iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** XII **Month of publication:** December 2021

DOI: https://doi.org/10.22214/ijraset.2021.39297

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue XII Dec 2021- Available at www.ijraset.com

A Comparison of the Effectiveness of the RRT, PRM, and Novel Hybrid RRT-PRM Path Planners

Jacqueline Jermyn
Department of Electrical and Computer Engineering, Florida Agricultural and Mechanical University-Florida State University College of Engineering, Tallahassee, Florida, United States of America

Abstract: ***Sampling-based path planners develop paths for robots to journey to their destinations. The two main types of sampling-based techniques are the probabilistic roadmap (PRM) and the Rapidly Exploring Random Tree (RRT). PRMs are multi-query methods that construct roadmaps to find routes, while RRTs are single-query techniques that grow search trees to find paths. This investigation evaluated the effectiveness of the PRM, the RRT, and the novel Hybrid RRT-PRM methods. This novel path planner was developed to improve the performance of the RRT and PRM techniques. It is a fusion of the RRT and PRM methods, and its goal is to reduce the path length. Experiments were conducted to evaluate the effectiveness of these path planners. The performance metrics included the path length, runtime, number of nodes in the path, number of nodes in the search tree or roadmap, and the number of iterations required to obtain the path. Results showed that the Hybrid RRT-PRM method was more effective than the PRM and RRT techniques because of the shorter path length. This new technique searched for a path in the convex hull region, which is a subset of the search area near to the start and end locations. The roadmap for the Hybrid RRT-PRM could also be re-used to find pathways for other sets of initial and final positions.***

Keywords: Path Planning, Sampling-based algorithms, search tree, roadmap, single-query planners, multi-query planners, Rapidly Exploring Random Tree (RRT), Probabilistic Roadmap (PRM), Hybrid RRT-PRM

I. INTRODUCTION

Robots use sampling-based path planners to generate routes to travel from their starting to their goal positions, while avoiding obstructions that may be found along their pathways [1]. Sampling-based algorithms, to include probabilistic roadmaps (PRM) and rapidly exploring random trees (RRT), are probabilistically complete. They will find a route if enough iterations have been accomplished and if a path exists [2-6]. PRMs are multi-query planners that create roadmaps to find routes. These roadmaps could be re-used to find paths for other sets of starting and final locations [7-12]. RRTs are single-query techniques that build search trees to output paths, but they must be re-executed to generate routes for new pairs of initial and final positions [10].

The RRT algorithm grows a search tree by selecting points from its environment and connecting them to this search tree if these points do not cause collisions with objects [9]. When the search tree reaches the goal location, a route is produced by working backwards to pinpoint the parent node for each node in the pathway until it backtracks to its starting point [7, 10]. The runtime for RRTs is usually shorter than other types of planners because RRTs terminate when a route is found, but the path may not be optimized [9]. The PRM planner chooses points from its search area and connects them to form a roadmap [7,10,12,13]. A graph search method, such as Dijkstra's algorithm or A*, will then be utilized to search this roadmap for the most direct path to the destination [8,14]. However, parts of this roadmap could become entrapped inside obstacles because points located inside objects are not discarded [15]. A novel multi-query sampling-based path planner, the Hybrid RRT-PRM, was developed to improve the performance of the RRT and PRM techniques by shortening the path length from the starting to the end locations. The Hybrid RRT-PRM planner is a synthesis of these two techniques. It generates the first path by using the RRT planner, and a second path by employing the PRM algorithm. The outermost points of these paths define the perimeter of a polygon, also known as the convex hull region, which is a subset of the search area. Points are selected from the convex hull area because this region is close to the initial and final positions. In addition, points that are located outside objects are saved, while points that are inside objects are discarded to prevent the roadmap from being trapped within these obstructions. A roadmap is created inside the convex hull area by connecting these saved points. A graph search technique looks for a path to the destination by using this roadmap. Since the Hybrid RRT-PRM is also a multi-query planner, its roadmap can be re-purposed to generate paths for other sets of initial and final positions. Experiments were conducted to determine the effectiveness of the Hybrid RRT-PRM method when compared to the RRT and PRM planners. The performance metrics that were measured included the runtime, number of iterations, number of nodes in the search tree or roadmap, number of nodes in path, and the path length. The relationship between the RRT, PRM, and Hybrid RRT-PRM path planners is provided in Fig. 1.

The remainder of the paper is structured as follows: Section 2 discusses the RRT, PRM, and Hybrid RRT-PRM planners. Section 3 provides the setup for the experiments. Section 4 reviews the experimental findings. Section 5 concludes this study and briefly describes future experiments.

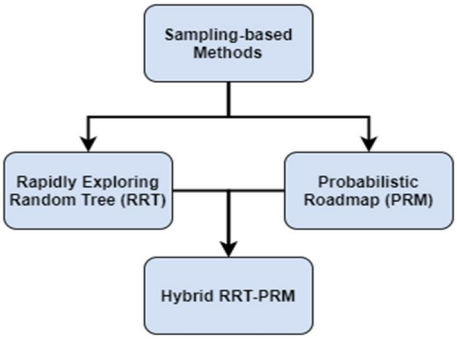


Fig. 1 The Relationship Among the RRT, PRM, and Hybrid RRT-PRM Path Planners

II. DISCUSSION OF RRT, PRM, AND HYBRID RRT-PRM PLANNERS

This section of the paper reviews the activity diagrams for the RRT, the PRM, and the Hybrid RRT-PRM algorithms to explain how they generate routes from the starting to end locations.

A. *Rapidly Exploring Random Tree (RRT)*

The RRT technique begins by initializing the variables. These include the obstacle list, the start and goal locations, the maximum edge expansion distance, the search area boundaries, and the search tree. Next, the starting location is added to the search tree, and a determination is made to see if the destination has been reached. If this is the case, this end point is appended to the search tree. A path is generated, the search tree is outputted, and the algorithm is concluded.

If the destination has not been found, the RRT selects a point from the search area. The distance from this point to the nearest node in the search tree is calculated by using the distance formula as shown as follows [4]:

$$d_{nearest,j} = \sqrt{\left(node_{j,y} - rnd_y\right)^2 + \left(node_{j,x} - rnd_x\right)^2} \quad (1)$$

where $d_{nearest,j}$ is the distance between the node with index j in the list of nodes in the search tree and the random point, $node_{j,y}$ is the y coordinate of the j^{th} node in the node list, rnd_y is the y coordinate of the randomly generated point, $node_{j,x}$ is the x coordinate of j^{th} node in the node list, and rnd_x is the x coordinate of the randomly generated point. The nearest node in the tree to the random point has the minimum $d_{nearest,j}$ value.

The RRT then uses the maximum edge expansion distance to calculate the location of a new node. If the randomly generated point is within the maximum edge expansion distance to the nearest node in the tree, this random point will become the new node, but it will not be connected to the search tree yet [4].

If the random point is further away from the nearest node in the tree, the new node is found by interpolating along the line segment between the nearest node in the tree and the randomly generated point [15]. The coordinates for the new node are calculated as follows:

$$new_y = d_{exp} \sin\theta + nearest_y \quad (2)$$

$$new_x = d_{exp} \cos\theta + nearest_x \quad (3)$$

where new_y is the y coordinate of the new node, new_x is the x coordinate of the new node, θ is the angle between the random point and the nearest node, d_{exp} is the maximum edge expansion distance, $nearest_y$ is the y coordinate of the nearest node in the tree, and $nearest_x$ is the x coordinate of the nearest node in the tree. The angle between the nearest node and the random point, θ, is calculated as follows:

$$\theta = \text{atan2}(dy, dx) \quad (4)$$

$$dy = rnd_y - nearest_y \quad (5)$$

$$dx = rnd_x - nearest_x \quad (6)$$

where d_y is the change in position along the y axis, while d_x is the change in position along the x axis. In addition, rnd_y is the random point's y coordinate, $nearest_y$ is the nearest node's y coordinate, rnd_x is the random point's x coordinate, and $nearest_x$ is the nearest node's x coordinate [15]. After a new node has been generated, it is checked to determine if connecting this new node to the nearest node in the tree would cause a collision. If this is the case, this new node will be discarded, and the RRT method advances to the next iteration. If the line segment between the nearest node in the tree and the new node does not touch any obstacles, the new node is added to the tree by connecting it to the nearest node. This planner advances to the next iteration. The RRT continues to add collision-free nodes to the search tree until it reaches the destination. The goal position is then added to the search tree. The search tree and path are outputted, and the algorithm terminates. The RRT activity diagram is provided in Fig. 2.

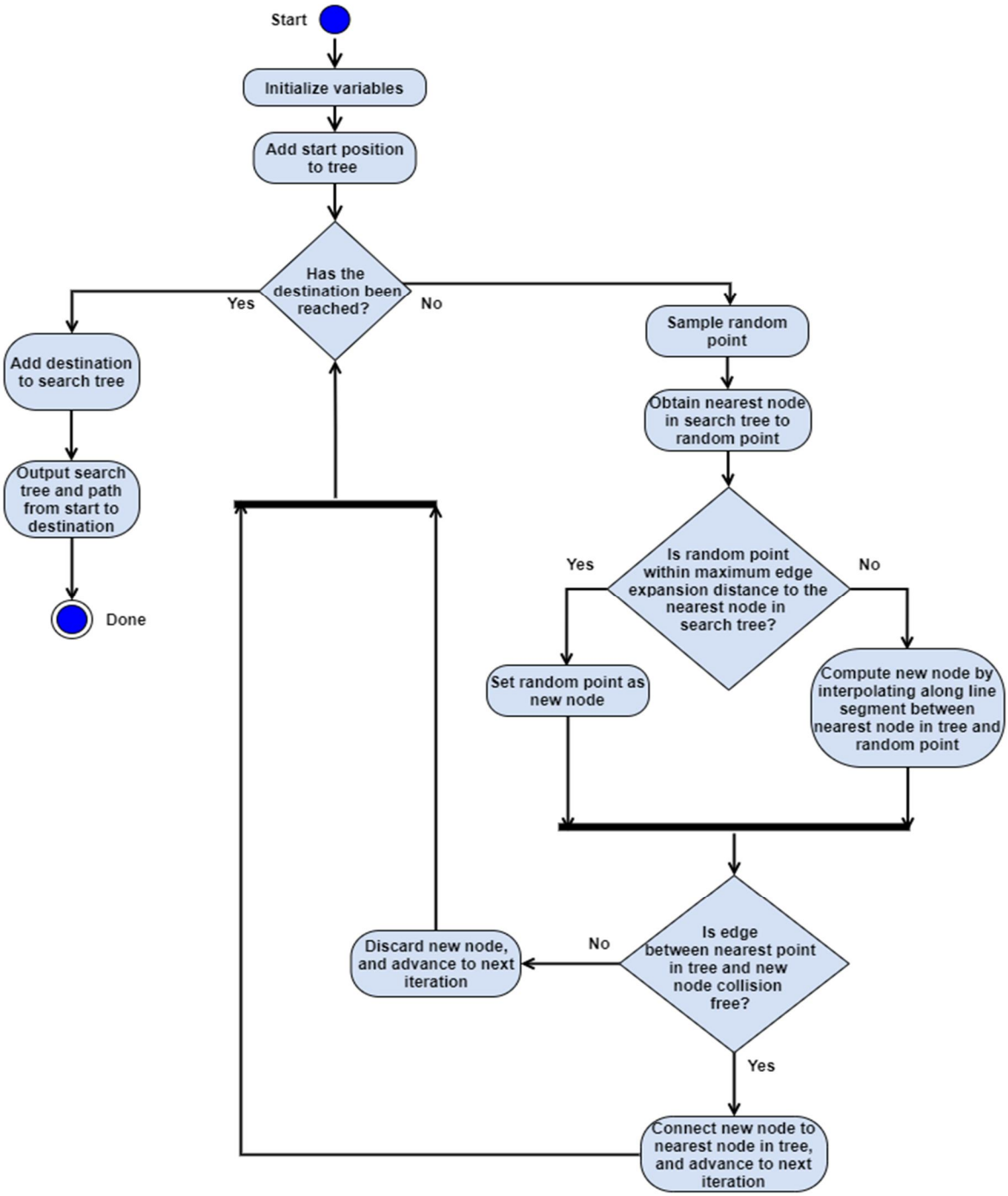


Fig. 2 Activity Diagram for RRT

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue XII Dec 2021- Available at www.ijraset.com

B. Probabilistic Roadmap (PRM)

The PRM method begins by initializing variables. These include the start and end positions, search area boundaries, number of iterations, the maximum number of nearest neighboring nodes for each node in the roadmap, the maximum edge expansion distance, and the list of obstacles. It then checks to see if the pre-specified number of iterations have been completed. If this is not the case, the PRM randomly selects points until the pre-specified number of points have been obtained from the search area. A K-Dimensional (K-D) tree is formed from these points. A K-D tree is a data-structure where points are grouped into a binary tree to facilitate performing nearest-neighbor queries on points in this tree [16-18]. For a K-D tree, K represents the number of dimensions of the search area [18]. For each node in the K-D tree, its nearest neighboring nodes are identified. Collision-free nearby nodes will be connected to this node until all these neighboring nodes that are located within the maximum edge expansion distance or if the maximum number of nearest neighboring nodes have been linked.

When all nodes in the K-D tree have been connected, a roadmap is formed. A graph search algorithm searches it for the shortest path from the start to the goal location. When this path is found, the PRM outputs this route and the roadmap. This planner then terminates. Fig. 3 shows the PRM activity diagram.

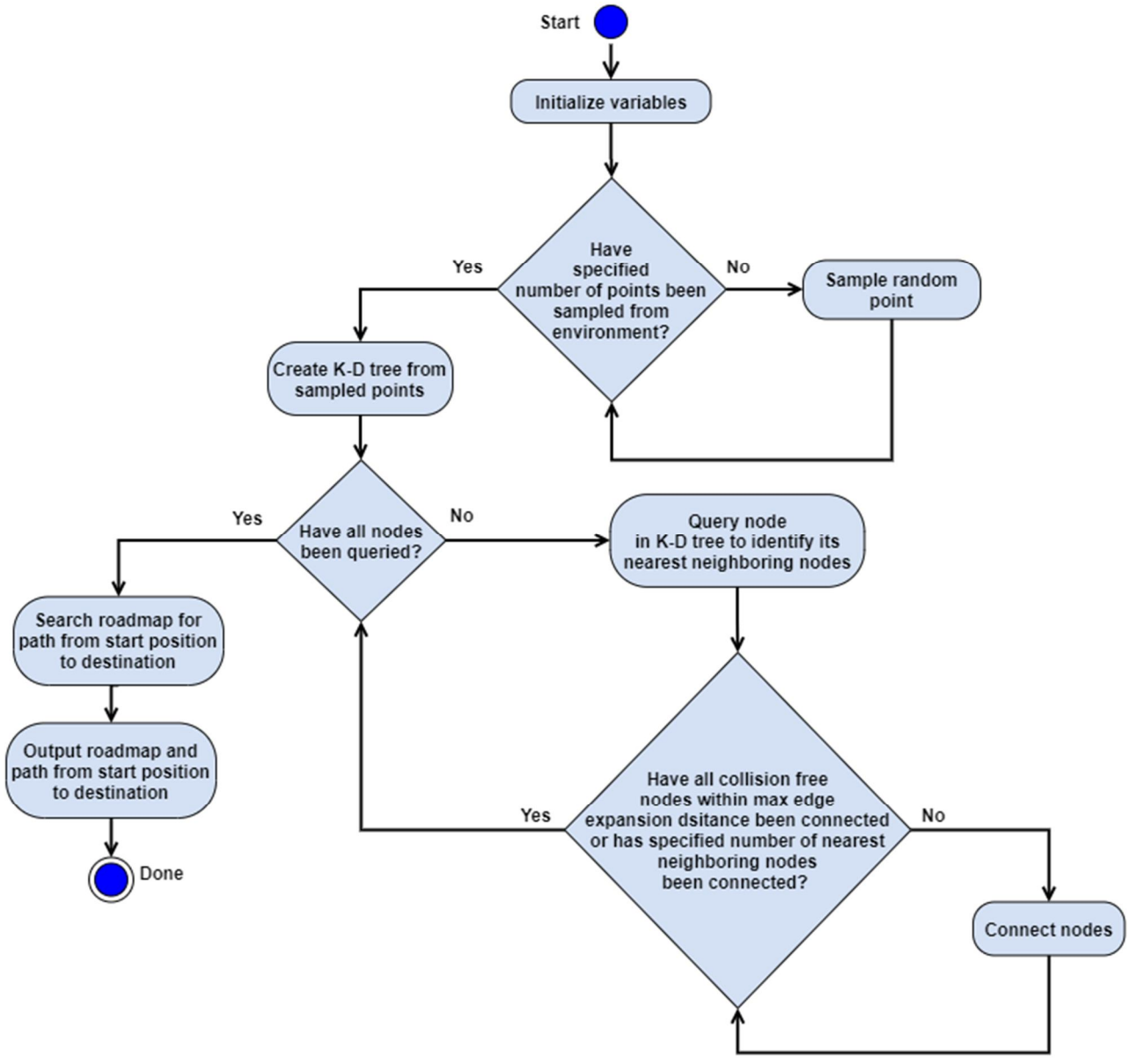


Fig. 3 Activity Diagram for PRM

C. Hybrid RRT-PRM

The Hybrid RRT-PRM technique is a new type of sampling-based algorithm. It combines the RRT and the PRM techniques to generate shorter paths. The Hybrid RRT-PRM algorithm begins by initializing variables, which include the start and end positions, the limits of the search area, the number of iterations, and the maximum number of nearest neighboring nodes for each node in the roadmap, the maximum edge expansion distance, and the obstacle list. It then plans two routes. The RRT method generates the first path; after which, the PRM planner finds a second path. The equations for these paths are provided as follows:

$$path_{rrt} = \left[[r_{x1}, r_{y1}], [r_{x2}, r_{y2}], \dots, [r_{xn}, r_{yn}] \right] \quad (7)$$

$$path_{prm} = \left[[p_{x1}, p_{y1}], [p_{x2}, p_{y2}], \dots, [r_{xm}, r_{ym}] \right] \quad (8)$$

where $path_{rrt}$ is the list of nodes in the RRT path, r_{x1} and r_{y1} are the x and y coordinates of the first node in the RRT path, and n is the number of nodes in this path. For the PRM path, $path_{prm}$ is the list of nodes in the PRM path, p_{x1} and p_{y1} are the x and y coordinates of the first node in the PRM path, and m is the number of nodes in the path. The paths obtained by the RRT and PRM methods are then merged into a combined list of nodes, which is provided as follows:

$$path_{combined} = \left[[r_{x2}, r_{y2}], \dots, [r_{xn-1}, r_{yn-1}], [p_{x1}, p_{y1}], [p_{x2}, p_{y2}], \dots, [p_{xm}, p_{ym}] \right] \quad (9)$$

where $path_{combined}$ is the combined list of nodes. The first and final nodes of the RRT path are excluded from the combined list because they are already included in the PRM path. The number of points in the combined list of nodes is computed as follows:

$$s = n + m - 2 \quad (10)$$

where s is the number of points in the combined list, n is the number of nodes in the RRT path, and m is the number of nodes in the PRM path. The combined list of nodes is then converted into an array.

The Hybrid RRT-PRM then determines if the pre-specified number of points have been obtained from the subsection of the search area near to the start and goal locations. This region of the search area, known as the convex hull, is obtained by connecting the outermost points in the combined list of nodes [19]. If the specified number of points have not been obtained from the convex hull, a random point is sampled from the entire rectangular search area. Linear programming is now applied to determine if the random point is located within the convex hull [20]. Linear programming is a mathematical technique that optimizes a function that is subjected to constraints [21-22]. The general linear programming problem is defined as

$$\min_{\boldsymbol{x}} \boldsymbol{cx} \text{ such that } A_{eq}\boldsymbol{x} = \boldsymbol{b}_{eq} \text{ and } \mathbf{x} \geq 0 \quad (11)$$

where $\boldsymbol{c}$ is a row vector whose elements are the coefficients of the objective function, while $\boldsymbol{x}$ is the column vector that is the solution to the linear programming problem. The $\boldsymbol{x}$ vector has the same number of elements as the $\boldsymbol{c}$ vector. In addition, A_{eq} is the equality constraint matrix, and $\boldsymbol{b}_{eq}$ is the equality constraint column vector [23-24].

When linear programming is applied in the Hybrid RRT-PRM algorithm, the objective function vector is a row vector of zeros that has s elements because there is no function to be optimized. The equation for the equality constraint matrix, A_{eq}, is provided as follows:

$$A_{eq} = \begin{bmatrix} r_{x2} & \dots & r_{xn-1} & p_{x1} & p_{x2} & \dots & p_{xm} \\ r_{y2} & \dots & r_{yn-1} & p_{y1} & p_{y2} & \dots & p_{ym} \\ 1 & 1 & 1 & 1 & 1 & \dots & 1 \end{bmatrix} \quad (12)$$

For this matrix, every column contains a point in the combined list of nodes followed by a one to fulfill the requirement for obtaining a convex combination [20]. The equality constraint vector, $\boldsymbol{b}_{eq}$,is provided as follows:

$$\boldsymbol{b}_{eq} = \begin{bmatrix} rnd_x \\ rnd_y \\ 1 \end{bmatrix} \quad (13)$$

where rnd_x is the x coordinate of a randomly sampled point from the entire rectangular search area, while rnd_y is the y coordinate of this randomly sampled point.

Linear programming then performs the simplex method to determine if the linear program is feasible and to obtain a solution [24]. A linear program is infeasible when at least one of its artificial variables is a positive number. It is feasible when none of its artificial variables are positive numbers [25].

The simplex algorithm consists of two phases. The first phase minimizes the sum of the artificial variables, determines the feasibility of the linear program, and calculates a base form solution. The second phase uses this base form solution to solve the linear program [25]. Applying linear programming to determine if a point is located inside the convex hull is beneficial because the convex hull does not need to be calculated to determine whether a randomly generated point is located within this region of the search area [20].

For the Hybrid RRT-PRM PRM method, if phase I of the simplex method finds at least one artificial variable that is a positive number, the randomly sampled point is located outside the convex hull search area. This point is then discarded. When none of the artificial variables are positive numbers, the randomly sampled point is located within the convex hull. It is saved to a list for further processing [25-26]. The second phase of the simplex method uses the base form solution to find an optimal solution, but phase II is not implemented because there is no objective function to be optimized.

When the specified number of points have been obtained, this planner iterates over these points to identify the ones that are located inside obstacles. These points will be discarded to prevent a roadmap from being created within them. Points that are outside obstacles will be saved to a list to make a K-D tree. The nearest neighboring nodes are identified for each node in the K-D tree. Collision-free nodes will be connected to this node until all neighboring nodes within the maximum edge expansion distance or if the maximum number of nearest neighboring nodes have been attached to it. When all nodes have been connected, a roadmap is formed from them. A graph search technique will search this roadmap for the shortest pathway to the destination. The Hybrid RRT-PRM algorithm then outputs the roadmap and the path. This algorithm then terminates. Fig. 4 provides the activity diagram for the Hybrid RRT-PRM algorithm.

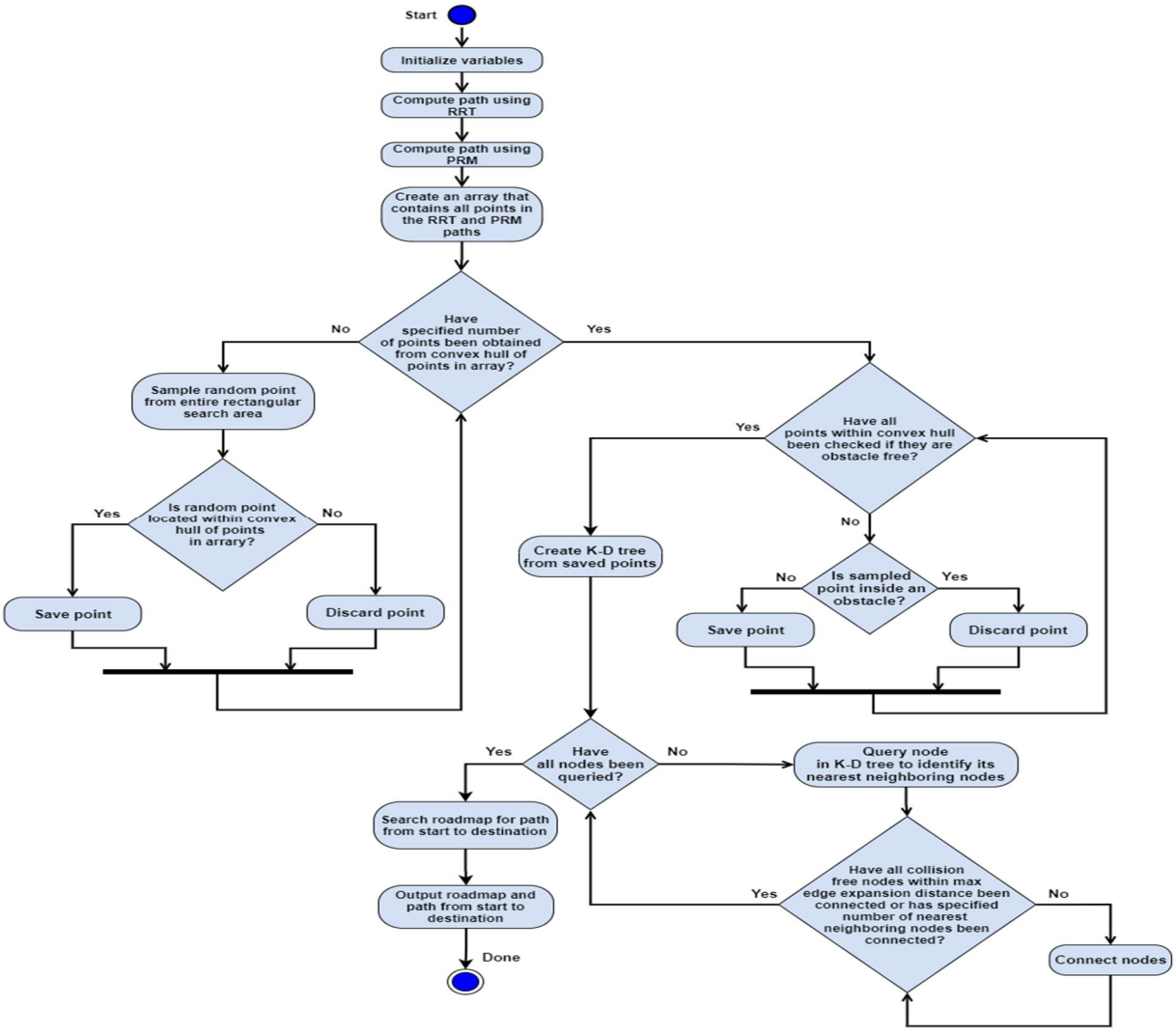


Fig. 4 Activity Diagram for Hybrid RRT-PRM

III.SETUP FOR EXPERIMENTS

This investigation evaluated the effectiveness of the Hybrid RRT-PRM planner, when compared with the RRT and PRM techniques. Single-query experiments were accomplished for these three planners in a two-dimensional square search area with corners located at (0,0), (0, 40), (40, 40), and (40, 0). The beginning and end positions of the paths were initialized at (10, 10) and (30, 30) respectively. Fifteen circular obstacle sets were placed in this environment. Each set consisted of 12 randomly generated obstacles. The size of the radii for these circles and the positions of the circle centers were found using uniform probabilistic distributions. The circle radii ranged from one to four units in size. The placement of the x and y coordinates of the circle centers ranged from four to thirty-six so that the circular obstacles would be contained entirely within the square search area. Each algorithm was executed 450 times with a maximum of 500 iterations per trial.

The Matplotlib, NumPy, PythonRobotics, and SciPy toolboxes were used to conduct these experiments [15, 27-31]. Conducting simulations to evaluate the effectiveness of these algorithms to generate paths removed errors associated with obstacle localization, robot localization, and robot motion execution. The maximum edge expansion distance for these path planning experiments was five units. For the RRT algorithm, the maximum edge expansion distance is used to calculate the locations of the new nodes that would be added to the search tree [15].

For the PRM and Hybrid RRT-PRM algorithms, the maximum edge expansion distance is the maximum separation between two nodes for them to be connected to each other in a roadmap [8]. For the Hybrid RRT-PRM and the PRM methods, the maximum number of nearest neighboring nodes that could be connected to a node in the roadmap was 100 nodes. These two path planners also used Dijkstra's Shortest Path Algorithm to search their roadmaps for paths [15]. This graph search algorithm was implemented because it required all edges of the roadmaps to have positive values [32-34]. The roadmaps met this requirement because the distances between nodes are positive numbers [32].

The performance metrics that were recorded were the path length, runtime, number of nodes in the path, number of nodes in the roadmaps or search trees, and the number of iterations for each path planner to output a path. The path length was obtained by adding the distances between each consecutive pair of nodes in the route from the initial position to the destination. The path length is also the dominant criterion for establishing the most efficient path planning algorithm since robots with shorter paths arrive at their destinations quicker.

The runtime is the amount of time for each path planner to output a path, but it did not include the time to display the search trees, roadmaps, and paths. The number of nodes in the paths was obtained by counting the starting position, the waypoints, and the end position. The number of nodes in the search tree or roadmap corresponded to the number of points in the search tree for the RRT technique. It corresponded to the number of points in the roadmap for the PRM and Hybrid RRT-PRM planners. For all path planning methods, the number of iterations was the number of points sampled from the environment.

After the simulations were completed for each type of planner, the data for each performance metric was stored to a list so that the sample statics could calculated for further analysis. The sample statistics included the mean, standard deviation, mode, median, maximum, and minimum values for each performance metric. Images were also captured for search trees, roadmaps, and paths outputted by these path planning algorithms.

IV.EXPERIMENTAL FINDINGS

For this investigation, the performance measures that were collected to evaluate the effectiveness of these three types of sampling-based planners were the path length, runtime, nodes in path and nodes in tree, and the number of iterations that were required to generate a path. In addition, sample statistics were also compiled so that meaningful conclusions could be drawn from the data. The statistical values that were saved include the mean, the standard deviation, the mode, the median, the maximum, and the minimum. To facilitate analysis of the effectiveness of these types of algorithms, images of the search trees, roadmaps, and paths were captured. For the search tree and roadmap diagrams, the red "x's" are the starting and the end positions. The blue circles are the round obstacles. For the RRT experiments, the green lines are the search tree. For the PRM and Hybrid RRT-PRM methods, the green lines are the roadmap. For the path diagrams, the path is shown in red. All paths begin at (10,10) and end at (30,30).

For the RRT method, the mean path length was 39.054 and the runtime to obtain a path was 0.007 s. The number of nodes in the path and in the tree were 9.593 and 24.027 respectively. The RRT planner outputted a path after 37.173 iterations because it terminates immediately when the search tree connects to the destination. The path, however, was not optimized. The RRT planner results are provided in Table I.

TABLE I
Results for RRT Experiments

Quantity	Mean	Standard Deviation	Mode	Median	Max	Min
Path Length	39.054	7.065	none	37.613	63.973	28.408
Runtime (s)	0.007	0.010	0.000	0.000	0.047	0.000
Number of Nodes in Path	9.593	1.706	9.000	9.000	16.000	7.000
Number of Nodes in Tree	24.027	19.224	10.000	19.000	148.000	7.000
Number of Iterations	37.173	30.789	23.000, 15.000	27.000	222.000	7.000

For this RRT investigation, images for the search tree and path were acquired. Fig. 5a displays the RRT search tree for Obstacle set 1, Trial 1, while Fig. 5b shows the path from the start to the end locations. As can be seen in Fig. 5a, the search tree was not dense because it only took an average of 37.173 iterations to obtain a path. The path also remained outside obstacles because the search tree was grown from the starting location by iteratively connecting to nodes that were found outside obstacles. Points generated inside an obstacle were discarded, and they were not connected to the search tree. Fig. 5b shows that the RRT path meandered because the path was not optimized.

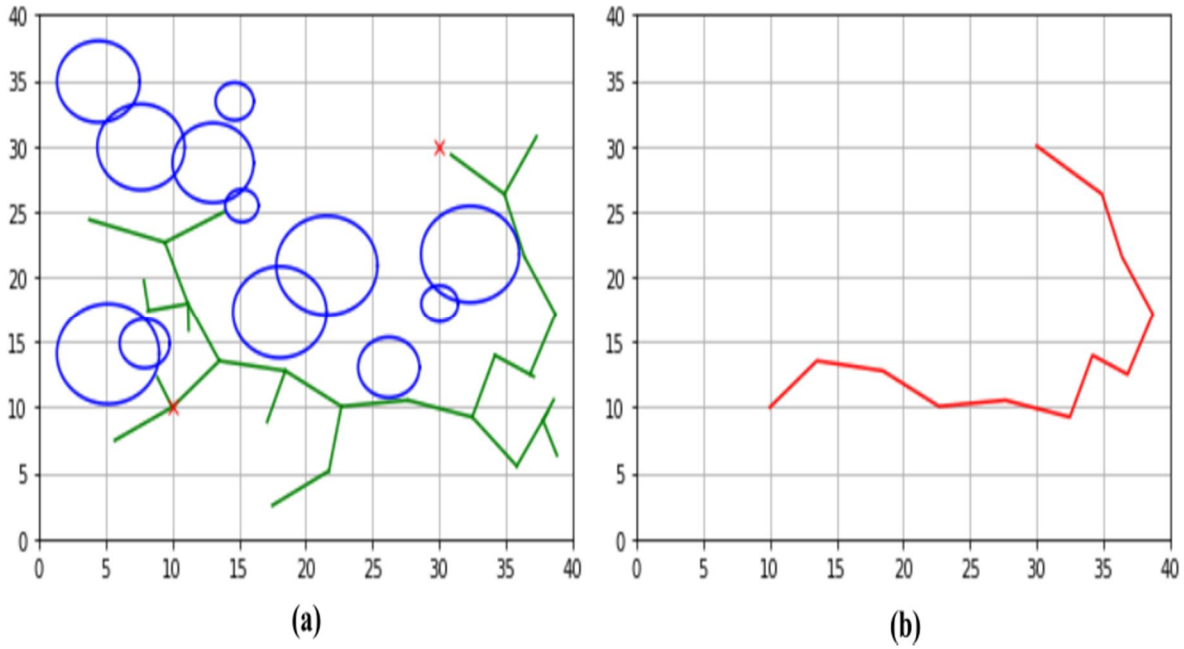


Fig. 5a. Search Tree for RRT Obstacle Set 1, Trial 1. Fig. 5b. Path for RRT Obstacle Set 1, Trial 1.

For the PRM experiments, the average length of the path was 30.693, and the runtime was 0.482 s. The number of nodes in the path and in the roadmap were 9.611 and 500 respectively. The number of nodes in the roadmap was 500 because points that were located inside obstacles were not removed. It also took 500 iterations to complete each trial because this number was pre-set during the initialization of the program. This planner sampled points until the pre-specified number of iterations were reached. The results for these experiments are provided in Table II.

TABLE III
Results for PRM Experiments

Quantity	Mean	Standard Deviation	Mode	Median	Max	Min
Path Length	30.693	1.565	none	30.399	39.042	28.465
Runtime (s)	0.482	0.268	none	0.472	0.997	0.004
Number of Nodes in Path	9.611	0.827	10.000	10.000	12.000	8.000
Number of Nodes in Roadmap	500	0.000	500	500	500	500
Number of Iterations	500	0.000	500	500	500	500

The diagrams for the roadmap and for the path were also captured for the PRM experiments. Fig. 6a shows the roadmap for Obstacle set 1, Trial 1. Fig. 6b displays the path from the starting point to the destination. Fig. 6a shows that the roadmap was very dense, and it covered most of the search area. In addition, portions of the roadmap were trapped inside the round obstacles, which could not be accessed to travel to the destination. Fig. 6b shows that the path was more direct because the graph search algorithm searched this highly interconnected roadmap for a shorter path.

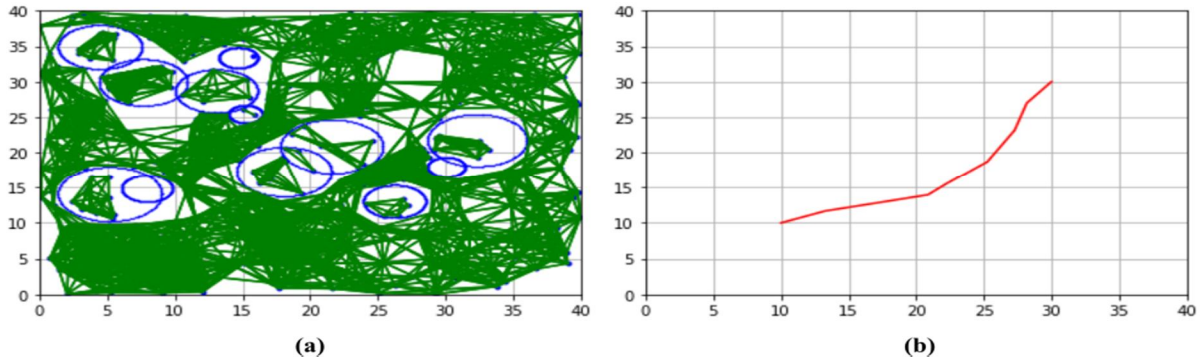


Fig. 6a. Search Tree for PRM Obstacle Set 1, Trial 1. Fig. 6b. Path for PRM Obstacle Set 1, Trial 1.

The Hybrid RRT-PRM technique had a mean path length of 29.610. Its runtime was 0.473 s. The number of nodes in the path and in the roadmap were 10.798 and 380.384 respectively. The number of nodes in the roadmap was less than 500 because the randomly generated points that were located inside obstacles were eliminated. The number of iterations, however, was 500 because this planner continued to iterate until the prespecified number of iterations were accomplished. The results for the Hybrid RRT-PRM method are listed in Table III.

TABLE IIIII
Results For Simulations for Hybrid RRT-PRM Method

Quantity	Mean	Standard Deviation	Mode	Median	Max	Min
Path Length	29.610	1.439	none	29.285	37.992	28.306
Runtime (s)	0.473	0.280	none	0.449	0.997	0.003
Number of Nodes in Path	10.798	1.349	11.000	11.000	15.000	8.000
Number of Nodes in Roadmap	380.384	52.722	362.000, 371.000, 372.000, 388.000	378.500	477.000	231.000
Number of Iterations	500	0.000	500	500	500	500

The plots for the roadmap and for the path were also taken for the Hybrid RRT-PRM experiments. Fig. 7a depicts the roadmap for Obstacle set 1, Trial 1, and Fig. 7b exhibits the path from the beginning to the final locations. Figure 7a shows that the roadmap was extremely dense, and it was entirely contained within the convex hull region of the search area and outside the obstacles. This convex hull was created by finding two paths using the RRT and PRM methods and then joining the outermost points in these paths. Sampling points within the convex hull generated shorter paths because the search area was smaller. This area was also near the start and goal locations. The Hybrid RRT-PRM method discarded points that were sampled inside obstacles, and this further reduced the number of nodes in the roadmap. Fig. 7b shows that the route was very direct, and it only had a slight turn to go around obstacles that were found within the convex hull area.

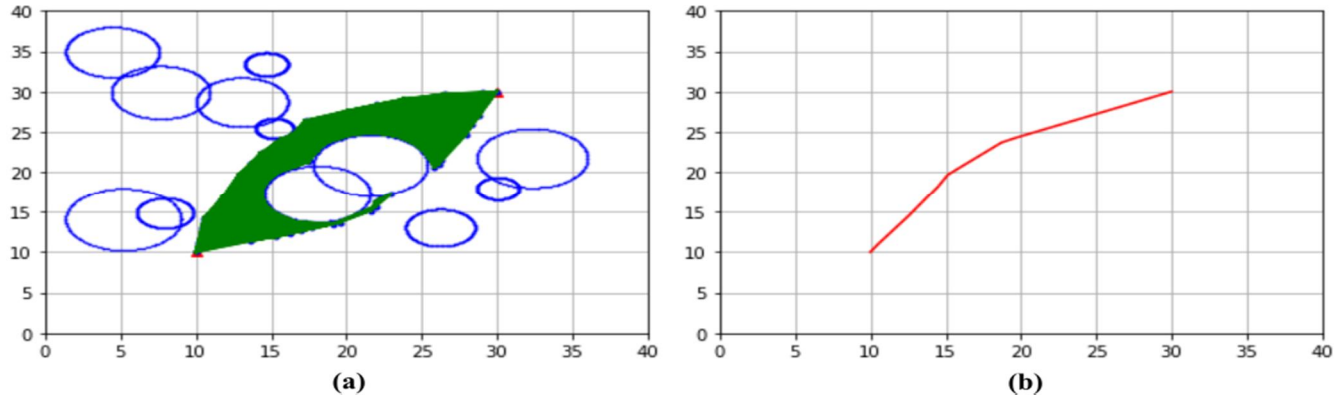


Fig. 7a. Search Tree for Hybrid RRT-PRM Obstacle Set 1, Trial 1. Fig. 7b. Path for Hybrid RRT-PRM Obstacle Set 1, Trial 1

The results for the Hybrid RRT-PRM, RRT, and PRM methods were compared to evaluate their effectiveness to find paths. These metrics include the length of the path, runtime, number of nodes in the path, number of nodes in the tree or roadmap, and the number of iterations. Table IV provided a summary of the performance metrics for these three types of algorithms.

TABLE IVV
Summary of Results

Algorithm	Path Length	Runtime	Number of Nodes in Path	Number of Nodes in Tree or Roadmap	Number of Iterations
RRT	39.054	0.007	9.593	24.027	37.173
PRM	30.693	0.482	9.611	500	500
Hybrid RRT-PRM	29.610	0.473	10.798	380.384	500

The RRT, PRM, and Hybrid RRT-PRM path planners successfully outputted paths. When the RRT method was compared to the Hybrid RRT-PRM technique, it had a shorter runtime, fewer nodes in the tree, and fewer iterations. The path was not optimized because this algorithm terminated as soon as a path was found. The Hybrid RRT-PRM method yielded a shorter path length. Since the length of the path is the overriding factor to establish the efficacy of a path planner, the Hybrid RRT-PRM planner had a better performance than the RRT. Furthermore, the Hybrid RRT-PRM algorithm is a multi-query technique so its interconnected roadmap could be re-purposed to obtain paths for different pairs of initial and final positions. By contrast, the RRT method is a single-query technique, and this algorithm must be re-executed to obtain routes for different sets of beginning and end points.

When the PRM planner was compared to the Hybrid RRT-PRM algorithm, it had fewer nodes in its path, but the Hybrid RRT-PRM algorithm had fewer nodes in its roadmap resulting in shorter runtimes. The Hybrid RRT-PRM also outputted shorter paths making it more effective than the PRM. In addition, both these planners are multi-query methods, so their roadmaps could be re-utilized. The PRM method does not discard points that are found inside obstacles, and part of its roadmap could become entrapped. The Hybrid RRT-PRM, on the other hand, discards points within obstacles thereby reducing the number of nodes in the roadmap and shortening the runtime.

V. CONCLUSIONS

This paper introduced and evaluated the performance of a new type of multi-query sampling-based path planner, the Hybrid RRT-PRM method that integrates the RRT and PRM techniques. The goal of this novel algorithm is to improve the performance by decreasing the path length. Experiments were conducted to compare the efficacy of the Hybrid RRT-PRM method with the RRT and PRM planners. The performance metrics that were recorded were the path length, the runtime, the number of nodes in the path, the number of nodes in the roadmap or search tree, and the number of iterations needed to output a route. Results showed that the Hybrid RRT-PRM technique is more effective than the other two methods because it yielded shorter path lengths since it only searched for paths in the convex hull region that was a subset of the search area. The Hybrid RRT-PRM planner is also a multi-query method, and its roadmap could be re-applied to output routes for different sets of beginning and end points. Future research will entail developing and evaluating new variations of the Hybrid RRT-PRM algorithm to include the Hybrid RRT*-PRM* and the Hybrid Informed RRT*-PRM* techniques.

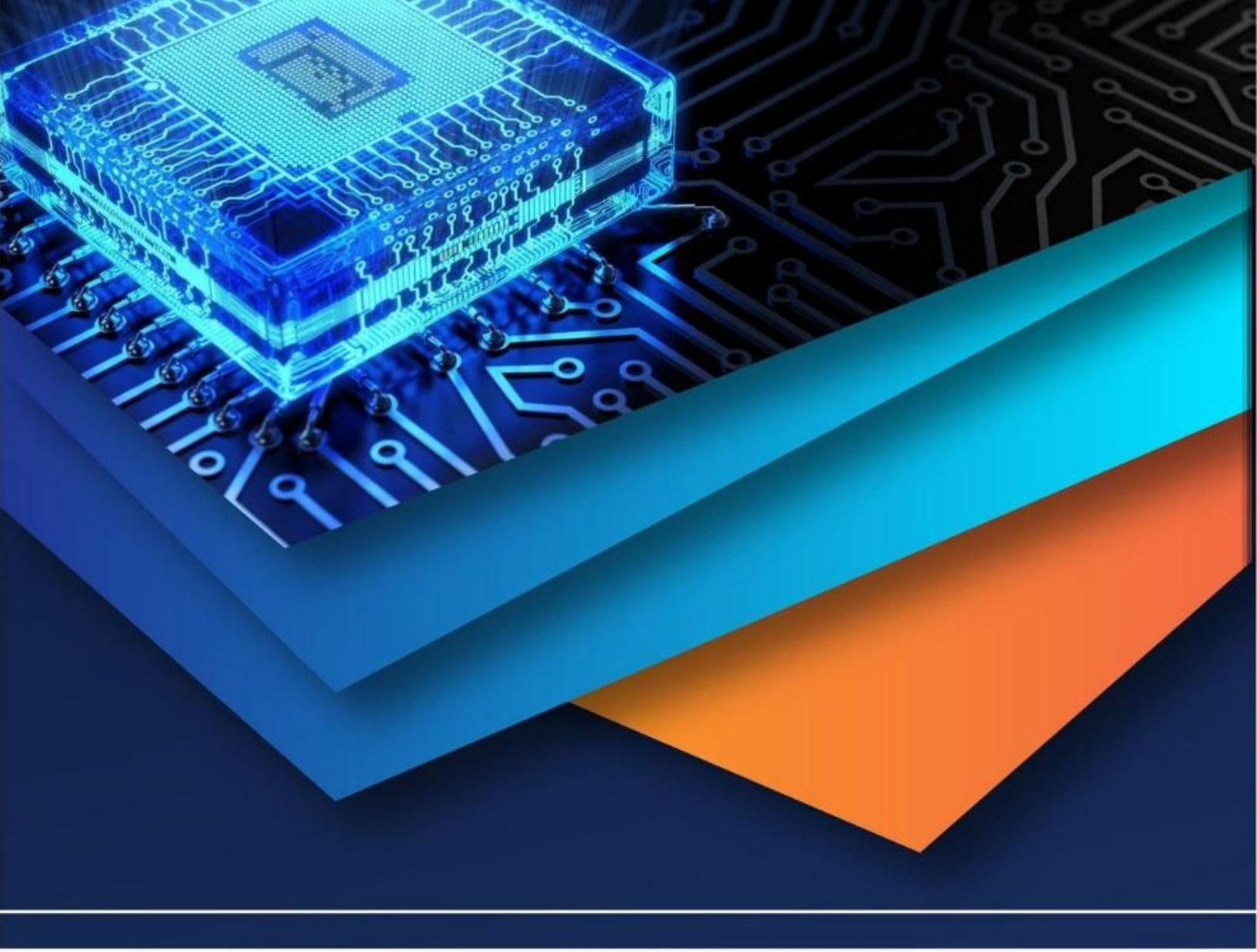
VI. ACKNOWLEDGMENT

The author would like to thank the SMART Scholarship program, the FAMU-FSU College of Engineering Dean's Fellowship program, and the Electrical and Computer Engineering Department for their support.

REFERENCES

[1] S. Murray, W. Floyd-Jones, Y. Qi, D. Sorin, and G. Konidaris, "Robot Motion Planning on a Chip," in Proc. Of the 2016 Robotics: Science and Systems Conf., 2016, doi: 10.15607/RSS.2016.XII.004.
[2] J. D. Gammell, S. S. Srinivasa and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 2997-3004, doi: 10.1109/IROS.2014.6942976.
[3] J. D. Gammell, S. S. Srinivasa and T. D. Barfoot, "Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 3067-3074, doi: 10.1109/ICRA.2015.7139620.
[4] S. Karaman, and E. Frazzoli, "Incremental Sampling-based Algorithms for Optimal Motion Planning," in Proc. of the 2010 Robotics: Science and Systems Conf., Zaragoza, Spain, Jun. 27-30, 2010.
[5] S. LaValle, "Continuous Discrete," in Planning Algorithms, 1st ed. New York, USA: Cambridge University Press, 2006, pp. 79-80.
[6] A. Perez, R. Platt, G. Konidaris, L. Kaelbling and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," 2012 IEEE International Conference on Robotics and Automation, 2012, pp. 2537-2542, doi: 10.1109/ICRA.2012.6225177.
[7] S. M. LaValle, "Motion Planning," in IEEE Robotics & Automation Magazine, vol. 18, no. 1, pp. 79-89, March 2011, doi: 10.1109/MRA.2011.940276.
[8] T. Chin. "Robotic Path Planning: PRM & PRM*." Medium.com. https://theclassytim.medium.com/robotic-path-planning-prm-prm-b4c64b1f5acb (accessed Nov. 15, 2021).
[9] T. Chin. "Robotic Path Planning: RRT and RRT*." Medium.com. https://theclassytim.medium.com/robotic-path-planning-rrt-and-rrt-212319121378 (accessed Nov. 10, 2021).
[10] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," in Int. J. Robotics Research, vol. 20, pp. 378-400, doi:10.1177/02783640122067453.
[11] F. Dellaert. "PRM, RRT, and RRT*." cc.gatech.edu. https://www.cc.gatech.edu/~dellaert/11S-AI/Topics/Entries/2011/2/21_PRM,_RRT,_and_RRT__files/06-RRT.pdf (accessed Nov. 8, 2021).
[12] L. E. Kavraki, P. Svestka, J. -. Latombe and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in IEEE Transactions on Robotics and Automation, vol. 12, no. 4, pp. 566-580, Aug. 1996, doi: 10.1109/70.508439.
[13] P. Abbeel. "Sampling-Based Motion Planning." people.eecs.berkely.edu. https://people.eecs.berkeley.edu/~pabbeel/cs287-fa12/slides/SamplingBasedMotionPlanning.pdf (accessed Nov. 7, 2021).
[14] "PRM Path Planning Algorithm." ProgrammerSought.com. https://www.programmersought.com/article/21893996265/ (accessed Nov. 5, 2021).
[15] A. Sakai, D. Ingram, J. Dinius, K. Chawla, A. Raffin and A. Paques, "PythonRobotics: a Python code collection of robotics algorithms," 2018. [Online]. Available: arXiv:1808.10703.
[16] P. Abbeel, "Sampling-Based Motion Planning," presented to CS287 Advanced Robotics, University of California Berkely, Berkley, CA, USA, Sep. 27, 2012. [PowerPoint slides]. Available: https://people.eecs.berkeley.edu/~pabbeel/cs287-fa12/slides/SamplingBasedMotionPlanning.pdf, Accessed on: Nov. 1, 2021.
[17] M. I. Chowdhury and D. G. Schwartz, "UUV On-Board Path Replanning using PRM-A," Global Oceans 2020: Singapore – U.S. Gulf Coast, 2020, pp. 1-8, doi: 10.1109/IEEECONF38699.2020.9389384.
[18] J. L. Bentley, "Multidimensional Binary Search Tress used for Associative Searching," Commun. ACM, vol. 18, no. 9, pp. 509-517, Sep. 1975, doi: 10.1145/361002.361007.
[19] R. L. Graham and F. F. Yao, "Finding the Convex Hull of a Simple Polygon," J. Algorithms, vol. 1, no. 2, pp. 324-331, Dec. 1982, doi: 10.1016/0167-8655(82)90016-2.
[20] "What's an Efficient Way to Find if a Point Lies in the Convex Hull of a Point Cloud?" stackoverflow.com. https://stackoverflow.com/questions/16750618/whats-an-efficient-way-to-find-if-a-point-lies-in-the-convex-hull-of-a-point-cl (accessed Dec. 4, 2021).
[21] M. J. Fryer, An Introduction to Linear Programming and Matrix Game Theory, 1st ed. New York, NY, USA: Wiley, 1978.
[22] R. Sekhon and R. Bloom, Applied Finite Mathematics, 3rd ed. Cupertino, CA, USA: De Anza College, 2016.
[23] C. Griffin, Linear Programming: Penn State Math 484 Lecture Notes. (2009-2014). Math 484, State College: Pennsylvania State.
[24] J. C. Nash, "The (Dantzig) simplex method for linear programming," in Computing in Science & Engineering, vol. 2, no. 1, pp. 29-31, Jan.-Feb. 2000, doi: 10.1109/5992.814654.

[25] F. Fischer, “The Two-Phase Simplex Method,” presented to Optimization, University of Cambridge, Cambridge, UK, May 15, 2015. Available: http://www.maths.qmul.ac.uk/~ffischer/teaching/opt/notes/notes8.pdf, Accessed on: Dec. 5, 2021.
[26] H. Hui, “On the Feasibility of a Generalized Linear Program,” Systems Optimization Lab, Stanford Univ., Stanford, CA, USA, Tech. Rep. AD-A207402, May. 1, 1989.
[27] P. Barrett, J. Hunter, T. Todd and J. C. Hsu, “Matplotlib – A Portable Python Plotting Package,” in Astronomical Data Analysis Software and Systems XIV, Pasadena, CA, USA, Oct. 24-27, 2004, pp. 91-95.
[28] N. Ari and M. Ustazhanov, "Matplotlib in Python," 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), 2014, pp. 1-6, doi: 10.1109/ICECCO.2014.6997585.
[29] J. D. Hunter, "Matplotlib: A 2D Graphics Environment," in Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, May-June 2007, doi: 10.1109/MCSE.2007.55.
[30] C. R. Harris et al., “Array Programming with NumPy,” Nature, vol. 585, pp. 357-362, Sep. 2020, doi: 10.1038/s41586-020-2649-2.
[31] P. Virtanen et al., “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” vol. 17, pp. 261-272, Feb. 2020, doi:10.1038/s41592-019-0686-2.
[32] E. W. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” Numerische Mathematik, vol. 1, pp. 269-271, Dec. 1959, doi:10.1007/BF01386390.
[33] G. Van Noord, G. Bouma, R. Koeling, and M. J. Nederhof, “Robust Grammatical Analysis for Spoken Dialogue Systems,” Natural Language Engineering, vol. 5, pp. 45-93, Mar. 1999, doi:10.1017/S1351324999002156.
[34] W. S. Xi, “The Improved Dijkstra's Shortest Path Algorithm and Its Application,” Procedia Engineering, vol. 29, pp. 1186-1190, 2012, doi:10.1016/j.proeng.2012.01.110.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY