ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20221015080721
Original scientific paper

Simplified and Smoothed Rapidly-Exploring Random Tree Algorithm for Robot Path Planning

Ayhan GÜLTEKİN*, Samet DİRİ, Yaşar BECERİKLİ

Abstract: Rapidly-exploring Random Tree (RRT) is a prominent algorithm with quite successful results in achieving the optimal solution used to solve robot path planning problems. The RRT algorithm works by creating iteratively progressing random waypoints from the initial waypoint to the goal waypoint. The critical problem in the robot movement is the movement and time costs caused by the excessive number of waypoints required to be able to reach the goal, which is why reducing the number of waypoints created after path planning is an important process in solving the robot path problem. Ramer-Douglas-Peucker (RDP) is an effective algorithm to reduce waypoints. In this study, the Waypoint Simplified and Smoothed RRT Method (WSS-RRT) is proposed which reduces the distance costs between 8.13% and 13.36% by using the RDP algorithm to reduce the path into the same path with fewer waypoints, which is an array of waypoints created by the RRT algorithm.

Keywords: distance minimization; optimization approach; path planning; reduction algorithm; simplification

1 INTRODUCTION

In recent years, studies on robot path planning have increased massively with the number of fields with industrial fields taking the lead [1-3]. The advantages such as the ability to complete the assigned task flexibly and quickly, discarding the risks that could appear with the human factor, and being able to complete tasks like search-and-rescue, recon missions, discovering unknown environments, and health, logistics, etc. increase the demand for robots. Path planning is a vital component of autonomous systems [4]. Autonomous robots have the ability to locate themselves in the environment, the ability to map their surroundings, and being able to make use of their sensors to detect nearby objects. Autonomous robots sustain their movements from their initial waypoint to their goal waypoint by finding the best path, avoiding any obstacles. The efforts to find the best path to their goal during the movement cause path planning problems trying to be solved with lots of algorithms, to appear in the literature.

Robot path planning algorithms are divided into two basic classes according to the problem: Classical algorithms that try to provide the best solution possible in an acceptable finite time frame, and heuristic algorithms that try to find the best solution in a group of solutions [5]. The heuristic methods in this classification require excessive computing power for the path planning solutions, especially in complex environments [6].

A path planning algorithm can be local or global depending on the information available regarding the operating environment. Global algorithms, as there is information available about the environment, work with current obstacles while local algorithms continuously update the path planning strategy according to the detected obstacles and objects during the robot's movement [7].

Path planning algorithms can be classified differently in many ways. The path planning problem is an optimization problem at its roots. Classical approaches have excessive computing costs, especially in multi-dimensional environments. Because of that, heuristic algorithms are used extensively in path planning problems' solutions. Heuristic algorithms try to find the best solution in all possible solutions. Sampling-based algorithms from heuristic algorithms have quite successful results as they are quick and have low costs in complex environments [8].

Rapidly-exploring Random Trees (RRT) is an important sampling-based heuristic algorithm used intensively in robot path planning, developed by LaValle [9]. In their studies, LaValle and Kuffner present that the RRT algorithm allows a tree to be generated from the initial waypoint to the goal waypoint through randomly chosen waypoints in a multi-dimensional space and that the random waypoints generate a new waypoint to the tree in every iteration according to the waypoint that is closest to the tree and has a constant epsilon distance [10]. Ioannis et al. used B-Spline curves to increase the performance of the convolutional off-line path planning algorithm in a three-dimensional space they presented and to make the plan smoother. The problem of not being fully used has been prevented [11]. Ngoc et al. used B-spline to parameterize the path created using RRT or A* algorithms in a non-convex environment, and a Bézier curve equivalent to the B-spline curve was used to increase the power of the presented method [12]. Jian et al., in the study they presented, defined the path formed as a curve for the optimization of the path formed as a result of path planning, so that the curve parameters created for the defined curve were optimized and made smoother and the high dimension problem that would occur as a result of the optimization was reduced [13]. Saini et al. used RDP to increase the power of the methods used for trajectory classification in their study, so that the classification curve was created with fewer points [14]. In their study, Liangbin and Guoyou used RDP to reduce the rate by compressing very large data generated by the automatic identification system. As a result of the compression, the most suitable compression process of the original trajectory has been achieved at low cost in terms of time [15]. Marino and Manic have provided the FastTray algorithm they have presented by using RDP to reduce the data used to create the trajectory and to make the trajectory smoother with splines. Thus, they ensured that the GPS data necessary for a healthier traffic analysis was obtained from the original trajectory with a very small margin of error [16]. Saux and Daniel performed the data reduction of the polygonal curves with the fitting method applied over the B-spline with the method they presented [17].

This study differs from the publications in the literature especially with the use of RRT and RDP smoothing methods together. In addition, the evaluation of different smoothing methods within the scope of the study provided successful results.

In this study, the Waypoint Simplified and Smoothed RRT Method (WSS-RRT) which uses the RDP algorithm to reduce the path which is created as an array of waypoints as an artifact of the RRT algorithm, is presented. This study consists of the following sections: after the introduction, detailed information about RRT and RDP algorithms in the 2nd and 3rd sections, respectively, the method and its application in the 4th section, and the results and future studies are given in the 5th section.

2 BASIC THEORY

2.1 Rapidly-Exploring Random Trees

One of the algorithms used to solve the path planning problem is the Rapidly-exploring Random Trees (RRT) approach. The RRT algorithm provides a sampling-based solution for path planning [18]. The RRT algorithms work by creating a random tree from the initial waypoint to the goal waypoint and searching for the best path on this tree. After the initial waypoint is decided, the closest waypoint on the tree to the randomly decided waypoint is determined, taking collisions into account. Then a new waypoint between the randomly decided waypoint and the closest waypoint is added to the tree taking the predefined distance into account and this is repeated until an epsilon distance [19].

The RRT algorithm operates in an environment sterilized from obstacles, called the workspace.

C = Workspace (Configuration Space)
C_{free} = Free Space
C_{o} = Obstacle Space
$C_{\text{free}} = C/C_{\text{o}}$

The obstacles inside the workspace are expressed with

$$O = \{O_1, O_2, O_3, ..., O_n\} \subset D_2 \tag{1}$$

The new waypoint to be decided after the robot's movement should be inside the C_{free} space. While $q_0 \in C_{\text{free}}$ and $q_{\text{goal}} \in C_{\text{free}}$, T represents the tree generated after the algorithm has been run, V represents the waypoints inside the tree and E represents the inter-point connections, $T_x = (V_x, E_x)$ is the obtained expression. The tree's expansion continues until a possible path between q_0 and q goal has been found. The RRT algorithm is a quick and simple-to-apply algorithm. Although it is not always possible to find the optimal path due to its nature

RRT is an iterative algorithm. Euler integration method or similar methods can be used for phase-shift equality, to be used to decide the new waypoint to be added to the tree according to the waypoint that will be randomly generated in the workspace.

$$x_i = x_{i-1} + f(x_{i-1}, \varepsilon) \tag{2}$$

In this equation, x_{i-1} being the waypoint inside the tree that is closest to the random waypoint, the new waypoint is decided by adding a fixed step size of epsilon (ε) to the closest waypoint.

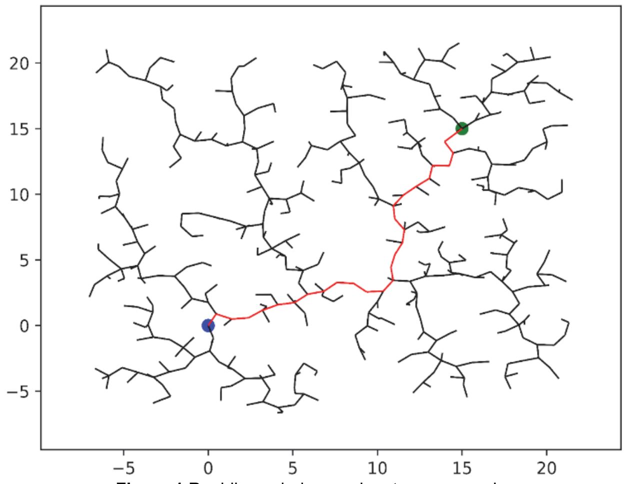


Figure 1 Rapidly-exploring random trees expansion

During the movement to the goal waypoint in the workspace, in each iteration the designated waypoints are designated statistically randomly which can be seen in Fig. 1. The RRT algorithm, due to its structure, is at the level of asymptotic complexity. The random waypoint count to be designated to approach the optimal solution will approach infinity [20] in most cases, sharp turns are required to avoid objects in the environment to reach the goal waypoint. The required number of iterations to reach the goal waypoint is usually decided according to the time limit or the previously determined random waypoint count [21].

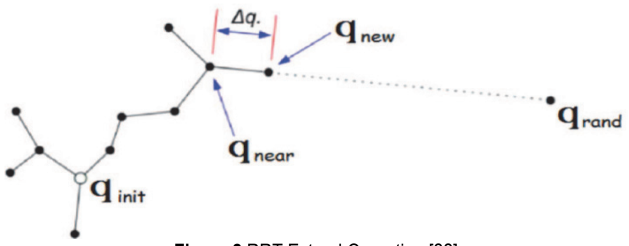


Figure 2 RRT Extend Operation [22]

We can graphically express the change in the tree structure that is generated during the runtime of the RRT algorithm like shown in Fig. 2. The initial waypoint is determined first, then a new waypoint is randomized and the closest waypoint on the tree is determined relative to the new random waypoint, then between those two waypoints a new waypoint, avoiding collisions, will be determined according to its distance of epsilon to the closest waypoint on the tree and added to the tree and this will keep on iterating until the goal waypoint is reached. The RRT algorithm is a quick algorithm due to its structure. Especially remarkable improvements in runtime can be made by running the RRT algorithm in architectures where parallel processing techniques can be applied. Also, the RRT algorithm can be run in parallel with the MapReduce programming principle. With this, by using this structure the closest waypoint on the tree to the random waypoint can be determined quickly [22].

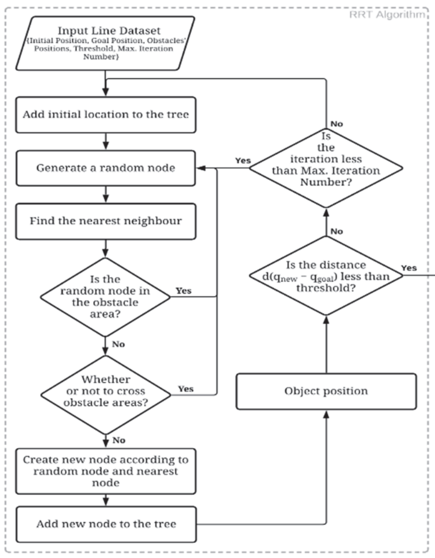

RRT Algorithm
Input Line Dataset (Initial Position, Goal Position, Obstacles' Positions, Threshold, Max. Iteration Number)
Add initial location to the tree
Generate a random node
Find the nearest neighbour
Is the random node in the obstacle area?
Yes
No
Whether or not to cross obstacle areas?
Yes
No
Create new node according to random node and nearest node
Add new node to the tree
Object position
Is the distance d(q_new − q_goal) less than threshold?
Yes
No
Is the iteration less than Max. Iteration Number?
Yes
No
RRT Generated Path

Figure 3 The flowchart of the RRT algorithm [23]

RRT works according to the flow in Fig. 3. Initially, the initial waypoint, the goal waypoint, and the obstacles are defined in the configuration space and the number of waypoints to be generated randomly is determined. The random waypoint is added to the tree if there is no obstacle collision between itself and the closest waypoint on the tree.

2.2 Ramer-Douglas-Peucker Algorithm

It is a line simplifying algorithm developed by Ramer-Douglas-Peucker (RDP) [24]. The RDP algorithm simplifies the lines made of straight-line segments by the number of waypoints. A recursive divide and combine strategy which can provide fast, compact and precise compression for time-critical systems, is used [25]. This algorithm is used to reduce the number of waypoints in a curve made of an array of waypoints. It simplifies by detecting if the first and last elements of the array of waypoints forming the curve are a straight line and the waypoint farthest from the virtual line. If the waypoints between the first and last waypoints are closer than a distance of epsilon, all those waypoints are removed. On the other hand, if this "deviant waypoint" is farther than the epsilon distance from our imaginary curve, the curve is split into two, the first one being from the first waypoint to the extreme waypoint, and the second one being from the extreme waypoint to the last waypoint. So, the formed curve's degree of coarsening is controlled by the ε parameter which defines the maximum distance between the original waypoints and the simplified curve.

The process is recursively called in both resulting curves and both of the reduced forms of the curve are combined. The simplified curve preserves the coarse shape of the original curve but must be composed of a subset of the set of waypoints defining the original curve [26].

How the RDP algorithm works is shown in Fig. 4.

The RDP algorithm is widely used in computer graphics, reducing trajectory data and especially cartography. As the method detects the deviant waypoints in the path or smoothing in the given line, the subject line mentioned in RDP can be a line in computer graphics, a waypoint in the coordinate plane, or a path in a robot's movement.

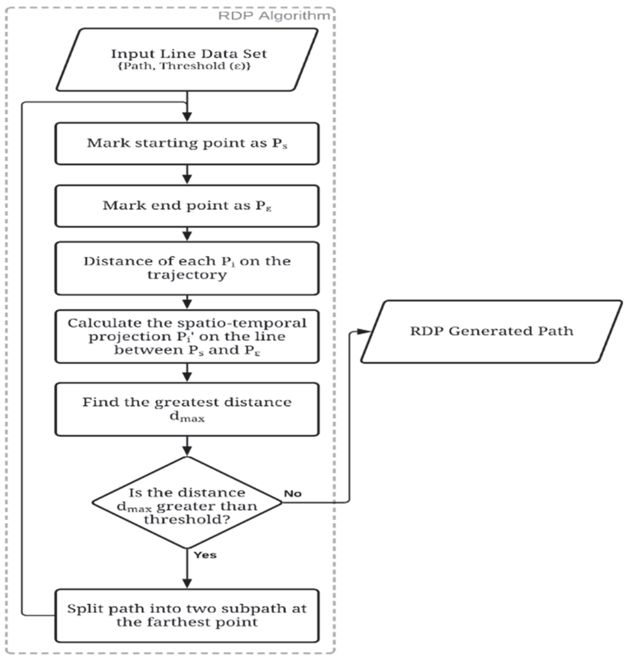

RDP Algorithm
Input Line Data Set (Path, Threshold (ε))
Mark starting point as P_s
Mark end point as P_E
Distance of each P_i on the trajectory
Calculate the spatio-temporal projection P_i' on the line between P_s and P_E
Find the greatest distance d_max
Is the distance d_max greater than threshold?
No
Yes
Split path into two subpath at the farthest point
RDP Generated Path

Figure 4 Ramer-Douglas-Peucker flowchart of the method [15]

A curve is as the following:

$$C = (P_1, P_2, P_3, \ldots, P_n) \tag{3}$$

where: C - curve; P - set of waypoints.

In each iteration, the farthest waypoint to the line is determined;

$$d_{\max} = \max d(P_i, P_n),\ i = 2, \ldots, n-1 \tag{4}$$

where: d - distance between two waypoints; P - coordinates of waypoints.

Euclidean distance is used to find the maximum distance with the equation below;

$$distance = \sqrt{\left(p_x - q_x\right)^2 + \left(p_y - q_y\right)^2} \tag{5}$$

where: (p_x, p_y) - coordinates of P point; (q_x, q_y) - coordinates of Q point.

In each iteration the new waypoint that has the maximum distance to the line is determined, then a new line segment is created for waypoints after that waypoint and for each line segment the waypoints with the euclidean distance to the line segment $< \varepsilon$ are ignored, and waypoint reduction in the curve is done. The epsilon parameter, which controls the threshold distance to the virtual line between the first and last waypoints being $\varepsilon > 0$.

3. PROPOSED METHOD

3.1 Waypoint Simplified and Smoothed RRT (WSS-RRT) Method

The RRT algorithm, due to its working principles, tries to find the shortest path on a tree made of randomly created waypoints in the search space that are hierarchically connected. It is used widely as it is able to find good solutions quickly during the path planning phase [27].

Although it is quick to produce solutions, there is the problem of not being able to guarantee the optimal solution because of the randomness used during the creation of waypoints. Though this is a problem that exists in most heuristic methods [28], there are multiple methods to improve the RRT algorithm. But these solutions usually either do not provide the desired improvements or drastically increase the runtime of the algorithm [29] in real environments. The runtime of the algorithm for path planning during the robot's movement has vital importance. The computing taking too long can cause the robot to crash or stop until the path planning is complete. Likewise, the planned path being too long can cause the energy required for the movement of the robot to be wasted, decrease its lifetime or increase the maintenance costs. If this vehicle is a combatant in a tactical field, too much cost in time or movement can cause the vehicle to be detected by the hostile elements or even to be destroyed by them. Reducing the time and the movement path will reduce the possibility of being detected by the hostile elements.

Algorithm 1. The algorithm of proposed method

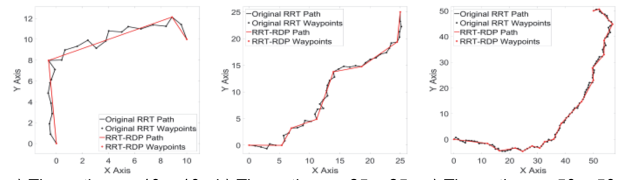
```
function RRT($x_{init}$, $x_{goal}$)
    V ← {$x_{init}$};
    E ←∅;
    G = (V, E);
    for i = 1, . . . , n do
        $x_{rand}$← SampleFree$_i$;
        $x_{nearest}$← Nearest(G ← (V, E), $x_{rand}$);
        $x_{new}$← Steer($x_{nearest}$, $x_{rand}$);
        if ObstacleFree($x_{nearest}$, $x_{new}$) then
            V ← V ∪ { $x_{new}$ };
            E ← E ∪ {($x_{nearest}$, $x_{new}$)};
        end if
    end for
    G = (V, E);
    return G;
end function

function RDP(G, ε)
    end = G.length;
    i, max$_d$ = Farthest(G, Line(G[0], G[end]));
    if max$_d$>ε then
        line$_{left}$← RDP(G[0, . . . i], ε);
        line$_{right}$ ← RDP (G[i, . . . end], ε);
        line ← line$_{left}$∪ line$_{right}$;
    else
        line ← {G[0], G[end]};
    end if
end function

function Main()
    $G_{RRT}$ = RRT($x_{init}$, $x_{goal}$);
    $G_{RDP}$ = RDP($G_{RRT}$ , ε);
    $G_{smooth}$ = SmoothPath($G_{RDP}$ ,ρ);  //Desired method
    return $G_{smooth}$;
end function
```

In this study, the RRT algorithm and the RDP algorithm are run hybrid, attempting to reduce the planned path to the shortest path possible without ramping up the time costs too much. The flow of the method is shown in Algorithm 1.

The search space, the obstacles in the space, and the outputs regarding the waypoints produced as a result of running the algorithm are shown in Fig. 5. The method working with randomly generated obstacles has been tested on a computer with an Intel Core i7-10750H CPU @ 2.60 GHz processor, 16 GB of RAM, and Windows 11 Pro Operating System.



a) The path on a 10 × 10 plane b) The path on a 25 × 25 plane c) The path on a 50 × 50 plane

Figure 5 The comparison of the resulting waypoints of the RRT algorithm and the RRT-RDP algorithm without smoothing

As a result of the waypoints of the path generated by the **WSS-RRT** algorithm being added randomly to the workspace, sawteeth can appear between waypoints.

Because of restrictions in robot kinematics, robots might have difficulties trying to accomplish their goal with paths with sawteeth, which brings the need for the path to be smoothed. In this study, 4 different smoothing methods were used to smooth the path appropriately to the 1-D interpolation structure. These methods are pchip, makima, cubic, and spline methods.

3.2 Smoothing Methods

3.2.1 Pchip Method

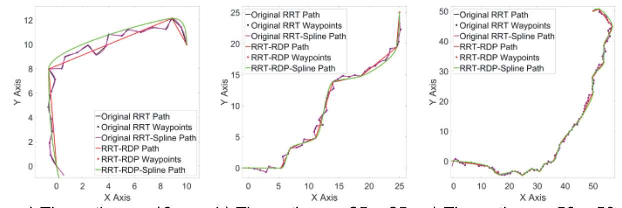
At least 4 waypoints are required, the monotony in the data is preserved, and the smoothing process will not generate extremes even if the data is not distributed uniformly. The derivatives are calculated on waypoints X_k and f'_k with Pchip [30].

$$\frac{w_1 + w_2}{f'_k} = \frac{w_1}{d_{k-1}} + \frac{w_2}{d_k} \tag{6}$$

$$w_1 = 2h_k + h_{k-1},\ w_2 = h_k + 2h_{k-1} \tag{7}$$

where: w - weighted harmonic mean; h - slope of x - coordinate at waypoint; d - slope of y - coordinate at waypoint; f - derivation on waypoint.

The end slopes are set by using a one-sided scheme [31] when Pchip is used as the smoothing method at WSS-RRT, the outputs are shown in Fig. 6.



a) The path on a 10 × 10 plane b) The path on a 25 × 25 plane c) The path on a 50 × 50 plane

Figure 6 The comparison of the resulting waypoints of the RRT and WSS-RRT algorithm using pchip method for smoothing

3.2.2 Makima Method

At least 4 waypoints are required, it is developed by Akima to prevent extreme local fluctuations in interpolation [32].

$$d_i = \frac{|\delta_{i+1} - \delta_i|\,\delta_{i-1} + |\delta_{i-1} - \delta_{i-2}|\,\delta_i}{|\delta_{i+1} - \delta_i| + |\delta_{i-1} - \delta_{i-2}|} \tag{8}$$

$$d_i = \frac{w_1}{w_1 + w_2}\delta_{i-1} + \frac{w_2}{w_1 + w_2}\delta_i \tag{9}$$

where: w - weighted harmonic mean; d - derivation of waypoint; δ - slope of waypoint.

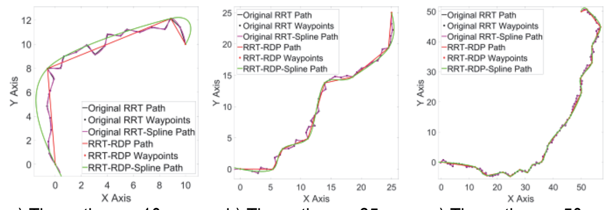
Though Akima's derivative formula has to be modified to prevent extreme cases where both the numerator and the denominator equal 0.

$$d_i = \frac{w_1}{w_1 + w_2}\delta_{i-1} + \frac{w_2}{w_1 + w_2}\delta_i \tag{10}$$

To prevent that the Makima method was developed by taking the average of lower and upper slopes [31].

$$d_i = \left(\delta_{i-1} + \delta_i \,|\, / 2\right) \tag{11}$$

When the Makima method is used as the smoothing method at WSS-RRT, the outputs are shown in Fig. 7.



a) The path on a 10 × 10 plane b) The path on a 25 × 25 plane c) The path on a 50 × 50 plane

Figure 7 The comparison of the resulting waypoints of the RRT and WSS-RRT algorithm using makima method for smoothing

3.2.3 Cubic Interpolation Approach

At least 3 waypoints are required for the method. A uniform distribution of waypoints is required for interpolation to be applied.

$$x_i < x < x_{i+1}\ ,\ \ x = x_i - \alpha t \tag{12}$$

$$\alpha = x_{i+1} - x_i \ \ \forall i \tag{13}$$

If the intervals are of equal length the continuity conditions of the derivative are same for x and t.

$$\begin{gathered} Y_{i-1}(1) = y_i,\ Y_i(0) = y_i,\ Y'_{i-1}(1) = Y'_i(0), \\ Y''_{i-1}(1) = Y''(0) \end{gathered} \tag{14}$$

For each interval the function is defined as:

$$Y_i(t) = a_i + b_i t + c_i t^2 + d_i t^3 \tag{15}$$

unknown coefficients in the function is solved with y_i and through $D_i = Y'_i(0)$.

$$Y_i(0) = y_i = a_i \tag{16}$$

$$Y_i(1) = y_{i+1} = a_i + b_i + c_i + d_i \tag{17}$$

$$Y'_i(0) = D_i = b_i \tag{18}$$

$$Y'_i(1) = D_{i+1} = b_i + 2c_i + 3d_i \tag{19}$$

a_i, b_i, c_i, d_i are found using the following equations [33].

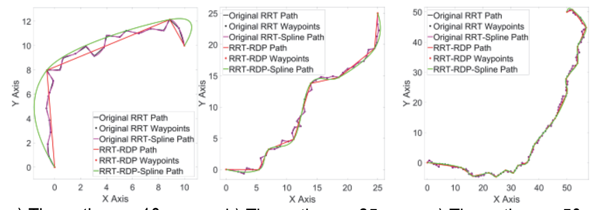
$$a_i = y_i \tag{20}$$

$$b_i = D_i \tag{21}$$

$$c_i = 3(y_{i+1} - y_i) - 2D_i - D_{i+1} \tag{22}$$

$$d_i = 2(y_i - y_{i+1}) + D_i + D_{i+1} \tag{23}$$

When Cubic method is used as the smoothing method at WSS-RRT, the outputs are shown in Fig. 8.



a) The path on a 10 × 10 plane b) The path on a 25 × 25 plane c) The path on a 50 × 50 plane

Figure 8 The comparison of the resulting waypoints of the RRT and WSS-RRT algorithm using cubic interpolation approach for smoothing

3.2.4 Spline Method

At least 4 waypoints are required for the spline method. It is used in cases where the waypoints are not distributed uniformly [34].

$$Y_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{24}$$

$$Y_i(x_i) = y_i = a_i \tag{25}$$

$$Y_i(x_i + 1) = y_{i+1} = a_i + b_i\alpha_i + c_i\alpha_i^2 + d_i\alpha_i^3 \tag{26}$$

$$Y'_i(x_i) = D_i = b_i \tag{27}$$

$$Y'_i(x_i + 1) = D_{i+1} = b_i + 2c_i\alpha_i + 3d_i\alpha_i^2 \tag{28}$$

For each interval the function is defined as:

$$Y_i\left(t\right) = a_i + b_i t + c_i t^2 + d_i t^3 \tag{29}$$

a_i, b_i, c_i, d_i is determined according to these equations:

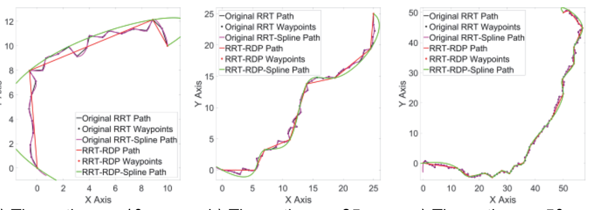
$$a_i = y_i \tag{30}$$

$$b_i = D_i \tag{31}$$

$$c_i = 3\frac{y_{i+1} - y_i}{\alpha_i^2} - 2\frac{D_i}{\alpha_i} - \frac{D_{i+1}}{\alpha_i} \tag{32}$$

$$d_i = 2\frac{y_i - y_{i+1}}{\alpha_i^3} + \frac{D_i}{\alpha_i^2} + \frac{D_{i+1}}{\alpha_i^2} \tag{33}$$

When Spline is used as the smoothing method at WSS-RRT, the outputs are shown in Fig. 9.



a) The path on a 10 × 10 plane b) The path on a 25 × 25 plane c) The path on a 50 × 50 plane

Figure 9 The comparison of the resulting waypoints of the RRT and WSS-RRT algorithm using spline method for smoothing

4 RESULTS AND DISCUSSION

The test results of the proposed WSS-RRT method are shown in Tab. 1. The method was applied on 3 different landscape maps which are respectively 10 × 10 plane with 7 obstacles, 25 × 25 plane with 19 obstacles, and 50 × 50 plane with 44 obstacles.

Table 1 The average data of the WSS-RRT algorithm for different landscapes

	Map 1	Map 2	Map 3
Map Area	10 × 10	25 × 25	50 × 50
Number of Iterations	250	500	1000
Obstacle Count	7	19	44
RRT Runtime / s	2.84	23.22	182.37
RDP Runtime on RRT Waypoints / s	0.007	0.009	0.016
Smoothing Runtime / s	0.001	0.002	0.007
RRT Path Length	23.52	44.76	108.17
RRT-RDP Path Length	20.75	40.71	100.04
RRT Waypoint Count	25	45	109
RRT-RDP Waypoint Count	4	8	18
Distance Difference	2.77	4.05	8.13
Distance difference / %	13.36	9.96	8.13

In Map 1, which contains 7 obstacles on a 10 × 10 plane, the average path length obtained as a result of running only the RRT algorithm after a maximum of 250 iterations is 23.52 units and the running time is 2.84 seconds. On the other hand, when the WSS-RRT algorithm is run on the same path, the average path length obtained is 20.75 units and the running time is 2.848 seconds. It has been observed that WSS-RRT provides a gain of approximately 2.77 units, which is about 13.36% in terms of path length, and this is achieved with a loss of only 0.008 second.

In Map 2, which contains 19 obstacles on a 25 × 25 plane, the average path length obtained as a result of running only the RRT algorithm after a maximum of 500 iterations is 44.76 units and the running time is 23.22 seconds. On the other hand, when the WSS-RRT algorithm is run on the same path, the average path length obtained is 40.71 units and the running time is 23.231 seconds. It has been observed that WSS-RRT provides a gain of approximately 4.05 units, which is about 9.96% in terms of path length, and can achieve this with a loss of only 0.011 second.

In Map 3, which contains 4 obstacles on a 50 × 50 plane, the average path length obtained as a result of running only the RRT algorithm after a maximum of 1000 iterations is 108.17 units and the running time is 182.37 seconds. On the other hand, when the WSS-RRT algorithm is run on the same path, the average path length obtained is 100.04 units and the running time is 182.393 seconds. It has been observed that WSS-RRT provides a gain of approximately 8.13 units, which is about 8.13% in terms of path length, and can achieve this with a loss of only 0.023 second.

The most important reason for the gain is the smoothing of the waypoints in the working space due to the randomness due to the working principle of the RRT, by simplification with the RDP algorithm and smoothing of the waypoints in the sharp corners. This is the novelty of the proposed method which differs from studies in literature.

5 CONCLUSION

In this study the Waypoint Simplified and Smoothed RRT Method (WSS-RRT) which allows a path to be represented with fewer waypoints by reducing a path generated by the RRT algorithm using the RDP algorithm is proposed. As a result of the method being applied on maps with different sizes, it is observed that there are improvements compared to the current RRT algorithm in terms of path length, runtime, and waypoint count. It will be experimented on with other reduction methods which are quicker or with a higher rate of reduction, besides the Ramer-Douglas-Peucker method. The RRT algorithm due to its nature progresses with creating a tree structure with a probabilistic approach. This structure of the RRT algorithm causes the waypoints from the initial waypoint to the goal waypoint to be numerous. With the proposed method there are also improvements in probability-caused non-optimal parameters.

The WSS-RRT method presented within the scope of the study was applied in a 2-dimensional workspace. The test results data obtained in the study were analyzed in accordance with the limitations of the proposed method. The operation of the RRT applied in the proposed method in 3-dimensional workspace will increase the computation time and algorithm complexity. It is thought that the computation time and algorithm complexity will increase during the operation of RDP in 3-dimensional workspace, but it will not be costly since it will still work at relatively few points.

In future work, the method's results will be evaluated on three-dimensional maps. Also, different obstacle formations will be placed on maps to evaluate how the method works for different obstacle types. In future, work will be done on reducing path waypoints according to the robot's kinematic structure by taking its movement dynamics into account. The presented study is about reducing the path determined with the RRT algorithm and avoiding static obstacles in a previously defined movement space. In future work, the current method will be improved, and waypoint reduction will be done in real-time during the RRT algorithm's movement in the workspace.

6 REFERENCES

[1] Karaman, S. & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, *30*(7), 846-894. https://doi.org/10.1177/0278364911406761

[2] Banjanovic-Mehmedovic, L., Karabegovic, I., Jahic, J., & Omercic, M. (2021). Optimal path planning of a disinfection mobile robot against COVID-19 in a ROS-based research platform. *Advances in Production Engineering & Management, 16*(4), 405-417. https://doi.org/10.14743/apem2021.4.409

[3] Muthukumaran, S., Sivaramakrishnan, R. (2019). Optimal Path Planning for an Autonomous Mobile Robot Using Dragonfly Algorithm. *International Journal of Simulation Modelling, 18*(3), 397-407. https://doi.org/10.2507/IJSIMM18(3)474

[4] Qureshi, A. H. & Ayaz, Y. (2016). Potential functions based sampling heuristic for optimal path planning. *Autonomous Robots*, *40*(6), 1079-1093. https://doi.org/10.1007/s10514-015-9518-0

[5] Elbanhawi, M. & Simic, M. (2014). Sampling-based robot motion planning: A review. *IEEE Access: Practical Innovations, Open Solutions*, *2*, 56-77. https://doi.org/10.1109/access.2014.2302442

[6] Tan, C. S., Mohd-Mokhtar, R., & Arshad, M. R. (2021). A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms. *IEEE Access: Practical Innovations, Open Solutions*, *9*, 119310-119342. https://doi.org/10.1109/access.2021.3108177

[7] Huang, C., Lan, Y., Liu, Y., Zhou, W., Pei, H., Yang, L., Cheng, Y., Hao, Y., & Peng, Y. (2018). A new dynamic path planning approach for unmanned aerial vehicles. *Complexity*, *2018*, 1-17. https://doi.org/10.1155/2018/8420294

[8] Aggarwal, S. & Kumar, N. (2020). Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications*, *149*, 270-299. https://doi.org/10.1016/j.comcom.2019.10.014

[9] LaValle, S. M. (1998). *Rapidly-exploring random trees : a new tool for path planning*. The annual research report.

[10] Kuffner, J. J. & LaValle, S. M. (2002). RRT-connect: An efficient approach to single-query path planning. Proceedings 2000 ICRA. Millennium Conference. *IEEE International Conference on Robotics and Automation. Symposia Proceedings*, (Cat. No.00CH37065). https://doi.org/10.1109/ROBOT.2000.844730

[11] Nikolos, K., Tsourveloudis, I., & Valavanis, C. (2001). Evolutionary Algorithm Based Off-Line Path Planner for UAV Navigation. *Automatika, 42*, 3-4.

[12] Nguyen, N. T., Schilling, L., Angern, M. S., Hamann, H., Ernst, F., & Schildbach, G. (2021). B-spline path planner for safe navigation of mobile robots. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 339-345. https://doi.org/10.1109/IROS51168.2021.9636612

[13] Jian, Z., Zhang, S., Zhang, J., Chen, S., & Zheng, N. (2022). Parametric Path Optimization for Wheeled Robots Navigation. *2022 International Conference on Robotics and Automation (ICRA)*, 10883-10889. https://doi.org/10.1109/ICRA46639.2022.9812167

[14] Saini, R., Pratim Roy, P., & Prosad Dogra, D. (2018). A segmental HMM based trajectory classification using genetic algorithm. *Expert Systems with Applications*, *93*, 169-181. https://doi.org/10.1016/j.eswa.2017.10.021

[15] Zhao, L., & Shi, G. (2018). A method for simplifying ship trajectory based on improved Douglas-Peucker algorithm. *Ocean Engineering*, *166*, 37-46. https://doi.org/10.1016/j.oceaneng.2018.08.005

[16] Marino, D. L. & Manic, M. (2016). Fast trajectory simplification algorithm for natural user interfaces in Robot programming by demonstration. *2016 IEEE 25th International Symposium on Industrial Electronics (ISIE)*, 905-911. https://doi.org/10.1109/ISIE.2016.7745011

[17] Saux, E. & Daniel, M. (1999). Data reduction of polygonal curves using B-splines. *Computer Aided Design*, *31*(8), 507-515. https://doi.org/10.1016/s0010-4485(99)00049-4

[18] Urmson, C. & Simmons, R. (2003). Approaches for heuristically biasing RRT growth. *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, *2*, 1178-1183. https://doi.org/10.1109/IROS.2003.1248805

[19] Wang, X., Luo, X., Han, B., Chen, Y., Liang, G., & Zheng K. (2020). Collision-Free Path Planning Method for Robots Based on an Improved Rapidly-Exploring Random Tree Algorithm. *Applied Science*, *10*, 1381. https://doi.org/10.3390/app10041381

[20] Bialkowski, J., Karaman, S., & Frazzoli, E. (2011). Massively parallelizing the RRT and the RRT. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3513-3518. https://doi.org/10.1109/IROS.2011.6095053

[21] Eriksson, U. (2018). *Dynamic Path Planning for Autonomous Unmanned Aerial Vehicles, Master in Robotics and Autonomous Systems*. KTH School of Electrical Engineering and Computer Science (EECS).

[22] Abou El Majd, Y., Ghazi, H. E., & Nahhal, T. (2017). PaRRT: Parallel rapidly exploring random tree (RRT) based on MapReduce. *2017 International Conference on Electrical and Information Technologies (ICEIT)*, 1-5. https://doi.org/10.1109/EITech.2017.8255234

[23] Zhang, Z., Wu, D., Gu, J., & Li, F. (2019). 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*, *7*(5), 132. https://doi.org/10.3390/jmse7050132

[24] Douglas, D. H. & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica The International Journal for Geographic Information and Geovisualization*, *10*(2), 112-122. https://doi.org/10.3138/fm57-6770-u75u-7727

[25] Cebrian, P. & Moure, J. C. (2020). GPU-Accelerated RDP Algorithm for Data Segmentation. *Computational Science - ICCS 2020. ICCS 2020. Lecture Notes in Computer Science*, 12137. https://doi.org/10.1007/978-3-030-50371-0_17

[26] Fernández-García, N. L., Del-Moral Martínez, L., Carmona-Poyato, A., Madrid-Cuevas, F. J., & Medina-Carnicer, R. (2016). A new thresholding approach for automatic generation of polygonal approximations. *Journal of Visual Communication and Image Representation*, *35*, 155-168. https://doi.org/10.1016/j.jvcir.2015.12.013

[27] Yang, F., Fang, X., Gao, F., Zhou, X., Li, H., Jin, H., & Song, Y. (2022). Obstacle avoidance path planning for UAV based on improved RRT algorithm. *Discrete Dynamics in Nature and Society*, *2022*, 1-9. https://doi.org/10.1155/2022/4544499

[28] Qi, J., Yang, H., & Sun, H. (2021). MOD-RRT*: A sampling-based algorithm for robot path planning in dynamic environment. *IEEE Transactions on Industrial Electronics (1982)*, *68*(8), 7244-7251. https://doi.org/10.1109/tie.2020.2998740

[29] Min, Y., Wu, K., & Li, K. (2021). Online Trajectory Planning Strategy for UAV in Dynamic Threats Based on Bug-Rapidly-exploring Random Tree Algorithm. *2021 China Automation Congress (CAC)*, 4814-4820. https://doi.org/10.1109/CAC53003.2021.9728317

[30] Fritsch, F. N. & Butland, J. (1984). A method for constructing local monotone piecewise cubic interpolants. *SIAM Journal on Scientific and Statistical Computing*, *5*(2), 300-304. https://doi.org/10.1137/0905021

[31] Moler, C. B. (2004). *Numerical Computing with Matlab*. SIAM, Philadelphia. https://doi.org/10.1137/1.9780898717952

[32] Akima, H. (1970). A new method of interpolation and smooth curve fitting based on local procedures. *Journal of the ACM*, *17*(4), 589-602. https://doi.org/10.1145/321607.321609

[33] Moler, C. (2019, April 29). *Makima piecewise cubic interpolation*. Cleve's Corner: Cleve Moler on Mathematics and Computing.

[34] Turley, R. S. (2018). *Cubic Interpolation with Irregularly-Spaced waypoints in Julia 1.4*. Brigham Young University, Faculty Publications.

Contact information:

Ayhan GÜLTEKİN, PhD Candidate
(Corresponding author)
Kocaeli University,
Department of Computer Engineering, Kocaeli University, Kocaeli, Turkey
E-mail: ayhangultekin38@gmail.com

Samet DİRİ, PhD Candidate
Kocaeli University,
Department of Computer Engineering, Kocaeli University, Kocaeli, Turkey
E-mail: sametdiri@gmail.com

Yaşar BECERİKLİ, Full Professor
Kocaeli University,
Department of Computer Engineering, Kocaeli University, Kocaeli, Turkey
E-mail: ybecerikli@gmail.com