Pertanika J. Sci. & Technol. 32 (5): 2327 - 2342 (2024)

PERTANIKA JOURNALS

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Smoothing RRT Path for Mobile Robot Navigation Using Bio-inspired Optimization Method

Izzati Saleh[1], Nuradlin Borhan[1] and Wan Rahiman[1,2,3]*

[1]School of Electric & Electronic Engineering, Universiti Sains Malaysia Engineering Campus, 14300 Nibong Tebal, Pulau Pinang, Malaysia
[2]Cluster of Smart Port and Logistics Technology (COSPALT), Universiti Sains Malaysia Engineering Campus, 14300 Nibong Tebal, Pulau Pinang, Malaysia
[3]Daffodil Robotics Lab, Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh

ABSTRACT

This research addresses the challenges of using the Rapidly Exploring Random Tree (RRT) algorithm as a mobile robot path planner. While RRT is known for its flexibility and wide applicability, it has limitations, including careful tuning, susceptibility to local minima, and generating jagged paths. The main objective is to improve the smoothness of RRT-generated trajectories and reduce significant path curvature. A novel approach is proposed to achieve these, integrating the RRT path planner with a modified version of the Whale Optimization Algorithm (RRT-WOA). The modified WOA algorithm incorporates parameter variation ($\vec{C}$) specifically designed to optimize trajectory smoothness. Additionally, Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) instead of conventional splines for point interpolation further smoothes the generated paths. The modified WOA algorithm is thoroughly evaluated through a comprehensive comparative analysis, outperforming other popular population-based optimization algorithms such as Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Firefly Algorithm (FA) in terms of optimization time, trajectory smoothness, and improvement from the initial guess. This research contributes a refined trajectory planning approach and highlights the competitive advantage of the modified WOA algorithm in achieving smoother and more efficient trajectories compared to existing methods.

Keywords: Bio-inspired optimization, mobile robot navigation, obstacle avoidance, optimization, path planning, path smoothing, RRT

ARTICLE INFO

Article history:
Received: 15 November 2023
Accepted: 28 February 2024
Published: 26 August 2024

DOI: https://doi.org/10.47836/pjst.32.5.22

E-mail addresses:
izzatisaleh@student.usm.my (Izzati Saleh)
nuradlinnadhirahb@student.usm.my (Nuradlin Borhan)
wanrahiman@usm.my (Wan Rahiman)
* Corresponding author

ISSN: 0128-7680
e-ISSN: 2231-8526

INTRODUCTION

There are four navigation difficulties in robotics: sensing, localization, motion control, and path planning. Path planning may be argued to be the most significant element for navigation procedures. It is the process of determining a collision-free path in a given environment, which is often cluttered in the real world (Alam & Rafique, 2015; Dao et al., 2016; Karur et al., 2021).

Since mobile robots are utilized in a variety of applications, researchers have devised ways to effectively adapt to their needs and overcome some of the key obstacles encountered while implementing fully or partially autonomous navigation in a cluttered environment (Dosoftei et al., 2021; Galli et al., 2017; Mac et al., 2016; Zhou et al., 2019). In order to simplify the path planning problem and ensure that the robot runs/moves smoothly in a cluttered environment while avoiding obstacles, the configuration space must be matched with the algorithm. Multiple path-planning and path-finding techniques exist, with their usefulness depending on the system's kinematics, the environment's dynamics, the computational capabilities of the robot, and the availability of input from sensors and other sources. The trade-offs between algorithm performance and complexity also rely on the use case.

Research in robotics, particularly in the subfield of autonomous navigation, significantly emphasizes the concept of robot path planning. Path planning algorithms build trajectories for robots to follow so that they can safely and effectively reach their destinations.

The Rapidly Exploring Random Tree (RRT) method is one of the most well-known approaches to path design. This sampling-based motion planning algorithm employs a "randomized" technique to explore the space for obstacles and generate a tree of possible paths. RRT was given its name because it uses a "randomized" approach. In the first step of the process, points in the space are sampled at random, and then a tree is constructed by linking the sampled points with edges. The tree's potential to reach new spots that have never been sampled increases as the tree continues to mature. As soon as it reaches its objective, it navigates the tree in reverse, looking for the most direct route from the beginning to the end.

Numerous enhancements to RRT, such as RRT-Connect, RRT*, and Bi-directional RRT (Bi-RRT), have been developed and implemented. RRT* is an extension of RRT that employs heuristics to select nodes more likely to lead to the objective. RRT was originally developed to determine which nodes are most likely to do so. It chooses which nodes are more promising based on a cost function and saves time by avoiding portions of the space that have already been searched. It is done to maximize the likelihood of finding useful information (Jeong et al., 2019; Li et al., 2014; Naderi et al., 2015; Yu & Xiang, 2021). Similarly, Bi-RRT is an RRT extension that generates two distinct trees, one starting at the start point and one starting at the goal point. It then looks for a path between the two

trees to identify the most efficient route from the beginning to the end of the maze. This method is more effective than the RRT technique since it only needs to search half of the space, which significantly increases speed (Xinyu et al., 2019).

Overall, RRT and its variants are powerful and flexible algorithms that can be used in a wide variety of applications. They are a great choice for path planning in unknown or dynamic environments. RRT's strength is that it uses minimal heuristics and arbitrary parameters and does not require state-to-state linkages. This facilitates the application of RRTs to non-holonomic and kinodynamic planning (Lavalle & Kuffner, 2001).

However, RRT may have several downsides. First, RRTs can be challenging to tune properly and may necessitate numerous tries to determine the ideal configuration. Second, RRTs are susceptible to local minima, which means that the algorithm may become stalled in a local optimum and unable to progress to a more optimal solution. Lastly, the path generated by RRT is typically jagged and not smooth (Abbadi & Matousek, 2014).

Paper Scopes and Objectives

The primary objective of this paper is to enhance the smoothness of the trajectory generated by the RRT path planner. A secondary objective is to minimize significant curvature along the path, particularly in challenging areas characterized by tight turns or narrow passages. The final objective of this study is to ensure path validity and collision avoidance by devising an approach that guarantees the generated trajectory remains clear of obstacles within the given map environment.

RRT Path Smoothing Strategies

The RRT path smoothing strategies will be discussed in depth—the strategy employed is population-based optimization. Figure 1 shows the flowchart of the proposed method. Each step in the flowchart will be explained in great detail in the next subsections.

Generating Initial Path Using RRT

Figure 2 shows the sequence of RRT. The x_{tree} is initialized at the start point. While searching for the goal point, the algorithm will sample a random point x_{rand} inside a search space (in this case, an occupancy map). The algorithm will find the nearest node x_{near} inside x_{tree} to the sample point x_{rand}. Then, a new node, x_{new}, is generated along the line connecting x_{rand} and the nearest node, with a distance less than the calculated distance. The path between x_{new} and x_{near} is then checked for collision. If no collision is detected, the new point x_{new} is added to the tree x_{tree}. These steps are repeated until the goal point or the maximum iteration is reached. If a goal point is reached, the path $\overrightarrow{P_{RRT}}$ from the start point is traced until the goal point.

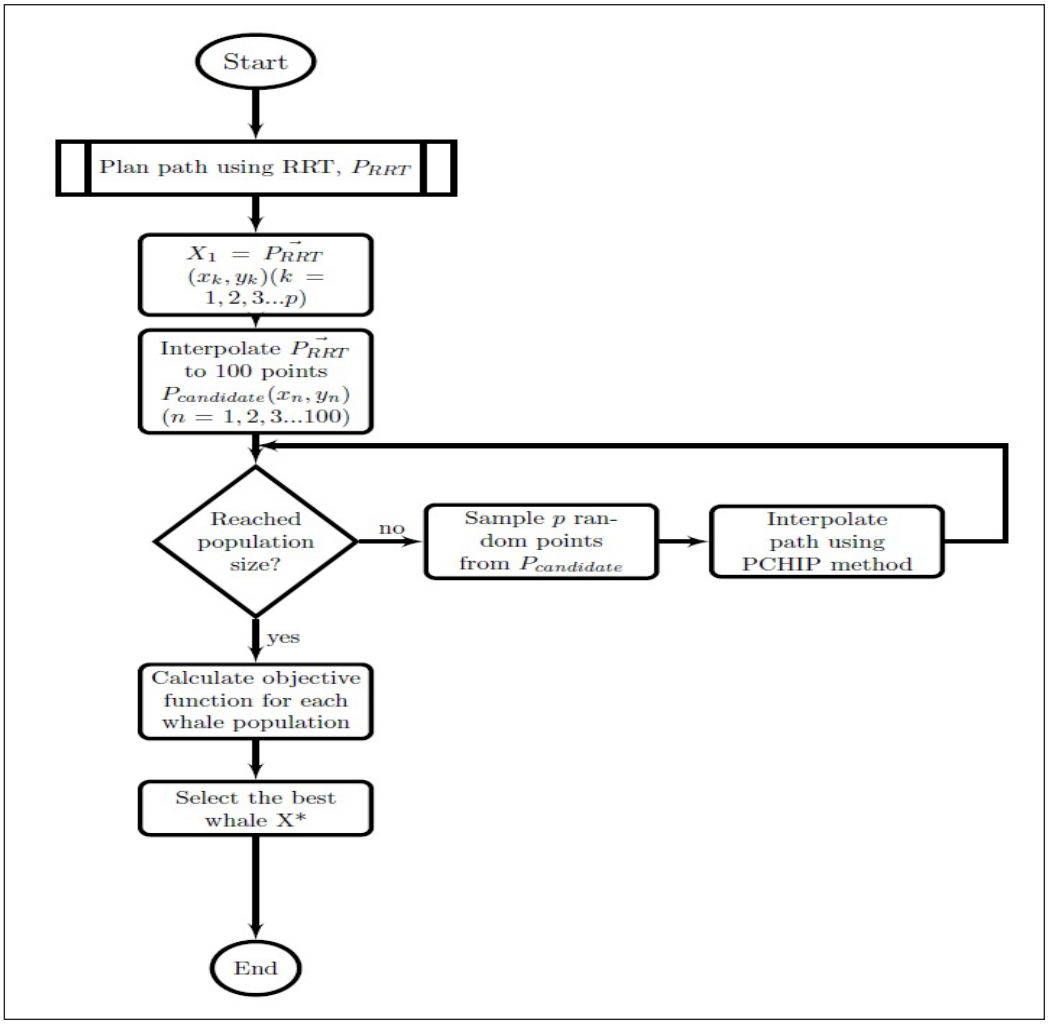


Figure 1. Flowchart of the proposed solution

1: **while** $x_{goal} \neq$ reached **do**
2: Sample random position x_{rand} inside a map
3: Find nearest point x_{near}
4: Find a new point x_{new} along the line between x_{near} and x_{rand}
5: **if** isCollision (path) **then**
6: **continue**
7: **end if**
8: Add x_{new} to x_{tree}
9: **end while**
10: return $\overrightarrow{x_{tree}}$
11: return $\vec{P}_{RRT}$

Figure 2. RRT algorithm

Population Initialization

Population-based optimization methods constitute a category of computational algorithms that derive inspiration from the collective behavior witnessed within natural populations, including animal groups and species evolution. The underlying principle intrinsic to these methods is the meticulous preservation of a population of candidate solutions, commonly referred to as individuals or agents, to methodically explore the vast search space and discern the globally optimal solution.

The population X with a solution size of n is generated. In our case, the first potential solution $\overrightarrow{X_1}$ will be the original RRT path $\overrightarrow{P_{RRT}}$. The original path $\overrightarrow{P_{RRT}}$ consists of p number of points (x_k, y_k), $(k = 1,2,3...p)$ from start to goal point. In order to generate other possible solutions inside the population, the $\overrightarrow{P_{RRT}}$ points were interpolated linearly to produce 100 (x,y) points $P_{candidate} = \{(x_1, y_1) \ldots (x_{100}, y_{100})\}$ where the points were ranked from start point to goal point. This step samples the search space for the optimization algorithm. For the next consecutive solutions $\overrightarrow{X_2}$ - $\overrightarrow{X_n}$, p number of points will be randomly selected from the search area. Figure 3 shows the initialization of population X, where the first individual solution is assigned to $\overrightarrow{P_{RRT}}$.

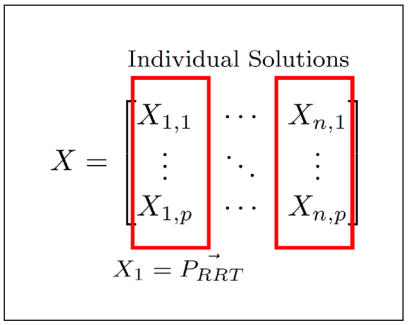


Figure 3. Initialization population solution

Piecewise Cubic Hermite Interpolating Polynomial (PCHIP)

Minimizing significant curvatures around corners, especially in confined spaces, is important as they can lead to inaccuracies to achieve a smooth trajectory path. Using the MATLAB function, Piecewise Cubic Hermite Interpolating Polynomial (PCHIP), instead of the conventional spline approach for point interpolation, as shown in Figure 4, helps

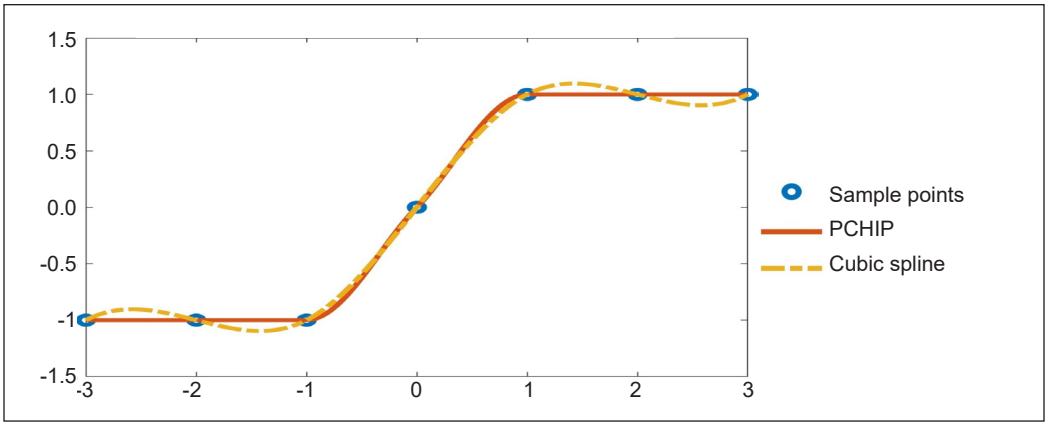


Figure 4. Comparison of different types of interpolation modes in MATLAB (adapted from MATLAB, 2020)

address this issue effectively. The cubic interpolant $P(x)$ maintains the shape of the data. The slopes at points x_j are chosen so that $P(x)$ preserves the data's shape and respects monotonicity. On intervals where the data are uniform, $P(x)$ is also uniform, and at intervals where the data have a local extremum, $P(x)$ also has a local extremum (MATLAB, 2020).

Objective Functions

Objective functions are devised to evaluate the fitness quality of solutions in the population. They comprise three essential components for improving global trajectory planning: smoothness cost ($f_1(\overrightarrow{X_i})$), fuel cost ($f_2(\overrightarrow{X_i})$), and safety cost ($f_3(\overrightarrow{X_i})$), where $\overrightarrow{X_i}$ represents the solution within the population.

It is important to note that the primary objective of this paper is to improve the smoothness of $\overrightarrow{P_{RRT}}$. However, to prevent the solution from getting trapped in the global minimum, it is crucial to consider the fuel cost and safety cost objectives. Without incorporating these two objectives, the resulting solution path may appear smooth but not accurately represent a valid trajectory from the start to the goal point.

It emphasizes the significance of a holistic approach, considering both smoothness and practical constraints. The proposed methodology ensures a smooth path and a viable and reliable trajectory by addressing the fuel cost and safety cost objectives alongside the smoothness objective.

Path Smoothness, $f_1(\overrightarrow{X_i})$

Equation 5 can calculate the smoothness of each path. The path is viewed as a sequence of segments, and the angle of the triangle it forms is calculated. There is a total segment of $p - 1$, and $cos\ \alpha$ for each segment i can be determined (Equations 1 to 3).

$$A = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad [1]$$

$$B = \sqrt{(x_{i+2} - x_i)^2 + (y_{i+2} - y_i)^2} \quad [2]$$

$$cos\ \alpha_i = \frac{A^2+B^2}{2AB} \quad [3]$$

The smoothness of the whole path can be calculated by taking the sum of the smoothness of all the segments (Equation 4).

$$\alpha(\overrightarrow{X_i}) = \sum_{k=1}^{p-1} cos\ \alpha_i \quad [4]$$

The path smoothness function is then determined by taking the average value of all the $\alpha(X_i)$ (Equation 5).

$$f_1(\overrightarrow{X_i}) = \frac{\alpha(\overrightarrow{X_i})}{p-1} \quad [5]$$

It is important to note that a lower value of the smoothness function indicates a smoother path. The smoothness function does not have a specific unit of measurement.

Fuel Cost, $f_2(\overrightarrow{X_i})$

The fuel cost is determined by Equation 6.

$$f_2(\overrightarrow{X_i}) = \sum_{k=1}^{p-1} \sqrt{(x_{k+1} - x_j) + (y_{k+1} - y_k)} \quad [6]$$

where (x_k, y_k), $(k = 1,2,3 \ldots p)$ is the sequence of points inside path $\overrightarrow{P_{RRT}}$ with a total number of sequences p. The fuel cost is measured in meters, m.

Valid Cost, $f_3(\overrightarrow{X_i})$

The validity of the path sequence is examined according to Equation 7 to ensure the robot's safety during navigation. A considerable penalty β is imposed on the cost function if the distance between the point (x_j, y_j) and the obstacle d_{obs} is smaller than the specified safety distance d_s. There is no unit for this objective function.

$$f_3(\overrightarrow{X_i}) = \begin{cases} \beta & d_{obs} \leq d_s \\ 0 & d_{obs} \geq d_s \end{cases} \quad [7]$$

The total cost function for each solution $\overrightarrow{X_i}$ is determined by Equation 8.

$$f(\overrightarrow{X_i}) = f_1(\overrightarrow{X_i}) + f_2(\overrightarrow{X_i}) * \left(1 + f_3(\overrightarrow{X_i})\right) \quad [8]$$

The resultant path example solution for X_1 to X_3 is shown in Figure 5.

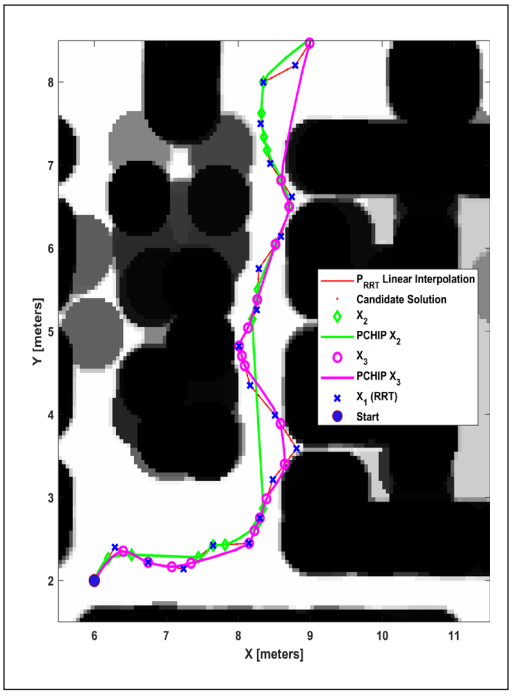


Figure 5. Example of initialization of population for X_1 to X_3 (zoomed)

Whale Optimization Algorithm (WOA)

The optimization algorithm can be employed after initializing the population and specifying the objective functions. This paper presents the application of the WOA as the chosen optimization method.

WOA was initially proposed by Mirjalili and Lewis (2016). It is an optimization

algorithm that inspires humpback whales. Once they have discovered their target, humpback whales can encircle it. The WOA approach operates under the premise that the current best candidate solution is either the prey of interest or is very near to the optimal because the placement of the optimal design inside the search space is a priori unknown. Once the best search agent has been selected, the remaining search agents will attempt to draw closer to it. This behavior is demonstrated through Equations 9 and 10:

$$\vec{D} = \left|\vec{C} \cdot \overrightarrow{X^*}(t) - \vec{X}(t)\right| \quad [9]$$

$$\vec{X}(t+1) = \overrightarrow{X^*}(t) - \vec{A}.\vec{D} \quad [10]$$

where t indicates the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors, X^* is the position vector of the best solution obtained so far, and $\vec{X}$ is the position vector. It is important to note that X^* should be adjusted after each iteration if a better solution exists.

The vectors $\vec{A}$ and $\vec{C}$ are calculated as Equations 11 and 12:

$$\vec{A} = 2\vec{a} \cdot rand(\,) - \vec{a} \quad [11]$$

$$\vec{C} = 2 \cdot rand(\,) \quad [12]$$

where $\vec{a}$ is linearly decreased from 2 to 0 throughout iterations (in both exploration and exploitation phases), and the $rand(\,)$ is a random vector in [0,1].

Bubble-net Aattacking Method (Exploitation Phase)

In order to mathematically model the bubble-net behavior of humpback whales, two approaches are designed as follows:

Shrinking Encircling Mechanism

It is accomplished by reducing the value of $\vec{a}$ n in Equation 12. Note that the fluctuation range of $\vec{A}$ is likewise decreased by $\vec{a}$. In other words, $\vec{A}$ is a random number in the interval [–a,a] where a is decreased from 2 to 0 throughout iterations.

Spiral Updating Position

A spiral equation is established between the whale's position and its prey to simulate the helix-shaped movement of humpback whales (Equation 13).

$$\vec{X}(t+1) = \overrightarrow{D'} \cdot exp^{bl} \cdot cos(2\pi l) + \overrightarrow{X^*}(t) \quad [13]$$

where $\overrightarrow{D'}$ is represented in Equation 14 and indicates the distance between the i-th whale to the prey (best solution obtained so far), b is a constant for defining the shape of the logarithmic spiral, l is a random number in [–1,1] and $\cdot$ is an element-by-element multiplication.

$$\overrightarrow{D'} = \left|\overrightarrow{X^*}(t) - \vec{X}(t)\right| \quad [14]$$

Humpback whales swim simultaneously in a diminishing circle and spiral pattern around their prey. It is assumed that there is a 50% likelihood of picking either the shrinking encirclement mechanism or the spiral model to update the whales' position to characterize this concurrent behavior. In addition to the bubble-net method, humpback whales search for prey randomly. The mathematical model of the search is as follows.

Search for Prey (Exploration Phase)

Value $\left|\vec{A}\right| > 1$ emphasizes exploration and permits the WOA algorithm to perform a global search. The mathematical model looks like this (Equations 15 and 16):

$$\vec{D} = \left|\vec{C} \cdot \overrightarrow{X_{rand}}(t) - \vec{X}(t)\right| \quad [15]$$

$$\vec{X}(t+1) = \overrightarrow{X_{rand}}(t) - \vec{A} \cdot \vec{D} \quad [16]$$

where X_{rand} is a random position vector (a random whale) chosen from the current population.

Proposed Modification

In order to further improve the smoothness of the path, an adaptive value of $\vec{C}$ is proposed based on the occupancy of the X^* points instead of a constant number. The occupancy of a grid point inside an occupancy map is shown in Equation 17, where 0 is considered free space, 1 is considered occupied space and –1 is considered an unknown space. The constant value 2 is replaced with Equation 18, which means that $\vec{D}$ and consequently $\vec{X}(t+1)$ is only updated when the occupancy of the point is free space or an unknown space. The modified equation of C is shown in Equation 19.

$$\overrightarrow{occ_{X^*(t)}} \cup M = \begin{cases} 0 & free\ space \\ 1 & occupied\ space \\ -1 & unknown\ space \end{cases} \quad [17]$$

$$\vec{w} = 1 - \overrightarrow{occ_{X^*(t)}} \quad [18]$$

$$\vec{C} = \vec{w} \cdot rand(\,) \quad [19]$$

The algorithm for the WOA is shown in Figure 6. The maximum iteration number t_{max} is set to 20 for all the experiments.

EXPERIMENTAL SETUP AND PERFORMANCE EVALUATION

An experimental setup was devised to assess the proposed method's robustness. Two factors were varied in the environment: the start-goal points and the randomness of RRT

1: Initialize the whales population $X_i(i = 1,2…n)$
2: Calculate the fitness of each search agent
3: X^* = the best search agent
4: **while** $t < t_{max}$ **do**
5: **for** each search agent **do**
6: Update a, A, C, l and p
7: **if** $p < 0.5$ **then**
8: **if** $|A| < 1$ **then**
9: Update the position of the current search agnt by eq. (10)
10: **else if** $|A| \geq 1$ **then**
11: Select random search agent $\boldsymbol{x}_{rand}$
12: Update the position of the current search agent by Eq. (17)
13: **end if**
14: **else if** $p \geq 0.5$ **then**
15: Update the position of the current search by using Eq. (14)
16: **end if**
17: **end for**
18: Check if any search agent goes beyond the search space and amend it
19: Plot path using PCHIP
20: Calculate the fitness of each search agent
21: Update X^* if there is a better solution
22: **end while**
21: return X^*

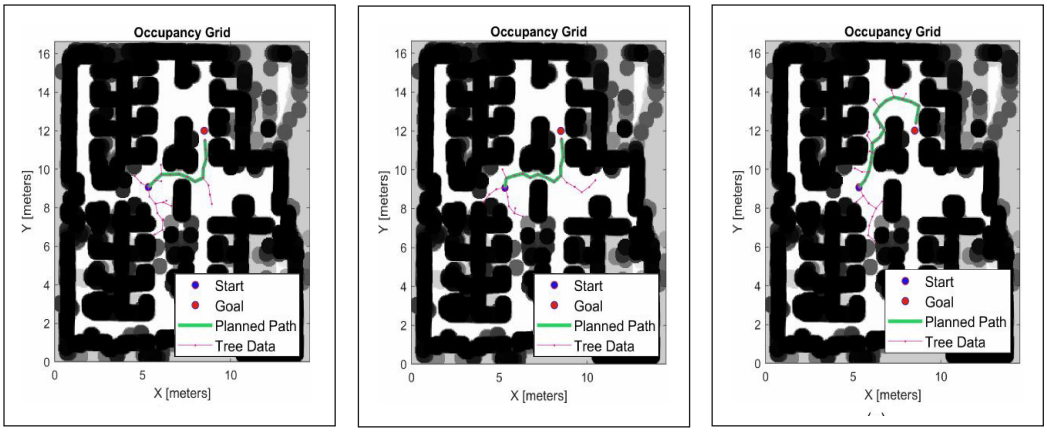
Figure 6. RRT-WOA Algorithm

path solutions. The algorithm's versatility and robustness in accommodating different configurations can be evaluated by altering the start-goal points.

The RRT algorithm generates paths based on random sampling, resulting in potential variations with each execution. The Random Number Generator (RNG) seeds were saved for data collection to ensure the planned path's reproducibility. Three sets of RNG seeds were selected to test the algorithm's robustness. The initial solution within the population, denoted as X, was recorded to facilitate a fair performance comparison among optimization algorithms. This guarantees that all optimization algorithms start with the same set of solutions and the same population size for each trial.

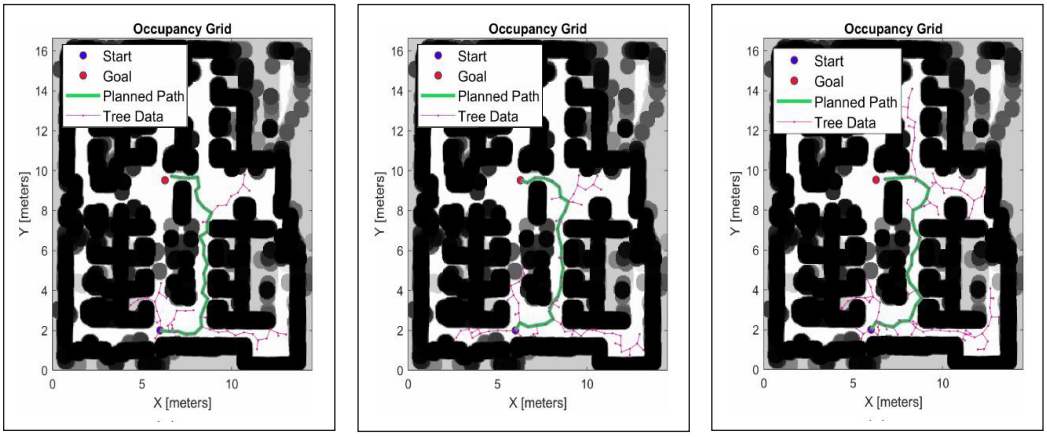
Environment Setup

Figures 7 and 8 illustrate the six different environmental setups used in the experiment. The occupancy map is based on the Engineering Design Lab in the USM Engineering Campus. It is inflated with a radius of 0.4m, corresponding to the physical size of the hardware robot. This inflation ensures that the initial planned path generated by the RRT algorithm is feasible for the robot to navigate. The simulation experiment was implemented using MATLAB ver. 2020b and executed on a laptop with an Intel Core i7-1065G7 CPU @1.30GHz.



(a) (b) (c)

Figure 7. Generated RRT for Path 1 (Start point:(5.3,9.0) and Goal point: (8.5,12.0)) using (a) RNG 1, (b) RNG 2, and (c) RNG 3 *for ease of viewing, all plot legend used the same indicator



(a) (b) (c)

Figure 8. Generated RRT for Path 2 (Start Point: (6.0,2.0) and Goal point: (6.3,9.5)) using (a) RNG 1, (b) RNG 2, and (c) RNG 3 *for ease of viewing, all plot legend used the same indicator

Benchmark Test

A performance evaluation and comparison were conducted to determine whether the proposed method (denoted as WOA (modified) onwards) performs better in smoothing the RRT path than other population-based optimization algorithms, namely PSO, ABC, and FA. Table 1 presents the parameters used in the performance comparison of these algorithms.

The evaluation focused on key metrics to assess the effectiveness and efficiency of these algorithms. The metrics analyzed included the average optimization time, the mean smoothness of the resulting paths, and the percentage improvement achieved in path smoothing.

Table 1
Benchmark test parameters

Algorithm	Parameters	Values
Common	Population size, n	30
	Number of iterations, t_{max}	20
WOA (Mirjalili & Lewis, 2016)	Coefficient vector, $\vec{C}$	$2 \cdot rand( )$
	Coefficient vector, $\vec{A}$	$2\vec{a} \cdot rand( ) - \vec{a}$
	Logarithmic spiral, l	$[-1,1]$
WOA (modified)	Coefficient vector, $\vec{C}$	$\vec{w} \cdot rand( )$
	Coefficient vector, $\vec{A}$	$2\vec{a} \cdot rand( ) - \vec{a}$
	Logarithmic spiral, l	$[-1,1]$
PSO (Heris, 2017)	Cognitive component, c_1	1.5
	Social component, c_2	1.5
	Inertia weight, w_i	1
ABC (Heris, 2020)	Number of employed bees	n
	Number of onlooker bees	n
	Number of scout bees	n
	Trial limit, L	$0.6mn^*$
FA (Yang, 2009)	Mutation number, ϕ	$rand([-1,1])$
	Attraction coefficient, β0	2
	Absorption coefficient, γ	1

RESULTS AND DISCUSSION

Benchmark Performance Analysis

The first benchmark test evaluated the optimization time of multiple algorithms. Figure 9 shows that WOA (modified) achieves the lowest median and second smallest spread for optimization time, indicating superior and consistent performance. WOA follows with the second-lowest median and a smaller spread. PSO demonstrates a slightly higher median and the smallest spread, indicating relatively better and consistent performance. ABC performs competitively with a higher median and small spread. FA exhibits the highest median and largest spread, indicating lower performance and greater variability.

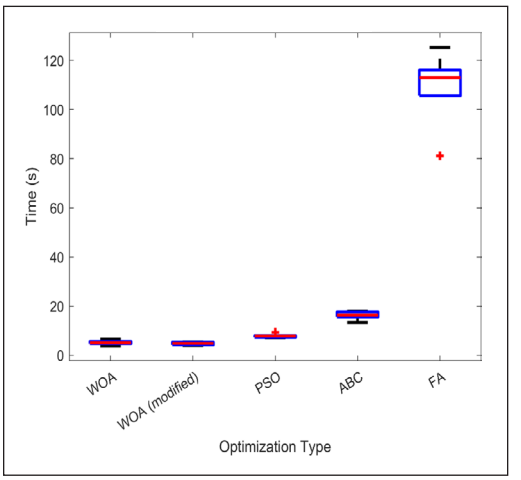


Figure 9. Boxplot analysis of optimization time for various optimization algorithms (lower values indicate better performance)

The second benchmark test evaluated multiple optimization algorithms based on path smoothness. From the boxplot shown in Figure 10, WOA (modified) demonstrates the lowest median and the smallest spread,

indicating better and more consistent performance. PSO shows a slightly higher median with a small spread, while ABC exhibits the highest median and the largest spread, indicating relatively poorer performance with more variability. FA falls in between, with a median higher than PSO but lower than ABC and a slightly larger spread than PSO.

The third benchmark test evaluated multiple optimization algorithms based on their ability to improve the smoothness of the initial RRT path. From boxplot data in Figure 11, WOA (modified) demonstrates the highest median and the smallest spread, indicating the best improvement percentage and most consistent performance. On the other hand, FA shows the lowest median with a smaller spread, suggesting relatively poorer but more consistent performance. PSO and ABC exhibit larger spreads, indicating more variability in their results, with PSO having a higher percentage of improvement compared to ABC.

The results of the benchmark performance test were visualized using a radar chart, as depicted in Figure 12. The chart concisely summarizes the algorithm rankings based on their overall performance. According to the plot, the algorithm that

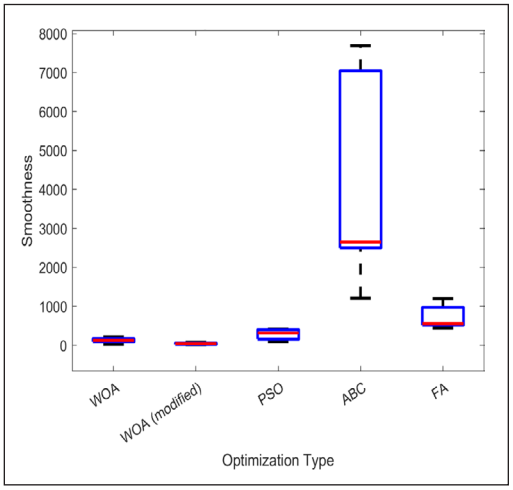


Figure 10. Boxplot analysis of optimization results in path smoothness for various optimization algorithms (lower values indicate better performance)

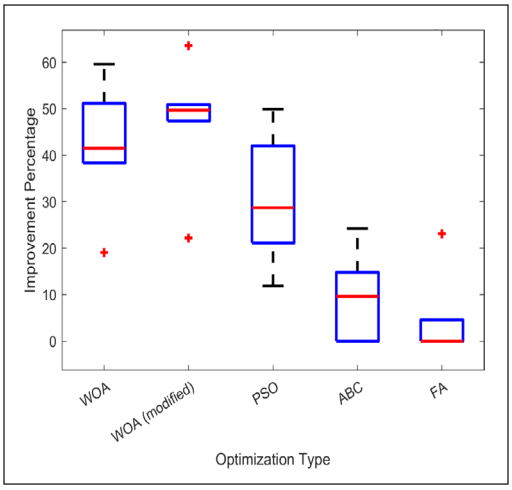


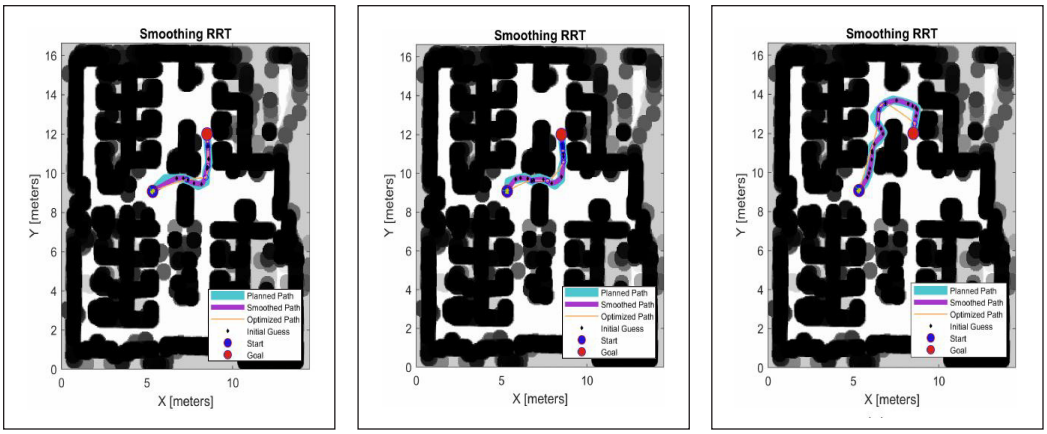
Figure 11. Boxplot analysis of optimization percentage improvement in path smoothness for various optimization algorithms (higher values indicate better performance)

Figure 12. Radar chart for performance ranking (best to worst: WOA (modified), WOA, PSO, ABC & FA)

achieved the highest overall performance was WOA (modified), closely followed by WOA. PSO attained the third rank, while ABC and FA demonstrated the lowest performance among the tested algorithms. These rankings offer a clear representation of the relative performance of each algorithm in the benchmark test.

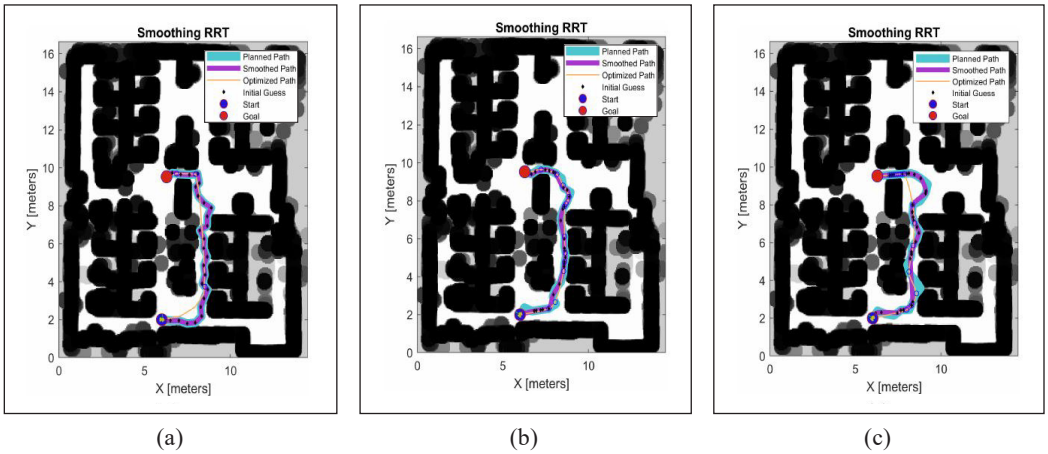
Results of Smoothed RRT Path

The result of the smoothed RRT path using WOA is shown in Figure 13 for Path 1 and Figure 14 for Path 2. From Figures 13 and 14, it could be observed that the optimized path (labeled with an orange line) is able to smooth out the original RRT path while still maintaining its path validity.



(a) (b) (c)

Figure 13. Smoothed RRT for Path 1 (Start point:(5.3,9.0) and Goal point: (8.5,12.0)) using (a) RNG 1, (b) RNG 2, and (c) RNG 3 *for ease of viewing, all plot legend used the same indicator



(a) (b) (c)

Figure 14. Smoothed RRT for Path 1 for Path 2 (Start Point: (6.0,2.0) and Goal point: (6.3,9.5)) using (a) RNG 1, (b) RNG 2, and (c) RNG 3 *for ease of viewing, all plot legend used the same indicator

CONCLUSION

In conclusion, this research has successfully addressed the challenges of using the RRT algorithm as a path planner. By integrating the RRT path planner with a modified version of the WOA (RRT-WOA), significant improvements have been made in trajectory smoothness and path curvature reduction. The novel approach of incorporating parameter variation ($\vec{C}$) in the modified WOA algorithm has effectively optimized trajectory smoothness. Additionally, using Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) for point interpolation has further contributed to smoothing generated paths. The modified WOA algorithm has demonstrated its superiority over popular population-based optimization algorithms such as PSO, ABC, and FA through a comprehensive comparative analysis. The WOA (modified) algorithm outperformed these alternatives in terms of optimization time, trajectory smoothness, and improvement from the initial guess.

FUTURE WORKS

In future research, we hope to fine-tune the updated equations and parameters so that the proposed equation applies to all path types.

ACKNOWLEDGEMENT

This work was supported by Collaborative Research in Engineering, Science, and Technology (CREST), Malaysia, with grant no. 304/PELECT/6050423/C121.

REFERENCES

Abbadi, A., & Matousek, R. (2014, November). *Path planning implementation using MATLAB*. [Paper presentation]. International Conference of Technical Computing Bratislava 2014, Bratislava, Slovakia. https://doi.org/10.13140/2.1.3324.5767

Alam, M. S., & Rafique, M. U. (2015). Mobile robot path planning in environments cluttered with non-convex obstacles using particle swarm optimization. In *2015 International Conference on Control, Automation and Robotics* (pp. 32-36). IEEE Publishing. https://doi.org/10.1109/ICCAR.2015.7165997

Dao, T. K., Pan, T. S., & Pan, J. S. (2016). A multi-objective optimal mobile robot path planning based on whale optimization algorithm. In *International Conference on Signal Processing Proceedings* (pp. 337-342). IEEE Publishing. https://doi.org/10.1109/ICSP.2016.7877851

Dosoftei, C. C., Popovici, A. T., Sacaleanu, P. R., Gherghel, P. M., & Budaciu, C. (2021). Hardware in the loop topology for an omnidirectional mobile robot using matlab in a robot operating system environment. *Symmetry*, *13*(6), Article 969. https://doi.org/10.3390/sym13060969

Galli, M., Barber, R., Garrido, S., & Moreno, L. (2017). Path planning using Matlab-ROS integration applied to mobile robots. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)* (pp. 98-103). IEEE Publishing. https://doi.org/10.1109/ICARSC.2017.7964059

Heris, M. K. (2020). *Artificial bee colony in Matlab*. Yarpiz. https://yarpiz.com/297/ypea114-artificial-bee-colony

Heris, M. K. (2017). *Particle swarm optimization*. Yarpiz. https://yarpiz.com/50/ypea102-particle-swarm-optimization

Jeong, I. B., Lee, S. J., & Kim, J. H. (2019). Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate. *Expert Systems with Applications*, *123*, 82-90. https://doi.org/10.1016/j.eswa.2019.01.032

Karur, K., Sharma, N., Dharmatti, C., & Siegel, J. E. (2021). A survey of path planning algorithms for mobile robots. *Vehicles*, *3*(3), 448-468. https://doi.org/10.3390/vehicles3030027

Lavalle, S. M., & Kuffner, J. J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, *20*(5), 378-400. https://doi.org/10.1177/02783640122067453

Li, R., Liu, J., Zhang, L., & Hang, Y. (2014). LIDAR/MEMS IMU integrated navigation (SLAM) method for a small UAV in indoor environments. In *2014 DGON Inertial Sensors and Systems (ISS)* (pp. 1-15). IEEE Publishing. https://doi.org/10.1109/InertialSensors.2014.7049479

Mac, T. T., Copot, C., Tran, D. T., & De Keyser, R. (2016). Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems*, *86*, 13-28. https://doi.org/10.1016/j.robot.2016.08.001

MATLAB. (2020). *Piecewise cubic hermite interpolating polynomial (PCHIP)*. MathWorks. https://www.mathworks.com/help/matlab/ref/pchip.html

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51-67. https://doi.org/10.1016/j.advengsoft.2016.01.008

Naderi, K., Rajamaki, J., & Hamalainen, P. (2015). RT-RRT∗: A real-time path planning algorithm based on RRT∗. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games (MIG 2015)* (pp. 113-118). Association for Computing Machinery https://doi.org/10.1145/2822013.2822036

Xinyu, W., Xiaojuan, L., Yong, G., Jiadong, S., & Rui, W. (2019). Bidirectional potential guided rrt* for motion planning. *IEEE Access*, *7*, 95046-95057. https://doi.org/10.1109/ACCESS.2019.2928846

Yang, X. S. (2009). Firefly algorithms for multimodal optimization. In O. Watanabe & T. Zeugmann (Eds.), *Stochastic Algorithms: Foundations and Applications* (pp. 169-178). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04944-6_14

Yu, Z., & Xiang, L. (2021). NPQ-RRT ∗: An improved RRT ∗ Approach to hybrid path planning. *Complexity*, *2021*(1), Article 6633878. https://doi.org/10.1155/2021/6633878

Zhou, X., Gao, Y., & Guan, L. (2019). Towards goal-directed navigation through combining learning based global and local planners. *Sensors*, *19*(1), Article 176. https://doi.org/10.3390/s19010176