SINERGI Vol. 29, No. 2, June 2025: 473-484
http://publikasi.mercubuana.ac.id/index.php/sinergi
http://doi.org/10.22441/sinergi.2025.2.017

A Bidirectional-RRT*-Connect-Assisted RRT*-Smart for a path planning algorithm

Heru Suwoyo, Yudi Hastomi*, Julpri Andika
Department of Electrical Engineering, Faculty of Engineering, Universitas Mercu Buana, Indonesia

Abstract
Although Rapidly Exploring Random Tree Star (RRT) has been considered to be able to achieve convergence to an optimal solution, this method has a slow convergence speed and requires an infinite amount of time to produce a truly optimal solution. For this reason, RRT*-Smart which includes path optimization and intelligent sampling processes was introduced. Although the addition of these methods can directly complete infinite-duration RRT* searches, they will work once the initial path obtained with RRT* is available. The effectiveness of reducing the optimality time is determined by the initial path formed. If this path is not close to optimal, the path optimization and intelligent sampling process will take a long time, and vice-versa. For this reason, RRT*-Connect, which has the advantage of searching from two directions, is proposed in this study. The goal is to replace the RRT* algorithm to produce a more optimal initial formed path. Based on this approach, this method will be named Connect-RRT*-Smart. Several methods, such as RRT, RRT*, RRT*-Connect, and RRT*-Smart, are compared to see their performance in producing the feasible path. Regarding this comparative result, the proposed method shows better performance in terms of convergence speed and path optimality.*

Keywords:
Bidirectional-Connect-RRT-Smart;*
Global Path Planning;
RRT-Connect;*
RRT-Smart;*

Article History:
Received: October 2, 2024
Revised: November 23, 2024
Accepted: December 14, 2024
Published: May 15, 2025

Corresponding Author:
Yudi Hastomi
Electrical Engineering Department, Universitas Mercu Buana, Indonesia
Email: yudi.hastomi@gmail.com

INTRODUCTION

Path planning is one of the important techniques in developing autonomous robot navigation [1][2]. By solving this problem, the robot will be helped in determining the movement to reach a certain destination point [3]. There have been several types of path planning methods proposed so far. However, RRT*-Smart is claimed to be a method that can offer convergence speed and optimality simultaneously [4]. Different from its predecessors, RRT [5] or RRT* [6], RRT*-Smart involves path optimization and intelligent sampling methods [4, 5, 7]. The concept of these two additional methods is to explore an informed search space so that it is not strange if the path solution is better than RRT*. However, the two additional methods will only work if the path originally formed is available. Where this initial path will be obtained by applying conventional RRT*. Thus, the optimality that is not necessarily provided by RRT* will greatly affect the working speed of the two methods. In brief, if the initial path formed is sub-optimal, then there is a longer duration with a concentrated search space. This will improve the optimality of RRT*-Smart. However, this will very rarely happen because the RRT* exploration process applies to random sampling throughout the search area which searches undirected.

As the use of mobile robots increases in human life, the number of studies on the theme of path planning also increases. This is proven by the many solving methods introduced by researchers. There are two classifications based on how expansion is carried out, namely sampling-based methods and search-based methods. Search-based methods such as Breath First Search (BFS) [8], Depth First Search (DFS) [1][8], Dijkstra [9],

[10], and A* [11][12], offer a high degree of path optimality. However, it has a slow convergence speed. Meanwhile, sampling-based methods tend to have low optimality but can work quickly. The researchers consider overcoming the speed of the search method to be more difficult than increasing the optimality of the sampling-based method. This is relatively the basis that influences the rapid development of sampling-based methods. Rapidly Exploring Random Trees (RRT) is the method that it is considered to have initiated this development. As the name suggests, RRT works by generating a random tree consisting of vertices that represent positions in the robot or vehicle configuration space. This tree is generated randomly by adding new nodes based on their randomness in the configuration space, then connecting these nodes with the closest node that was created before. Even though it has been able to solve the path planning problem, the solution provided is far from optimal. This is caused by an undirected random sampling process. So, by implementing the node reconnection stage, RRT develops with the name RRT* [13]. With this reconnection process, RRT* can improve the solution with a recursive iteration process. However, in a large environment, the time required to produce an optimal solution is as if infinite. This became a problem that was then paid attention to. In an effort to increase speed, this method is often combined with methods such as Potential Field [14, 15, 16], Genetic Algorithm [12, 17, 18, 19], Ant Colony Optimization [20][21], Artificial Bee Colony [21], Particle Swarm Optimization [23][24], and Neural Network [25], to produce a directed search process. In addition, there is also a fast method called RRT*-Connect [26], which adopts the working principle of RRT*-Connect expansion while maintaining the node reconnection process. Meanwhile, to increase optimality, RRT* was developed by implementing triangle inequality. The use of a loose method like this is intended to optimize node determination when connection is made. So, the node connection provides the shortest path. Both methods specifically address certain limitations of the RRT* algorithm, focusing on improving either convergence speed or path optimality. However, they do not encompass a full solution to the broader challenges of RRT* path planning in complex environments. Based on this problem, a new method was introduced, namely RRT*-Smart.

Unlike RRT*, RRT*-Smart implements Path Optimization and intelligent sampling, which works on the second layer when the initial path formed is given. For the record, the initial path formed is the path generated by RRT* which is then used as the basis for the application of the two methods. However, only a path that is at least sub-optimal will allow RRT*-Smart to work fast with a high level of optimality. Thus, to ensure that the initial path formed has high optimality, there needs to be development with a focus on replacing RRT* in the initial RRT*-Smart process. Where in this proposed research, RRT* will be replaced with RRT*-Connect in providing the initial path. RRT-Connect (Rapidly Exploring Random Tree-Connect) is a path planning algorithm designed to accelerate the exploration of configuration spaces in single-query problems. It works by constructing two trees from the starting and destination nodes until they are connected, providing higher efficiency than traditional RRT. Its development includes variants such as RRT*-Connect that introduce asymptotic optimality, allowing the discovered path to get closer to the optimal one with more iterations. Recent research has also integrated goal bias and node optimization strategies to improve search efficiency and path quality, which have been widely applied in mobile robotics and dynamic environments. However, the challenges in finding optimal paths have encouraged further development, such as the integration of artificial intelligence to improve real-world adaptability [15, 26, 27].

Basically, RRT*-Smart improves the path quality when the initial path is obtained. However, this initial path was obtained using the conventional method adopted from RRT* which tends to be very slow [28]. Thus, RRT*-Connect and Bi-Directional expansion are applied to carry out the initial path determination process. The RRT*-Connect in question is a combined method between RRT-Connect and RRT*. This approach is feasible and scalable, considering the ability of RRT-Connect to perform two-way line generation from the starting point and the target point simultaneously. In addition, there is a rewiring stage adopted from RRT* in this approximated method, so that the optimality of the formed path is still considered. So, replacing RRT* with RRT-Connect and bi-directional expansion at the early exploration stage of RRT*-Smart, will naturally optimize paths and increase planning speed. This description implies the contribution of this research, namely, improving RRT*-Smart. Initially, a random node is generated by surrounding the environment. Based on this node, the expansion of the start and goal point is carried out simultaneously. The result of this expansion process will be a series of nodes managed at Ts and Tg. The success of this expansion is marked by the existence of new nodes that can be connected to each of the nearest nodes at Ts and Tg without collisions with any obstacles in the

environment. This process is done by adopting RRT* expansion work while maintaining rewiring processes. In addition, the connecting process is carried out referring to Tg. This is done by making a new node in Ts, q_{newA}, a reference for steering the process to get a new node as a child of the last new node in Tg, q_{newB}. This process is the existing Greedy Extending. Thus, this expansion will remain centralized at the same meeting point. This also becomes the basis for stopping the expansion process in one cycle. This process then repeats until the initial path is found. Furthermore, the stages in RRT*-Smart, path optimization and intelligent sampling, were adopted and maintained in operation. With stages like this, you get RRT*-Smart with a faster expansion process because there is bidirectional expansion and connection. Regarding this achievement, several methods including the proposed method are then performed to solve the same problem of global path planning. They are then analyzed and compared in terms of convergence speed and path optimality.

The rest of the paper is organized as follows: section II presents material and method supporting the work; section III discusses the proposed method and algorithm; section IV presents the result and discussion; Section V concludes this study.

METHOD

At this part, some methods are discussed including the Bidirectional-RRT*-Connect, RRT*-Smart. However, before they are presented the problem definition of global path planning is given initially. Sequentially, RRT*-Smart and RRT*-Connect are then compactly presented.

Problem Statement

Assuming that $X \in \mathbb{R}^n$ is the state space representation of the path planning problem, where $n \in \mathbb{N}$ is the dimensions of this space, then $X = \{X_{obs}, X_{free}\}$ or in another representation $X_{obs} \in X$ which is the obstacle area and $X_{free} \in X$ which represents free area in search space X. Furthermore, knowing the starting point $x_{init} \in X_{free}$ and the ending point/destination $x_{goal} \in X_{free}$, then the ideal path or feasible path can be defined as $\sigma: [0,T] \rightarrow X_{free}$: where $\sigma(0) = x_{initial}$ and $\sigma(T) \in X_{goal}$; where $X_{goal} = \{x \in X | \|x - x_{goal}\| < r\}$ where r is the radius of the calculated termination area from x_{goal}. Therefore, with reference to the description above, the feasibility of the path planning can be described as follows: Given a search space $X \in \mathbb{R}^n$, barrier-free area X_{free}, knowledge of the location of all obstacles X_{obs}, states at the starting point $x_{initial} \in X_{free}$, ending point/destination $x_{goal} \in X_{free}$ and the termination area around the end point $X_{goal} \in X_{free}$, how can the freeway be found $\sigma = [0,T] \in X_{free}$ with $\sigma(0) = x_{init}$ and $\sigma(T) \in X_{goal}$. Path Cost or path weight σ_{cost} is an important observation in a path planning whose value must be positive. Furthermore, referring to this defined problem and to providing a basis for the proposal, some basic theories covering RRT*-Connect, and RRT*-Smart are presented.

RRT*-Connect

RRT*-Connect as discussed in [24], is a combined form of RRT* and RRT-Connect. So, RRT*-Connect and RRT-Connect have differences. Even though they are different, both have the same exploration process, namely bidirectional search to form separate trees, namely Ta (Tree built starting from the start point) and Tb (Tree built starting from the goal point). The difference lies in the exploration, in RRT*-Connect every time q_{newA} and q_{newB} are obtained and added to each of Trees A and Trees B, the connection is updated by applying the rewiring process found in RRT*. This is done to increase the optimality of the resulting path. The algorithm 1 can be seen in Figure 1, where EXTEND* can be described as shown in Algorithm 2 (see Figure 2) and CONNECT* is presented as Algorithm 3 as shown in Figure 3.

The second important feature that RRT*-Smart introduced is intelligent sampling. Because this sampling is focused on ideal path beacon nodes, it is not like random sampling. Using a Biasing Radius b, it establishes a radius for intelligent exploration around beacons. As soon as RRT*-Smart finds a shorter path, it repeats the path optimization process to construct more beacon nodes.

RRT*-Smart thereby increases path cost and quickens path convergence. During the first path-finding phase, RRT*-Smart uses a traditional sampling method. Using the maximum and minimum bounds of its representation, this strategy will produce nodes in the search space at random. Applying a Euclidean distance evaluation to the set of nodes based on these random nodes yields the nearest node. Then, as a reference direction for creating new nodes, the direction from the closest node to a random node is ascertained. The computed distance between the random node and the closest node is used to determine the new node locations in addition to the reference direction. The new node is positioned as far away from the closest node as the calculated distance

permits if the distance is less than or equal to the designated reference distance.

However, the new node is positioned so that it is equal to the designated reference distance from the closest node. After the sampling procedure in line 7 is completed, Algorithm 2 in lines 9 and 10 represents this process. Following that, RRT*-Smart will keep running by keeping an eye on the new node. The new node is connected to the nearest node as an Edge E if its location differs from one of the obstacle's points of position and if the line that connects it to that node does not cross any lines from the obstruction. In RRT, this procedure is known as the wiring process. Additionally, the parent node z_{near} of the new node z_{new} will be temporarily assumed to be the nearest node. Next, the closeness of each of the new node's neighbours is assessed. As the parent node of the new node, the node with the closest distance will be referred to as z_{min}he new node is connected to the nearest node as an Edge E if its location differs from one of the obstacle's points of position and if the line that connects it to that node does not cross any lines from the obstruction. In RRT, this procedure is known as the wiring process. Additionally, the parent node z_{near} of the new node z_{new} will be temporarily assumed to be the nearest node. Next, the closeness of each of the new node's neighbors is assessed. As the parent node of the new node, the node with the closest distance will be referred to as z_{min}his new set of nodes is referred to as beacons ($z_{beacons}$) because it is located on line 23. All of these beacons are connected by a path that connects them all, and the costs associated with this path are referred to as $directcosts$. When this initial path is found, several samples that are spawned around z_beacons will initiate intelligent sampling. By optimizing new nodes that may be superior to the beacon position without requiring laborious sampling, this is done to lower path costs. Once the lower direct cost is identified, the path correction is finally carried out in the global looping. The pseudocode of RRT*-Smart can be seen from Algorithm 4 in Figure 4.

Algorithm 1: *RRT*-Connect*

$T_a \leftarrow \{x_{init}\}, E_a \leftarrow empty$
$T_b \leftarrow \{x_{goal}\}, E_a \leftarrow empty$
for $i = 1:n$ **do**
  $x_{rand} \leftarrow$ **sampleFree**
  if $EXTEND(G_a = (T_a, E_a), x_{rand}, x_{newA}) \neq TRAPPED$ **then**
    if $EXTEND^*(G_b = (T_b, E_b), x_{rand}, x_{newB}) \neq TRAPPED$ **then**
      $CONNECT^*(G_b = (T_b, E_b), x_{newA})$
    $SWAP(G_a = (T_a, E_a), G_b = (T_b, E_b))$;
return $G = (T_a \cup T_b, E_a \cup E_b)$;

Figure 1. Algorithm 1: RRT*-Connect

Algorithm 2: *EXTEND**
Function EXTEND*$(G = (T, E), x, x_{new})$

// Extend G toward x with local optimization, creating x_{new}
$x_{nearest} \leftarrow NEAREST(G = (T_a, E_a), x)$;
$x_{new} \leftarrow STEER(x_{nearest}, x)$
if $collisionFree(x_{nearest}, x)$ **then**
  T $\leftarrow T \cup \{x_{new}\}$
  $x_{min} \leftarrow x_{nearest}$
  Xnear$\leftarrow NEAR(G = (T, E), x_{new}, \min(\gamma_{RRT} * (\log(|T|)/(T)^{\frac{1}{d}}), \eta)$
  $C_{min} \leftarrow COST(x_{nearest}, G) + c(LINE(x_{nearest}, x_{new}))$
  // Connecting along minimum cost path among nearest neighbor
  foreach $x_{near} \in X_{near} \backslash x_{near}$ **do**
    If $collisionFree(x_{near}, x_{new}) \wedge COST(x_{near}, G) + c(LINE(x_{near}, x_{new})) < C_{min}$ **then**
      $x_{min} \leftarrow x_{near}$
      $C_{min} \leftarrow COST(x_{near}, G) + c(LINE(x_{near}, x_{new}))$;
  E=E $\cup \{(x_{min}, x_{new})\}$
  // rewire the tree
  foreach $x_{near} \in X_{near} \backslash x_{min}$ **do**
    if $collisionFree(x_{new}, x_{near}) \wedge COST(x_{new}, G) + c(LINE(x_{new}, x_{near})) < COST(x_{near}, G)$ **then**
      $x_{parent} \leftarrow PARENT(x_{near}, G)$;
      E$\leftarrow (E\{(x_{parent}, x_{near})\}) \cup \{(x_{new}, x_{near})\}$
  if $x_{new} = x$ **then**
    return *REACHED*
  else
    return *ADVANCED*
return TRAPPED

Figure 2. Algorithm 2: EXTEND

Algorithm 3: *CONNECT**

Function $CONNECT^*(G_{other} = (V_{other}, E_{other}), x)$
  //repeatedly extend G_{other} toward x
  Repeat
    S $\leftarrow$ EXTEND$^*(G_{other}, x, x_new)$
  until $S \neq ADVANCED$
  // if $S = REACHED, x$ *will be in both trees*
  return S

Figure 3. Algorithm 3: CONNECT*

Algorithm 4: *RRT*Smart Algorithm* ~ $T = (V, E) \leftarrow RRT^*Smart(z_{init})$

1 $T \leftarrow InitializeTree()$
2 $T \leftarrow InsertNode(\emptyset, z_{init}, T)$
3 **for** $i = 1$ **to** N **do**
4 **if** $i = n + b, n + 2b, n + 3b \ldots$ **then**
5 $z_{rand} \leftarrow intelligent\_sampling(i, z_{beacon})$
6 **else**
7 $z_{rand} \leftarrow sampling(i)$
8 **endif**
9 $z_{nearest} \leftarrow nearest(T, z_{rand})$
10 $(z_{new}, u_{new}, T) \leftarrow steer(z_{nearest}, z_{rand})$
11 **if** $obstaclefree(z_{new})$ **then**
12 $z_{near} \leftarrow near(T, z_{new}, |V|)$
  $z_{min} \leftarrow choosenparent(z_{near}, z_{nearest}, z_{new})$
  $T \leftarrow insertnode(z_{min}, z_{new}, T)$
13 $T \leftarrow rewire(T, Z_{near}, Z_{min}, Z_{new})$
14 **endif**
15 **if** $initialPathfound$ **then**
16 $n \leftarrow i$
17 $(T, directcost) \leftarrow pathOptimization(T, z_{init}, z_{goal})$
18 **endif**
19 **if** $(directcostnew < directcostold)$ **then**
20 $Z_{beacons} \leftarrow pathOptimization(T, z_{init}, z_{goal})$
21 **endif**
22 **return** T
23 **endfor**

Figure 4. Algorithm 4: RRT*-Smart

Proposed Method (Bidirectional-RRT*-Connect-Assisted RRT*-Smart)

Since the RRT*-connect shows better optimality and convergence speed in producing the initial path, logically it can be applied on the RRT*-Smart. It is utilized to replace the RRT* on RRT*-Smart. Initially, two trees represented the trees for expansion process begin from starting and goal point are respectively defined as T_a and

T_b. At this initialization, there is no edge belongs to the trees. Once this initialization is conducted, the search process is initiated by generating a random node, x_{rand}. This random node is then used as the base for expanding and getting the new node. It is done by applying directly EXTEND* step as presented previously. Regarding this process, there will be two new nodes, one is for new nodes used for T_a expansion and one is for new nodes used for T_b expansion. For this reason, in its application, they are named as $qnewA$ and $qnewB$. The expansion is sequentially performed. T_a expansion is conducted and if the qnewA is successfully obtained, expansion process for T_b is begun. Once the process is done, CONNECT* is performed. It is done by applying Algorithm 3 as presented before. This function aims to make the T_b expansion in the same direction with T_a. According to this process, the termination criteria is set to further used as indicator to measure whether the initial path is found or not. This process shows how the first step in this proposed method is just the same as RRT*-Connect applied.

Furthermore, n is set to be equal to number of iterations i when the initial path is found. It is then used as a factor when the intelligent sampling is performed. As can be seen in Algorithm 5 line 4, when n is add with biasing radius b return the same values as current iteration gives, the intelligent sampling is performed. The intelligent sampling will improve the path by more dominant conducting the free sampling of node around beacons $z_{beacons}$. The beacons here are the nodes consisted of the tree after its optimized. Simultaneously, the $directcost$ are also calculated by measuring the distance of nodes from the starting to the goal point. This cost is then saved and named *prevCost* immediately when the initial path is found. It is further compared with the new beacon cost obtained. if the current beacon cost is better, *prevCost* will be updated and vice versa. The use of RRT*-Connect with bidirectional search speeds up the new path to be obtained. Therefore, the convergence speed of RRT*-Smart must be enhanced. Following these ideas, the algorithm of the proposed method is defined as follows

As can be seen from Algorithm 5, the only difference between RRT*-Smart and this proposed method lies on line 9-16 in Algorithm 4 which is replaced by Algorithm 1 and mentioned as shown in line 9 in Algorithm 5 in Figure 5.

RESULTS AND DISCUSSION

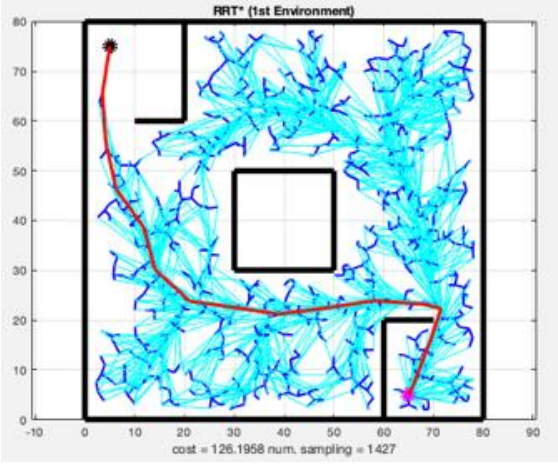
To evaluate the performance of the proposed algorithm, a series of experiments were carried out. As mentioned before, the proposed method RRT*-Smart-Bidirectional-RRT*-Connect can quickly find the initial path and is faster than RRT*-Smart. This claim is due to the presence of Bidirectional-RRT*-Connect in the proposed new method which replaces RRT* in RRT*-Smart. Therefore, RRT*-Smart-Bidirectional-RRT*-Connect and RRT* are first compared with reference to the number of samplings required to find the initial path. The fewer the number of sampling nodes required, the better the performance in this determination, and vice versa. In this comparison study, there will be four different scenarios of environment, namely simple and complex environment. The following representative result shows the performance of determining initial path in the simple environment.

Figure 6 illustrates the efficiency of Bidirectional-RRT*-Connect to determine the initial path in terms of search time. Compared to RRT* which requires 1427 sampling nodes, the Bidirectional-RRT*-Connect search time is faster by only requiring 374 sampling nodes to determine the initial path. Apart from that, the optimality provided is also better, shown by lower costs by a quite high margin. However, the validity of path search efficiency needs to be tested in a more complex environment. As shown in Figure 7, the environment is set with moderate complexity.

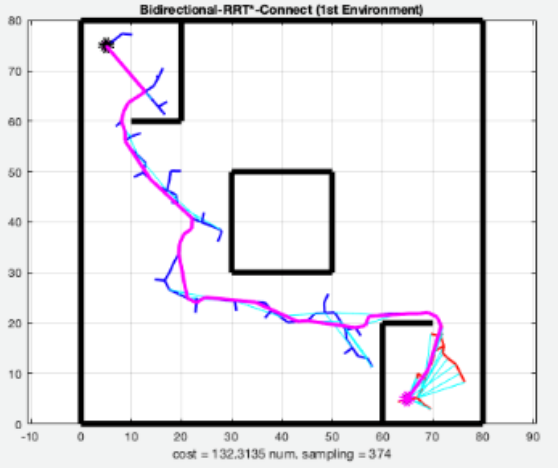
Algorithm 5. RRT*Connect – Assisted RRT*Smart

1 $T_a \leftarrow \{x_{init}\}, E_a \leftarrow empty$
2 $T_b \leftarrow \{x_{goal}\}, E_a \leftarrow empty$
3 **for** $i = 1$ **to** N **do**
4 **if** $i = n + b, n + 2b, n + 3b \ldots$ **then**
5 $x_{rand} \leftarrow intelligent\_sampling(i, z_{beacon})$
6 **else**
7 $x_{rand} \leftarrow samplingFree_i$
8 **endif**
9 $RRT^*Connect(T_a, T_b, x_{rand})$
10 **if** $initialPathfound$ **then**
11 $n \leftarrow i$
12 $(T, directcost) \leftarrow pathOptimization(T, z_{init}, z_{goal})$
13 **endif**
14 **if** $(directcostnew < directcostold)$ **then**
15 $Z_{beacons} \leftarrow pathOptimization(T, z_{init}, z_{goal})$
16 **endif**
17 **return** T
18 **endfor**
19

Figure 5. Algorithm 5: RRT*Connect-Assisted RRT*-Smart

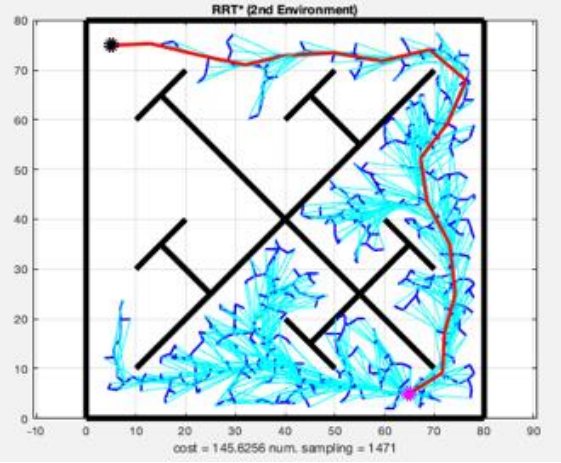


(a)

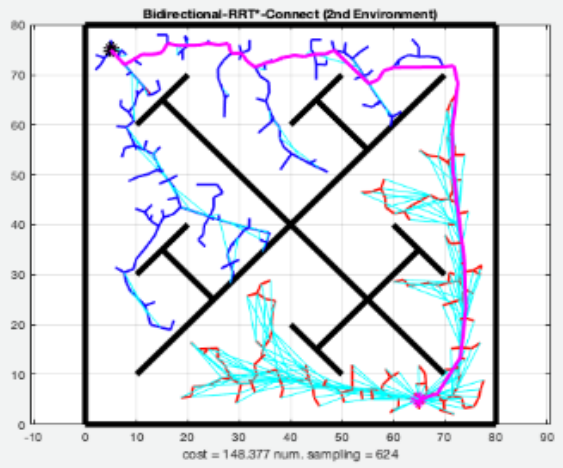


(b)

Figure 6. Performance of RRT* (a) and Bidirectional-RRT*-Connect (b) for Determining Initial Path in 1st Environment

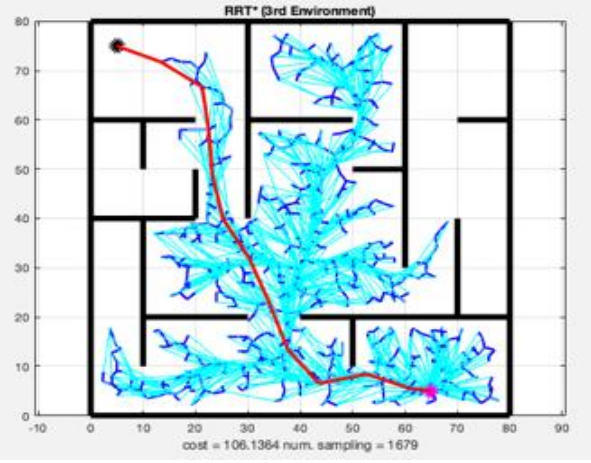


(a)

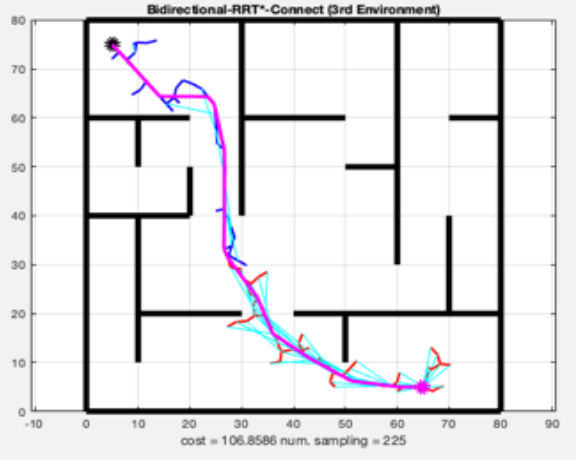


(b)

Figure 7. Performance of RRT* (a) and Bidirectional-RRT*-Connect (b) for Determining Initial Path in 2nd Environment



(a)



(b)

Figure 8. Performance of RRT* (a) and Bidirectional-RRT*-Connect (b) for Determining Initial Path in 3rd Environment

Compared with RRT*, the search time represented by the number of sampling nodes of Bidirectional-RRT*-Connect is significantly reduced as shown in Figure 7. It can be seen that there are several ways to connect the starting point, and the destination point in the case of the second experiment. This means that the performance of the initial Bidirectional-RRT*-Connect exploration method still has the potential for failure if it is limited by the number of options available. Thus, the third and fourth experiments were carried out. Referring to the results of the third test shown in Figure 8, the consistency of Bidirectional-RRT*-Connect in maintaining optimality and search time has been fulfilled. This is indicated by the low-cost value and the small number of sampling nodes, respectively.

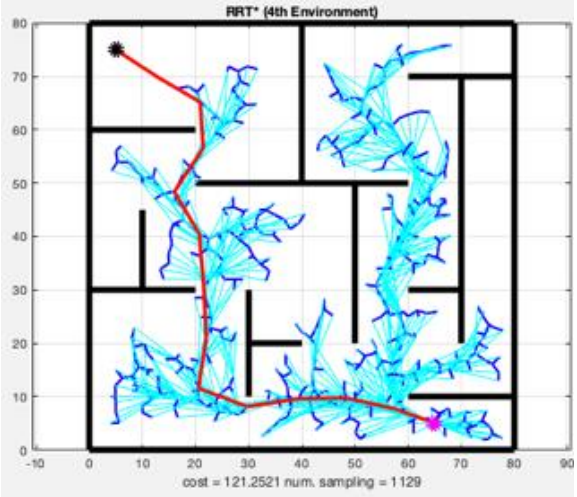
The claim above is also supported by the results of the fourth test with increased environmental complexity. Referring to Figure 9, the search time of Bidirectional-RRT*-Connect is reliable with a low number of sampling nodes compared to RRT*.

This is then used as a basis for confidently replacing the conventional method of RRT*-Smart with bidirectional-RRT*-Connect to determine the initial path. With implementation in this way, an integrated method named Bidirectional-RRT*-Connect-RRT*-Smart is proposed. Supported by the test results during research time, further testing of the proposed method was carried out. Different from previous testing and analysis, in this study the proposed method will be compared with its predecessor method, namely RRT*-Smart in terms of optimality represented by the cost path. Furthermore, to give the clearance of these performances, the ability of determining the initial path is also compared based on the time in second. This comparison is shown in Table 1.

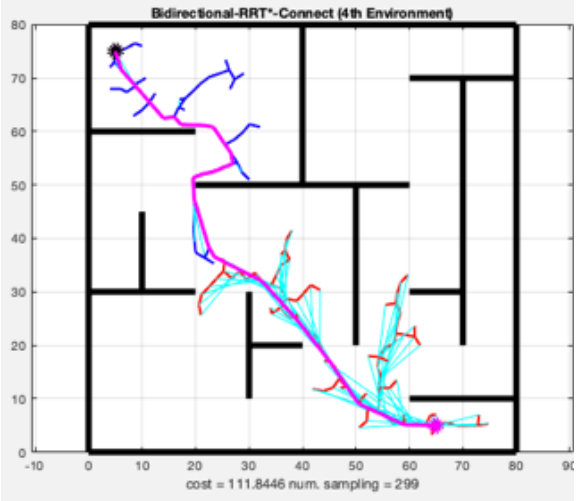
There are two environments used for this test with different complexities. To test optimality which includes the path optimization process for each method, this test will maintain the same number of samplings, namely 5000 times.

Table 1. Comparative Result in Term of Search Time.

Environment	RRT* (second)	Bidirectional-RRT*-Connect (second)
1st	0.70012	0.50272
2nd	0.79222	0.67019
3rd	0.50307	0.39783
4th	0.62637	0.47851



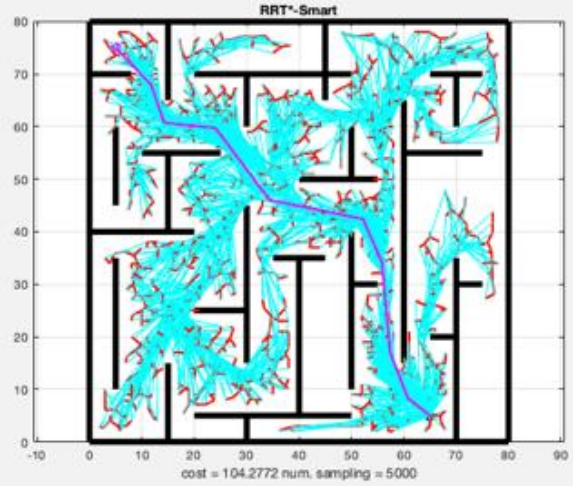
(a)



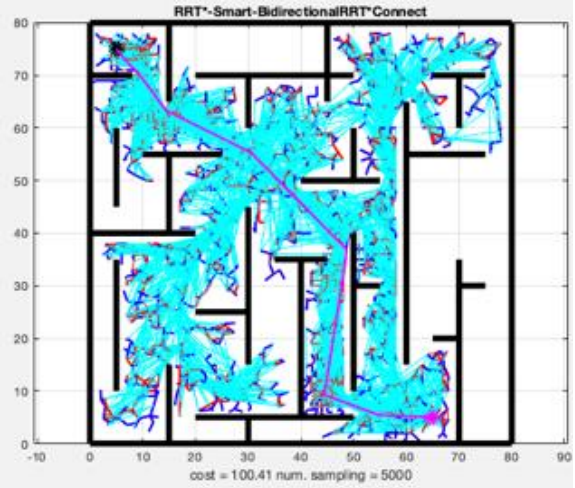
(b)

Figure 9. Performance of RRT* (a) and Bidirectional-RRT*-Connect (b) for Determining Initial Path in 4th Environment

The comparative result of RRT*-Smart and the proposed method in the first environment is shown in Figure 10. As can be seen from Figure 1, the start and goal point are assumed to be known by robot which are located on (65,5) and (5,75), respectively. Before the simulation is conducted, parameterization is conducted. it includes to defining EPS as the distance allowed to place the newly generated node to the trees. According to this information, the following results are obtained.



(a)

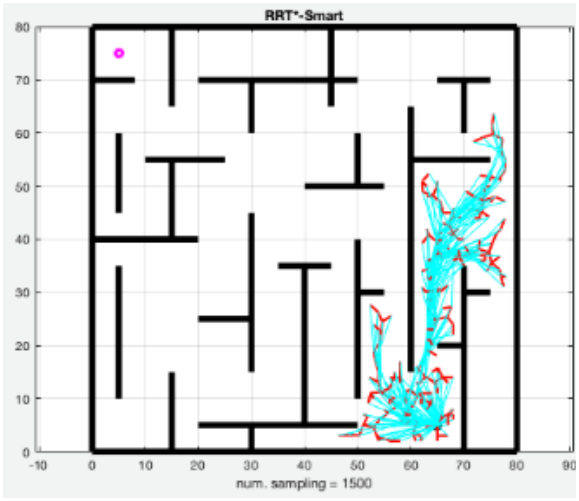


(b)

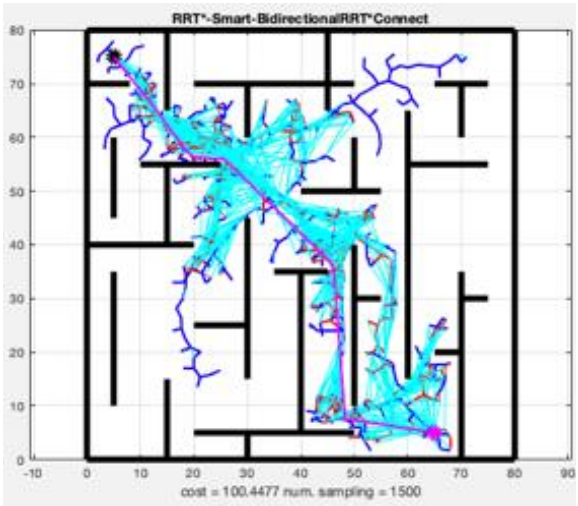
Figure 10. Optimality RRT*-Smart (a) and Bidirectional-RRT*-Connect-Assisted RRT*-Smart (b) for 5000 Number of Sampling Node

As shown in Figure 10, RRT*-Smart and Bidirectional-RRT*-Connect-Assisted RRT*-Smart successfully determined a feasible path. However, based on the path costs obtained from the limited sampling number of 5000, the optimality of RRT*-Smart is limited. In theory, this is caused by the high sampling requirements in determining the initial path. So that this point is clear, the initial hypothesis that increasing the optimality of RRT*-Smart can be done by replacing the exploration process using Bidirectional-RRT*-Connect has been accepted. This is supported by the achievements of two methods when the number of samplings is only limited to 1500 which is shown in Figure 11.

It can be seen in Figure 11 that RRT*-Smart cannot solve path planning problems caused by paths with narrow aisles and the starting point being far from the goal point. On the other hand, the proposed method can solve the problem with a good optimality of 100.4 which is similar to its achievement when the number of sampling nodes is increased up to 5000. In Figure 11, the environment is designed with a hallway that is moderately narrow but there are few alternatives for connecting the starting point with the goal point. Therefore, representative results are not enough to prove that the optimality of the proposed method is better. Based on this, the two methods were compared again by applying them to an environment consisting of narrower passages with the alternative that the start-goal point connection was increased. This performance was first observed when the number of samples allowed was 4000 times. These results are shown in Figure 12.

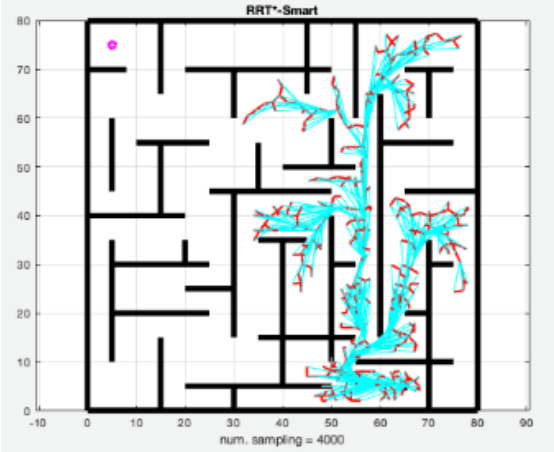


(a)

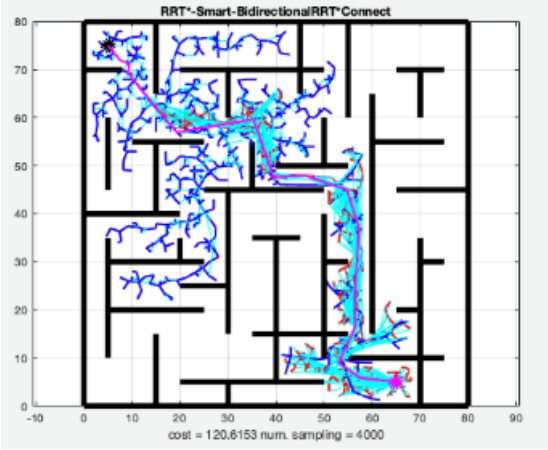


(b)

Figure 11. Optimality RRT*-Smart (a) and Bidirectional-RRT*-Connect-Assisted RRT*-Smart (b) for 1500 Number of Sampling Node (5th Environment)



(a)



(b)

Figure 12. Optimality RRT*-Smart (a) and Bidirectional-RRT*-Connect-Assisted RRT*-Smart (b) for 4000 Number of Sampling Node (6th Environment)

Figure 12 shows a comparison between RRT*-Smart (a) and the proposed method (b). It can be seen that RRT*-Smart cannot solve the problem when the number of samples allowed is only 4000. While the proposed method can optimally produce the expected path. This shows that in a complex environment with narrow channels, the role of RRT* in Smart-RRT* cannot expand properly. Instead of obtaining an optimal path, the solution cannot be provided by RRT*-Smart.

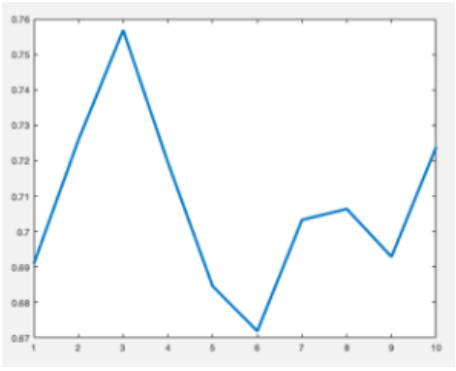
With the same environment shown in Figure 12, the next experiment was observed based on the number of nodes in trees, elapsed time, and path cost solving the problem. It was conducted in order to prove that the proposed method offers the convergence rate and enough optimality as the new alternative algorithm to solve the global path planning. This result can be seen in Table 2.

Table 2. Performance of Bidirectional-RRT*-Connect-Assisted RRT*-Smart Based on 10 Trial Runs

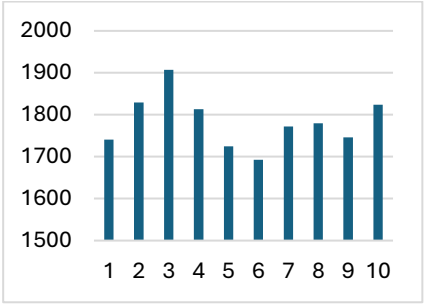
No	Elapsed Time (Seconds)	Number of Nodes in Tree	Path Cost
1	0.6909	1741	133.9696
2	0.7259	1829	129.7226
3	0.7568	1907	125.6812
4	0.7195	1813	128.5003
5	0.6846	1725	131.7774
6	0.6719	1693	139.3332
7	0.7033	1772	135.3682
8	0.7064	1780	135.9793
9	0.6929	1746	133.3820
10	0.7239	1824	129.8406

The last experiment used the number of sampling nodes as 4000, and there were 10 trial runs using MATLAB R2017b through a computer with a specification as follows: 3,6 GHz Quad-Core Intel Core i3, 8 GB 2667 MHZ DDR4. According to the results presented in Table 2, the representative graph is given in Figure 13.

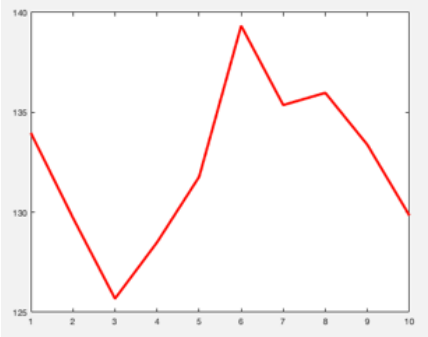
Figure 13 shows a visualization of Table 2 and represents the consistency in solving path problems in the 6th environment. This can be seen in the range (0.67, 0.75) for the search time, [1693,1907] for the number of nodes, and [125.68, 139.33] in finding an initial path that is linear with the resulting path cost.



(a)



(b)



(c)

Figure 13. Performance of Bidirectional-RRT*-Connect-Assisted RRT*-Smart After 10 Trial Runs with 4000 Sampling Nodes Allowed, in Term of (a) Elapsed Time, (b) Number of Nodes in Tree, (c) Path Cost

CONCLUSION

The RRT*-Smart consists of smart sampling and path optimization that makes it better than RRT*. However, the optimality of the resulting path does not only depend on this optimization process, but also on the length of time to carry out the optimization. Where the ideal duration will be obtained when the initial path is obtained faster. Therefore, the exploration process and determining the initial path greatly influences the optimality that can be provided by RRT*-Smart. This is different from the characteristics of RRT* exploration with uniform sampling found in the initial RRT*-Smart process. Referring to this phenomenon, a bidirectional-RRT*-connect method with a good convergence rate is proposed in determining the initial path. This method is applied to RRT*-Smart by replacing the sampling and wiring process methods at the beginning of the stage. Therefore, the duration that can be used for path optimization increases at limited sampling times. This is in line with the desire of most researchers to design an RRT* that can work effectively and optimally without relying on an infinite sampling process.

ACKNOWLEDGMENT

This research is supported by the Ministry of Education, Culture, Research, and Technology of the Republic of Indonesia through its competitive 2023 research funding, Schema of Postgraduate Research and supported by Universitas Mercu Buana.

REFERENCES

[1] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, “Path planning techniques for mobile robots: Review and prospect,” *Expert Systems with Applications*, vol. 227, pp. 120254–120254, May 2023, doi: 10.1016/j.eswa.2023.120254.

[2] 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, pp. 945–945, Jul. 2019, doi: 10.3390/sym11070945.

[3] H. Suwoyo, A. Adriansyah, J. Andika, A. Ubaidillah, and M. F. Zakaria, “An Integrated RRT*SMART-A* Algorithm for solving the Global Path Planning Problem in a Static Environment,” *IIUM Engineering Journal*, vol. 24, no. 1, pp. 269–284, Jan. 2023, doi: 10.31436/iiumej.v24i1.2529.

[4] S. Li and C. Zhu, “Improved RRT*-Smart Algorithm for UGV Path Planning in Emergency Rescue Scenarios,” *2024 7th International Conference on Robotics,*

Control and Automation Engineering (RCAE), pp. 212–218, Oct. 2024, doi: 10.1109/rcae62637.2024.10833983.

[5] Y. Huang and C. Jin, "Path Planning Based on Improved RRT Algorithm," *2023 2nd International Symposium on Control Engineering and Robotics (ISCER)*, Hangzhou, China, 2023, pp. 136-140, doi: 10.1109/ISCER58777.2023.00030.

[6] I. B. Jeong, S. J. Lee, and J. H. Kim, "Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate," *Expert Systems with Applications,* vol. 123, pp. 82–90, 2019, doi: 10.1016/j.eswa.2019.01.032.

[7] H. Suwoyo, A. Adriansyah, J. Andika, A. Ubaidillah, and M. F. Zakaria, "An Integrated RRT*SMART-A* Algorithm for solving the Global Path Planning Problem in a Static Environment," *IIUM Engineering Journal*, vol. 24, no. 1, pp. 269–284, Jan. 2023, doi: 10.31436/iiumej.v24i1.2529.

[8] S. E. Ginting and A. S. Sembiring, "Comparison of Breadth First Search (BFS) and Depth-First Search (DFS) Methods on File Search in Structure Directory Windows," *Login*, vol. 13, no. 1, pp. 26–31, 2019.

[9] R. Haq, D. Purwanto and R. Mardianto, "Microcontroller-Based Multi-Objective Genetic Algorithm for Mobile Robots Path Planning," *2023 3rd International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*, Yogyakarta, Indonesia, 2023, pp. 122-126, doi: 10.1109/ICE3IS59323.2023.10335453.

[10] D. Rachmawati and L. Gustin, "Analysis of Dijkstra's Algorithm and A∗ Algorithm in Shortest Path Problem," *Journal of Physics: Conference Series,* vol. 1566, no. 1, 2020, doi: 10.1088/1742-6596/1566/1/012061.

[11] G. Yi, C. Zhou, Y. Cao, and H. Hu, "Hybrid assembly path planning for complex products by reusing a priori data," *Mathematics*, vol. 9, no. 4, pp. 1–13, 2021, doi: 10.3390/math9040395.

[12] Z. Tang and H. Ma, "An overview of path planning algorithms," *IOP Conference Series: Earth and Environmental Science*, vol. 804, no. 2, p. 022024, Jul. 2021, doi: 10.1088/1755-1315/804/2/022024.

[13] 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*, vol. 28, no. 2, p. 265, Apr. 2024, doi: 10.22441/sinergi.2024.2.007

[14] T. Huang, K. Fan, W. Sun, W. Li, and H. Guo, "Potential-Field-RRT: A Path-Planning Algorithm for UAVs Based on Potential-Field-Oriented Greedy Strategy to Extend Random Tree," *Drones*, vol. 7, no. 5, p. 331, May 2023, doi: 10.3390/drones7050331.

[15] D. Muriyatmoko, A. Djunaidy, and A. Muklason, "Heuristics and Metaheuristics for Solving Capacitated Vehicle Routing Problem: An Algorithm Comparison," *Procedia Computer Science*, vol. 234, pp. 494–501, Jan. 2024, doi: 10.1016/j.procs.2024.03.032.

[16] B. Zhang and C. Li, "The Optimization and Application Research of the RRT-APF-Based Path Planning Algorithm," *Electronics*, vol. 13, no. 24, pp. 4963–4963, Dec. 2024, doi: 10.3390/electronics13244963.

[17] B. K. Patle, N. Pagar, D. R. K. Parhi and S. Sanap, "Hybrid FA-GA Controller for Path Planning of Mobile Robot," *2022 International Conference on Intelligent Controller and Computing for Smart Power (ICICCSP)*, Hyderabad, India, 2022, pp. 1-6, doi: 10.1109/ICICCSP53532.2022.9862422.

[18] H. Yang, J. Zhang, S. Liu, J. Liang and W. Zhang, "UAV path planning based on GA search," *2023 IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI)*, Changchun, China, 2023, pp. 784-788, doi: 10.1109/ICETCI57876.2023.10176629.

[19] W. -H. Li, T. Zhao and S. -Y. Dian, "A Multi-Robot Coverage Path Planning Method Based On Genetic Algorithm," *2021 International Conference on Security, Pattern Analysis, and Cybernatics (SPAC)*, Chengdu, China, 2021, pp. 13-18, doi: 10.1109/SPAC53836.2021.9539901.

[20] R. Nian, Z. Shen and W. Ding, "Research on Global Path Planning of Unmanned Sailboat Based on Improved Ant Colony Optimization," *2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE)*, Dalian, China, 2021, pp. 428-432, doi: 10.1109/CACRE52464.2021.9501353.

[21] J. Chen, X. Zhang and L. Xu, "Path planning algorithm based on hybrid A* and adaptive ant colony optimization," *2022 18th International Conference on Computational Intelligence and Security (CIS)*, Chengdu, China, 2022, pp. 43-48, doi: 10.1109/CIS58238.2022.00017.

[22] S. Dereli and R. Köker, "A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm," *Artificial*

Intelligence Review, vol. 53, no. 2, pp. 949–964, 2020, doi: 10.1007/s10462-019-09683-x.

[23] 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," *Journal of Applied Engineering Science*, vol. 19, no. 3, pp. 592–599, 2021, doi: 10.5937/jaes0-30557.

[24] A. Adriansyah, H. Suwoyo, and Y. Tian, "Improving the Wall-Following Robot Performance using PID-PSO Controller," *Jurnal Teknologi,* vol. 81, no. 3, pp. 119–126, 2019, doi: 10.11113/jt.v81.13098

[25] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep Learning Approach for Intelligent Intrusion Detection System," *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[26] R. Mashayekhi, M. Y. I. Idris, M. H. Anisi, I. Ahmedy, and I. Ali, "Informed RRT∗-Connect: An Asymptotically Optimal Single-Query Path Planning Method," *IEEE Access*, vol. 8, pp. 19842–19852, 2020, doi: 10.1109/ACCESS.2020.2969316.

[27] *J*. Wang, J. Li, Y. Song, Y. Tuo, and C. Liu, "FC-RRT*: A modified RRT* with rapid convergence in complex environments," *Journal of Computational Science,* vol. 77, p. 102239, Apr. 2024, doi: 10.1016/j.jocs.2024.102239.

[28] 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 Systems with Applications*, vol. 184, no. June, pp. 115457–115457, 2021, doi: 10.1016/j.eswa.2021.115457.