Journal of Robotics and Control (JRC)
Volume 6, Issue 1, 2025
ISSN: 2715-5072, DOI: 10.18196/jrc.v6i1.22062

An Effective Way for Repositioning the Beacon Nodes of Fast RRT Results Utilizing Grey Wolf Optimization

Heru Suwoyo [1], Andi Adriansyah [2*], Julpri Andika [3], Abu Ubaidah Shamsudin [4], Yingzhong Tian [5]
[1, 2, 3] Department of Electrical Engineering, Universitas Mercu Buana, Jakarta, Indonesia
[4] Department of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia
[5] School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China
Email: [1] heru.suwoyo@mercubuana.ac.id , [2] andi@mercubuana.ac.id, [3] julpri.andika@mercubuana.ac.id,
[4] ubaidah@uthm.edu.my, [5] troytian@shu.edu.cn
*Corresponding Author

***Abstract*—Conceptually, Fast-RRT applies fast sampling and random steering which makes the initial path quickly obtained. Referring to the initial path, the optimality of the path is improved by applying path fusion and path optimization. Theoretically, path fusion will only be optimal if there is always a unique/different path to be fused with the previously obtained path. However, in the conditions of solving path planning problems in narrow corridors, the potential for obtaining a different path from the previous one is very small. So that fusion does not run properly, but checking the relationship between nodes to nodes still occurs. Instead of getting an optimal path in conditions like this, the computation will increase, the solution time will be long, and the resulting path will still be sub-optimal. As an effort to solve this problem, Grey Wolf Optimization (GWO) is involved through this study. While an initial path is found, the beacons are repositioned. From the path, the number of nodes is unpredictable, causing the decision variables in optimization to become large. For this reason, the GWO is chosen because it is independent of population representation and is not affected by the number of decision variables. This proposed method is claimed to be more effective in solving path planning problems in terms of convergence rate and optimality. Therefore, the proposed method is evaluated and compared with previous methods and gives the result that the average working speed of Fast-RRT is improved by 90.25% and the optimality average increased by 5.67%.**

Keywords—Fast-RRT; Grey Wolf Optimization; Path Planning; Convergence Rate; Optimality.

I. INTRODUCTION

In some scenarios, the robot performs the exploration task at the same time as solving the Simultaneous Localization and Mapping (SLAM) [1], [2], [3], [4], [5]. Generally, SLAM will let the robot know the marginalized poses as well as the initially unknown map. Thus, it must be supporting the performance of the robot to be fully autonomous. The reason behind this assumption is that since these tasks performed well, the robot needs to only decide where the goal point from any initial point is to be later traced [5], [6], [7], [8]. Finding the feasible and safe path in this case is known as global path planning and the tracing process is called path tracking [9], [10], [11], [12], [13], [14]. Before the robot moves from the current to any desired pose, path planning should be performed well to generate the reference path. Therefore, it must be designed and prepared as well as possible.

There are two types of methods commonly used for solving global path planning, which are the searching-based and sampling-based algorithms [15], [16], [17], [18], [19], [20], [21], [22]. The popular methods for searching-based algorithms include A* [8], [23], [24], [25] and Dijkstra algorithm [13], [14], [15], [16], [17]. For the sampling-based algrorithm include the Rapidly-Exploring Random (RRT) [8], [32], [33] and RRT* [34], [35], [36], [37]. Generally, the searching-based offers the quality and resolution of the safely planned path compared to the sampling-based algorithm. Nevertheless, the sampling-based algorithm possesses the high-speed process of finding and generating this path which cannot be found in sampling-based algorithms [38], [39], [40]. This can be seen clearly from how the RRT or RRT* works. Armed with knowledge about the environment, starting points, and planned ending points, RRT started its work by expanding from node to node. In this expansion process, nodes are randomly generated and then selected as the next node. Then the randomly generated nodes are checked for whether they are free-collision or not if the node is connected to the closest available node named a vertex. If a vertex is free from obstacles, the node is considered as an available node, and vice versa if it is not free from obstacles, the process of randomly generating nodes is repeated. The process is then repeated whenever the single vertex is made up. In the end, the shortest connected vertexes are found by comparing one to the other. Moreover, the RRT* also performs the same process as RRT does with the improvement on the process of generating the vertex. It is conducted by applying the rewiring step that again checks the connection to any rounded nodes to get the shortest vertex.

Although RRT and its successor, RRT*, satisfy the objective of global path planning, they suffer from the ability to generate vertex connecting the nodes that are separated by tunnel-like obstacles [23], [41], [42], [43], [44], [45]. Additionally, the expansion process has also been consuming much time since the randomly generated node has the potential to be in the area around the available one [46], [47], [48], [49], [50]. For these limitations, several methods have been proposed. More Quickly-RRT* (MQ-RRT*) has been introduced with the aim of improving optimality and convergence speed. The convergence speed is improved by

applying the sparse sampling mechanism that reduces repetitive sampling. Moreover, optimality is achieved by biasing the *Chooseparent* before the rewiring stage is conducted. With the same focus, to speed up the initial path found, the hybrid-RRT combining the informed-RRT* [51], [52] and RRT-Connect [34], [41] has also been introduced. This combination sequentially gains the initial path faster because of bidirectional exploration from RRT-Connect and gives an optimal path offered by informed-RRT* after the initial path found. Another improvement is to combine A* and RRT* [53]. The sampling of RRT* is conducted by following the manner how A* finds the optimal path. Even though this hybrid algorithm is still classified as sampling-based algorithm. The new node is wired to the most potential node in the trace. It means the near node is chosen based on the lowest cost. This cost is the sum of the path cost from the starting node to the examined node and the path cost from the goal node to the examined node. For this reason, this hybrid algorithm gives better optimality and is faster compared to RRT*. Not only these methods, the development of RRT* can be noticed from RRT*-Smart [54] which utilizes the biasing method to improve optimality. Regarding the beacons found from the initial path, the rest of sampling focuses on the area around the beacon. Not only this intelligence sampling, but the path optimization is also conducted to reduce the number of beacons as the basis for intelligence sampling. They are sequentially conducted to accelerate the optimality of algorithm. However, the initial path is found by applying the procedure of RRT* which takes time in the complex and large area. Moreover, Fast-RRT has been introduced [55]. In the Fast-RRT, the fast sampling is initiated to tackle the first limitation, and the random steering process is used to overcome the second one. These two processes are carried out sequentially and are referred to as improved RRT. Apart from this improvement, Fast-RRT also has a Fast-Optimal stage which contains commands for fusion and tuning. The fusion stage is carried out in the second stage after the new path is available. This is done with the assumption that the previously formed path is still sub-optimal. Every time there is a new path, the sub-path will be fused with the another sub-path. Which means that fusion occurs when there is a unique path that is different from the previous path. However, in the conditions of solving path planning problems in narrow corridors, the potential for obtaining a path that is different from the previous one is very small. So that fusion does not run properly, but checking the relationship between nodes to nodes still occurs. Instead of getting an optimal path in conditions like this, the computation will increase, the solution time will be long, and the resulting path will still be the same, just suboptimal. Furthermore, in the tuning stage, the relationship between nodes is checked again and will be rewired by trimming the vertices, it is found that the two outcrop nodes are free of obstacles. While logically this would provide an improvement, Fast-Optimal still relies on Improved-RRT sub-paths only. Therefore, the optimal Fast-RRT solution is highly dependent on the repetitions performed. So, the optimality of the solution requires a long time [11].

Regarding this problem, in this paper an improvement is being carried out. This development involves Grey Wolf Optimization (GWO) [56], [57], [58], [59] which will be the core method for improving the performance of the Fast-Optimal path. The application of GWO begins by generating a wolf pack at a position around the target node $x_{targeted}$. Next, the fitness value of each wolf is calculated based on the overall path length connecting the beacon nodes. From there, $\vec{X}_{\alpha}$, $\vec{X}_{\beta}$, and $\vec{X}_{\delta}$ are determined based on the three lowest values. Then, $\vec{X}_1$, $\vec{X}_2$, and $\vec{X}_3$ are calculated using the established equations, which results in $\vec{X}_{new}$. This process is followed by greedy selection and updating of the wolf pack position. This stage is repeated as many times as the number of iterations (maximum iteration). This maximum iteration is calculated by subtracting the number of samples when the initial path is known from the maximum number of samples, then dividing the result by the number of wolves in the pack. Due to its more nature-inspired workings, Grey Wolf Optimization (GWO) has several advantages over other algorithms such as Particle Swarm Optimization (PSO) [60], [61], [62], [63], [64], [65], [66], [67], [68] and Genetic Algorithm (GA) [69], [70], [71], [72], [73], [74], [75], [76]. GWO mimics the hunting behavior of wolves; the pack leader (alpha), second leader (beta), and third leader (delta) work together to chase prey, balancing exploration and exploitation of the solution space [77], [78], [79], [80]. This method makes GWO easy to adapt to difficult problems, especially those with large or non-linear solution spaces. The simple structure and the small number of parameters make GWO computation less heavy and more efficient. This effective computation underlies the ability of GWO convergence to be fast without requiring many repetitions and iterations to provide an optimal solution. This supports the acceleration of GWO work in supporting the optimization of beacons nodes on a predetermined path. GWO also uses greedy selection, which helps the pack choose the best solution based on previous experience, so they don't get stuck in bad local solutions. This provides the basis for selecting GWO over other related methods.

The rest of this paper is organized as follows: Section II presents the materials and methods, including the problem statement, the basic concepts of Fast-RRT, and Grey Wolf Optimization. In Section III, the proposed method is discussed, followed by the analysis of the results in Section IV. Finally, the conclusion is provided in the last section.

II. MATERIAL AND METHOD

A. Problem Statement

Let $X \in \mathbb{R}^n$ is representation of state space for a path planning problem, with $n \in N$ is space dimension, thus $X = \{X_{obs}, X_{free}\}$ is state space with $X_{obs} \in X$ refers to obstacle coordinates and $X_{free} \in X$ refers to the free space. Moreover, if the starting node $x_{init} \in X_{free}$ and the goal node $x_{goal} \in X_{free}$ are given, then referring to X_{obs}, the path planning algorithm has to find the ideal path from-to those nodes, denoted as $\sigma = [0, T] \rightarrow X_{free}$ with $\sigma(0) = x_{init}$ and $\sigma(T) = x_{goal}$. Where the area closed to the goal node x_{goal} is denoted by X_{goal}, which is defined as $\{x \in X | x - x_{goal}| < r\}$, where r is radius around x_{goal}.

B. Fast-RRT

Conceptually, Fast-RRT works by implementing two core stages, namely Improved-RRT, and Fast-Optimal. In improved-RRT there are two differentiators that distinguish Fast RRT from RRT, namely fast sampling and random steering. Likewise, path optimization is different from conventional methods, namely implementing path fusion and path optimization, both of which are found in Fast-Optimal. In short, the working of Fast-RRT is shown in Algorithm 1.

Algorithm 1 – Fast-RRT

1 Input: x_{init}, x_{goal}, Map
2 Output: A path T from x_{init} to x_{goal}
3 for $= i \ldots N$ do
4 $T_{init} \leftarrow improvedRRT(x_{init}, X_{goal}, Map)$
5 If T_{init} isnot None then
6 $T_{opt} \leftarrow FastOpt(T_{opt}, T_{init})$

Although there are similarities in the general concept of exploration, in $improvedRRT$ step there is fast sampling. Where this step is done by limiting random nodes which are only allowed in areas that have never been touched. This coverage refers to the radius of nodes that have been obtained by Fast-RRT. Obviously, there can be no repeated generation of nodes around the old node which makes the method speed up. In addition to fast-sampling on the improved RRT, there is also random-steering which does the steering process to random points so that the vertices are free from obstacles. The sampling process on the improved RRT stops when the newest node is in the goal node area. But instant solutions like this only provide sub-optimality. So that there is a repetition of path generation with the same method. However, whenever a path is formed, Fast-RRT will process the two best paths in the next process, namely Fast-Optimal. Where at this stage, there are two core jobs, namely Fusion and Fast-Tuning. In fusion nodes with very close distances are considered to coincide. Then all vertices are reconnected, with reference to the cost of each sub-optimal path. Furthermore, in order to cut the high-cost value, fast-tuning is done. Where in this process, the direct connection between nodes is barrier-free, replacing the vertices between the previous nodes. In general, fast-optimal on Fast-RRT can be seen in the following algorithm.

Algorithm 2 – $improvedRRT$ *and Fast Optimal*

1 Input: x_{init}, x_{goal}, Map
2 Output: A path T_{opt} from $x_{init}$to x_{goal}
3 for $= i \ldots N$ do
4 $x_{rand} \leftarrow FastSample(Map)$
5 $x_{near} \leftarrow nearest(T, x_{rand})$
6 $x_{new} \leftarrow randomsteering(x_{near}, x_{rand})$
7 $E_t \leftarrow Edge(x_{new}, x_{near})$
8 if $collisionFree(E_t, Map)$ then
9 $T \leftarrow addNode(x_{new})$
10 if $x_{new} \in X_{goal}$ then
11 $T_{init} \leftarrow T$
12 if $isnotempty(T_{init})$
13 $T_{opt} \leftarrow pathfusion(T_{init}, T_{opt})$
14 $T_{opt} \leftarrow pathOptimize(T_{opt})$
15 endif
16 endif
17 endif
18 endfor

Referring to Algorithm 2, $improvedRRT$ is shown in the command in lines 4 to 11. Meanwhile, $FastOptimal$ is shown in commands 12 up to 17. To find out details about these, Fast-RRT in [55] can be used as a reference.

C. Grey Wolf Optimization

The leadership structure and hunting strategy of grey wolves in nature are modeled by the GWO algorithm. For the purpose of mimicking the leadership hierarchy, four different varieties of grey wolves, including alpha, beta, delta, and omega, are used. In addition, the three essential components of hunting—looking for prey, surrounding prey, and attacking prey—are used.

The grey wolf, which hunts enormous prey in packs and depends on interpacket cooperation, inspired this algorithm. This behavior has two intriguing aspects: social hierarchy and the hunting mechanism. The grey wolf has a complicated social hierarchy due to being a highly gregarious animal. The term "dominance hierarchies" refers to ranking wolves based on their size and power. There are the alphas, betas, deltas, and omegas as a result. The pack is led by the alpha male and female, who are at the top of the hierarchy. Every member of the pack is ranked according to their position. The wolf pack's hierarchy helps weaker members of the pack who are unable to hunt for themselves and is not just about hostility and power. The beta wolf comes next, who aids the alpha wolf in making choices and maintains order in the pack. The delta wolf is ranked beneath the beta wolf. They are frequently powerful but lack leadership abilities or self-assurance to assume leadership roles. Finally, the omega wolf has no power at all, and other wolves will run after him right away. Omega wolf is also in charge of keeping an eye on the young wolves. The three best solutions will be denoted by alpha, beta, and delta, respectively, at each stage when we apply the approach previously discussed to our optimization problem, and the remaining solutions will be denoted by omega. It basically means that the three best solutions guide the optimization process. The prey will also be the best possible outcome of the optimization.

Most of the logic follows the equations:

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right| \quad (1)$$

$$\vec{X}_p(t+1) = \vec{X}_p(t) - \vec{A}.\vec{D}\ \vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right| \quad (2)$$

where t denotes the current iteration, the vector $\vec{A}$ and $\vec{C}$ represent the coefficient vectors, $\overrightarrow{X_p}$ refers to the position vector of the prey and $\vec{X}$ is the position of the wolf. Vectors $\vec{A}$ and $\vec{C}$ are generally given as follows.

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \quad (3)$$

$$\vec{C} = 2\vec{r}_2 \quad (4)$$

where the linear decrement $\vec{A}$ is from 2 to 0 through iteration and $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are random vector in range [0,1], calculated for each wolf at each iteration. Whereas $\vec{a}$ can be calculated using (5).

$$\vec{a} = 2\left(1 - \frac{iteration}{maxiter}\right) \quad (5)$$

Vector $\vec{A}$ controls the trade-off between exploration and exploitation while $\vec{C}$ always adds some degree of randomness. This is necessary because our agents can get stuck in the local optima and most of the metaheuristics have a way of avoiding it. Since $\vec{a}$ is calculated, $\vec{A}$ and $\vec{C}$ will be computed using (3) and (4), respectively. Since, we don't know the real position of the optimal solution, $\overrightarrow{X_p}$ depends on the 3 best solutions and the formulas for updating each of the agents (wolfs) are:

$$\vec{D}_\alpha = \left|\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}\right| \quad (6)$$

$$\vec{D}_\beta = \left|\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}\right| \quad (7)$$

$$\vec{D}_\gamma = \left|\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}\right| \quad (8)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \quad (9)$$

$$\vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \quad (10)$$

$$\vec{X}_3 = \vec{X}_\gamma - \vec{A}_3 \quad (11)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (12)$$

where $\vec{X}(t)$ represents the current position of the agent and $\vec{X}(t+1)$ is the updated one (called $\vec{X}_{new}$ in Introduction). According to the algorithm above, the wolf's position will be changed in accordance with the top three wolves from the previous iteration. The result will not exactly match the average of the top three wolves because the vector $\vec{C}$ introduces a slight random shift.

III. PROPOSED METHOD

In general, the proposed method combines RRT*, Fast-RRT, and GWO in some ways. This framework is based on $improvedRRT$ preserving workflow adopted from Fast-RRT and involves the RRT* rewiring process. GWO is used to find the best position for beacons on the formed path. The beacon in question is a node obtained after path reduction optimization is applied (it is called the target node). In this study, the reduction in question is by applying triangular inequality. This aims to provide a reasonable and feasible path variant. Triangular inequality and continued with GWO replaces fusion which is heavily influenced by the alternative suboptimal paths obtained. Therefore, in addition to being fast, it also guarantees an increase in optimality. The advantage of this technique is that it is fast because it does not require a lot of sampling and is also not affected by how optimal the initial path is obtained. The flow diagram of this process can be seen in Fig. 1.

Referring to Fig. 1, RRT with fast sampling and random direction is initially performed. This RRT is adopted from $improvedRRT$ in Fast-RRT and is performed to obtain the initial path in a certain number of sampling times. In this step, the use of adopted rewiring is also applied. This is to support that the results obtained also show good optimality. In this study, based on this rewiring, it is expected that the suboptimal path is obtained so that the optimization work will be better. The process of applying these methods continues until the initial path is found. After the initial path is found, the next process is path optimization. With the focus no longer depending on the number of samplings allowed or even the number of samplings approaching infinity, the optimization carried out in this study minimizes sampling and focuses on moving the node position to a better place. To reduce the computational cost, in this study the initial path is optimized by applying triangular inequality before GWO optimization is carried out. The optimization process using GWO takes place with the number of generations adjusted to the number of remaining samplings. The beacons in question are nodes that connect the starting point to the destination point. Simultaneously, a number of nodes obtained are optimized with the aim of determining the best position that produces the shortest obstacle-free path. At the beginning of GWO, one by one the node positions are identified, as a reference in generating points around the beacon randomly, at the distribution radius r_{disb}. Where the number of beacons determines the number of decision variables at this stage. By determining a number of candidate solutions and random generation, the initial population is obtained. The number of this population is defined as $Npop$. After obtaining this initial population, the optimization process runs with each generation.

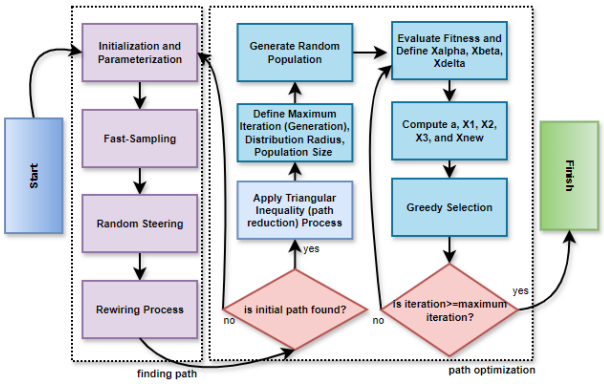


Fig. 1. Flowchart of the proposed method

Algorithm 3 – Proposed Method

Input: $x_{init}, x_{goal}, Map, Npop$
Output: A path T_{opt} from x_{init} to x_{goal}
for $= i \ldots N$ do
    $x_{rand} \leftarrow FastSample(Map)$
    $x_{near} \leftarrow nearest(T, x_{rand})$
    $x_{new} \leftarrow randomsteering(x_{near}, x_{rand})$
    $E_t \leftarrow Edge(x_{new}, x_{near})$
    if $collisionFree(E_t, Map)$ then
        $T \leftarrow addNode(x_{new})$
        if $x_{new} \in X_{goal}$ then
            $T_{init} \leftarrow T$
            $pathfoundat = i;$
            state=1
            $iteration = Npop$
        endif
    endif
endfor
$maxit = (N - pathfoundat)$
If state==1
    $idx \leftarrow getrandIdx(T_{init})$
    $x_{selected} \leftarrow getnode(T_{init}(idx))$
    $T_{opt} \leftarrow GWO(T_{init})$
    $T_{opt} \leftarrow pathOptimize(T_{opt})$
end

As a note, not only considering the distance, the fitness or objective function applied to this GWO also considers the feasibility of the path. Each time the fitness is evaluated, the candidate solution is considered invalid when there is a collision with any obstacle. Based on the generated random population, $\vec{X}_\alpha, \vec{X}_\beta, \vec{X}_\gamma$ are determined by first evaluating the fitness for all individuals (i.e. candidate solution, which collects information on all beacon coordinates. Where $\vec{X}_\alpha$ is the best gray wolf in the pack, $\vec{X}_\beta$ is the second best, $\vec{X}_\gamma$ is the third best. Based on these results, the next determination of a is done by applying the formula described in (2). With a, $\vec{X}_1, \vec{X}_2, \vec{X}_3$ can then be determined using (9), (10), and (11), respectively. By maintaining this process, the final solution is obtained. In each generation, the increase is indicated by a decrease in the cost path. And as a termination criterion, the maximum iteration is used as the main reference for stopping the optimization process or not. If the iteration is the same as the maximum iteration, the optimization process is complete, and the final path is obtained. All steps of the proposed method can be clearly seen in Algorithm 3. Moreover, the optimization step is conducted when GWO obtained a new solution. Conceptually, the path reduced in this optimization step by firstly checking the connection from the current waypoint to the next one. This method iterates by starting at x_{goal} and moving to x_{init} while sequentially observing the direct connection to the parent up until the connection of two separate nodes is deemed to be a collision with the obstruction. No additional nodes can be joined directly if the order of observations reaches x_{init}. This method follows the rule of triangular inequality and can be seen in Algorithm 5.

Algorithm 4 – GWO

```
Input: T_init, iteration, maxit, Npop
Output: T_opt
agents ← ballshapedDist(x_selected, Npop)
while iteration < maxit
        X_α, X_β, X_δ ← evaluate(agents)
        for i = 1: length (agents)
                D_α, D_β, D_δ
                X_1, X_2, X_3
        end
        iteration = interation + Npop
        agents ← greddySel(X_new)
end
T_opt ← agents
```

Assuming the initial path as input, the first time before GWO is applied, the first step in this optimization is to determine a node with a random index, called $x_{selected}$, which is a node that is not a starting point and not a goal point. This determination is shown in command line 21 in Algorithm 3. Referring to the maximum iterations allowed, $x_{selected}$ is optimized using GWO. In the initial setting, this maximum iteration is the number of samples that are still left and can be used which is proportional to the specified $Npop$. $Npop$ is the number of candidate solution swarms that can be determined. $Npop$ is the number of candidate solution swarms that can be determined. The larger the specified value, the fewer optimization iterations using GWO. This is because $Npop$ becomes an addition to the iteration when GWO is successfully applied in one cycle. The termination criteria used to represent the GWO iteration limitation is $maxit$ which is determined previously based on the sampling iterations required to obtain the initial path (see line 18 in Algorithm 3. By knowing $x_{selected}$, agents who are candidate solution flocks (representing wolves) are generated by applying Uniform Spherical Distribution centred on $x_{selected}$.

Algorithm 5 – $pathOptimize$

```
Input: T_opt, x_init, x_goal
Output: T_final
wolves ←
T_final(1) ← x_goal
x_current = x_goal
for i = 2: length(T_opt) do
        if obstacleFree(x_current, T_opt(i))
                continue
        else
                T_final = append(T_opt(i − 1))
                x_current = T_opt(i − 1)
        end
end
return T_final
```

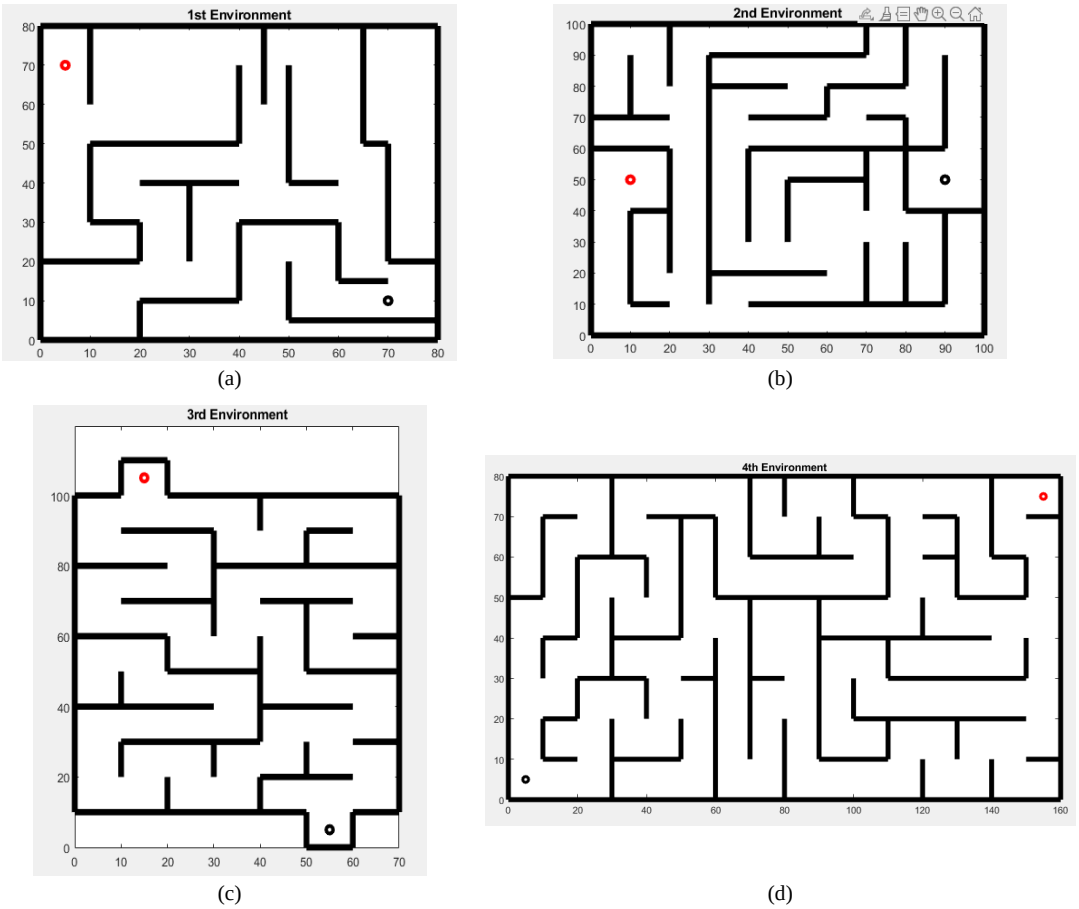
The number of wolves is determined by the desired population size, in this case termed $Npop$. Furthermore, by considering the termination criteria and (1) – (11), GWO repositions $x_{selected}$. This step begins by determining the first best wolf, second best, third best with the notation $\vec{X}_\alpha, \vec{X}_\beta, \vec{X}_\delta$, respectively (see command line 5 in Algorithm 4). This determination applies to the objective function which is the path cost when $x_{selected}$ is substituted with the candidate solution with the lowest value from the agents after optimization. Not only referring to the cost path, candidate solutions are considered valid if the solution node does not collide if connected with $x_{nieg1} \leftarrow T_{init}(idx + 1)$ and does not collide with obstacles in the environment if connected with $x_{nieg2} \leftarrow T_{init}(idx - 1)$. idx referred to is the index used to determine $x_{selected}$ in Algorithm 3, line 20. Furthermore, when new $agents$ are successfully determined, $\vec{X}_{new}$ determined by applying (12). And finally in one optimization cycle, greedy selection is performed before $\vec{X}_{new}$ will be substituted into $\vec{X}_{certain}$ in $agents$. By maintaining this stage, when the termination criteria are met, GWO provides agents whose one of the candidate solutions is the best node to replace $x_{selected}$. These steps result in a new node arrangement with better optimality and give an optimized path after GWO is operated (note line 23 in Algorithm 3 and its description in Algorithm 5). This path planning algorithm can improve the efficiency of autonomous robots in exploration, especially for global path planning in static environments. The perception of the environment and the robot's position must be generated by SLAM well before the algorithm is applied. If the environment is dynamic, the robot must also be able to avoid obstacles. Ethical aspects such as security, data privacy, and the freedom of algorithms from bias need to be considered to avoid risks to humans and ensure fair decisions.

IV. RESULT AND DISCUSSION

At this section, the robot is assumed to know all the information of the environment and the starting as well as the goal point are given initially. Moreover, the environment is considered static, and the algorithm will solve the global path planning. There will be several environments used to evaluate the work of algorithms including the RRT*, Fast-RRT, RRT*-Smart, and GWO-tuned Fast-RRT as the proposed method. These environments are presented on Fig. 2.

All static environments shown in Fig. 2 are environments that researchers usually use in their research to test the methods that have been developed. 1st Environment in Fig. 2(a) is used to examine the ability of the method to pass through the narrow channel. Environment shown in Fig. 2(b) is the environment used to test RRT*-Connect on [34]. Environment 2(c) is one of the environments used to test RRT*-Smart on [54]. While the environment shown in Fig. 2(d) is an environment for testing methods to solve path planning problems in a large environment and there is only one possible solution.

Based on the level of complexity, the 1st Environment to 4th Environment have different characteristics from each other. 1st Environment complexity is represented by a narrow corridor and there is only one unique solution to produce a path connecting the starting and goal points. In 2nd Environment, its complexity is shown by the existence of an indirect path that requires more challenging exploration compared to 1st Environment. 3rd Environment has the same level of complexity as 2nd Environment, namely there are narrow corridors with lots of space that have the potential to make exploration spread. While in 4th Environment, not only does it show a long distance between the starting position and the goal, but it also only has 1 unique solution. As for the initial and final positions for each environment are determined as follows: in the 1st Environment, the initial position is at position (70, 10) and the final position is at position (5, 70); in the 2nd Environment, the initial position is placed at coordinates (90, 50) and the final position is placed at coordinates (10, 50); in the 3rd Environment the initial and final positions are placed at (155, 5) and (15, 105), respectively; the initial and final positions in the 4th Environment are placed at (5, 5) and (155, 75), respectively. Testing of all methods was conducted on a PC with the following specifications: Intel Core i7-14700K 3.40 GHz processor, 64.0 GB RAM, 64-bit operating system, RTX 3080 10GB GPU, and 1TB SSD. In the test, all methods were run alternately to solve the path planning problem in the 1st – 4th Environment with a sampling rate of 10000. All methods were compared based on the working speed, indicated by the completion time in seconds, and the optimality of the resulting path, represented by the cost path in units of length. Furthermore, eps, the maximum distance for placing x_{new} in each method is determined to be 5 units of length.



(a) (b) (c) (d)

Fig. 2. Used environment

In the first experiment, RRT*, Fast-RRT, Informed-RRT*, RRT*-Smart, and Proposed methods were applied to solve the 1st environmental problem. The results of the application are visually shown in Fig. 3. Based on the solution of the 1st Environment problem, as seen in Fig. 3(a) and Fig. 3(c), RRT* and RRT*-Smart cannot solve the problem. This is because both apply conventional exploration that is not directed and limited to narrow corridors. Meanwhile, with the presence of random steering after applying fast sampling, Fast-RRT and the proposed method can easily and quickly pass through narrow corridors when conducting exploration (see Fig. 3(b) and Fig. 3(d)). The effectiveness of both approaches used to carry out the exploration process is also shown in the number of sampling repetitions of 5000 to get the initial path. Based on this test, not only the optimality of the path is indicated by a shorter cost path than Fast-RRT, the overall planning work speed for the proposed method is also better than Fast-RRT. This is shown by the time consumed in planning, namely, 8.9079 seconds, while Fast-RRT requires 104.5422 seconds.

Next, testing and comparison of all methods are carried out in solving the problem in the 2nd Environment. While maintaining the same parameterization stages as the testing in the 1st Environment, the test results of RRT*, Fast-RRT, RRT*-Smart, and the proposed method are shown in Fig. 4. Referring to the test on the 2nd Environment, it can be seen that all methods can solve the path planning problem.

However, if observed, each method shows different test results from one another. Based on the results shown in Fig. 4(a), RRT* can solve the problem in 665.0656 seconds with a path cost of 278.9263 units of length. While for Fast-RRT, as seen in Fig. 4.(b) shows a longer cost path, namely 287.1462 with a time required of 197.7759 seconds.

Although fast, the path produced by Fast-RRT is not more optimal than RRT*. This strengthens the statement at the beginning that the path fusion in Fast-RRT is not able to work well. While RRT* by utilizing the rewiring and conventional exploration stages can provide improvements to the initial path, path fusion cannot. This is because the possibility of obtaining a unique path other than the path that was previously obtained is very small. Logically, this statement can be accepted. In narrow corridors, by only maintaining fast sampling and random steering in exploration, the potential for obtaining random nodes in areas that approach the path is very low. This phenomenon is also supported by the results shown by RRT*-Smart in Fig. 4(c). The optimization process of RRT*-Smart that applies smart sampling is more ideal when the environment is narrow corridors. Thus, this approach is more ideal than maintaining the limitation of sampling areas such as path fusion with the exploration process applying fast-sampling and random steering. In short, the experiments and tests in the second environment show that in the case of narrow corridors, Fast-RRT can only provide suboptimal paths. Based on the analysis of the three methods, the basis for the development of the proposed method is getting stronger. This is also proven by the results showing that the path cost generated by the proposed method is 267.7008 which is very short compared to the previous results. In addition, the speed of getting the final path also shows that the proposed method has a good convergence rate, with a very fast completion time of 14.4501 seconds.

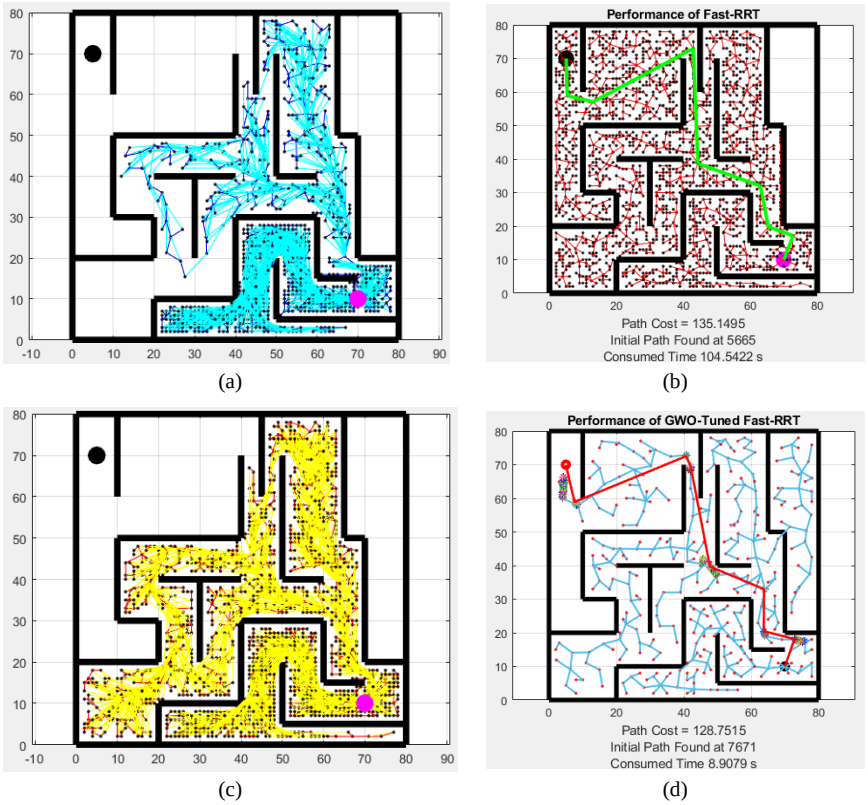


Fig. 3. The performance of different algorithms to solve path planning problem in 1st Environment (a) RRT*, (b) Fast-RRT, (c) RRT*-Smart, (d) Proposed Method

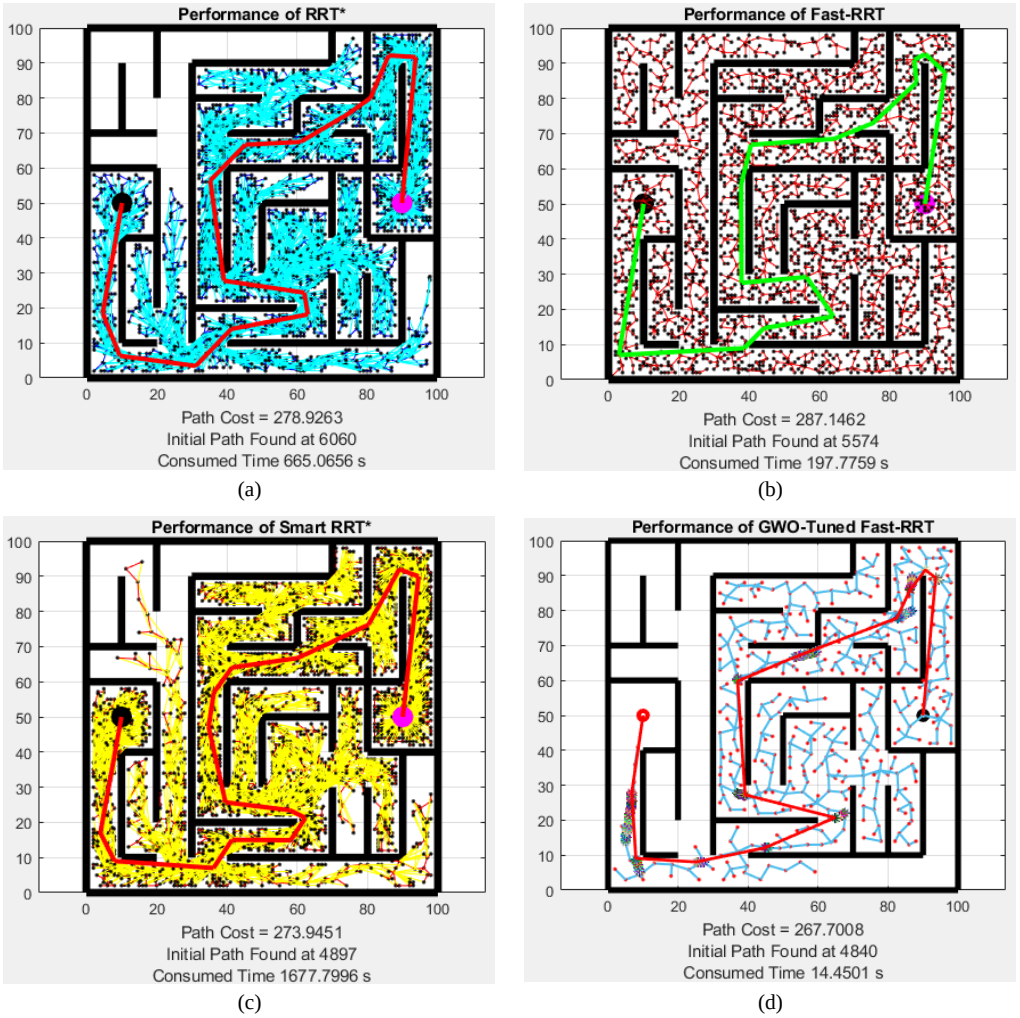


Fig. 4. The performance of different algorithms to solve path planning problems in 2nd Environment (a) RRT*, (b) Fast-RRT, (c) RRT*-Smart, (d) Proposed Method

Next, all methods are tested to solve the problem in the 3rd Environment. Based on this test, the results are shown in Fig. 5. The results show that all methods can solve the path planning problem. However, each method has different performance from each other. Referring to Fig. 5(a), RRT* produces a cost path of 182.7306 with a completion time of 1545.9285 seconds. This duration shows that RRT* is slow in solving path planning problems in a relatively large area. The slowness of this solution is also indicated by the number of samples of 4617 needed to determine the initial path. This result is like RRT*-Smart which also takes a long time to solve the problem. RRT*-Smart takes 2383.5985 seconds to solve the problem where the initial path is found in 4322 sample repetitions. The cost path generated through the application of RRT*-Smart is 183.5356 units of length. These two results again show that exploration in a conventional and undirected way makes the initial path slow to find. Nevertheless, the results given have shown that path optimization can run in sufficient time. On the other hand, it is seen that smart sampling, which is carried out periodically in sufficient duration, turns out to show that smart sampling is not significantly different from conventional optimization methods. This statement supports the similar results obtained between RRT* and RRT*-Smart. Although at first glance, the results obtained from RRT* or RRT*-Smart are sufficient, the time required for planning is a fundamental concern. The effectiveness of the method considers the speed in carrying out the planning. Referring to this statement, Fast RRT and the proposed method are relevant to meet the criteria for the required working time.

Although Fast-RRT significantly shows faster working time and better path cost results, its performance is not better than the proposed method. Fast-RRT takes 179.7521 seconds to produce a path with a cost of 180.3921. While the proposed method shows an extremely better performance. The resulting Path Cost is 161.4539 units of Length, with the time required being 6.572 seconds. Thus, the proposed method outperforms the RRT*, Fast-RRT, and RRT*-Smart methods in the 3rd Environment.

By conducting tests on the 1st, 2nd, 3rd Environment, the ability to solve problems in narrow channels, and areas with many corridors and partitions, the proposed method has shown very good performance and outperforms other methods. Furthermore, to test its ability to solve problems in areas where there is only one unique and tricky solution (because many corridors and partitions interfere with exploration), testing on the 4th Environment was carried out. This test still uses the same parameterization as the previous test. Based on the results shown in Fig. 6, RRT* and RRT*-

Smart cannot solve the problem of path planning in a large area with partitions, while the number of samples taken is limited. This shows that the exploration process of both methods is slow due to the uncertainty of the distribution direction. Slowness in the exploration process certainly has a negative impact when the search space faced is large. Repetition in areas that have been explored potentially occurs due to this uncertainty. This statement causes RRT* and RRT*-Smart to be unable to solve the problem. Although RRT*-Smart has the advantage of being able to perform better path optimization, this step can only be done when the initial path is obtained. Unlike the results of RRT* or RRT*-Smart, Fast-RRT and the proposed method successfully solve the problem with a cost path of 290.4575 and 288.118 units of length, respectively. In the process of obtaining this path, Fast-RRT requires a total duration of 166.3818 seconds, and the proposed method requires 30.6501 seconds. Thus, the proposed method offers a better path cost, which means that optimality is met. Considering the difference in path cost between Fast-RRT and the proposed method, the average increase in path optimality is 5.67%. Furthermore, referring to the time required is also short, the proposed method has a better convergence speed. And considering the time difference to Fast-RRT, the proposed method shows an average increase in working time of 90.25%. Up to this point, the proposed method consistently shows its superiority compared to the previous methods, RRT*, Fast-RRT, and RRT*-Smart both in terms of convergence rate and optimality.

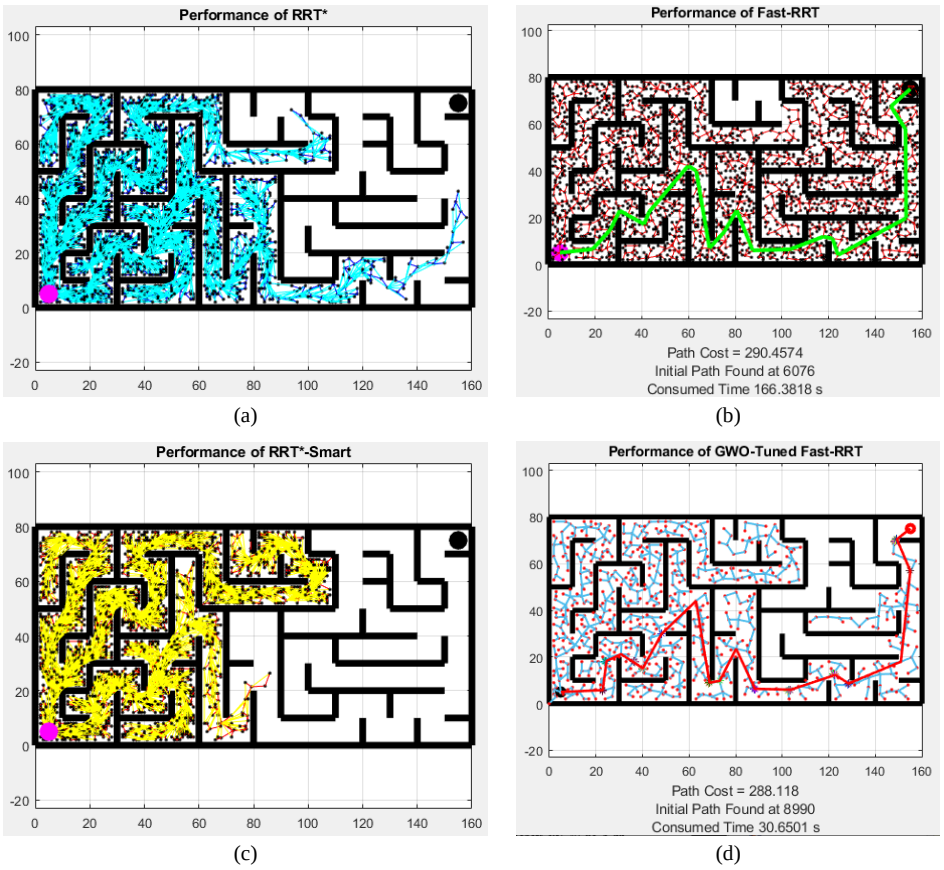


Fig. 5. The performance of different algorithms to solve path planning problem in 4th Environment (a) RRT*, (b) Fast-RRT, (c) RRT*-Smart, (d) Proposed Method

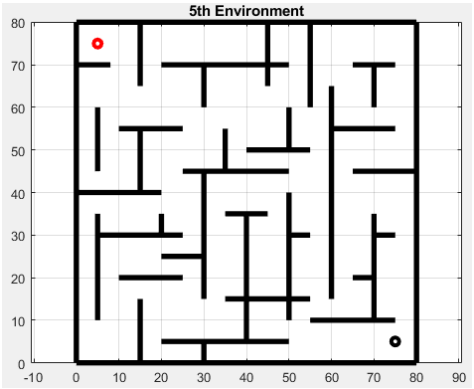


Fig. 6. 5th Environment

To further demonstrate the efficiency of GWO-tuned Fast RRT. It was performed to solve the 5th Environment as shown in Fig. 7. In this test, the starting and goal points were placed at (75, 5) and (5, 75), respectively. Parameterization was done by applying the test to other previous environments. The GWO-tuned Fast RRT test was performed with 14 repetitions to measure the consistency of the method in solving the path planning problem. Not only referring to the working speed, the optimality of the path was considered in this test. The results of the test are shown in Fig. 7 and explained in Table I.

Referring to Table I and Fig. 7, the time required to solve the problem is uncertain. This is because the fast-sampling and random steering contained in the proposed method make exploration have no definite direction (and only focus on reaching unexplored areas). In addition to time, the resulting cost path also shows differences between one test and another. Based on the theory, this condition occurs due to the need for different sampling amounts in determining the initial path. As discussed earlier, the faster the initial path is determined, the more freedom the method provides to perform optimization. Maximum optimization is obtained when the number of samples is high, and vice versa. So naturally, the initial path that is found earlier has the potential to show more optimal results. Furthermore, to measure how effectively the method can be applied, the results of the tests carried out 14 times were analyzed. Based on the overall performance, the average time required to solve the problem is 7.754335714 seconds and produces the path cost that varies with the range [120.1041, 134.7460].

TABLE I. THE PERFORMANCE OF GWO-TUNED FAST-RRT FOR 15 REPETITIVE SOLVING PATH PLANNING PROBLEM OF 5TH ENVIRONMENT

No	Path Cost	Initial Path Found	Consumed Time
1	134.1819	1463	2.6529
2	130.6204	6833	6.5168
3	126.3837	3449	5.8087
4	131.3522	3372	10.513
5	129.4169	2960	10.9536
6	132.6336	7172	12.3564
7	120.1041	5198	7.3613
8	133.9939	6978	11.5184
9	120.8056	3937	8.9688
10	134.7460	1217	2.3513
11	133.0216	2888	6.8522
12	122.8171	3400	7.1297
13	132.7946	3921	7.0770
14	129.5982	3333	8.5006

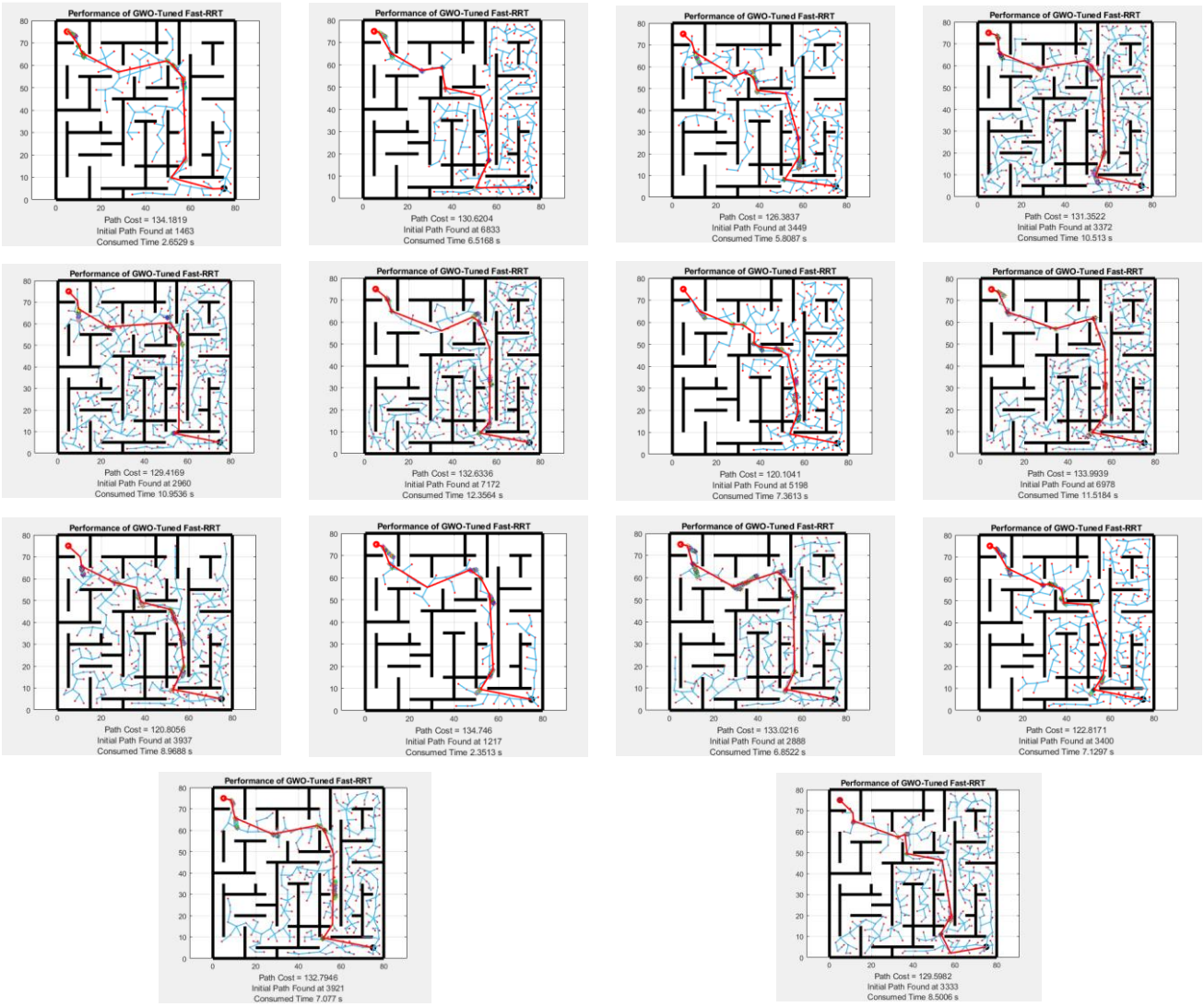


Fig. 7. Performance of GWO-Tuned Fast RRT for Solving Path Planning Problem in 5th Environment for 14 times of repetitio

V. CONCLUSION

This study successfully generates optimal paths in a short time by applying GWO to Fast RRT. The experimental results show that the path cost increases by an average of 5.67%, while the overall working speed increases by 90.25% compared to the previous method. For future work, this study can be further tested in dynamic and real-time environments and improve its scalability. The theoretical contribution of this study is to add the ability to adjust beacons that have been generated by $improvedRRT$ (on Fast RRT) and process rewiring, by utilizing GWO. Instead of using fast-optimal which includes path fusion and path optimization, after the initial path is found, GWO is applied to perform optimization. First, the node is determined based on the index obtained randomly instead of the starting and goal points. This node is called $x_{selected}$ which will be optimized by GWO by involving the previous and subsequent nodes based on the path that has been found. In other words, GWO changes the position of beacon nodes with a focus on improving the optimality of the path (shortening the path).

ACKNOWLEDGMENT

This research is supported by Universitas Mercu Buana, Jakarta.

REFERENCES

[1] H. Suwoyo *et al.*, "Maximum likelihood estimation-assisted ASVSF through state covariance-based 2D SLAM algorithm," *Telkomnika Telecommun. Comput. Electron. Control*, vol. 19, no. 1, pp. 327–338, 2020, doi: 10.12928/TELKOMNIKA.V19I1.16223.

[2] Y. Tian, H. Suwoyo, W. Wang, D. Mbemba, and L. Li, "An AEKF-SLAM Algorithm with Recursive Noise Statistic Based on MLE and EM," *J. Intell. Robot. Syst. Theory Appl.*, vol. 97, pp. 339-355, 2020, doi: 10.1007/s10846-019-01044-8.

[3] Y. Tian, H. Suwoyo, W. Wang, and L. Li, "An ASVSF-SLAM Algorithm with Time-Varying Noise Statistics Based on MAP Creation and Weighted Exponent," *Math. Probl. Eng.*, vol. 2019, pp. 28–34, 2019, doi: 10.1155/2019/2765731.

[4] T. Bailey and H. Durrant-whyte, "Simultaneous Localisation and Mapping (SLAM): Part II State of the Art," *Robotics and Automation Magazine*, pp. 1–10, 2006.

[5] W. Burgard, C. Stachniss, K. Arras, and M. Bennewitz, "Introduction to Mobile Robotics SLAM : Simultaneous Localization and Mapping What is SLAM ?," *University of Freiburg*, 2012.

[6] I. Karabegović and V. Doleček, *Mobile Robotics*. in Detecting and Mitigating Robotic Cyber Security Risks, 2017, doi: 10.4018/978-1-5225-2154-9.ch016.

[7] R. Kuemmerle. *State Estimation and Optimization for Mobile Robot Navigation*. Doctoral dissertation, Verlag nicht ermittelbar, 2013.

[8] I. Noreen, A. Khan, and Z. Habib, "A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 16, no. 10, pp. 20–27, 2016.

[9] J. Cong, J. Hu, Y. Wang, Z. He, L. Han, and M. Su, "FF-RRT*: a sampling-improved path planning algorithm for mobile robots against concave cavity obstacle," *Complex Intell. Syst.*, vol. 9, no. 6, pp. 7249–7267, Dec. 2023, doi: 10.1007/s40747-023-01111-6.

[10] L. Zhu, P. Duan, L. Meng, and X. Yang, "GAO-RRT*: A path planning algorithm for mobile robot with low path cost and fast convergence," *AIMS Math.*, vol. 9, no. 5, pp. 12011–12042, 2024, doi: 10.3934/math.2024587.

[11] N. Kumar and A. Kumar, "Multi-Point Path Planning Using Bidirectional Path Search In Static And Dynamic Environments," *Educ. Adm. Theory Pract.*, Apr. 2023, doi: 10.53555/kuey.v29i4.6468.

[12] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Syst. Appl.*, vol. 227, p. 120254, Oct. 2023, doi: 10.1016/j.eswa.2023.120254.

[13] A. Orthey, C. Chamzas, and L. E. Kavraki, "Sampling-Based Motion Planning: A Comparative Review," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 7, no. 1, pp. 285–310, Jul. 2024, doi: 10.1146/annurev-control-061623-094742.

[14] G. Sotirchos and Z. Ajanovic, "Search-based versus Sampling-based Robot Motion Planning: A Comparative Study," J *arXiv preprint arXiv:2406.09623*, 2024, doi: 10.48550/arXiv.2406.09623.

[15] Z. M. Al-Zubaidi, S. Ay, and M. Al-Khafaji, "A Comparative Study of Various Path Planning Algorithms for Pick-and-Place Robots," *Research Square*, pp. 1-27, 2023, doi: 10.21203/rs.3.rs-2808265/v1.

[16] Z. Feng, L. Zhou, J. Qi, and S. Hong, "DBVS-APF-RRT*: A global path planning algorithm with ultra-high speed generation of initial paths and high optimal path quality," *Expert Syst. Appl.*, vol. 249, p. 123571, Sep. 2024, doi: 10.1016/j.eswa.2024.123571.

[17] S. Ganesan and S. K. Natarajan, "A novel directional sampling-based path planning algorithm for ambient intelligence navigation scheme in autonomous mobile robots," *J. Ambient Intell. Smart Environ.*, vol. 15, no. 3, pp. 269–284, Sep. 2023, doi: 10.3233/AIS-220292.

[18] X. Jiang, Z. Wang, and C. Dong, "A Path Planning Algorithm Based on Improved RRT Sampling Region," *Comput. Mater. Contin.*, vol. 80, no. 3, pp. 4303–4323, 2024, doi: 10.32604/cmc.2024.054640.

[19] X. Wang, Y. Feng, J. Tang, Z. Dai, and W. Zhao, "A UAV path planning method based on the framework of multi-objective jellyfish search algorithm," *Sci. Rep.*, vol. 14, no. 1, p. 28058, Nov. 2024, doi: 10.1038/s41598-024-79323-0.

[20] M. Faroni, N. Pedrocchi, and M. Beschi, "Adaptive hybrid local–global sampling for fast informed sampling-based optimal path planning," *Auton. Robots*, vol. 48, no. 2–3, p. 6, May 2024, doi: 10.1007/s10514-024-10157-5.

[21] Y. Shi, S. Huang, and M. Li, "An Improved Global and Local Fusion Path-Planning Algorithm for Mobile Robots," *Sensors*, vol. 24, no. 24, p. 7950, Dec. 2024, doi: 10.3390/s24247950.

[22] I. A. Hassan, I. A. Abed, and W. A. Al-Hussaibi, "Path Planning and Trajectory Tracking Control for Two-Wheel Mobile Robot," *J. Robot. Control JRC*, vol. 5, no. 1, pp. 1–15, Dec. 2023, doi: 10.18196/jrc.v5i1.20489.

[23] H. Suwoyo, A. Adriansyah, J. Andika, A. U. Shamsudin, and M. F. Zakaria, "An Integrated Rrt*Smart-a* Algorithm for Solving the Global Path Planning Problem in a Static Environment," *IIUM Eng. J.*, vol. 24, no. 1, pp. 269–284, 2023, doi: 10.31436/iiumej.v24i1.2529.

[24] S. Al-Ansarry and S. Al-Darraji, "Hybrid RRT-A*: An Improved Path Planning Method for an Autonomous Mobile Robots," *Iraqi J. Electr. Electron. Eng.*, vol. 17, no. 1, pp. 1–9, 2021, doi: 10.37917/ijeee.17.1.13.

[25] W. Xin, L. Wanlin, F. Chao, and H. Likai, "Path Planning Research Based on An Improved A* Algorithmfor Mobile Robot," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 569, no. 5, p. 052044, Jul. 2019, doi: 10.1088/1757-899X/569/5/052044.

[26] D. Fan and P. Shi, "Improvement of Dijkstra's algorithm and its application in route planning," in *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 1901–1904, 2010, doi: 10.1109/FSKD.2010.5569452.

[27] M. A. Javaid, "Understanding Dijkstra Algorithm," *SSRN Electron. J.*, 2013, doi: 10.2139/ssrn.2340905.

[28] M. A. Khan, "A Comprehensive Study of Dijkstra's Algorithm," *SSRN Electron. J.*, 2023, doi: 10.2139/ssrn.4559304.

[29] A. Tilanterä, J. Sorva, O. Seppälä, and A. Korhonen, "Students Struggle with Concepts in Dijkstra's Algorithm," in *Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1*, pp. 154–165, 2024, doi: 10.1145/3632620.3671096.

[30] D. Rachmawati and L. Gustin, "Analysis of Dijkstra's Algorithm and A∗ Algorithm in Shortest Path Problem," *J. Phys. Conf. Ser.*, vol. 1566, no. 1, 2020, doi: 10.1088/1742-6596/1566/1/012061.

[31] A. Rk, P. Reddy, and M. Yamuna, "Research On The Optimization Of Dijkstra's Algorithm And Its Applications," *International Journal of Science, Technology & Management*, vol. 4, no. 1, pp. 304-309, 2015.

[32] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, and I. Ahmedy, "Hybrid RRT: A semi-dual-tree RRT-based motion planner," *IEEE Access*, vol. 8, pp. 18658–18668, 2020, doi: 10.1109/ACCESS.2020.2968471.

[33] H. Suwoyo, A. Burhanudin, Y. Tian, and J. Andika, "Problem solving path planning and path tracking in a 3 DOF hexapod robot using the RRT* algorithm with path optimization and Pose-to-Pose," *Sinergi Indones.*, vol. 28, no. 2, pp. 265–276, 2024, doi: 10.22441/sinergi.2024.2.007.

[34] S. Klemm *et al.*, "RRT*-Connect: Faster, asymptotically optimal motion planning," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1670–1677, 2015, doi: 10.1109/ROBIO.2015.7419012.

[35] I. B. Jeong, S. J. Lee, and J. H. Kim, "RRT*-Quick: A motion planning algorithm with faster convergence rate," *Adv. Intell. Syst. Comput.*, vol. 345, pp. 67–76, 2015, doi: 10.1007/978-3-319-16841-8_7.

[36] I. Noreen, A. Khan, K. Asghar, and Z. Habib, "A path-planning performance comparison of RRT*-AB with MEA* in a 2-Dimensional Environment," *Symmetry*, vol. 11, no. 7, 2019, doi: 10.3390/sym11070945.

[37] B. Liao, F. Wan, Y. Hua, R. Ma, S. Zhu, and X. Qing, "F-RRT*: An improved path planning algorithm with improved initial solution and convergence rate," *Expert Syst. Appl.*, vol. 184, pp. 115457–115457, 2021, doi: 10.1016/j.eswa.2021.115457.

[38] A. Perez, S. Karaman, A. Shkolnik, E. Frazzoli, S. Teller, and M. R. Walter, "Asymptotically-optimal path planning for manipulation using incremental sampling-based algorithms," *IEEE Int. Conf. Intell. Robots Syst.*, pp. 4307–4313, 2011, doi: 10.1109/IROS.2011.6048640.

[39] A. H. Qureshi and Y. Ayaz, "Potential functions based sampling heuristic for optimal path planning," *Auton. Robots*, vol. 40, no. 6, pp. 1079–1093, 2016, doi: 10.1007/s10514-015-9518-0.

[40] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2997-3004, 2014, doi: 10.1109/IROS.2014.6942976.

[41] J. Chen, Y. Zhao, and X. Xu, "Improved RRT-Connect Based Path Planning Algorithm for Mobile Robots," *IEEE Access*, vol. 9, pp. 145988–145999, 2021, doi: 10.1109/ACCESS.2021.3123622.

[42] X. Li and Y. Tong, "Path Planning of a Mobile Robot Based on the Improved RRT Algorithm," *Appl. Sci.*, vol. 14, no. 1, p. 25, Dec. 2023, doi: 10.3390/app14010025.

[43] E. H. Bes. *Path Planning with Homotopic Constraints for Autonomous Underwater Vehicles*. Doctoral dissertation, Universitat de Girona, 2012.

[44] L. G. D. O. Veras, F. L. L. Medeiros, and L. N. F. Guimaraes, "Systematic Literature Review of Sampling Process in Rapidly-Exploring Random Trees," *IEEE Access*, vol. 7, pp. 50933–50953, 2019, doi: 10.1109/ACCESS.2019.2908100.

[45] W. Burzyński and W. Stecz, "Trajectory planning with multiplatform spacetime RRT*," *Appl. Intell.*, vol. 54, no. 19, pp. 9524–9541, Oct. 2024, doi: 10.1007/s10489-024-05650-4.

[46] A. Almalaq, K. Alqunun, R. Abbassi, Z. M. Ali, M. M. Refaat, and S. H. E. Abdel Aleem, "Integrated transmission expansion planning incorporating fault current limiting devices and thyristor-controlled series compensation using meta-heuristic optimization techniques," *Sci. Rep.*, vol. 14, no. 1, p. 13046, Jun. 2024, doi: 10.1038/s41598-024-63331-1.

[47] O. Maruyama and A. Chihara, "NWE: Node-weighted expansion for protein complex prediction using random walk distances," *Proteome Sci.*, vol. 9, 2011, doi: 10.1186/1477-5956-9-S1-S14.

[48] A. Felner *et al.*, "Partial-Expansion A* with Selective Node Generation," *Proc. AAAI Conf. Artif. Intell.*, vol. 26, no. 1, pp. 471–477, Sep. 2021, doi: 10.1609/aaai.v26i1.8137.

[49] J. Qi, Q. Yuan, C. Wang, X. Du, F. Du, and A. Ren, "Path planning and collision avoidance based on the RRT*FN framework for a robotic manipulator in various scenarios," *Complex Intell. Syst.*, vol. 9, no. 6, pp. 7475–7494, Dec. 2023, doi: 10.1007/s40747-023-01131-2.

[50] H. Qin, S. Shao, T. Wang, X. Yu, Y. Jiang, and Z. Cao, "Review of Autonomous Path Planning Algorithms for Mobile Robots," *Drones*, vol. 7, no. 3, p. 211, Mar. 2023, doi: 10.3390/drones7030211.

[51] C. Li, C. Wang, J. Wang, Y. Shen, and M. Q. H. Meng, "Sliding-window informed RRT∗: A method for speeding up the optimization and path smoothing," *2021 IEEE Int. Conf. Real-Time Comput. Robot. RCAR 2021*, pp. 141–146, 2021, doi: 10.1109/RCAR52367.2021.9517672.

[52] D. Wu, L. Wei, G. Wang, L. Tian, and G. Dai, "APF-IRRT*: An Improved Informed Rapidly-Exploring Random Trees-Star Algorithm by Introducing Artificial Potential Field Method for Mobile Robot Path Planning," *Appl. Sci. Switz.*, vol. 12, no. 21, 2022, doi: 10.3390/app122110905.

[53] Q. Zhou and G. Liu, "UAV Path Planning Based on the Combination of A-star Algorithm and RRT-star Algorithm," in *2022 IEEE International Conference on Unmanned Systems (ICUS)*, pp. 146–151, 2022, doi: 10.1109/ICUS55513.2022.9986703.

[54] F. Islam, J. Nasir, U. Malik, Y. Ayaz, and O. Hasan, "RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution," *2012 IEEE Int. Conf. Mechatron. Autom. ICMA 2012*, pp. 1651–1656, 2012, doi: 10.1109/ICMA.2012.6284384.

[55] Z. Wu, Z. Meng, W. Zhao, and Z. Wu, "Fast-RRT: A RRT-based optimal path finding method," *Appl. Sci. Switz.*, vol. 11, no. 24, 2021, doi: 10.3390/app112411777.

[56] Q. Gu, X. Li, S. Jiang, and H. Mora, "Hybrid Genetic Grey Wolf Algorithm for Large-Scale Global Optimization," *Complexity*, vol. 2019, 2019, doi: 10.1155/2019/2653512.

[57] J. Liu, X. Wei, and H. Huang, "An Improved Grey Wolf Optimization Algorithm and its Application in Path Planning," *IEEE Access*, vol. 9, pp. 121944–121956, 2021, doi: 10.1109/ACCESS.2021.3108973.

[58] N. Mittal, U. Singh, and B. S. Sohi, "Modified Grey Wolf Optimizer for Global Engineering Optimization," *Appl. Comput. Intell. Soft Comput.*, vol. 2016, 2016, doi: 10.1155/2016/7950348.

[59] J. Zhao and Z. M. Gao, "An improved grey wolf optimization algorithm with multiple tunnels for updating," *J. Phys. Conf. Ser.*, vol. 1678, no. 1, 2020, doi: 10.1088/1742-6596/1678/1/012096.

[60] J. A. Abdor-Sierra, E. A. Merchán-Cruz, F. A. Sánchez-Garfias, R. G. Rodríguez-Cañizo, E. A. Portilla-Flores, and V. Vázquez-Castillo, "Particle swarm optimization for inverse kinematics solution and trajectory planning of 7-dof and 8-dof robot manipulators based on unit quaternion representation," *J. Appl. Eng. Sci.*, vol. 19, no. 3, pp. 592–599, 2021, doi: 10.5937/jaes0-30557.

[61] S. Dereli and R. Köker, "IW-PSO approach to the inverse kinematics problem solution of a 7-DOF serial robot manipulator," *Sigma J Eng Nat Sci*, vol. 36, no. 1, pp. 77–85, 2018.

[62] R. Havangi, "Mobile robot localization based on PSO estimator," *Asian J. Control*, vol. 21, no. 4, pp. 2167–2178, 2019, doi: 10.1002/asjc.2004.

[63] B. Song, Z. Wang, and L. Zou, "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve," *Appl. Soft Comput.*, vol. 100, p. 106960, Mar. 2021, doi: 10.1016/j.asoc.2020.106960.

[64] J. Xin, Z. Li, Y. Zhang, and N. Li, "Efficient real-time path planning with self-evolving particle swarm optimization in dynamic scenarios," *Unmanned Systems*, vol. 12, no. 2, pp. 215-226, 2024.

[65] H. T. Najm, N. S. Ahmad, and A. S. Al-Araji, "Enhanced path planning algorithm via hybrid WOA-PSO for differential wheeled mobile robots," *Syst. Sci. Control Eng.*, vol. 12, no. 1, p. 2334301, Dec. 2024, doi: 10.1080/21642583.2024.2334301.

[66] A. J. Mohammed, K. I. Ghathwan, and Y. Yusof, "Optimal Robot Path Planning using Enhanced Particle Swarm Optimization algorithm," *Iraqi J. Sci.*, pp. 178–184, Jan. 2020, doi: 10.24996/ijs.2020.61.1.20.

[67] L. Zheng, W. Yu, G. Li, G. Qin, and Y. Luo, "Particle Swarm Algorithm Path-Planning Method for Mobile Robots Based on Artificial Potential Fields," *Sensors*, vol. 23, no. 13, p. 6082, Jul. 2023, doi: 10.3390/s23136082.

[68] I. D. Fahmizal, M. Arrofiq, H. Maghfiroh, H. P. Santoso, P. Anugrah, and A. Molla, "Path Planning for Mobile Robots on Dynamic Environmental Obstacles Using PSO Optimization," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)*, vol. 10, no. 1, pp. 166-172, 2024.

[69] T. Duckett, "A genetic algorithm for simultaneous localization and mapping," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 1, pp. 434–439, 2003.

[70] J. Ni, K. Wang, H. Huang, L. Wu, and C. Luo, "Robot path planning based on an improved genetic algorithm with variable length chromosome," *2016 12th Int. Conf. Nat. Comput. Fuzzy Syst. Knowl. Discov. ICNC-FSKD 2016*, pp. 145–149, 2016, doi: 10.1109/FSKD.2016.7603165.

[71] S. T. Lian, K. Marzuki, and Y. Rubiyah, "Tuning of a neuro-fuzzy controller by genetic algorithms with an application to a coupled-tank liquid-level control system," *Eng. Appl. Artif. Intell.*, vol. 11, no. 4, pp. 517–529, 1998, doi: 10.1016/s0952-1976(98)00012-8.

[72] Y. Li, J. Zhao, Z. Chen, G. Xiong, and S. Liu, "A Robot Path Planning Method Based on Improved Genetic Algorithm and Improved Dynamic Window Approach," *Sustainability*, vol. 15, no. 5, p. 4656, Mar. 2023, doi: 10.3390/su15054656.

[73] J. Wang, "Intelligent Path Planning of Mobile Robot Based on Genetic Algorithm," *J. Phys. Conf. Ser.*, vol. 2547, no. 1, p. 012001, Jul. 2023, doi: 10.1088/1742-6596/2547/1/012001.

[74] W. Rahmaniar and A. E. Rakhmania, "Mobile Robot Path Planning in a Trajectory with Multiple Obstacles Using Genetic Algorithms," *J. Robot. Control JRC*, vol. 3, no. 1, pp. 1–7, Jun. 2021, doi: 10.18196/jrc.v3i1.11024.

[75] F. Liu, S. Liang, and D. X. Xian, "Optimal Path Planning for Mobile Robot Using Tailored Genetic Algorithm," *TELKOMNIKA Indones. J. Electr. Eng.*, vol. 12, no. 1, pp. 1–9, Jan. 2014, doi: 10.11591/telkomnika.v12i1.3127.

[76] J. Liu, Z. Chen, Y. Zhang, and W. Li, "Path Planning of Mobile Robots based on Improved Genetic Algorithm," in *Proceedings of the 2020 2nd International Conference on Robotics, Intelligent Control and Artificial Intelligence*, pp. 49–53, 2020, doi: 10.1145/3438872.3439054.

[77] M. S. Abed, O. F. Lutfy, and Q. F. Al-Doori, "Adaptive weight grey wolf algorithm application on path planning in unknown environments," *Indones. J. Electr. Eng. Comput. Sci.*, vol. 27, no. 3, p. 1375, Sep. 2022, doi: 10.11591/ijeecs.v27.i3.pp1375-1387.

[78] L. Liu, L. Li, H. Nian, Y. Lu, H. Zhao, and Y. Chen, "Enhanced Grey Wolf Optimization Algorithm for Mobile Robot Path Planning," *Electronics*, vol. 12, no. 19, p. 4026, Sep. 2023, doi: 10.3390/electronics12194026.

[79] R. Kumar, L. Singh, and R. Tiwari, "Path planning for the autonomous robots using modified grey wolf optimization approach," *J. Intell. Fuzzy Syst.*, vol. 40, no. 5, pp. 9453–9470, Apr. 2021, doi: 10.3233/JIFS-201926.

[80] B. Tu, F. Wang, X. Han, and X. Fu, "Q-learning Guided Grey Wolf Optimizer for UAV 3D Path Planning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 7, 2024, doi: 10.14569/IJACSA.2024.0150747.