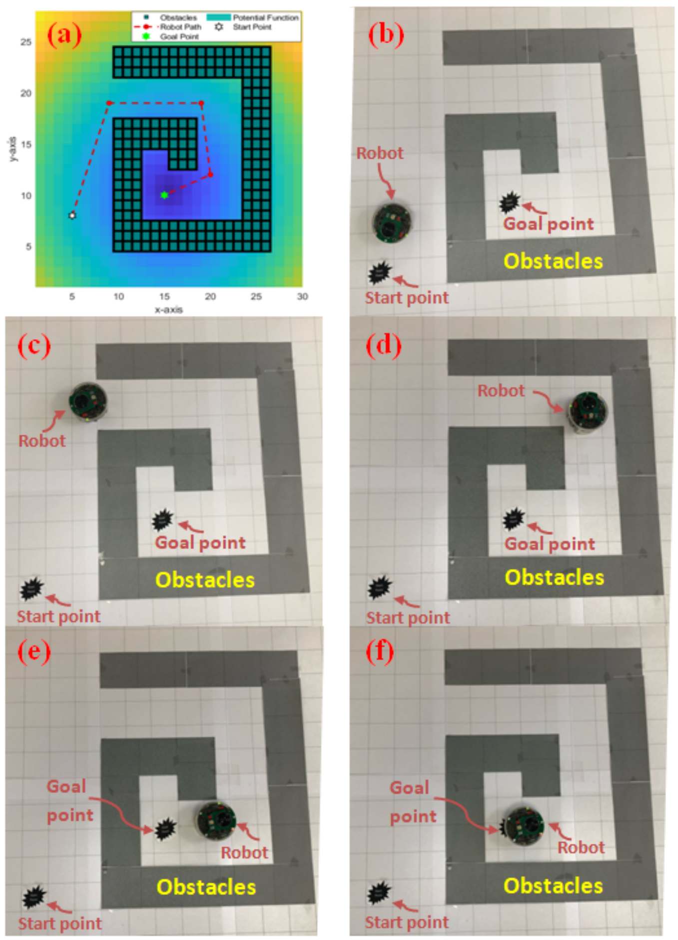

Figure 24: The simulation and experimental results. (a) the simulation results to generate IFP by using $BNM\&PEM$. (b) to (f) locations of the e-puck robot at different waypoints in the robot's working environment.

reducing the number of waypoints and the overall path length. In order to validate the performance of the developed method in comparison with existing path planning methods, several different scenarios with different complexity have been tested. The comparison reveals that the Boundary Node Method can solve the path planning problem effectively and efficiently in terms of the computational times and the path length. Finally, the cubic spline method has been used to generate a continuous smooth path that connects the starting point to the end point.

The developed method is used to solve the multi-robot path planning problem, and the simulation results showed that the developed method effective and useful for collision avoidance in multi-robot systems. Additionally, the performance of the developed method for generating a collision-free path is tested on a real robot. The experimental test shows that the proposed method is able to generate shortest path, and direct the real robot to the final destination point.

In the future work, we will address a number of research issues related to autonomous navigation of mobile robots in unknown environments, where the deployed robot does not have full knowledge about its environment. Another possible direction will explore an extension to the proposed method in order to deal with a dynamic scene.

Acknowledgements

The authors gratefully acknowledge Sardinia Regional Government for the financial support (Convenzione triennale tra la Fondazione di Sardegna e gli Atenei Sardi Regione Sardegna – L.R. 7/2007 annualità 2016 – DGR 28/21 del 17.05.2016, CUP: F72F16003030002). Moreover, this work has been supported by Sardinia Regional Government (P.O.R. Sardegna F.S.E. Operational Programme of the Autonomous Region of Sardinia, European Social Fund 2014-2020 - Axis IV Human Resources, Objective l.3, Line of Activity l.3.1.)

References

References

[1] N. Leena, K. Saju, A survey on path planning techniques for autonomous mobile robots, IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE) 8 (2014) 76–79 (2014).

[2] J. Han, Y. Seo, Mobile robot path planning with surrounding point set and path improvement, Applied Soft Computing 57 (2017) 35–47 (2017).

[3] P. Victerpaul, D. Saravanan, S. Janakiraman, J. Pradeep, Path planning of autonomous mobile robots: A survey and comparison, Journal of Advanced Research in Dynamical and Control Systems 9 (2017) 1535–1565 (2017).

[4] N. Chao, Y.-k. Liu, H. Xia, A. Ayodeji, L. Bai, Grid-based rrt for minimum dose walking path-planning in complex radioactive environments, Annals of Nuclear Energy 115 (2018) 73–82 (2018).

[5] A. Shum, K. Morris, A. Khajepour, Direction-dependent optimal path planning for autonomous vehicles, Robotics and Autonomous Systems 70 (2015) 202–214 (2015).

[6] Y. Zhang, D.-w. Gong, J.-h. Zhang, Robot path planning in uncertain environment using multi-objective particle swarm optimization, Neurocomputing 103 (2013) 172–185 (2013).

[7] A. Bakdi, A. Hentout, H. Boutami, A. Maoudj, O. Hachour, B. Bouzouia, Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control, Robotics and Autonomous Systems 89 (2017) 95–109 (2017).

[8] T. Lozano-Pérez, M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, Communications of the ACM 22 (10) (1979) 560–570 (1979).

[9] M. Brand, M. Masuda, N. Wehner, X.-H. Yu, Ant colony optimization algorithm for robot path planning, in: 2010 International Conference On Computer Design and Applications, Vol. 3, IEEE, 2010, pp. V3–436 (2010).

[10] S. M. LaValle, Planning algorithms, Cambridge university press, 2006 (2006).

[11] J. Rintanen, Introduction to automated planning, Lecture notes of the AI planning course, Albert-Ludwigs-University Freiburg (2006).

[12] C. Lamini, S. Benhlima, A. Elbekri, Genetic algorithm based approach for autonomous mobile robot path planning, Procedia Computer Science 127 (2018) 180–189 (2018).

[13] T. T. Mac, C. Copot, D. T. Tran, R. De Keyser, Heuristic approaches in robot path planning: A survey, Robotics and Autonomous Systems 86 (2016) 13–28 (2016).

[14] H.-C. Huang, C.-C. Tsai, Global path planning for autonomous robot navigation using hybrid metaheuristic ga-pso algorithm, in: SICE Annual Conference (SICE), 2011 Proceedings of, IEEE, 2011, pp. 1338–1343 (2011).

[15] O. Montiel, U. Orozco-Rosas, R. Sepúlveda, Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles, Expert Systems with Applications 42 (12) (2015) 5177–5191 (2015).

[16] M. A. Contreras-Cruz, V. Ayala-Ramirez, U. H. Hernandez-Belmonte, Mobile robot path planning using artificial bee colony and evolutionary programming, Applied Soft Computing 30 (2015) 319–328 (2015).

[17] W. Song, H.-x. LI, Y.-n. ZHANG, Path planning of mobile robot based on genetic bee colony algorithm, DEStech Transactions on Computer Science and Engineering 16 (5) (2016) 615–620 (2016).

[18] A. H. Karami, M. Hasanzadeh, An adaptive genetic algorithm for robot motion planning in 2d complex environments, Computers & Electrical Engineering 43 (2015) 317–329 (2015).

[19] H. Liu, B. Xu, D. Lu, G. Zhang, A path planning approach for crowd evacuation in buildings based on improved artificial bee colony algorithm, Applied Soft Computing (2018).

[20] J. K. Goyal, K. Nagla, A new approach of path planning for mobile robots, in: Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on, IEEE, 2014, pp. 863–867 (2014).

[21] Y. Hu, S. X. Yang, A knowledge based genetic algorithm for path planning of a mobile robot, in: Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, Vol. 5, IEEE, 2004, pp. 4350–4355 (2004).

[22] B. Fu, L. Chen, Y. Zhou, D. Zheng, Z. Wei, J. Dai, H. Pan, An improved a* algorithm for the industrial robot path planning with high success rate and short length, Robotics and Autonomous Systems (2018).

[23] N. A. Shiltagh, L. D. Jalal, Optimal path planning for intelligent mobile robot navigation using modified particle swarm optimization, International Journal of Engineering and Advanced Technology 2 (4) (2013) 260–267 (2013).

[24] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, L. Jurišica, Path planning with modified a star algorithm for a mobile robot, Procedia Engineering 96 (2014) 59–69 (2014).

[25] A. K. Guruji, H. Agarwal, D. Parsediya, Time-efficient a* algorithm for robot path planning, Procedia Technology 23 (2016) 144–149 (2016).

[26] S. S. Ge, Y. J. Cui, New potential functions for mobile robot path planning, IEEE Transactions on robotics and automation 16 (5) (2000) 615–620 (2000).

[27] G. E. Jan, K. Y. Chang, I. Parberry, Optimal path planning for mobile robot navigation, IEEE/ASME Transactions on mechatronics 13 (4) (2008) 451–460 (2008).

[28] R. Kala, A. Shukla, R. Tiwari, Robotic path planning in static environment using hierarchical multi-neuron heuristic search and probability based fitness, Neurocomputing 74 (14-15) (2011) 2314–2335 (2011).

[29] M. Davoodi, F. Panahi, A. Mohades, S. N. Hashemi, Clear and smooth path planning, Applied Soft Computing 32 (2015) 568–579 (2015).

[30] P. Raja, S. Pugazhenthi, Optimal path planning of mobile robots: A review, International Journal of Physical Sciences 7 (9) (2012) 1314–1320 (2012).

[31] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, P. Feyzeau, Path planning: A 2013 survey, in: Industrial Engineering and Systems Management (IESM), Proceedings of 2013 International Conference on, IEEE, 2013, pp. 1–8 (2013).

[32] A. Tuncer, M. Yildirim, Dynamic path planning of mobile robots with improved genetic algorithm, Computers & Electrical Engineering 38 (6) (2012) 1564–1572 (2012).

[33] Z. Bi, Y. Yimin, Y. Wei, Hierarchical planning approach for mobile robot navigation under the dynamic environment, in: Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on, IEEE, 2008, pp. 372–376 (2008).

[34] M. P. Garcia, O. Montiel, O. Castillo, R. Sepúlveda, P. Melin, Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation, Applied Soft Computing 9 (3) (2009) 1102–1110 (2009).

[35] B. Patle, D. Parhi, A. Jagadeesh, S. K. Kashyap, Matrix-binary codes based genetic algorithm for path planning of mobile robot, Computers & Electrical Engineering 67 (2018) 708–728 (2018).

[36] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, in: Robotics and Automation. Proceedings. 1985 IEEE International Conference on, Vol. 2, IEEE, 1985, pp. 500–505 (1985).

[37] J.-O. Kim, P. K. Khosla, Real-time obstacle avoidance using harmonic potential functions, IEEE Transactions on Robotics and Automation 8 (3) (1992) 338–349 (1992).

[38] A. H. Qureshi, Y. Ayaz, Potential functions based sampling heuristic for optimal path planning, Autonomous Robots 40 (6) (2016) 1079–1093 (2016).

[39] L. Janson, B. Ichter, M. Pavone, Deterministic sampling-based motion planning: Optimality, complexity, and performance, The International Journal of Robotics Research 37 (1) (2018) 46–61 (2018).

[40] W. Wang, L. Zuo, X. Xu, A learning-based multi-rrt approach for robot path planning in narrow passages, Journal of Intelligent & Robotic Systems 90 (1-2) (2018) 81–100 (2018).

[41] E. Di Mario, Z. Talebpour, A. Martinoli, A comparison of pso and reinforcement learning for multi-robot obstacle avoidance, in: 2013 IEEE Congress on Evolutionary Computation, Ieee, 2013, pp. 149–156 (2013).

[42] Z. Wang, J. Cai, Probabilistic roadmap method for path-planning in radioactive environment of nuclear facilities, Progress in Nuclear Energy 109 (2018) 113–120 (2018).

[43] K. Yakovlev, E. Baskin, I. Hramoin, Grid-based angle-constrained path planning, in: Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz), Springer, 2015, pp. 208–221 (2015).

[44] A. Ravankar, A. Ravankar, Y. Kobayashi, Y. Hoshino, C.-C. Peng, Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges, Sensors 18 (9) (2018) 3170 (2018).

[45] J. M. Hyman, Accurate monotonicity preserving cubic interpolation, SIAM Journal on Scientific and Statistical Computing 4 (4) (1983) 645–654 (1983).

[46] Q. Li, W. Zhang, Y. Yin, Z. Wang, G. Liu, An improved genetic algorithm of optimum path planning for mobile robots, in: Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on, Vol. 2, IEEE, 2006, pp. 637–642 (2006).

[47] C. Liu, X. Yan, C. Liu, G. Li, Dynamic path planning for mobile robot based on improved genetic algorithm, Chinese Journal of Electronics 19 (2) (2010) 2010–2014 (2010).

[48] I. Chaari, A. Koubaa, H. Bennaceur, A. Ammar, M. Alajlan, H. Youssef, Design and performance analysis of global path planning techniques for autonomous mobile robots in grid environments, International Journal of Advanced Robotic Systems 14 (2) (2017) 1729881416663663 (2017).