BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **24**, No 1
Sofia • 2024 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2024-0011

Memorized Rapidly Exploring Random Tree Optimization (MRRTO): An Enhanced Algorithm for Robot Path Planning

Dena Kadhim Muhsen[1], *Ahmed T. Sadiq*[2], *Firas Abdulrazzaq Raheem*[3]

[1,2]*Computer Science Department, University of Technology – Iraq, 10066 Baghdad, Iraq*
[3]*Control and Systems Engineering Department, University of Technology – Iraq, 10066 Baghdad, Iraq*
E-mails: dena.k.muhsen@uotechnology.edu.iq ahmed.t.sadiq@uotechnology.edu.iq
Firas.A.Raheem@uotechnology.edu.iq

Abstract: *With the advancement of the robotics world, many path-planning algorithms have been proposed. One of the important algorithms is the Rapidly Exploring Random Tree* (*RRT*) *but with the drawback of not guaranteeing the optimal path. This paper solves this problem by proposing a Memorized RRT Optimization Algorithm* (*MRRTO Algorithm*) *using memory as an optimization step. The algorithm obtains a single path from the start point, and another from the target point to store only the last visited new node. The method for computing the nearest node depends on the position, when a new node is added, the RRT function checks if there is another node closer to the new node rather than that is closer to the goal point. Simulation results with different environments show that the MRRTO outperforms the original RRT Algorithm, graph algorithms, and metaheuristic algorithms in terms of reducing time consumption, path length, and number of nodes used.*

Keywords: *Robot path planning, Rapidly exploring random tree, Memorized RRT Optimization Algorithm, Sampling-based method, Metaheuristic Algorithm.*

1. Introduction

Mobile robots are useful for many various tasks, such robots for cleaning floors [1], production factories [2], the medicine domain [3] healthcare sector [4], and security [5-6]. The basic property of an intelligent mobile robot is the ability to autonomously navigate from a starting position to a particular goal position without human interaction [7-8]. The robotic path-planning problem has received a significant amount of interest with the development of robotic fields. The problems of Path planning comprise finding a possible path from the starting point to the goal point [9]. Many algorithms adopted for path planning such as A* Algorithm, probabilistic road method, quadtrees, rapidly exploring random trees, Dijkstra Algorithm, D* and D* lite algorithms [10-12]. The environment type of robot motion is static, dynamic, or both, which determines the complexity of the problem [13]. A dynamic environment is where objects move and change their positions over time [14]. These types are categorized into completely known which comprises all information about

the environment, partially known when not all information about the obstacles exists at the path planning time and finally, there is nothing to know about the obstacles completely, named as totally unknown which requires some intelligent methods for making right decision [15].

The contribution of work in this paper is to solve the limitation of the previous version of the RRT Algorithm by enhancing it. The basic purpose is to reduce randomness, and execution time to get a solution near-optimal solution using memory as an optimization step. A single path is generated from two directions one from the start point and another from the target point to store only the last visited new node. In addition, the method for computing the nearest node depends on the position. In the improved algorithm when a new node is added RRT function checks if there is another node that is closer to the new node rather than the node that is closer to the goal point. This solution reduces path length, the number of nodes used, and time consumption.

2. Related work

There are many searches in the field on rapidly exploring random tree algorithm improvement. Some of the asymptotic studies are submitted in this section.

In 2015, an asymptotically optimal randomized motion planning algorithm is introduced [16], which merges between the RRT* and RRT-Connect to solve problems of single-query path planning through the use of a bidirectional search. It proves its efficiency by applying it to many applications in the real world such as planning car trajectories in a parking garage for the autonomous vehicle CoCar, creating cost-efficient trajectories for the multi-legged walking robot LAURON V in a planetary exploration scenario, and achieving mobile manipulation tasks for our highly actuated service robot HoLLiE. The results show the RRT∗_Connect Algorithm reduces the time for path planning and enhancing performance, which led to its being very useful for autonomous robots and vehicles.

In 2018, an Improved RRT Algorithm named RRT-Rectangular Algorithm was introduced [17] which solves the problem of 2D path planning that includes static obstacles. The basic idea is to pre-process the map by division into different sizes of rectangular elements and then the position of all points that generate path planning is in the center of safe rectangular elements; this led to robot motion only in safe environments. The results showed that the proposed RRT-Rectangular Algorithm outperforms than original RRT Algorithm in path length, which is closer to the optimal path. Especially in a complex environment with many obstacles, the shortest path is obtained.

In 2021, present a proposed improved RRT Algorithm [18] in which the basic idea depends on the strategy of selecting the priority of the parent point and the strategy of real-time optimization. The parent point was selected before creating a new point at first and combined the real-time optimization strategy secondly, to make a comparison about the distance of a new point, its parent point, and two ancestor points to the goal point. Results showed that the proposed algorithm has a high rate of success approximately near to 100% when compared to the original

RRT Algorithm in 3D path planning. The points' number, path planning time, and length of the path were reduced by about 93.25%, 91.49%, and 7.88%, respectively.

In 2021, the study [19] solves the problem of path planning for urban low-altitude UAV flights by improving on RRT-Connect Algorithm. The idea was to decrease the time planning of the algorithm through optimization for search step length and determine both the parent node and branch orientation. The search starts from the shortest parent node of the two trees, and the search is achieved in the Middle Direction for a path. The search step length is changed due to the field *f* view which is in an open field is cut down quickly. The direction of the main branch of the search tree is controlled by the angle to decrease the probability of the UAV turning sharply. The simulation results present that the improved RRT-connect produces less in both time and path length, which outperforms RTT-extend and Lazy-RTT.

In 2021, a proposed triangular inequality-based RRT-Connect Algorithm using principles of triangular inequality is presented [20], which is one of improvement on the Rapidly-Exploring Random Tree-Connect Algorithm to get closer to the optimal solution. The findings from simulations introduce more efficiency for the proposed algorithm by decreasing the number of nodes used, path length, and planning time compared to the original RRT Algorithm and RRT-Connect Algorithm.

In 2022, proposed a new sampling-based path planning method suitable for the autonomous navigation of quadruped robots named Multiple Informed RRT-Connect (MI-RRT-Connect) Algorithm [21]. It solves the limitation of the Informed RRT* Algorithm by using RRT-Connect to enhance the path by computing the cost of the parent node. The simulation results illustrate that the algorithm introduces the optimal path in less time outperforms on original Informed RRT* Algorithm and is well applied in a real quadruped robot.

In 2023, addressed the problem of the disadvantage of the Rapidly-exploring Random Tree Star Algorithm (RRT* Algorithm) which is low sampling effectiveness and slow convergence rate in the environment comprised of long corridors, according to a large iterations number are demanded in sampling critical nodes [22]. The search proposes the Expanding Path RRT* (EP-RRT*) depends on heuristic through merging the RRT* Algorithm with RRT-Connect. This combination led to generating a quick-finding path in the explored area and expanding the environment to produce the heuristic sampling area. This operation was repeated in the heuristic sampling area to obtain the solution close to the optimal path.

3. Rapidly Exploring Random Tree Algorithm (RRT Algorithm)

RRT algorithm is extensively used for robot path planning which depends on the generation of random numbers. It has a search ability in high-dimensional areas efficiently and works under many constraints to find the path for the robot. The disadvantage of the RRT Algorithm is that it has very high randomization and the resulting path is not optimal [23-25]. The RRT is one of the most widely used probabilistic planning algorithms [26]. RRT Algorithm is illustrated in Fig. 1 for generating path details, and in Fig. 2 – for algorithm steps, in which the tree expands toward a *q* random node and selects the nearest node vertex from the tree to continue

its expansion from this vertex by a small step toward q random node and define a new node which is determined radius r from q nearest point [27]. The Euclidean distance is used to compute the distance between two consecutive points, as in the next equation [28-30]:

(1) $$d=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}.$$

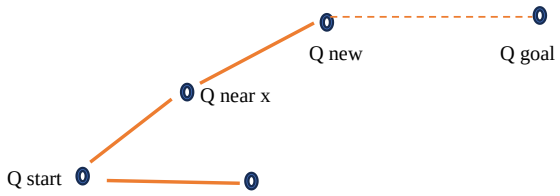


Fig. 1. Expanding of rapidly exploring random tree

Algorithm 1. Rapidly Exploring Random Tree Algorithm (RRT Algorithm)

Input parameters:
q_s starting point
q_g goal point
k length of step
H all frontier points locations in all obstacle (known)
S NO. of samples in random
Output:
p Resulted Path
Initialization:
T ⟵ **Null** tree<node, edge>
Begin
1 T ⟵ **Add** s node (root)<qs>
2 **While s** from 0 **To S Do**
3 Create **s**-th sample in random
4 q_{rand} node ⟵ position of s-th sample in random
5 q_{near} node ⟵ position of the closest node in T from q_{rand} node
6 **If not** is within (q_{near} node, q_{rand} node, K) **Then**
7 q_{new} node ⟵ juncture point between line segment linking q_{rand} node and q_{near}
n node, and circle which radius is k centered at q_{near} node
8 **Else** q_{new} node ⟵ q_{rand} node
9 **If not** isTrapped(q_{new} node, q_{near} node, H) **Then**
10 T ⟵ **Add** node<q_{new} node> & edge<q_{new} node, q_{near} node>
11 **If** is within (q_{new} node, q_g, K) **Then**
12 T ⟵ **Add** node<q_g> & edge<q_{new} node, q_g>
13 M ⟵ path from flast added node{q_g} to root node{q_s} in T
14 **If** [length of P] > [length of M] **Then** P ⟵ M
15 T ⟵ **remove** node<q_g> & edge<q_{new} node, q_g> from T End

Fig. 2. Algorithm 1. Rapidly Exploring Random Tree Algorithm

4. Proposed Memorized RRT Optimization Algorithm (MRRTO Algorithm)

The new improvement on RRT is by using memory as an optimization step to store the nodes. For every iteration, new nodes are created for two trees, there are two trees made one from the start point and one from the goal point. In the original algorithm

when a new node is added, the RRT function checks if the node is closer to the goal point. In the improved algorithm when a new node is added, the RRT function checks if there is another node that is closer to the new node. If there is a node in 0.1×0.1 diameter where a new node is added then there are two states, one is the node is in the same tree, and the other is a different tree. If same tree then we do not need to add a new node. If a different tree then we found the path(result) from the start point to the endpoint. For this, introduced the idea of visited memory values. Fig. 3 shows the work of MRRTO Algorithm.

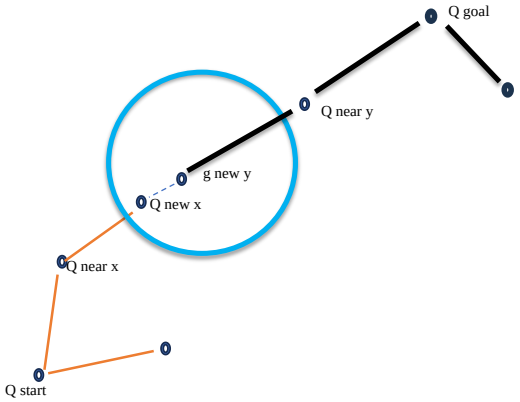


Fig. 3. MRRTO Algorithm

The nodes are saved as visited, if two points have the same value depending on point position in memory (computed as in the next equation) then the two points are very close:

(2) $$v = \text{int}(\text{newnode}(x) / 0.1) \times 100 + \text{int}(\text{newnode}(y) / 0.1),$$

where v is the get index of a node position with coordinates (x, y) in memory and then changes a position value (x, y) pair to a number. For example:

$$v\,(0, 0) = 0,$$
$$v\,(0.23, 0.45) = 2 \times 100 + 4 = 204,$$
$$v\,(0.55, 5.75) = 5 \times 100 + 57 = 557,$$
$$v\,(8.89, 9.55) = 88 \times 100 + 95 = 8895.$$

If two points have the same value with the v function then the two points are very closer:

$$v_1\,(3.03, 5.99) = 30 \times 100 + 59 = 3059,$$
$$v_2\,(3.01, 5.92) = 30 \times 100 + 59 = 3059,$$

(3.03, 5.99) and (3.01, 5.92) have the same index and are very close.

After a new node is found, get the position index and check if it is registered in the visited memory. If registered that means there was a node before that has been added to that position. If not registered then check collision-free and add a node to the tree (see (3)), where m is defined as a memory, v_1 index of the point in memory, and v_2 index of the new point. Then register the index of nodes_start or nodes_goal array to the visited memory with tree_id. Finally, when the node reaches the nearest node from the other tree returns the final path, which merges the resulting path from the start point with the resulting path from the target point. The dramatic

improvement comes from the meeting nodes. From the original RRT Algorithm, the tree from the start point is seeking a goal point by adding new nodes. However, in this improved RRT Algorithm, the new nodes from the start point and target point do not seek the start/target point. They are seeking a normal node from another tree.

Algorithm 2. Proposed Memorized Rapidly Exploring Random Optimization Algorithm (MRRTO Algorithm)

Input:
start ← start ^{point}
goal ← goal ^{point}
obstacles ← position set of all (measured) boundary points in all (known) obstacles
max_iters ← random samples number
step_size ← step length
Output:
path ← result of path from start point
nodes_start ← result node tree from start point
goal_path ← result of path from goal point
nodes_goal ← result node tree from goal point
Initialize:
Nodes_start ← [start]
Nodes_goal ← [goal]
visited ←{} Empty Memory
Begin
1 Visited ←**Register** start
2 Visited ←**Register** goal
3 **While** n ← 0 **To** max_iters **Do**
4 **Generate** n-th random sample
5 q_{rand} ← position of n-th random sample
6 q_{near} ← position of the nearest node in nodes_start from q_{rand}
7 q_{new} ← **Create** a new node with step_size away from q_{near} direction to q_{rand}
8 q_{new} ← **Set** q_{near} as parent
9 **If not** visited **and** is_collision_free(q_{new}, obstacles) **Then**
10 nodes_start ← **Insert** q_{new}
11 Visited ← **Register** q_{new}
12 **Elif** visited by a node from goal **Then**
13 path ← get path from q_{new} parent recursive
14 goal_path ← get path from registered node parent recursive
15 **Return** path, nodes_start, goal_path, nodes_goal
16 **Generate** n-th new random sample
17 q_{rand} ← position of n-th new random sample
18 q_{near} r ← position of the nearest node in nodes_goal from q_{rand}
19 q_{new} ← **Create** a new node with step_size away from q_{near} direction to q_{rand}
20 q_{new} ← **Set** q_{near} as parent
21 **If not** visited **and** is_collision_free(q_{new} , obstacles) **Then**
22 nodes_goal ← **Insert** q_{new}
23 Visited ← **Register** q_{new}
24 **Elif** visited by a node from start **Then**
25 path ← get path from registered node parent recursive
26 goal_path ← get path from q_{new} parent recursive
27 **Return** path, nodes_start, goal_path, nodes_goal

Fig. 4. Pseudocode of the proposed MRRTO Algorithm

That makes the finding/meeting possibility higher. Using memory takes $O(1)$ to find a near node. However, the original RRT Algorithm takes $O(n)$ for speed complexity [31]. Memory saves nodes not in add order; it saves in position order. Therefore, we can check easily if there is a node near a new node by using this memory. The essential work of the MRRTO Algorithm is illustrated in Algorithm 2, Fig. 4. The proposed algorithm assists in finding an efficient path with less consuming time,

$$m(v) = \begin{cases} v_2 & \text{if } v_2 = v_1 \,\&\, v_1 \in m, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

5. Simulation results

To verify the efficiency and performance of the proposed Memorized Rapidly Exploring Random Tree Optimization Algorithm (MRRTO Algorithm) in this paper, firstly, the proposed algorithm is compared with the original RRT Algorithm [32]. Secondly, RRT- Connect Algorithm [33], RRT* Algorithm [34], Informed RRT* Algorithm [35], Smart RRT*Algorithm [36], Extended RRT [37], Dijkstra Algorithm [38],
A* Algorithm [39], Particle Swarm Optimization (PSO) Algorithm [40], Ant Colony Optimization Algorithm (ACO Algorithm) [41], and the proposed algorithm, are compared with each other in a two-dimensional environment with many static obstacles shown in the simulation environment. The comparison criteria depend on the average path length, time-consuming average in seconds, and average of nodes used. Five runs for the original RRT Algorithm and the proposed algorithm were implemented at each specific start and end point; the e results are shown in Table 2. In addition, the proposed algorithm and all other algorithms used at each specific start and endpoint have been implemented ten times, and the average of these criteria is shown in Table 3. The results prove that the proposed algorithm reduces the average of these measurement criteria.

5.1. Simulation environment

This section submits the two-dimensional environment that is used in the simulation containing many static obstacles, Fig. 5. The black polygon refers to the obstacles. The border is 10 (horizontal) × 10 (vertical) pixels. The title of the implemented algorithm and the criteria of path length, time consumption, and number of nodes used appear in the screen below. The computer performance specifications used in the simulation environment are illustrated in Table 1. Python programming language was used to develop the simulation.

Table 1. Specifications of hardware

Hardware	Descriptions
CPU	Intel(R) Core (TM) i5-10210U CPU @ 1.60 GHz 2.11 GHz
RAM	8.00 GB (7.84 GB usable)
VGA	Intel ® UHD Graphics (4140 MB)

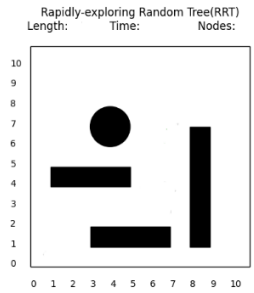
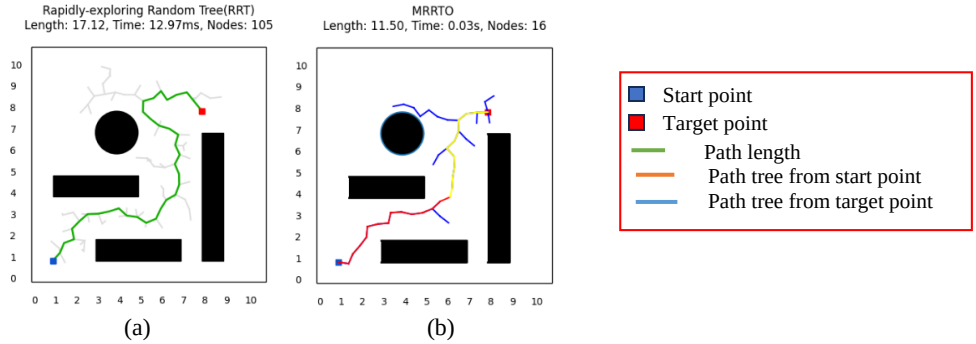


Fig. 5. Simulation environment

5.2. Analysis of simulation results

Many tries are done by the proposed MRRTO Algorithm and RRT Algorithms to check the performance of the robot path planning in many criteria to find the best paths from the start point to the target point. Table 2 illustrates the results of path planning by the proposed algorithm using many different points for both start and end for five tries for each point and how its efficiency is better than the original RRT Algorithm in measurements of path length, time consumption, and number of nodes used. Fig. 6 shows the first implementation of the first pair of points in Table 2, the RRT algorithm, and the proposed MRRTO Algorithm at the start point (1, 1) and a target point (8, 8) in an environment containing several obstacles. The path length average by using the MRRTO Algorithm for these five implementations of the first point is 12, the time-consuming is 0.04s and the number of nodes is 23 compared to the original RRT Algorithm where the average path length is 16, the time-consuming is 12.37 and the average number of nodes used is 73. All criteria are less by the proposed algorithm, which leads to better path quality and more path smoother. Other points were also tested and all got better results in reducing path length, time-consuming, and node numbers used.

In each result of the path planning in Fig. 6, the blue square refers to the start point, the red square refers to the target point, and the green line segment refers to the best path between the start point and the target point in the RRT Algorithm. In MRRTO Algorithm the red line is path generated from the start point and yellow line path generated from the target point. Gray and blue segments are paths (trees) that are excluded during the process of path planning.



(a) (b)

Fig. 6. First implementation of path planning with starting point (1, 1) and a target point (8, 8): RRT (a), MRRTO (b)

Table 2. RRT and proposed Mmemorized RRT Optimization algorithm

Start point; Goal point	RRT			MRRTO		
	Path length, m	Number of nodes	Time-consuming, s	Path length, m	Number of nodes	Time-consuming, s
(1, 1); (8, 8)	17	105	12.97	12	16	0.03
	15	60	18.99	12	33	0.04
	15	77	9.97	12	25	0.07
	15	65	13.95	12	16	0.04
	16	57	6.01	13	26	0.05
(2, 2); (4, 9)	15	77	10.96	11	18	0.09
	15	63	7.98	10	24	0.08
	14	66	6.98	10	19	0.09
	13	75	8.98	12	16	0.07
	14	70	5.96	10	15	0.07
(5, 3); (2, 8)	14	107	16.8	10	24	0.04
	13	80	14.69	8	31	0.06
	15	97	11.96	9	15	0.04
	13	95	9.97	11	26	0.05
	13	76	9.96	10	18	0.04
(2, 3); (9.5, 9)	14	118	11.66	12	22	0,04
	13	62	4.99	12	15	0.03
	16.	78	9.01	11	18	0.04
	16	101	11.97	13	18	0.03
	13	98	11.95	13	19	0.04
(1, 1); (4, 9)	14	92	15.78	10	24	0.02
	16	63	7.98	13	32	0.04
	15	57	8.96	13	18	0.06
	17	64	6.98	13	23	0.05
	16	79	17.17	14	38	0.06

5.3. Analysis of simulation results of the proposed MRRTO Algorithm with other path-planning algorithms

To evaluate the MRRTO Algorithm, it was compared with multiple benchmark algorithm scenarios for path planning including sampling-based methods, heuristic methods, and metaheuristic methods. The comparison was done in the same environment and same start point and target point for each algorithm and for 10 tries times for each point and taking the average of important criteria that path planning depends on such as Path Length average, Nodes Number average, and Time-consuming average as shown in Table 3. MRRTO Algorithm achieves a path optimization process when an initial path is found and eliminates redundant nodes from this path by generating two direction trees one from the start point and one from the target point. When a new node is added to the initial tree, the RRT function checks if there is another node that is closer to the new node. If there is a node in 0.1×0.1 diameter where a new node is added then there are two states, one is the node is in the same tree, and the other is a different tree. If same tree then we do not need to add a new node. If a different tree then the path (result) from the start point to the target point is found. The results of comparisons with other sample algorithms show that the proposed MRRTO Algorithm when compared with RRT* Algorithm and RRT-Connect Algorithm reduce in criteria of path length, less time-consuming significantly, and the number of nodes used and sometimes is equal in path length but in less than in nodes number and time-consuming, and the resulted path near the optimal path. In the case of informed RRT*, Smart RRT*, and Extended RRT

generate the same or shortest path than the MRRTO Algorithm but the latter gives superior results in reducing the time and nodes used in generating the path because RRT* and RRT*-Smart offer near neighbor search and rewiring operations, that cause to increase execution time and number of nodes used.

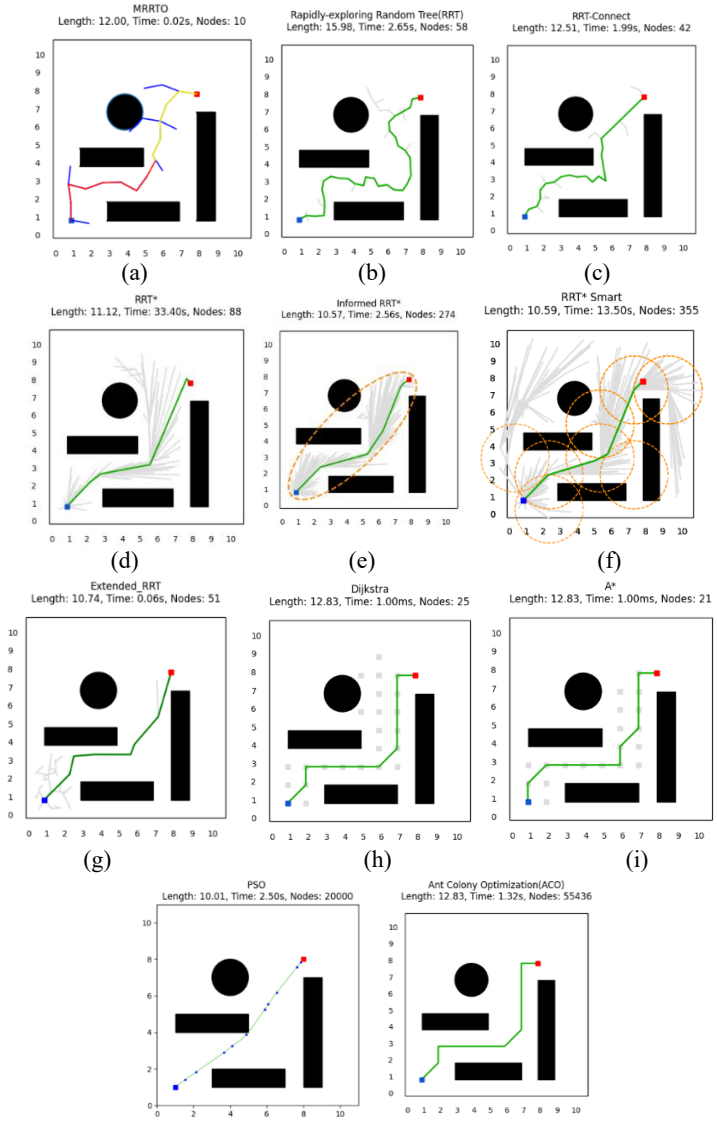
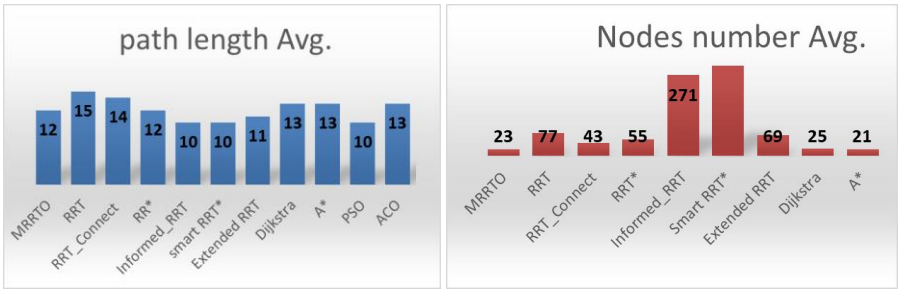


Fig. 7. Comparison of the performance of the proposed MRRTO Algorithm with other algorithms in many criteria (Path Length Avg., Nodes Number Avg., and Time-consuming Avg.) with starting point (1, 1) and a target point (8, 8): MRRTO Algorithm (a), RRT Algorithm (b), RRT_Connect (c), RRT* (d), Informed RRT (e), Smart RRT* (f), Extended RRT (g), Dijkstra Algorithm (h), A* Algorithm (i), PSO Algorithm (j), ACO Algorithm (k)

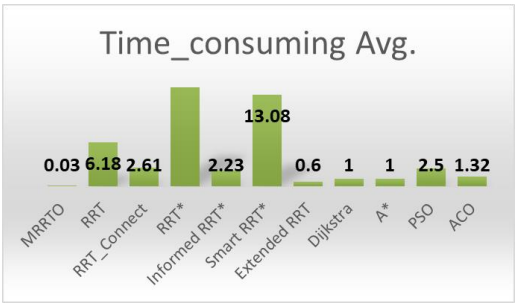
As for algorithms-based graphs such as Dijkstra and A*, the MRRTO Algorithm outperforms these algorithms in all criteria and gives the near shortest path for the

robot. Metaheuristic algorithms were also compared with the proposed algorithm and the results illustrate that the Particle Swarm Optimization Algorithm (PSO Algorithm) is a little less in path length but requires more time and a very large number of nodes used. These factors for time and a huge number of nodes used are critical and affect algorithm performance because when the PSO Algorithm uses a large number of nodes in its operations, this leads to consuming more memory. This easily falls in the local optimum in high-dimension space. In contrast, the proposed MRRTO Algorithm where it is very well in this situation. In addition, more time consumption is an important factor due to the consumption of energy by mobile robots which are powered limited operated by batteries, and the high cost as a result. This feature is of paramount importance for uses such as driverless cars, drone surveillance, and robotic exploration.

The MRRTO Algorithm is superior to the ACO Algorithm in all criteria and gets path planning near to an optimal solution. Fig. 7 shows a comparison of the performance of the proposed MRRTO Algorithm with other algorithms in many criteria (Path Length average, Nodes Number average, and Time-consuming average) with starting point (1, 1) and a target point (8, 8) for 10 implementations of each algorithm and show how MRRTO outperforms efficiently to give near-optimal solutions and smoother path.



(a) (b)



(c)

Fig. 8. Average comparisons result of MRRTO with other path planning algorithms for start point (1, 1) and target point (8, 8): path length average (a), nodes number average (b), time-consuming average (c)

In Fig. 8, the plots demonstrate all measurements of performance for comparisons of the proposed algorithm with other current algorithms for path planning and show how the MRRTO Algorithm outperforms efficiently in run time,

path length, number of nodes used, and low randomness. In (b) does not mention the number of nodes used in PSO Algorithm and ACO Algorithm because it is too large and other algorithms does not appear clearly. The proposed algorithm reduces time in a high ratio not accessed 1 s and for a number of nodes is about 50% less than other algorithms. This led to producing a path with high quality and more smother for mobile robots. All results are highly encouraging to utilize this proposed algorithm in many real-world scenarios for mobile robot path planning due to the advantages achieved.

Table 3. Average of Measurements comparison for MRRTO Algorithm performance with other algorithms

Start point; Goal point	Criteria	Sample-based methods							Graph methods		Metahurstic methods	
		Memorized RRT Optimization (proposed MRRTO)	RRT [32]	RRT-connect [33]	RRT* [34]	Informed RRT* [35]	Smart RRT* [36]	Extended RRT [37]	Dijkstra [38]	A* [39]	PSO [40]	ACO [41]
(1, 1); (8, 8)	Path Length average, m	**12**	15	14	12	10	10	11	13	13	10	13
	Nodes Number avg.	**23**	77	43	55	271	334	69	25	21	20,000	55,465
	Time-consuming average, s	**0.03**	6.18	2.61	18.8	2.23	13.08	0.6	1	1	2.5	1.32
(2, 3); (4, 9)	Path Length average, m	**10**	15	12	11	10	9	10	11.41	11.41	7.78	11.41
	Nodes Number avg.	**17**	72	37	56	294	293	57	29	19	20,000	49,719
	Time-consuming average, s	**0.02**	8.69	2.98	17.82	2.79	10.9	0.1	1.00	1.00	2.4	1.12
(6, 3); (2, 8)	Path Length average, m	**9**	13	11	10	9	9	10	10.41	10.41	6.56	10.41
	Nodes Number avg.	**16**	88	47	69	280	414	72	35	24	20,000	48,560
	Time-consuming average, s	**0.03**	10.30	4.99	31.46	3.70	19.854	0.02	0.99	0.96	2.48	1.12
(2, 1); (6, 9)	Path Length average, m	**12**	14	12	11	10	10	11	12	12	9.41	12
	Nodes Number avg.	**22**	59	38	53	265	232	59	24	16	20,000	55,537
	Time-consuming average, s	**0.04**	8.62	1.99	16.03	2.46	13.99	0.04	1.02	1	2.44	1.29
(2, 2); (7, 8)	Path Length average, m	**10**	12	10	10	9	9	9	10.41	10.41	8	11
	Nodes Number avg.	**19**	46	35	35	259	320	37	22	14	20,000	55,317
	Time-consuming average, s	**0.04**	6.92	2.01	13.38	2.34	11.03	0.03	1	1	2.45	0.99

6. Conclusion

The new MRRTO Algorithm is proposed for solving the limitation of old versions of the RRT Algorithm. It successfully demonstrates in various benchmarking simulations that it finds solutions earlier than the other current path-planning algorithms and that convergence towards the near-optimal solution is submitted. General simulation results show that the MRRTO outperforms the original RRT Algorithm, and other sampling-based algorithms, graph algorithms, and metaheuristic algorithms in terms of reducing time consumption, path length, and number of nodes used. This makes the MRRTO Algorithm by using memory as an optimization step to store the node produce a more smoothing path and it is a valuable instrument for a large diversity of planning scenarios for the robot world. Overall, the

results are encouraging and the advancements made possible by the MRRTO Algorithm have a major effect on the actual implementation of robots. The algorithm's versatility, efficiency, resilience, optimization of resources, and capacity to handle complicated real-world scenarios make it a great tool for improving intelligent robotic systems and autonomous navigation.

References

1. Yakoubi, M. A., M. T. Laskri. The Path Planning of Cleaner Robot for Coverage Region Using Genetic Algorithms. – Journal of Innovation in Digital Ecosystems, Vol. **3**, 2016, No 1, pp. 37-43. DOI: 10.1016/j.jides.2016.05.004.
2. Stączek, P., J. Pizoń, W. Danilczuk, A. Gola. A Digital Twin Approach for the Improvement of an Autonomous Mobile Robots (AMR's) Operating Environment – A Case Study. – Sensors, Vol. **21**, 2021, No 23, 7830. DOI: 10.3390/s21237830.
3. Baek, D., M. Hwang, H. Kim, D. Kwon. Path Planning for Automation of Surgery Robot Based on Probabilistic Roadmap and Reinforcement Learning. – In: Proc. of 2018 15th International Conference on Ubiquitous Robots (UR), Honolulu, USA, 2018, pp. 342-347. DOI: 10.1109/urai.2018.8441801.
4. Ortiz, E., B. Andres, F. J. L. Fraile, R. Poler, A. Ortiz. Fleet Management System for Mobile Robots in Healthcare Environments. – Journal of Industrial Engineering and Management, Vol. **14**, 2021, No 1, 55. DOI: 10.3926/jiem.3284.
5. Du, J., P. Zheng, Z. Xie, Y. Yang, H. Chu, G. Yu. Research on Path Planning Algorithm Based on Security Patrol Robot. – In: Proc. of 2016 IEEE International Conference on Mechatronics and Automation, Harbin, China, 2016, pp. 1030-1035. DOI: 10.1109/ICMA.2016.7558704.
6. Denk, M., S. Bickel, P. Steck, S. Goetz, H. Völkl, S. Wartzack. Generating Digital Twins for Path-Planning of Autonomous Robots and Drones Using Constrained Homotopic Shrinking for 2D and 3D Environment Modeling. – Applied Sciences, Vol. **13**, 2022, No 1. DOI: 10.3390/app13010105.
7. Muhammad, A., M. K. Ali, S. Turaev, I. H. Shanono, F. Hujainah, M. N. M. Zubir, M. A. Faiz, E. R. M. Faizal, R. Abdulghafor. Novel Algorithm for Mobile Robot Path Planning in Constrained Environment. – Computers, Materials & Continua, Vol. **71**, 2022, No 2, pp. 2697-2719. DOI: 10.32604/cmc.2022.020873.
8. Patle, B. K., B. L. Ganesh, A. Pandey, D. R. Parhi, A. Jagadeesh. A Review: On Path Planning Strategies for Navigation of Mobile Robot. – Defence Technology, Vol. **15**, 2019, No 4, pp. 582-606. DOI: 10.1016/j.dt.2019.04.011.
9. Raafat, S. M., F. A. Raheem. Intelligent and Robust Path Planning and Control of Robotic Systems. – In: Springer eBooks, Springer Nature, 2017, pp. 291-317. DOI: 10.1007/978-3-319-43901-3_13.
10. Xue, Y., J. Q. Sun. Solving the Path Planning Problem in Mobile Robotics with the Multi-Objective Evolutionary Algorithm. – Applied Sciences, Vol. **8**, 2018, No 9, p. 1425, DOI: 10.3390/app8091425.
11. Sadiq, A. T., A. N. Hasan. Robot Path Planning Based on PSO and D Algorithms in a Dynamic Environment. – In: Proc. of International Conference on Current Research in Computer Science and Information Technology (ICCIT'17), 2017. DOI: 10.1109/crcsit.2017.7965550.
12. Ahmed, T. S., F. A. Raheem, N. Abbas. Ant Colony Algorithm Improvement for Robot Arm Path Planning Optimization Based on D* Strategy. – International Journal of Mechanical &Mechatronics Engineering, Vol. **21**, 2017, No 1, pp. 96-111, 2021
13. Raheem, F. A., S. M. Raafat, S. M. Mahdi. Robot Path-Planning Research Applications in Static and Dynamic Environments. – In: J. N. Furze, S. Eslamian, S. M. Raafat, K. Swing, Eds. Earth Systems Protection and Sustainability. Cham, Springer, 2022. DOI: 10.1007/978-3-030-85829-2_12.
14. Wang, D., S. Chen, Y. Zhang, L. Liu. Path Planning of Mobile Robot in Dynamic Environment: Fuzzy Artificial Potential Field and Extensible Neural Network. – Artificial Life and Robotics, Vol. **26**, 2021, No 1, pp. 129-139. DOI: 10.1007/s10015-020-00630-6.

15. Raheem, F. A., U. I. Hameed. Interactive Heuristic D* Path Planning Solution Based on PSO for Two-Link Robotic Arm in Dynamic Environment. – World Journal of Engineering and Technology, Vol. **7**, 2019, No 1, pp. 80-99. DOI: 10.4236/wjet.2019.71005.
16. Klemm, S. O., J. Oberlander, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, R. Dillmann. – RRT-Connect: Faster, Asymptotically Optimal Motion Planning, 2015. DOI: 10.1109/robio.2015.7419012.
17. He, D., H. Wang, P. Li. Robot Path Planning Using Improved Rapidly-Exploring Random Tree Algorithm. – In: Proc. of 2018 IEEE Industrial Cyber-Physical Systems (ICPS), St. Petersburg, Russia, 2018, pp. 181-186. DOI: 10.1109/icphys.2018.8387656.
18. Tian, L., Z. Zhang, C. Zheng, Y. Tian, Y. Zhao, Z. Wang, Z, Y. Qin. An Improved Rapidly-Exploring Random Trees Algorithm Combining Parent Point Priority Determination Strategy and Real-Time Optimization Strategy for Path Planning. – Sensors, Vol. **21**, 2021, No 20. DOI: 10.3390/s21206907.
19. Jin, H., W. Cui, H. Fu. Improved RRT-Connect Algorithm for Urban Low-Altitude UAV Route Planning. – Journal of Physics, Vol. **1948**, 2021, No 1. DOI: 10.1088/1742-6596/1948/1/012048.
20. Kang, J. U., D. W. Lim, Y. S. Choi, W. D. Jang, J. W. Jung. Improved RRT-Connect Algorithm Based on Triangular Inequality for Robot Path Planning. – Sensors, Vol. **21**, 2021, No 2. DOI: 10.3390/s21020333.
21. Zhang, Y., H. Jiang, X. Zhong, X. Zhong, J. Zhao. MI-RRT-Connect Algorithm for Quadruped Robotics Navigation with Efficiently Path Planning. – Journal of Physics, Vol. **2402**, 2022, No 1. DOI: 10.1088/1742-6596/2402/1/012014.
22. Ding, J., Y. Zhou, X. Huang, K. Song, S. Lu, L. Wang. An Improved RRT* Algorithm for Robot Path Planning Based on Path Expansion Heuristic Sampling. – Journal of Computational Science, Vol. **67**, 2023. DOI: 10.1016/j.jocs.2022.101937.
23. Yamashita, T., T. Nishida. Path Planning Using Multilayer Neural Network and Rapidly-Exploring Random Tree. – In: Proc. of 18th International Conference on Control, Automation and Systems, Korea, 2018.
https://api.semanticscholar.org/CorpusID:210705126.
24. Kang, J. U., Y. Choi, J. Jung. A Method of Enhancing Rapidly-Exploring Random Tree Robot Path Planning Using Midpoint Interpolation. – Applied Sciences, Vol. **11**, 2021, No 18. DOI: 10.3390/app11188483.
25. Lonklang, A., J. Botzheim. Improved Rapidly Exploring Random Trees with Bacterial Mutation and Node Deletion for Offline Path Planning of Mobile Robots. – Electronics, Vol. **11**, 2022, No 9. DOI: 10.3390/electronics11091459.
26. Pohan, M. A. R., J. Utama. Efficient Sampling-Based for Mobile Robot Path Planning in a Dynamic Environment Based on the Rapidly-Exploring Random Tree and a Rule-Template Sets. – International Journal of Engineering. Transactions A: Basics, Vol. **36**, 2023, No 4, pp. 797-806. DOI: 10.5829/ije.2023.36.04a.16.
27. Kang, J. U., Y. Choi, J. Jung. A Method of Enhancing Rapidly-Exploring Random Tree Robot Path Planning Using Midpoint Interpolation. – Applied Sciences, Vol. **11**, 2021, No 18. DOI: 10.3390/app11188483.
28. Muhammad, S., Y. Zhou. Path Planning for EVs Based on RA-RRT* Model. – Frontiers in Energy Research, Vol. **10**, 2023. DOI: 10.3389/fenrg.2022.996726.
29. Seif, R. Mobile Robot Path Planning by RRT* in Dynamic Environments. – I. J. Intelligent Systems and Applications, Vol. **5**, 2015, pp. 24-30. DOI: 10.5815/ijisa.2015.05.04.
30. Rasheed, A. A., A. S. Al-Araji, M. N. Abdullah. Static and Dynamic Path Planning Algorithms Design for a Wheeled Mobile Robot Based on a Hybrid Technique. – International Journal of Intelligent Engineering and Systems, Vol. **15**, 2022, No 4, pp. 167-181.
http://dx.doi.org/10.22266/ijies2022.0831.16.
31. Zhang, Z., D. Wu, J. Gu, F. Li. A Path-Planning Strategy for Unmanned Surface Vehicles Based on an Adaptive Hybrid Dynamic Stepsize and Target Attractive Force-RRT Algorithm. – Journal of Marine Science and Engineering, Vol. **7**, 2019, No 5. DOI: 10.3390/jmse7050132.
32. LaValle, S. M. Rapidly-Exploring Random Trees: A New Tool for Path Planning. – The Annual Research Report, Computer Science Dept., Iowa State University, October 1998.

33. Kuffner, J. J., S. M. LaValle. RRT-Connect: An Efficient Approach to Single-Query Path Planning. – In: Proc. of 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, Vol. **2**, 2000, pp. 995-1001. DOI: 10.1109/ROBOT.2000.844730.
34. Karaman, S., M. R. Walter, A. Perez, E. Frazzoli, S. Teller. Anytime Motion Planning Using the RRT*. – In: Proc. of IEEE International Conference on Robotics and Automation, Shanghai, China, 2011, pp. 1478-1483. DOI: 10.1109/ICRA.2011.5980479.
35. Gammell, J. D., S. S. Srinivasa, T. D. Barfoot. Informed RRT*: Optimal Sampling-Based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic. – In: Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 2014, pp. 2997-3004. DOI: 10.1109/IROS.2014.6942976.
36. Nasir, J., F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, M. S. Muhammad. RRT*-SMART: A Rapid Convergence Implementation of RRT*. – International Journal of Advanced Robotic Systems, Vol. **10**, 2013, No 7, 299. DOI: 10.5772/56718.
37. Fragkopoulos, C., A. Graeser. Extended RRT Algorithm with Dynamic N-Dimensional Cuboid Domains. – In: Proc. of 12th International Conference on Optimization of Electrical and Electronic Equipment, Brasov, Romania, 2010, pp. 851-857. DOI: 10.1109/OPTIM.2010.5510401.
38. Zhou, M., N. Gao. Research on Optimal Path Based on Dijkstra Algorithms. – In: Proc. of 3rd International Conference on Mechatronics Engineering and Information Technology (ICMEIT'19), Advances in Computer Science Research, 2019. DOI: 10.2991/icmeit-19.2019.141.
39. Suwoyo, H., A. Adriansyah, J. Andika, A. Ubaidillah, 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**, 2023, No 1, pp. 269-284. DOI: 10.31436/iiumej.v24i1.2529.
40. Poli, R., J. Kennedy, T. Blackwell. Particle Swarm Optimization. – Swarm Intelligence, Vol. **1**, 2007, No 1, pp. 33-57. DOI: 10.1007/s11721-007-0002-0.
41. Dorigo, M., M. Birattari, T. Stützle. Ant Colony Optimization. Chapman and Hall/CRC, 2007, pp. 417-430. DOI: 10.1201/9781420010749-33.

Received: 22.01.2024; Second Version: 31.01.2024; Accepted: 07.02.2024 (fast track)