Trajectory Optimization and Obstacle Avoidance of Autonomous Vehicles Using Robust and Efficient Rapidly Exploring Random Tree (RE-RRT*)

Author

Kaynat Gul

Registration Number: 00000330764

Supervised By

Dr Naeem Ul Islam

DEPARTMENT OF ELECTRICAL ENGINEERING

COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING (E&ME),

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY (NUST),

ISLAMABAD

May 2023

Trajectory Optimization and Obstacle Avoidance of Autonomous Vehicles Using Robust and Efficient Rapidly Exploring Random Tree (RE-RRT*)

Author

Kaynat Gul

Registration Number: 00000330764

A thesis report submitted in partial fulfilment of the requirements
for the degree of MS in Electrical Engineering

Thesis Supervisor:

Dr Naeem Ul Islam

Supervisor's Signature:

_____

DEPARTMENT OF ELECTRICAL ENGINEERING

COLLEGE OF ELECTRICAL AND MECHANICAL ENGINEERING (E&ME),

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY (NUST),

ISLAMABAD

May 2023

DECLARATION

I declare that this research work titled, "*Trajectory Optimization and Obstacle Avoidance of Autonomous Vehicles Using Robust and Efficient Rapidly Exploring Random Tree (RE-RRT*)*" is my work and it has not been submitted for evaluation anywhere else. All the material from other sources used in this report has been appropriately cited.

_____

Student's Signature:

Kaynat Gul

2020-NUST-MS-EE-0000330764

LANGUAGE CORRECTNESS CERTIFICATE

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical, and spelling mistakes. Thesis is also according to the format given by the university.

_____

Signature of Student

Kaynat Gul

Registration Number

00000330764

_____

Signature of Supervisor

Dr. Naeem ul Islam

PLAGIARISM CERTIFICATE (TURNITIN REPORT)

This thesis report has been checked for Plagiarism. Attached is the Turnitin report checked by Supervisor.

_____

Signature of Student

Kaynat Gul

Registration Number

00000330764

_____

Signature of Supervisor

Dr. Naeem ul Islam

COPYRIGHT STATEMENT

ACKNOWLEDGEMENT

I owe Allah Almighty a huge debt of gratitude for his abundant blessings throughout this project. Indeed, this would not have been possible without his substantial guidance through every step, and for putting me across people who could drive me through this work in a superlative manner.

I am incredibly appreciative of my dear parents' love, prayers, support, and sacrifices in raising me and preparing me for the future. I also thank my siblings who encouraged me and prayed for me throughout the time of my research.

I would like to acknowledge my supervisor **Dr Naeem ul Islam** for his constant motivation, patience, enthusiasm, and immense knowledge. He has guided me and encouraged me to carry on and has contributed to this thesis with a major impact. For my MS study, I could not have asked for a greater mentor and advisor. He provided me with direction as I conducted my research and wrote this thesis.

My head of department and co-supervisor, **Dr. Fahad Mumtaz Malik,** deserve my gratitude for his incredible cooperation and for providing help at every phase of this thesis. He provided me with direction as I conducted my research and wrote this thesis.

I also want to express my gratitude to **Dr. Shahzad Amin Sheikh** and **Asst. Prof Kamran Aziz Bhatti** for serving on the committee that guided and evaluated my thesis. Their suggestions are extremely helpful in improving the work.

Last but not least, I would want to convey my gratitude to everyone who has helped my study in any way.

Thank you for your constant support!

Trajectory Optimization and Obstacle Avoidance of Autonomous Vehicles Using Robust and Efficient Rapidly Exploring Random Tree (RE-RRT*)

Kaynat Gul

Dissertation submitted to the Faculty of the Department of Electrical Engineering,

College of Electrical & Mechanical Engineering,

National University of Sciences & Technology (NUST)

in partial fulfilment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Dr. Naeem ul Islam, Supervisor

Dr. Fahad Mumtaz Malik, Co-Supervisor

Dr. Shehzad Amin Sheikh

Asst. Prof. Kamran Aziz Bhatti

May, 2023

COLLEGE OF ELECTRICAL & ELECTRICAL ENGINEERING,

NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY (NUST), ISLAMABAD

Dedicated to my loving parents, whose constant encouragement and guidance enabled me to accomplish this wonderful goal.

Abstract

One of the key issues in robotics is the motion planning problem. This study provides a local trajectory planning and obstacle avoidance strategy based on the Rapidly exploring Random Tree algorithm for autonomous vehicles to handle the issue of travelling in complicated surroundings. Rapidly Exploring Random Trees (RRT), a sampling-based pathfinding algorithm, has been extensively employed in motion planning issues. The RRT algorithm still has several limitations, including a sluggish convergence rate, significant search time volatility, a vast dense sample space, and unsmooth search routes. In this study, we suggest RE-RRT*(Robust and Efficient RRT*), a new RRT-based pathfinding algorithm. which extends Rapidly exploring Random Tree (RRT*), to identify a speedy path that is close to optimal. The Choose Parent and Rewire processes are used by RE-RRT* to continuously improve the path in succeeding cycles. The sample space is constrained during each stage of the random tree's growth., so reducing the number of pointless searches. The RE-RRT* algorithm can converge to a shorter path with a smaller number of iterations and be smoother, according to simulation and experimental results under diverse obstacle settings. The suggested method can increase search effectiveness, speed up convergence, and decrease processing time. Due to these advantages, our suggested RE-RRT* beats RRT* in experiments in terms of computational time, sampling space, speed, and stability

Keywords: Trajectory Planning**,** Obstacle Avoidance, RRT, RRT*, RE-RRT*.

Table of Contents

List of Figures

List of Tables

Chapter # 1

Introduction

Chapter 1
Introduction

This section provides a thorough introduction to the key ideas underlying our research, the current problem, and an overview of our solution. It is organized into five sub-sections. **Section 1.1** describes the background study; **Section 1.2** provides the goals and objectives of the thesis. **Section 1.3** discusses the motivation of thesis work. **Section 1.4** gives the problem statement of the research, **Section 1.5** discusses the proposed methodology, and thesis organization is presented in **Section 1.7**.

1.1 Background Study

The purpose of this section is to provide an overview of the research's background investigation. The main issue that needs to be resolved for AVs to function properly in complex environments is motion planning. Since the steam period, when they first appeared, cars have slowly but surely made their way into people's lives, where they now serve a crucial role in transportation and are an essential component of everyday life and economic activity. People benefit from this in terms of convenience and speed, as well as significant concerns relating to road safety. Modern autonomous technology, which can somewhat increase traffic safety, is developed from traditional automobile driving technology. As a result, driverless technology is starting to get the attention of experts and academia and has emerged as the primary area of study in the car sector. In order to comprehend the path tracking algorithm employed by autonomous vehicles, research based on the traditional RRT method is conducted in this work. Then, it suggests a Robust and Efficient RRT* algorithm that can successfully actualize the evolution of autonomous vehicle technology and act as a starting point for its further development.

One of the most widely used technologies today, autonomous cars have excellent potential both domestically and internationally. Research is also going around autonomous vehicle path tracking control technologies. Autos that can function without a human driver are known as autonomous vehicles. A technological advancement that is currently one of the biggest trends that aspires to change transportation. The adoption of autonomous automobiles may accelerate during the next ten

years, despite the fact that experiments with various countries and businesses began a few years ago. In the automotive industry, autonomous driving, connectivity, and electric vehicles are all directly related to one another. As a result, connection must be sufficiently strong to ensure the appropriate operation of autonomous vehicles, given that data transfer is crucial to their functioning. Although the development of autonomous vehicles has gained popularity over the past 20 years, it is true that it first began in the 1990s. In 1925, While the automobile was remote-controlled, Francis Houdina, an electrical engineer from New York, was the first to put the idea of an autonomous vehicle into practice. The prototype was displayed to the public in Manhattan and traveled roughly 19 kms between Broadway and Fifth Avenue before colliding with another vehicle and deviating from its intended course.

Houdini's automobile, Chandler, was however built between 1926 and 1930. The German Ernst Dickmanns, who is credited as the creator of the contemporary autonomous car, later converted a Mercedes-Benz van into one. A built-in computer was in charge of operating this vehicle. In 1987, the car was capable of 63 kilometre per hour via streets with no traffic.

Six stages of autonomous driving are currently recognized. 0 to 2 autonomous vehicles come equipped with driver assistance systems. The last three, numbered from 3 to 5, feature actual automation components, as described by the Society of Automotive Engineers (SAE).

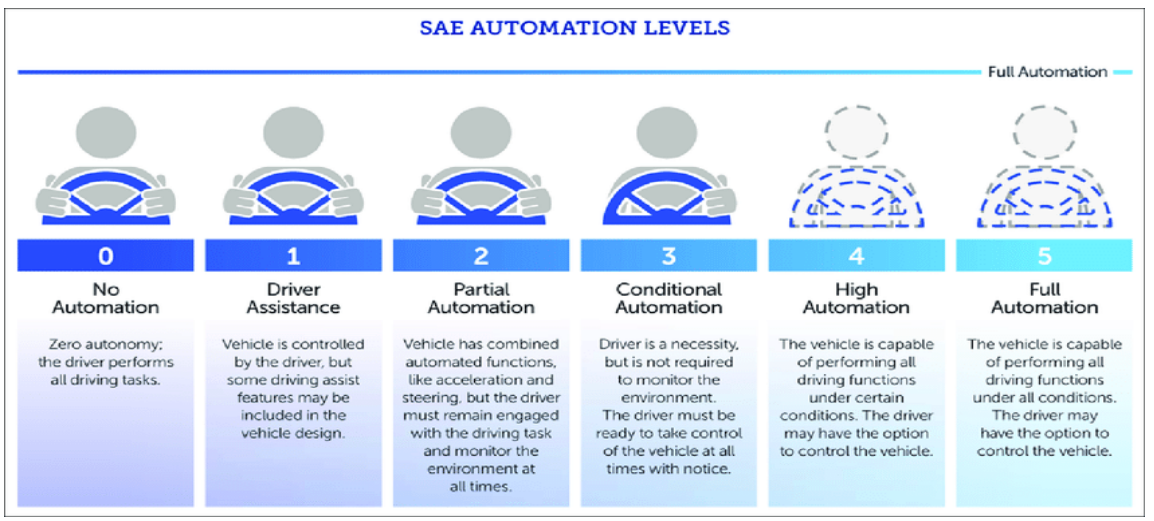


Figure1.1: introduces SAE J3016 Surface Vehicle Recommended

Practice's levels of automation [1]

1.1.1 Existing Algorithms of Path Tracking

There are numerous approaches to path optimization for autonomous vehicles. Model predictive control (MPC) is frequently used to steer an autonomous vehicle along a predetermined course. Commercial driving safety aid technologies, such adaptive cruise control and lane assist systems, are now frequently seen in high-end, mass-produced automobiles. These devices can reduce accidents brought on by driver error, drowsiness, or distraction. Yet, they are unable to engage in complicated driving behaviors like navigating around other vehicles, avoiding collisions, or reacting deftly to sudden dynamic impediments. Moreover, these systems still require ongoing human oversight. The DARPA Challenge [2], [3], in semi-structured environments [4], and in parking lots [5] have all seen the effective implementation of graph-searching-based algorithms like the Dijkstra and A algorithms family. Path planning has also been successfully accomplished using sampling-based techniques, one of which is the rapidly exploring random tree (RRT), which, along with its modifications, has become one of the most widely used algorithms in recent years [6], [7]. It has been demonstrated that curve interpolation is an effective method for creating reference paths. Many scholars have used polynomials [8], [9], Bezier curves [10], and B-splines [11]. To increase the autonomy of the vehicle and decrease the number of turns on the planned course, an adaptive ant colony algorithm (ACO) path planning approach is applied [12].

1.1.2 Importance of Automation in Vehicles

Let's examine some of the benefits driverless cars can offer society as the globe moves consciously toward a transportation system powered by them:

i. **Minimize Accidents**

   According to the USDOT website, "the promise of autonomous vehicle technology to minimize deaths and injuries on our roadways drives us to action" since human factors is to blame for 94% of fatal vehicle events.

ii. **Minimize Traffic Congestion**

   Because traffic accidents can account for up to 25% of congestion, fewer accidents would mean less congestion.

iii. Minimizes CO2 Emissions

Very likely, the lessening of traffic will also lead to a decrease in CO2 emissions.

iv. Increased Lane Capacity

AVs may enhance total vehicle travel in addition to encouraging higher driving throughput rates on the current highways. First off, because to their ability to constantly monitor the surrounding traffic and react with perfectly timed braking and acceleration adjustments, AVs should be capable of driving safely at faster speeds while maintaining a smaller headway (distance) between each vehicle. According to study, platooning AVs might increase lane capacity by 500%. (number of vehicles per lane per hour).

v. Reduced need of Fuel

Compared to a human driver, AV technologies can drive and descend more smoothly. It can also improve fuel efficiency by 4–10%. More advancements may result from expanding the capacity of the highways and reducing the distance between vehicles.

vi. Transportation Accessibility

Many elderly individuals and others with disabilities are currently unable to drive, with even car adaptations that make it less dangerous for others to drive. Many more people might be able to access the open road and independence thanks to autonomous vehicles. Self-driving cars may make tasks like travelling to work, seeing the doctors, and visiting family across town easier for elderly and people who have disabilities.

vii. Decreased Transportation Costs and Travel Time

AVs may reclaim up to 80 billion hours wasted to commutes and traffic, decrease the use of fuel by up to 40%, and reduce travel time by as much as to 40%. Another area where money could be saved is by decreasing the number of transporters and law enforcement personnel. [13].

1.2 Goals and Objectives

The primary goal of this study is to reduce human effort by implementing advanced technology. This study aims to offer a method for automatically track the optimum path and to avoid obstacles on the way between start position to goal position. Another main objective of this research is to improve search efficiency and the convergence speed and reduce processing time and the number of turns in the planned path. To locate a continuous line that avoids interacting with the nearby obstructions and travels from a given original state to a defined goal state. By limiting the sampling space thereby avoiding a lot of unnecessary searches

1.3 Motivation

The suggested strategy will assist in avoiding roadblocks while taking into consideration the rising number of accidents in daily life. The use of random sampling lengthens the algorithm's execution time and hinders convergence, and the algorithm's use of nearest neighbour selection frequently results in complex scenario planning. Our suggested Robust and Efficient (RE-RRT*) therefore, surpasses RRT* in all the aforementioned areas of weakness.

1.4 Problem Statement

One of the key issues in robotics is the trajectory optimization and path planning problem. In past, a lot of work has been done to cater this problem. Rapidly Exploring Random Trees (RRT), a sampling-based pathfinding algorithm, has been extensively employed in motion planning issues. The RRT algorithm still has several limitations, including a sluggish convergence rate, significant search time volatility, a vast dense sample space, and unsmooth search routes. Every time the random tree grows, there are numerous pointless searches conducted, which prolongs the convergence time.

1.5 Thesis Organization:

The report is organized as follows:

- **Chapter 1** gives the introduction about the proposed topic, aims, objectives, and motivation.
- **Chapter 2** presents the literature review of trajectory optimization and path planning of an autonomous vehicle using different techniques.
- **Chapter 3** discusses the proposed methodology of Robust and Efficient Rapidly Exploring Random Tree Star (RE-RRT*).
- **Chapter 4** gives the implementation details.
- **Chapter 5** discussed the experimentation including the setup used for implementation, results obtained, and their discussion.
- **Chapter 6** concludes the topic by suggesting some future work that is not under the scope of this research but can be implemented in the future.
- **Chapter 7** concludes the topic by suggesting some future work that is not under the scope of this research but can be implemented in the future.

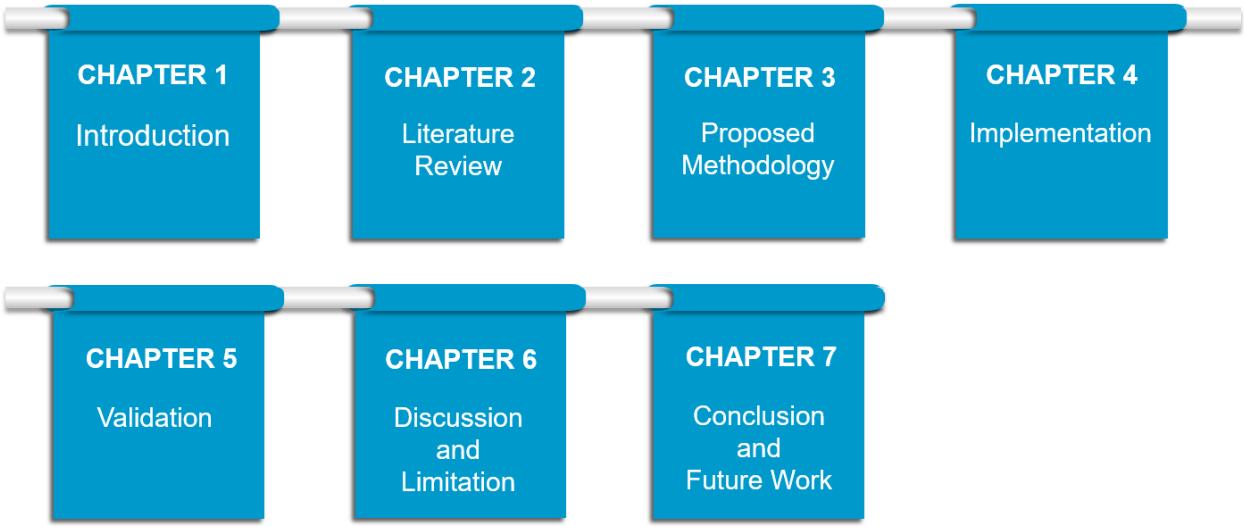


Figure1.2: Thesis Outline

Chapter # 2

Literature Review

Chapter 2

Literature Review

Finding a viable route to the destination in the observed environment with a certain update frequency is the job of a global path planner. For a safety assurance, a long enough ahead prediction horizon of the global path should be attained. This research project focuses on the motion planning and control of autonomous road vehicles in unexpected critical conditions, or circumstances when an accident is likely to happen.

The phrase "critical situation" refers to a circumstance that emerges abruptly due to internal or external reasons and in which there is a significantly elevated risk of an impending accident. Because autonomous driving moves at a faster pace than mobile robots, a lot of attention should be paid to both safety and comfort when driving. In order to handle the changing traffic situations in diverse on-road scenarios with computing efficiency, the motion planning issue for autonomous driving is often divided into a global reference route planning level and a local motion planning level [14]. These days, transportation, and industry all frequently use portable robots addressed by independent vehicles. The independent vehicle includes a framework for perception, direction, and control, and the study of the robot organizing framework has always been a major concern. One of the main ideas of contention is how to produce a proper way to arrive at an objective situation with next to no impact for autonomous vehicles [2][3][4]. Geometric search techniques and graph search methods may be discovered in earlier research [5]; the computation of a geometry approach is extremely straightforward, useful, and reasonable such as a curve of spline, and it is adaptable to the basic surroundings. Even if geometric approaches offer certain benefits, they cannot make up for their drawbacks. These path-planning techniques need to have their intelligence and adaptability improved in complicated contexts [6].

The ability of optimization techniques that are frequently employed in mobile robot route planning has been demonstrated while seeking paths based on graphs. Forward searching in continuous coordinates must be carried out using RRT [7]. Although this technique can search quickly, it cannot be widely applied in confined spaces when the situation is complicated. Based on specific decision criteria, the A* algorithm may be used to discover the shortest path that is obstacle-free [8], however, the resultant path is always made up of difficult-to-follow straight lines. It took a lot

of work to make the A* method perform more effectively. For instance, the A* method may be boosted up by an order of magnitude using the Jump Point Search algorithm [9]. The Jump Point Search technique was then expanded by Liu et al. [10] from a two-dimensional to a three-dimensional environment. By including dynamic limitations, the Hybrid A* [11] can create slick routes to placate the robots. The graph-based approach can identify an optimum path if a viable path already exists; if not, it will yield failure. This demonstrates the completeness and resolution excellence of the graph-based approach. But since the search space created by the graph-based deconvolution of the state space is too huge, graph-based techniques are unable to successfully address problems of a large scale (such as industrial robotic arms). The D* algorithm was designed to guide autonomous vehicles in a two-dimensional space. Its main advantage is that it can choose the best course while attempting to travel in a difficult environment [11]. Vehicle kinematics typically place limitations on this method.

The capacity of various research units to plan paths has been shown during the 2007 Defense Advanced Research Projects Agency (DARPA) urban challenge. The proposed plan and theoretical model lay the groundwork for additional investigation. The "Talos" vehicle was created by MIT, and a closed-loop RRT-based path planning technique is employed [12]. A well hybrid A* searching technique was created by Dolgov et al. [13]. Their method uses the vehicle's 3-dimensional kinematics state space as well as local planning using nonlinear optimization yield a local optimum. These path-planning planning techniques used on urban roads have not demonstrated their viability and efficacy in complicated challenging environments. In past years, Suresh et al. [14] have employed FSVM to guarantee an accident-free route while avoiding several dynamic impediments. The outcomes indicate that this approach is successful. The outcomes of simulations using this method, which generates fuzzy rules from plain evaluation data, are shown. Chu et al. [15] have implemented an algorithm for real-time route planning. The ideal function deals with choosing the ultimate safe and straightforward path after creating a few pathways based on predefined checkpoints.

The steering component of a method developed by Makarem and Gillet [28] is suitable for autonomous cars, however it ignores the effect of impediments. The "tentacle method" is yet another way proposed by Chebly et al. [29]. By treating the cars as the origin, the method generates a number of simulated branches that display probable pathways for the vehicle. The evaluation function is then used to select the best path.

In complex dynamic situations, Moreau et al[30] .'s method to curve design is superior. With an automated vehicle, all sensors needed for collision avoidance are taken into account. The planning problem is changed into an optimum problem by accounting for equality restrictions. Next, it is covered utilising gradient-based and Lagrangian methods. In [20], Tazir et al. develop real-time planning using two methods. The robot uses evolutionary algorithms and Dijkstra's algorithm to eliminate static obstructions. Travel in a local, dynamic area is made possible by the wait/accelerate principle. Although this approach is test-efficient, robot kinematic restrictions are not considered. An inventive integrated local trajectory planning and tracking control (ILTPTC) framework has been proposed by researchers to allow automated vehicles to travel along a basic track while also avoiding detection and meeting the vehicle kinematic limitations [16]. In this architecture, an MPC-based planning technique is employed, which can fulfil the need for vehicle kinematics but falls short of actual requirements. One significant kind of planning methodology is the random samples structured methodology. As opposed to separating the state space, sampling-based planning creates a graph or tree by choosing random locations within the state space. Sample-based organizing methods outperform graph-based planning algorithms in large-scale applications. The random samples planning system is probabilistically full, thus as the amount of samples grows exponentially, the likelihood of discovering an appropriate path approaches 1. Sampling-based planners employ RRT [22] and Probabilistic Roadmaps (PRM) [23] as two important techniques. A viable path is produced using the multi-query motion planning method PRM after a feasible graph reflecting spatial connection is obtained by random sampling in the state space. The PRM may be used to search various pathways after creating the graph. However, it takes a lot of effort to map out the entire area for a single search. RRT is quicker than PRM because, being a one-request route planning technique, by building a tree with its root at the starting point, it simply searches the state space. Three steps make up the RRT method: First, it randomly generates a state in state space; next, it chooses the nearest random tree nodes; and last, it grows a random state from the nearest neighbour selection point. The search is accomplished when the tree reaches the desired location, and RRT will then retrace its steps to a workable route.

RRT can identify an starting route in a high-dimensional area quickly, although it has several flaws. For instance, the variance of its runtime is so enormous due to the random sampling that it can take some time to find a suitable route. In circumstances with narrow channels, the RRT does not operate well [17]. Furthermore, since the path is generated arbitrarily, it's likely that the path found using RRT is not the best one [25]. Rapidly Exploring Random Tree Star(RRT), or RRT* was

considered a substantial improvement on RRT [26]. RRT* continues to optimize the original path after discovering one by continually sampling [18]. To determine the optimum path, RRT* adds the neighbour searching and rewiring tree processes. when there are infinitely many samples, it is demonstrable that the path generated by RRT* is the best one. RRT* consequently requires a significant amount of memory and time used to determine the optimum path [28][12]. Like this, RRT* is impacted by the problem of considerable search time volatility.

A lot of work was put into raising the caliber of the pathways that RRT and RRT* were able to find. For instance, equivalent Kino dynamic RRT* can achieve an ideal route that fulfils static constraints by expanding RRT* to Kino dynamic systems. Any location can be used as the starting point for a rapid path re-planning using Anytime-RRT*. As an alternative, increasing the search rate and reducing search time variance are important areas of attention in RRT algorithm-related research. For example, RRT Connect [19] builds two trees with roots from starting state to the destination state, and then causes the two trees to move in the same direction. In [30], a 2D Gaussian mix model is used to quickly find a good starting solution. Batch Informed Trees [20] swiftly locates a viable path by restricting the state space to a gradually growing subset. However, these techniques mainly worked effectively in specific settings. RRT can speed up the search process when combined with a variety of different path search strategies. To accelerate the convergence rate, the artificial potential field (APF) technique is incorporated into RRT* in [32], although, in complicated situations, the planning time may rise significantly.

Applying MATLAB to recreate the car shown in ADAMS [33], the effectiveness of the route regulator is confirmed. Zhou W suggested new infrastructure that depends on an updated (RRT) methodology and a continuous - time route planning and track control in order to evaluate the relationship among actual planning and monitoring control of smart cars. Depending on the LTV-MPC approach, the fundamental RRT algorithm is changed to provide the necessary intelligent transportation Path stability control, ensuring that the desired course coincides with the demands of the car's kinematics limitations and approaching the ideal result. Goal direction, node trimming, contour plots, and choosing the best path are a few of these adjustments. The effect of several factors, such as the vehicle's speed, the scheduling step, and the cycles, is then investigated [36]. Using a straightforward linear and time-invariant monorail framework computed using a uniform nominal longitudinal speed, Mata S proposes a method for planning the path of a moving object.

A tube-based robust (MPC) method is suggested to account for differences in system dynamics between the average car and this steady nominally simulation.

The A*-RRT* [21] technique considerably speeds up convergence by using the route created by the A* method to direct the RRT* planner's sampling process. However, when it comes to complex issues, A* takes a considerable amount of effort to identify a starting point. Although LM-RRT [22] uses reinforcement learning techniques to direct tree development, learning-based approaches might not function effectively in the novel environment.

It has been demonstrated that curve interpolation is an effective method for creating reference paths. Many academics have employed polynomials [23], [24], Bezier curves [25], and B-splines [26]. To increase the autonomy of the vehicle and decrease the number of turns on the planned course, an adaptive ant colony algorithm (ACO) path planning approach is applied [27].

In recent years, motion planning research has made extensive use of learning-based methodologies. Deep learning is used by Neural RRT* [30] to discover a distribution probability for sample selection. In order to bias tree growth in favor of the targeted area, RL-RRT [28] investigates deep reinforcement learning strategy as a local planner and employs a distance function that trains through deep learning. The cost function of the RRT* is learned using an approach that combines inverse reinforcement learning with RRT* in [29]. The virtual artificial potential field is used by the DL-P-RRT* method to understand the function of the artificial potential field before applying it to the RRT* algorithm. Learning-based approaches work well in specific situations, but they may struggle to generalize in unfamiliar situations. Most of the work has been done which is more like our proposed algorithm is on path planning and trajectory optimization of an autonomous vehicle using RRT, RRT*, Improved RRT*, Fast RRT* approaches.

2.1 RRT* over Basic RRT

Finding the relatively short, smoothest, and concussion route between the beginning and target locations is referred to as optimal path planning. Numerous robotic solutions, including autonomous vehicles, personnel security, agricultural robots, and exoplanet and space discovery operations, all depend on this task. RRT continues to experience multiple issues, but it can quickly discover an beginning route in a fast environment. For instance, the variance of its runtime is so enormous due to the random sampling that it can take some time to find a suitable route. In circumstances with

narrow channels, the RRT does not operate well [33]. Furthermore, since the path is generated arbitrarily, it's likely that the path found using RRT is not the best one [25]. Rapidly Exploring Random Tree Star, or RRT* was considered a substantial improvement on RRT [26]. RRT* continues to optimize the original path after discovering one by continually sampling [34]. To determine the optimum path, RRT* adds the neighbour searching and rewiring tree processes. when there are infinitely many samples, it is demonstrable that the path generated by RRT* is the best one. RRT* consequently requires a significant amount of memory and time used to determine the optimum path [28][12]. Like this, RRT* is impacted by the problem of considerable search time volatility. A well-known random samples planning system is quickly investigating Random Tree Star (RRT*). Due to its assistance for higher dimensional space complicated situations, it has experienced tremendous growth. Using RRT*-based techniques, a sizable research collection has addressed the issue of optimal route selection for mobile robots [46].

2.1.1 Basic Structure of RRT and RRT*

The RRT is a technique built on single query search that finds a viable path very rapidly. RRT creates a tree during the initiation step x_{init}, with the starting state acting as the node. RRT chooses the closest vertex x_{near}, after randomly sampling a state x_{free}, in state space for each iteration. The steering function will then create the RRT algorithm which generate x_{new}, as seen in Figure 2.1. If the edge c has no obstacles, then the set of nodes will be expanded by x_{new} and the set of edges will be expanded by $\{x_{near}, x_{new}\}$ The search is completed if x_{new} found at the desired location x_{goal}.

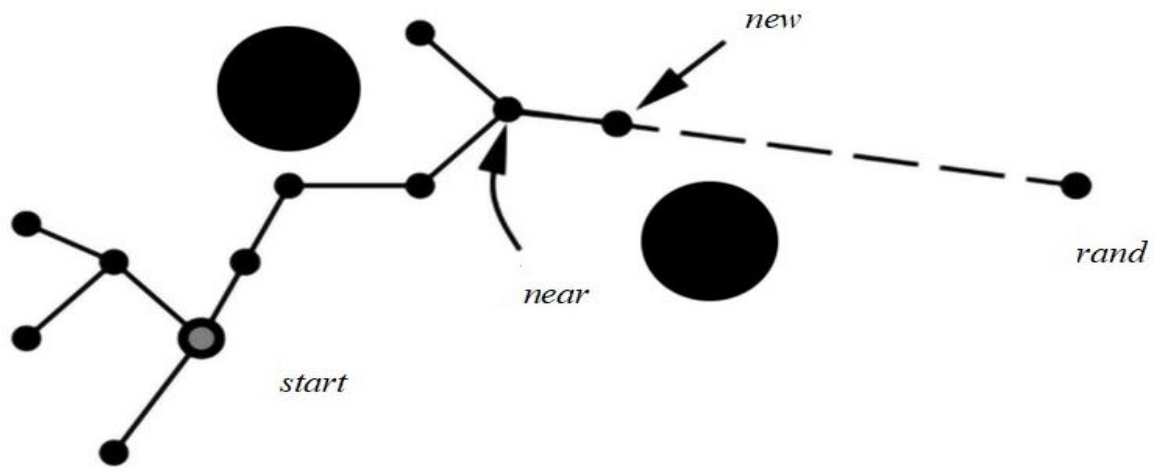


Figure 2. 1:Schematic result of RRT algorithm

The route that RRT chose might not be the best one. RRT* solves this issue by adding a rewire step. The best parent node for x_{new} will be found for nodes with a distance smaller than r surrounding it if the edge $\{x_{near}, x_{new}\}$ is free of obstacles. Additionally, RRT* considers x_{new} as a substitute parent node for existing nearby nodes in addition to adding it to the tree. Therefore, RRT* constantly modifies the random tree as the sample periods go closer to infinity until it discovers an ideal path. However, the RRT* takes a long time, making it unsuitable for systems that must immediately identify an optimal path. Figures 2.2 and 2.3 introduce the RRT and RRT* algorithms, accordingly.

Algorithm 1: RRT algorithm.

Input: x_{start} , x_{goal} , step, n

1. Initialize (x_{start})
2. For i= to n do
3. x_{rand} = sample ( )
4. $x_{near} = Near (x_{rand}, G)$
5. $x_{new} = steer (x_{rand}, x_{near} , step\_size)$
6. G. add node (x_{new})
7. G. node edge (X_{new}, x_{near})
8. If $x_{new} = x_{goal}$
   Success ( )

Figure2. 2. RRT algorithm.

Algorithm 2: RRT* algorithm.

1. Input: x_{init} , X_{goal}
2. Output: Tree T = (V, E)
3. $V \leftarrow x_{init}$, $E \leftarrow \varphi$;
4. For i=1….,N do
5. $X_{rand} \leftarrow$ random-sampling (i)
6. $x_{nearest} \leftarrow Nearest. (T, x_{rand})$;
7. $x_{new} \leftarrow steer. (x_{near} , x_{rand})$;
8. If obstacle free (x_{new} , $x_{nearest}$) then,
   $V \leftarrow V \cup \{ x_{new}\}$;
   $X_{near} \leftarrow Near (T, x_{new}, r)$;
   $choose. Parent (X_{near}, x_{nearest}, x_{new}, E)$;
   $Re - wiring (X_{near}, x_{new}, E)$
   end
   end

Figure 2. 3. RRT* algorithm.

2.2 Improved Bi-RRT

The study builds local pathways using the Frenet coordinates system. Wherever along the curve, the mathematical calculation of the Technology makes it possible frame coordinates is possible. The low-level model converts the vehicle location to Frenet coordinates based on the upgraded Bi-RRT route. Depending on the VFH results, the attempts to reach out from updated Bi-RRT route is moved to the ideal location. The polynomial planning method is used to generate a local path in Model can be created coordinates from the vehicles to the revised target position. The method outlined in this paper maintains path safety and smoothness while being insensitive to environmental change. Figure 2.4 displays the improved Bi-schematic RRT's diagram. Also, the graphic illustrates the improved Bi-RRT method. Figure 2.5 depicts the Improved Bi-RRT method.

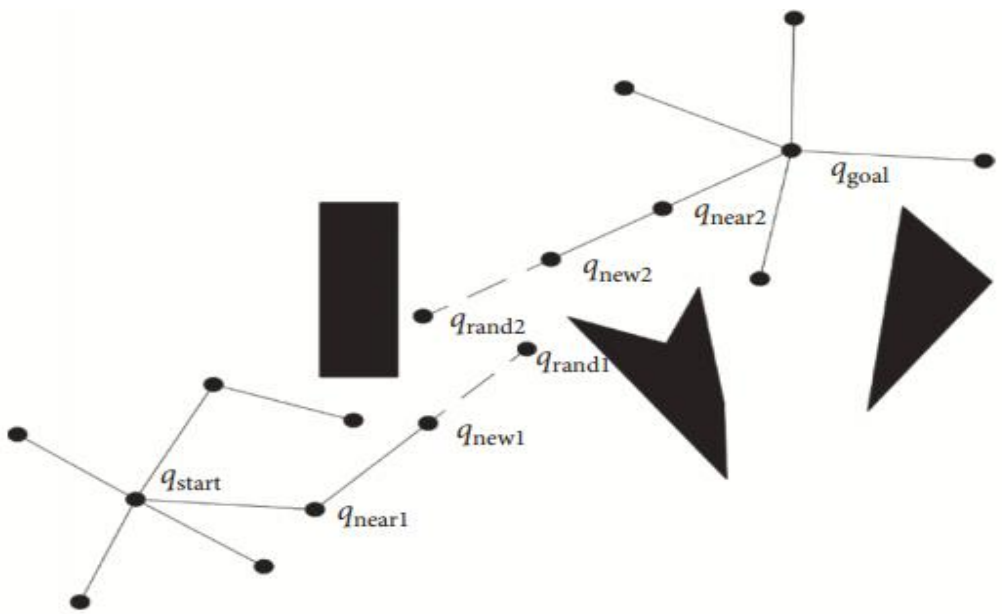


Figure 2. 4: Schematic diagram for Bi-RRT Algorithm

Algorithm 3 Improved Bi-RRT Algorithm

```
V1 ← {q_init}; E1 ← ϕ; G1 ← {V1, E1};
V2← {q_init}; E2 ← ϕ; G2 ← {V2, E2}; i ← 0;
Repeat
p ← random (0, 1);
if p > pprob then;
q_rand ← sample  (i) ; i ← i +1:
q_nearst ← Nearst (G1, q_rear);
q_new ← Extend (q_nearst, q_random);
else q_random = q_goal; i ← (i +1):
end if;
Same like Bi-RRT,
until find a collision-free path from q_start to q_goal;
return S ← PostProcession(G)
function NEARST_AREA (G, q_random)
G_Area ← Area (q′start, q_random);
Dist max ← –∞;
for all qi in G_Area do
Dist_max ← Dist (qi, q_random);
if Dist > Dist_max then
Dist_max ← Dist; q_nearst ← qi;
end if
end for
return q_nearst
end function
```

Figure 2. 5: Improved Bi-RRT algorithm [30]

2.2.1 Bi-RRT Algorithm Search Based on The Steering Constraints.

Quickly investigating the Random Tree planning technique or its growth during the random selection method is a randomized searching in the wide perspective, leading to numerous meaningless searches and wasting precious computer resources. The improved approach used in this study to deal with this problem and speed up the random tree's progression to the target position is as follows: through every stage of random tree growth, the sample space is constrained based on the steered limitation, reducing the number of fruitless searches. Figure 2.6 illustrates that it is not necessary to look at the entire T-tree when determining the closest neighbor for the whole tree.

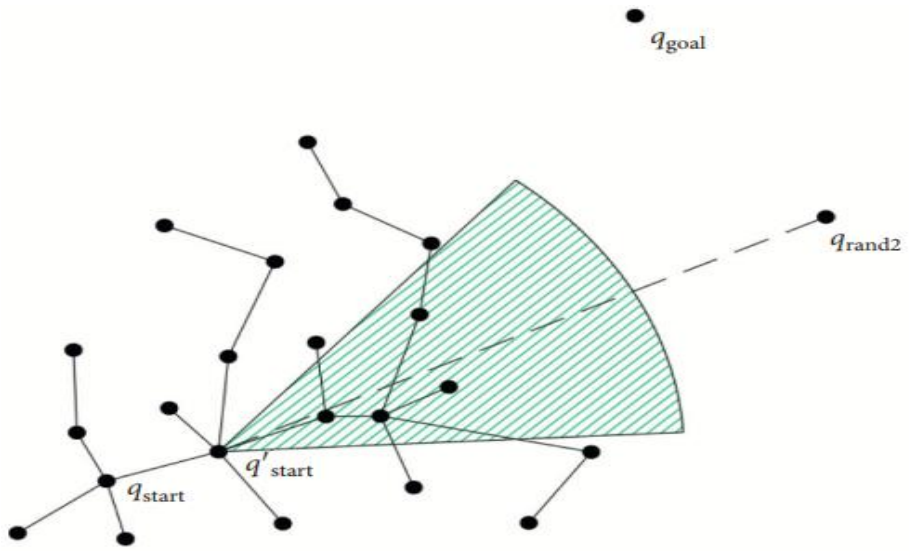


Figure 2. 6: An Improved Bi-RRT search based on the steering constraints

We discussed the relevant backgrounds for this study in this part. Following the introduction of the formal concept of motion planning issues, discussion of related algorithms like RRT, RRT* and Improved Bi-RRT* follows.

Two main issues of path planning using these algorithms are:

- convergence speed
- unnecessary sampling and searches

The following fundamental flaws in the RRT, RRT* and Improved Bi-RRT* method still exist:

(1) The use of random sampling lengthens the algorithm's execution time and hinders convergence.

(2) The application of the nearest node selection technique frequently results in complicated scenario planning.

(3) The planned path cannot be employed in the path planning of autonomous vehicles since it does not take vehicle kinematics restrictions into account.

To overcome these issues which we discussed above, we suggest Robust and Efficient-RRT* (RE-RRT*), an RRT-based motion planning technique. Compared to RRT, Improved-RRT and RRT*, RE-RRT* expedites the search for the optimum route, while limiting the number of randomly produced search nodes and minimizing the convergence rate. RE-RRT* is quicker than RRT* in searching for a path that is close to optimum.

Chapter # 3

Methodology

Chapter 3

Methodology

We introduce our Robust and Efficient Rapidly exploring Random Trees (RE-RRT*) algorithm in this Segment. The theoretical framework for the suggested method is laid out in Segment 3.1. Segments 3.2–3.3 introduce further information.

3.1 Model Structure

The decision-making system receives a grid map from the perception system. worldwide path, change direction, and target tracking in the decision-making system work together to enable the vehicle to manage a variety of situations. The decision-making system then generates a route and sends it to the vehicle's monitoring system. The suggested method's model structure is shown in Figure 3.1.

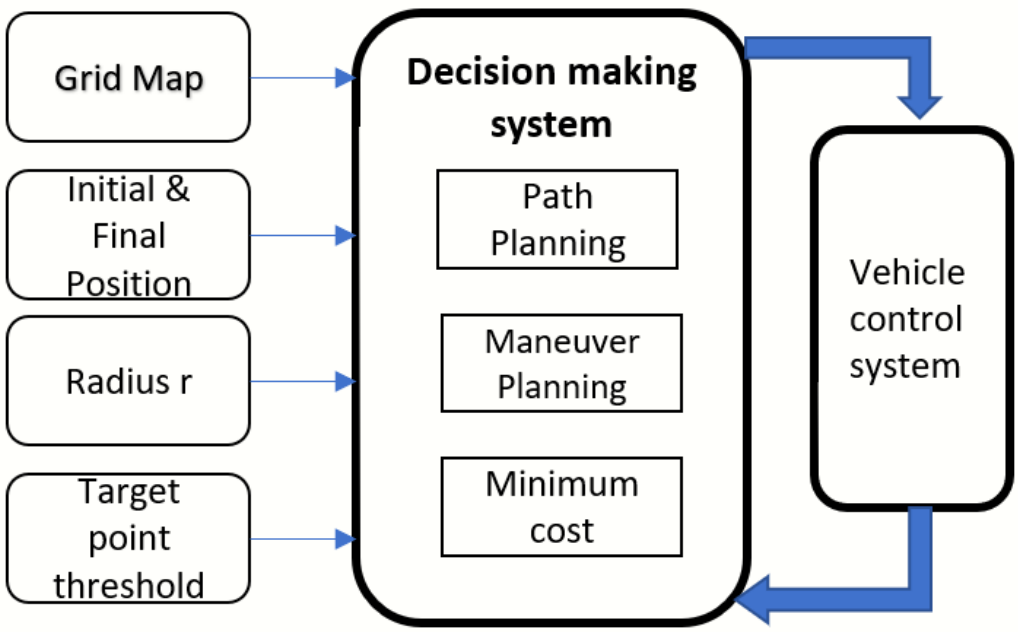


Figure 3.1 : The RRT algorithm's software architecture

This model takes a grid map which is 800m x 800m and information about the initial position and goal position as input. Also, we set the target point threshold, expansion steps, rewire range which

is radius r and maximum iteration The entire grid map is filled with all identified items. A collision-free route is swiftly generated using the steering constraint model.

A simplified vehicle model is shown in figure.3.2. The theta ′θ’ that defines the orientation of the vehicle, and the kinematic equation can be explained as follows.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}\cos\theta\\ \sin\theta\end{bmatrix} v \qquad (1)$$

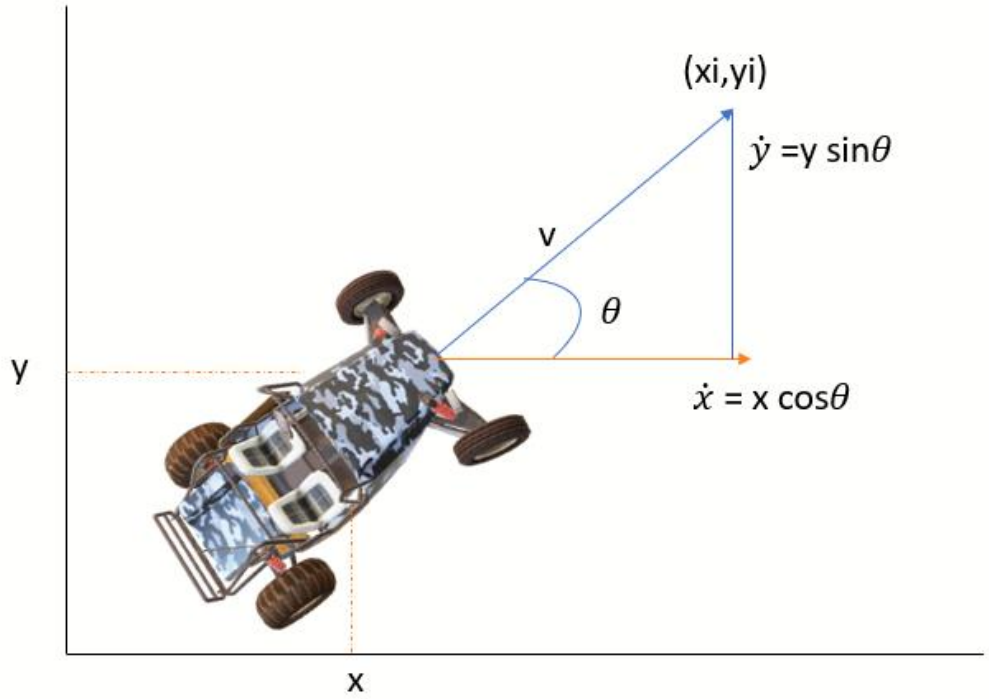


Figure 3. 2: Vehicle Kinematics Model

3.2 Framework of RE-RRT*

The complete framework and methodology of RE-RRT* is explained here. Which includes all steps from the start to the end of this algorithm. Our proposed algorithm is based on RRT* which is the extension of RRT. A small difference between RRT* and RE-RRT* makes the RE-RRT* more efficient and robust by limiting the random sampling and searches on the entire map. RE-RRT* limits unnecessary search and reduces the convergence time and makes the system efficient in this way.

3.2.1. Improvements of RE-RRT*

The RE-RRT* algorithm has two major enhancements.

- Firstly, random sampling is limited, which helps to avoid searching over the entire space. In this way, our proposed algorithm speeds up the convergence rate.
- The second improvement is to minimize random nodes by limiting their generation only around the obstacles. Otherwise, When travelling from the starting position to the destination, our vehicle will be travelling straight.

One of the principles of RRT is to build the random search tree concurrently in the starting position and the destination. The RE-RRT* initiative technique differs from the standard RRT and RRT* algorithms, as shown in algorithms 1 and 2. From the start position, a path is generated in a straight line towards the goal position. After creating random spots across the obstacle, the tree is extended if there will be a collision. After crossing the obstacle, the path leads to the goal position in a straight line until the next obstacle finds out on the vehicle path. Figure 3.3 displays the searching schematic diagram for the RE-RRT* algorithm.

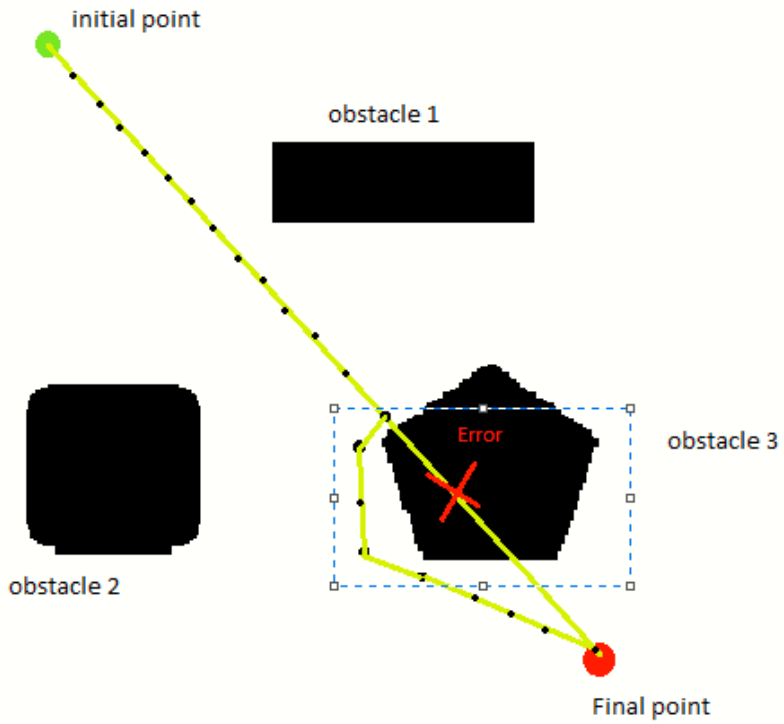


Figure 3. 3: RE-RRT* algorithm's searching schematic diagram

Keeping in mind figure 3.3, now we can visualize the RE-RRT* algorithm. The vehicle's starting point is in green colour and the ending point is in red colour. The huge black boxes are obstacles and we named them as obstacles 1, 2 and 3. Our vehicle started its path from the green signal moving forward towards the goal position in a straight line. When the vehicle reached near to the obstacle e.g., obstacle no 3, it stopped there. Until the RRT* algorithm finds the path where there would be no obstacle and it would be a collision-free path. The area of randomization and node formation is indicated by the blue dotted rectangle in figure 2.1. This is the area from where the random tree generates and ends until it crosses the obstacle area. At that point, the random sampling and node generation process would stop and from that point, let's say x_{new} to goal position straight line nodes will generate and our vehicle follow that path.

3.2.2. The Extension Strategy of RE-RRT*

The extension strategy of RE-RRT* is a bit different from RRT*. Unlike RRT*, RE-RRT* limits unnecessary searches and reduces the sampling space by adding a line equation method. The starting location of vehicle is our starting node. The next nearest node in a straight line will be chosen by the given formula.

$$\Delta_x = (x_{init} - x_{goal})/n \qquad (2)$$

$$\Delta_y = (y_{goal} - y_{init})/n \qquad (3)$$

Here Δ_x and Δ_y are the nodes generating along x and y-axis. The n defines the no of nodes which are generating on the straight path from starting location to the destination of the vehicle route. When the vehicle detects the obstacle on the path, it stops and waits until the algorithm generates the tree across the obstacle and finds the optimal solution to cross the obstacle without hitting the obstacle body. The tree generates the random nodes and the starting nodes of the tree will be the coordinate (x,y) where our vehicle detected the obstacle existence and stops there. Using the Euclidean distance formula, we are looking for the closest nodes. The method for choosing the closest neighbour nodes using the Euclidian equation is shown in figure 3.4.

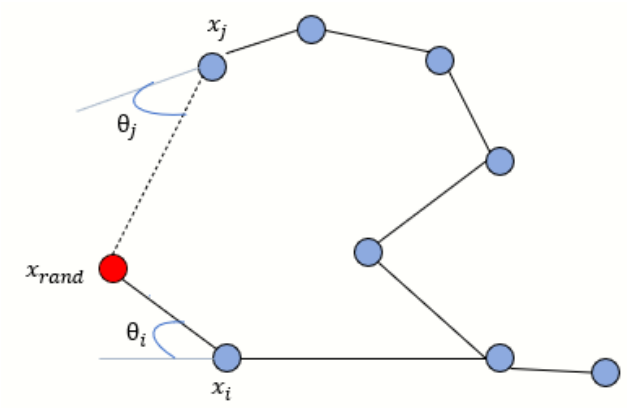


Figure 3. 4: Schematic diagram for selecting the nearest node.

The Euclidian distance between x_i and x_{rand} is closer, $\theta_i < \theta_j$ so, x_i chosen the closest neighbour node of the point which is under consideration. The parent node will be chosen based on whatever node produces the lowest cost to reach the random sample, and they are added appropriately. We calculated the distances between the random sample location and a nearby point using the following Euclidean distance equation:

$$\text{Dist} = \sqrt{(x_{rand} - x_i)^2 + (y_{rand} - y_i)^2} \qquad (4)$$

3.2.3. Vehicle Steering Angle

By expanding the tree to get new nodes, there must be a factor theta, our vehicle steer according to it. θ is basically our vehicle's steering angle.

$$\theta = \text{atan2}((y_{rand} - y_{near}), (x_{rand} - x_{near})) \qquad (5)$$

$$x_{new} = x_{near} + \cos(\theta) * \Delta_x \qquad (6)$$

$$y_{new} = y_{near} + \sin(\theta) * \Delta_y \qquad (7)$$

Equations 6 and 7 are written according to the vehicle's kinematic model shown in figure 3.2.

3.2.4. Pruning Process

The resulting pathways are typically exceedingly convoluted and uneven because of the RRT algorithm's random sampling, especially in situations with numerous complicated obstacles. It is challenging to locate them successfully. The ride comfort of the vehicle will be impacted by too many fold points, which is unsuitable for path planning. Consequently, it is necessary to prune and smooth the RRT and RRT* routes. To get a smooth and turn-free path, this algorithm deleted the useless nodes and updated them with useful nodes and the nodes which give optimal cost. It removes the number of turns in the path to achieve the pruning effect.

Figure 3.5 shows the pruning principle clearly. We have a solid obstacle and the number of random nodes generated around it. The irregular path sampling is shown in black lines. As we can see that the path between nodes n1 to n6, n8 to n12 are safe, so n1 to n6 and n8 to n12 can be directly connected and they are shown in the red line. The comparison impact prior to and following trimming is seen in the next segment.

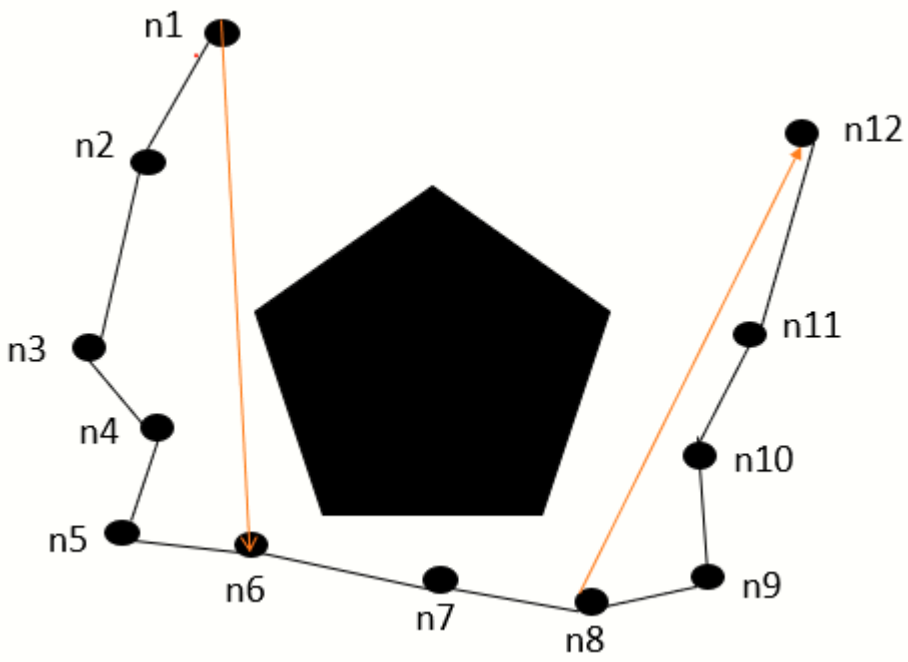


Figure 3. 5: Pruning Principle

Algorithm 3: RE-RRT* algorithm.

1. Input: X_{init} , Y_{new} , n, Imp
2. Output: Tree T = (V, E)
3. V←X_{init} , E←φ
4. $\Delta_x \leftarrow (x_{init} - x_{goal})/n$
5. $\Delta_y = (y_{goal} - y_{init})/n$
6. $plot$ (x_{init} , y_{init}
7. $x_{near} \leftarrow x_{init}$
8. $y_{near} \leftarrow y_{init}$
9. For I = 2 : 10
10. $X_{new} \leftarrow (X_{near} \pm \Delta_X)$;
11. If ˷collision checking (X_{new} , X_{near}, Imp)
    Break else
12. Plot (X_{new}) end
13. % If the obstacle area
14. For i=1…….N do
15. X_{rand} ← random sampling (i)
16. V ← X_{near}
17. $X_{new} \leftarrow steer\ (X_{near}, X_{rand})$
18. If ˷collision checking (X_{new} , X_{near}, Imp)
    % free of obstacle
    V ← V U { X_{new}}
    $X_{near} \leftarrow Near\ (T, X_{new}, r)$;
    chooseParent (X_{near}, X_{new}, E) ;
    Rewiring (X_{near}, X_{new}, E)
    end; End
19. % If cross the obstacle area
20. $X_{new1} \leftarrow X_{new}$;
21. $\Delta_1 = (X_{new1} \pm X_{goal})/n$
22. plot X_{new1}
23. if $X_{new1} \in X_{goal}$ then
24. Success ( );
25. End

Figure 3. 6: RE-RRT* algorithm

Figure 3.6 shows the RE-RRT* algorithm's steps. To speed up the convergence rate of the vehicle, two main improvements which we discussed in section 3.2.1 are clearly mentioned in figure 3.6. This is the complete pseudo-code for robust and efficient Rapidly-Exploring Random Tree Star (RE-RRT*).

Figure 3.7 is a schematic representation of the Robust and Efficient Rapidly Exploring Random Tree Star (RE-RRT*) method flow chart. It entails the following steps: It starts from the initial

points and then checks the direction towards the goal position. Nodes will be added in a straight line. Moves step one by one. When there would be an obstacle on the way, random nodes will be generated around the obstacle until the vehicle reaches the goal position. Besides this, it also checks the neighbour nodes and updates the shortest nodes by deleting the previous ones.

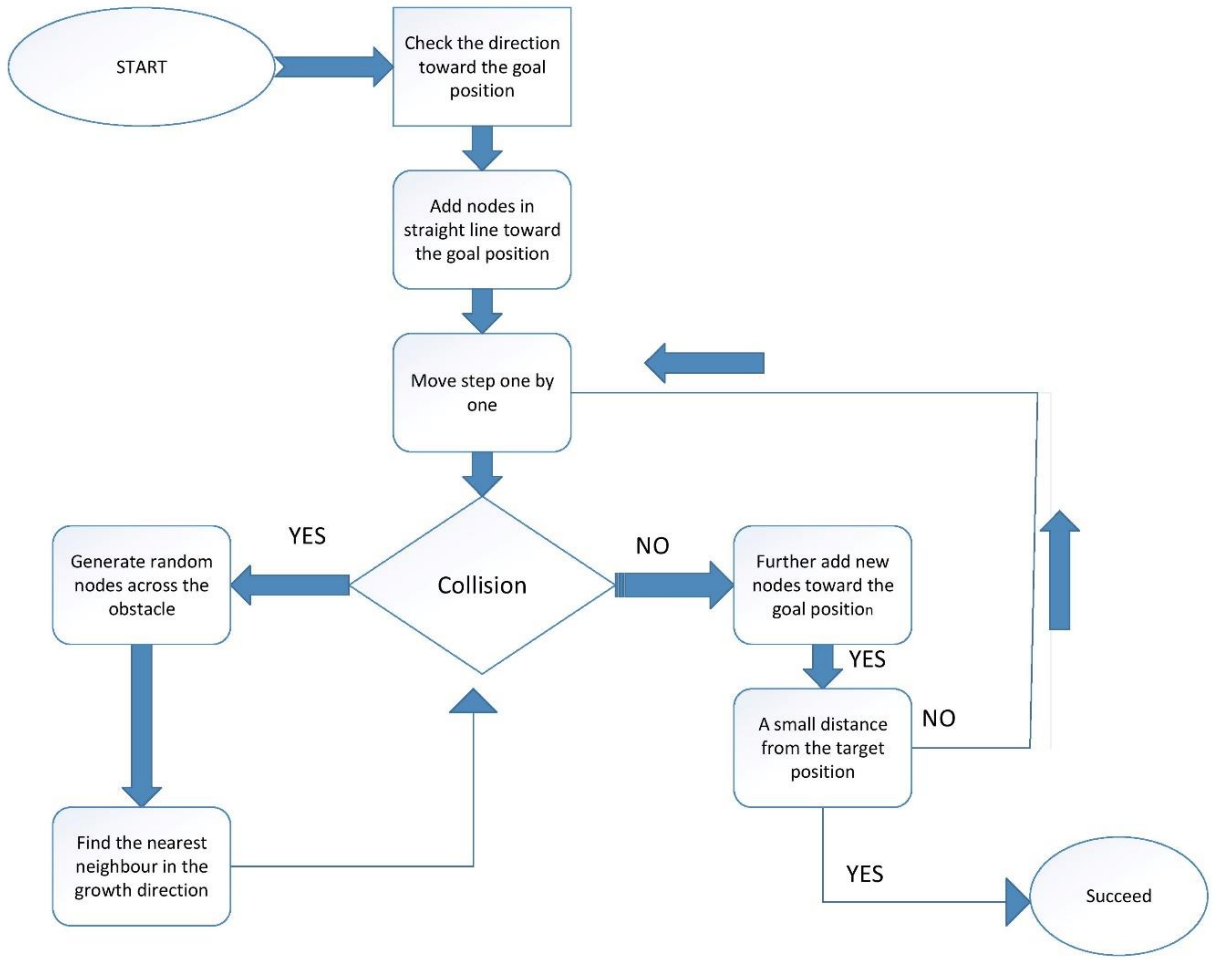


Figure 3. 7**:** Flow chart of RE-RRT* algorithm

Chapter # 4

Implementation

Chapter #4

IMPLEMENTATION

Chapter provides implementation details of our suggested framework. Section 4.1 gives the detail of the tools and languages used. 4.2 gives the overview of implementation of our proposed method.

4.1 TOOLS AND LANGUAGES

Based on the proposed algorithm, a complete framework of Robust and Efficient Rapidly Exploring Random Tree star (RE-RRT*) is implemented on software simulation using MATLAB 2022 language. The input of RE-RRT* is a grid map of 800 x 800 including different shapes of obstacle on the way. The output will be the optimal path tracked by the RE-RRT* based on Euclidean distance and neighbour check process as we discussed in previous chapter.
The process is summarized as follow:

- A grid map of 800 x 800, initial, and final points, delta distance and maximum iteration are given as an input, and we get an optimal path for vehicle by rapidly exploring random tree method.
- First of all, our vehicle is at its initial position x_{init} and will start moving toward the goal position in straight line until an obstacle hit the path.
- On each iteration our vehicle checks either its obstacle area or not.
- If it is obstacle area, then this is the area from where the random tree generates and ends until it crosses the obstacle area.
- At that point, the random sampling and node generation process would stop and from that point, let's say x_{new} to goal position, straight line nodes will generate, and our vehicle follow that path.
- As RE-RRT* processes are entirely automated, no manual involvement is needed.

We depict both a high-level and a low-level view of the suggested technique in section 3. The RE-RRT* is the proposed methodology we present in chapter 3 and its steps are summarized in Diagram 3.6. In the sections that follow, these steps of the procedure are explained in more detail. Figure 4-1 shows the 2-input grid map which we used to implement all the process of RE-RRT*.

Figure4. 1: Grid maps including solid obstacles.

4.2 Line Equation and RRT*

First of all, we imported the map to our MATLAB code and consider one obstacle in its way. We want our vehicle to move straight from initial to final position. Also, we wanted to remove unnecessary sampling from the path. The resultant output is shown in figure 4.2.

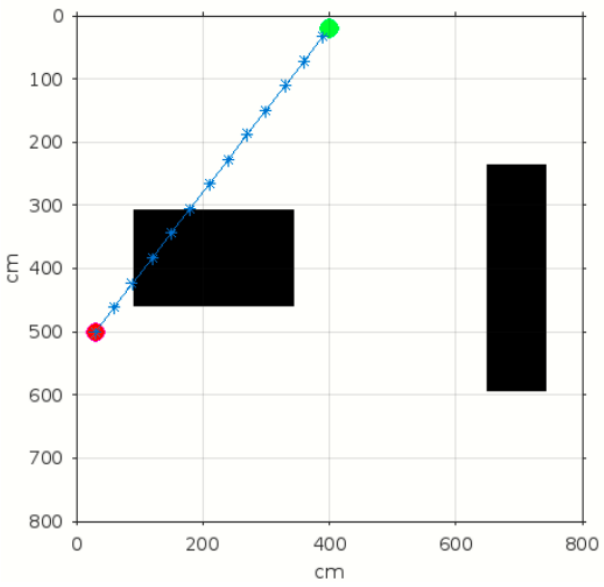


Figure4. 2: Result by line equation

The green dot is our starting point and red dot is our goal position. We used line equation to make our vehicle move in straight line from start to goal position, The blue line indicates the path followed by our vehicle. But the problem with this solution is that this algorithm does not check the obstacle area and pass directly towards the final position. This output is not like what we are

looking for. To cater this problem, we introduce collision detection check at every iteration just like RRT* does.

Moreover, we also introduce RRT sampling search only across the obstacle and our algorithm is working very well when we merge line equation with RRT sampling search across the obstacle. In this way we limit the unnecessary search over the whole area. By introducing collision detection check, we are capable to detect any solid obstacle in the path of our vehicle. And at the end we successfully track a smooth path which has optimal cost. We did a lot of experiments to achieve the desired goal. We will discuss the outputs of different scenarios in the upcoming chapter.

Chapter # 5

Validation

Chapter # 5

Validations

This section uses various experiments to demonstrate the adaptability of the suggested framework. Section **5.1** discusses the hyper parameters used as an input. Section **5.2** discusses the results obtained by the RE-RRT* for different environment. Results of scenario I, II and III are in detail discussed in sections **5.2.1**, **5.2.2** and **5.2.3.**

5.1 Hyper Parameters

In this research, MATLAB R2022a is utilized for the simulation experiment, and the size of the experimental map is 800 x 800. There are various obstacles throughout the map. The vehicle navigates the obstacles as it goes from the beginning position to the destination location. Table I displays the algorithm's parameter settings.

Hyper Parameters	Values
Size of image	800 x 800
Start Point	(50,200)
Goal Point	(600,700)
Segments on the line	10-15
Radius for neighbour node	30.0
Expansion step	30.0
Max Iteration	1000
Update time	50.0
Delay time	0.0

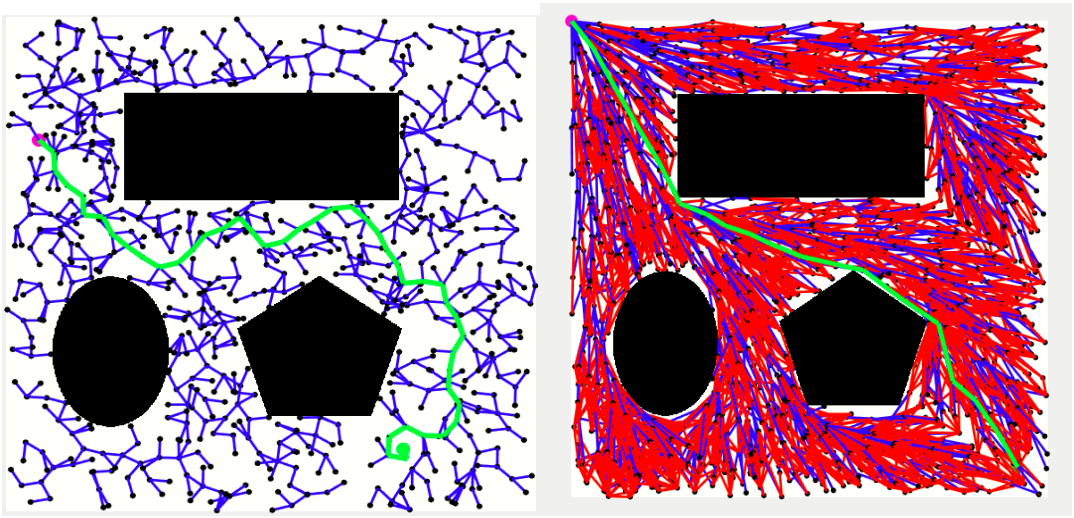
Table5. 1: The hyper parameters setting of RE-RRT*

We include obstacles in the map to test the performance of RF-RRT* in complicated spaces. Several tests are carried out to assess the performance of the suggested method, and we eventually compared the outcomes between RRT, RRT* and RE-RRT* in the next scenario. In the section 5.2, we discussed the results of scenario I, II, and III.

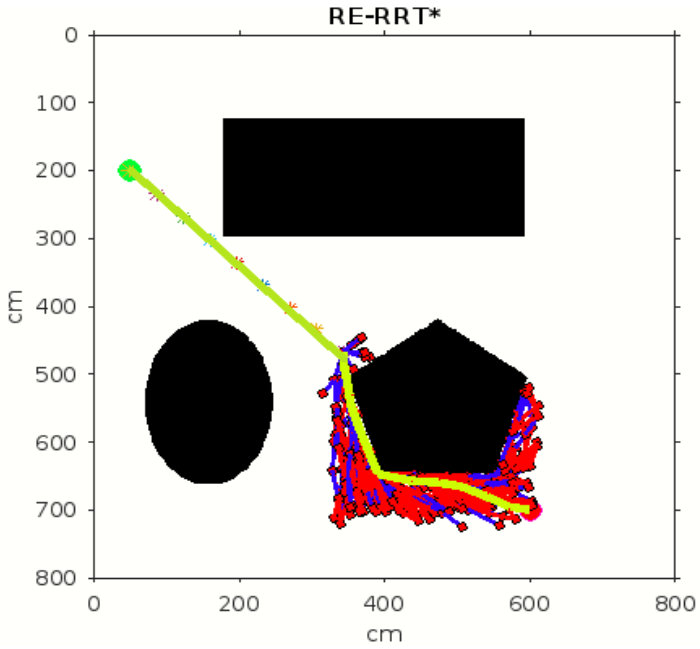
5.2 Results of RE-RRT*

We conduct a lot of experiments to get the desired output. In this section we show the two typical scenarios including different obstacles.

5.2.1 RESULTS OF SCENARIO I



(a) RRT **(b) RRT***



(c) RE-RRT*

Figure5. 1: Output of first simulated environment with several barriers inside.

(a) Basic RRT approach. (b) The RRT* approach. (c) The Suggested RE-RRT* approach

Fig 5.1. shows the schematic representation of simulated environment 1 with obstacles of different shapes. Fig 5.1. (a) shows the result of RRT where the blue lines show the random sampling over the whole map. The one green irregular line shows the path tracked by the RRT approach which is not smooth. Fig 5.1 (b) shows the result of RRT* approach, the improved version of the RRT method. The red lines show the rewired sampling. It generates a smooth path than RRT, but it takes a lot of time to execute the result because the sampling space is over the whole map. Fig 5.1 (c) shows the result of the suggested method RE- RRT* algorithm. This is the improved method of RRT* algorithm. We can see clearly in this picture how this method limits the random sampling and improve the execution time by 80% faster than RRT and RRT* by limiting the sampling space only around the obstacle. As our vehicle started moving straight from the initial towards the final position. When it reached the obstacle area, the RRT* algorithm is used to track the best route to pass the obstacle and after tracking the path the vehicle moves in a straight Line towards the goal position. Thereby this method avoids a lot of unnecessary searches and speeds up the convergence rate.

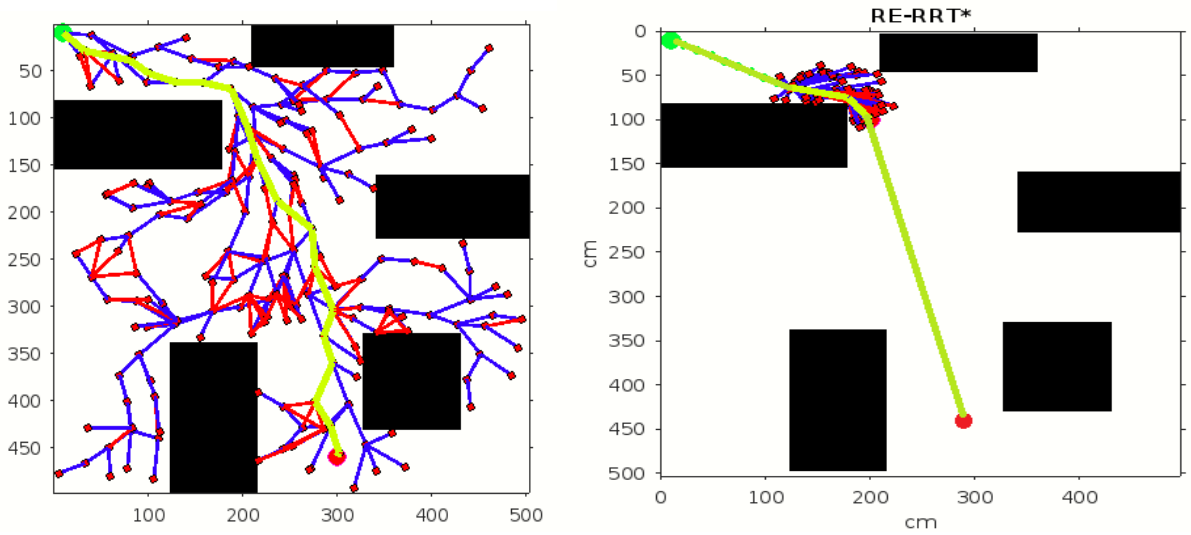
Scenario	**Total Count/No of Sampling**	**Distance (m)**	**Delete Index**	**Angle (φ)**	**Execution time (Sec)**
RRT	752	408	0	0.19	26
RRT*	738	256	270	-0.9	The first result in 22 sec Pruned output takes 3-4min
RE-RRT*	282	241	66	-1.2	11

Table5. 2: The main evaluation indicators of RRT, RRT* and RE-RRT* path planning results of scenario 1

Observing table 5.2, the number of samplings, distance, delete index, angle of curvature and execution time are shown. As I discussed earlier that our focus is on execution time. By comparing the results shown in this table, we can see that how much we minimize the execution time from 26 sec to 11 sec. As we know that the result of RRT* took 3 to 4 min to execute but

our proposed algorithm took only 11 sec to reach to the goal position by limiting the random sampling on the whole area.

5.2.2 RESULTS OF SCENARIO II



(a) (b)

Figure5. 2: Schematic representation of simulated scenario 2 with several obstacles.

(a) The RRT* approach. (b) The Suggested RE-RRT* approach

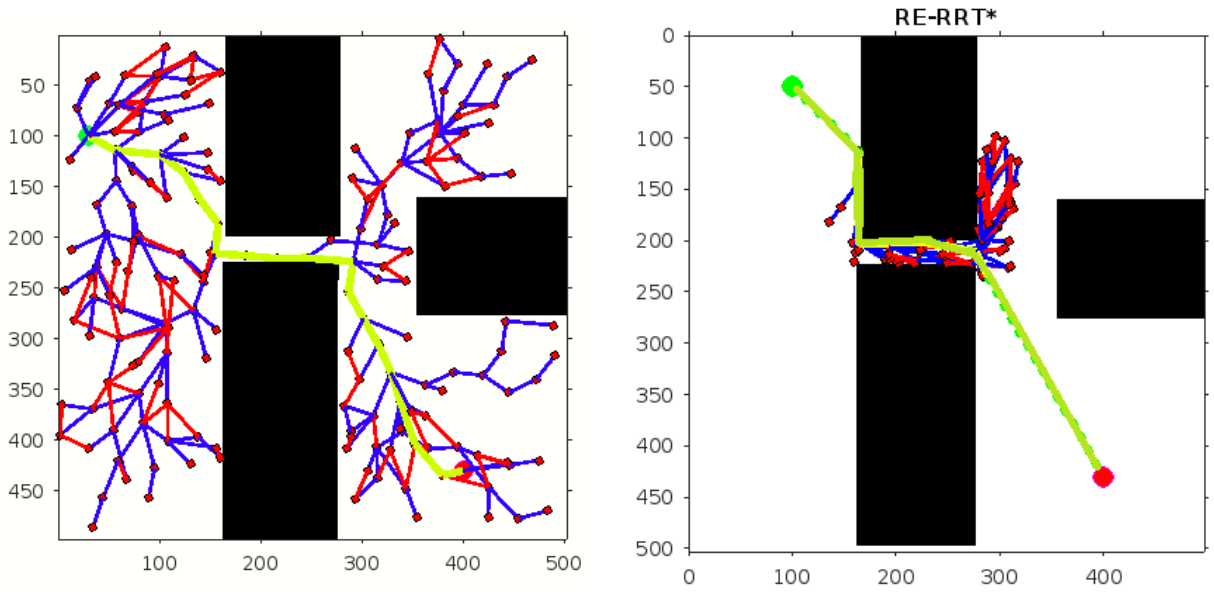
Figure 5.2 shows the schematic representation of simulated scenario 2 with different obstacles. As we can clearly see the difference between the results of RRT* and RE-RRT* environment. The basic RRT takes 26 sec to converge toward the goal position, while our proposed RE-RRT* method takes only 11 sec to converge to the goal position using the same parameters. In RRT*, the nodes are generated randomly over the whole map, and it checks every node whether the node is optimal or not. So, this process makes the system slow and increases the convergence time to meet the desired goal. On the other hand, RE-RRT* minimize the convergence rate even in a complex scenario to meet the desired goal by limiting random sampling only across the obstacles. Also, it reduces the number of turns by the pruning process. Table 5.3 shows the main evaluation indicators of RRT* and RE-RRT* path planning results of scenario 2.

Scenario	Total Count/No of Sampling	Distance (m)	Delete Index	Angle (φ)	Execution time (Sec)
RRT*	193	98	18	1.04	The first result in 22 sec Pruned output takes 3-4 min
RE-RRT*	76	54	3	0.6	11sec

Table5. 3: The main evaluation indicators of RRT* and RE-RRT* path planning results of scenario 2

Table 5-2, and 5-3 show the detailed results of the algorithm. With the help of tables, we can easily interpret that our suggested approach RE-RRT* is way much better than the previous algorithm. The no of sampling in RRT* is 193 because it covers whole space and 76 in the case of RE-RRT* because we restricted the unlimited sampling. Deleted index in the case of RRT* is 18 and for RE-RRT* is 3. The first result takes 22 sec and final result takes 3 to 4 min to execute in the case of RRT*. On the other hand, our proposed algorithm RE-RRT* takes only 11 to 12 sec to converge to the final position.

5.2.3 RESULTS OF SCENARIO III



(a) (b)

Figure 5.3: Schematic representation of simulated scenario 3 with narrow obstacles.

(a) The RRT* approach. (b) The Suggested RE-RRT* approach

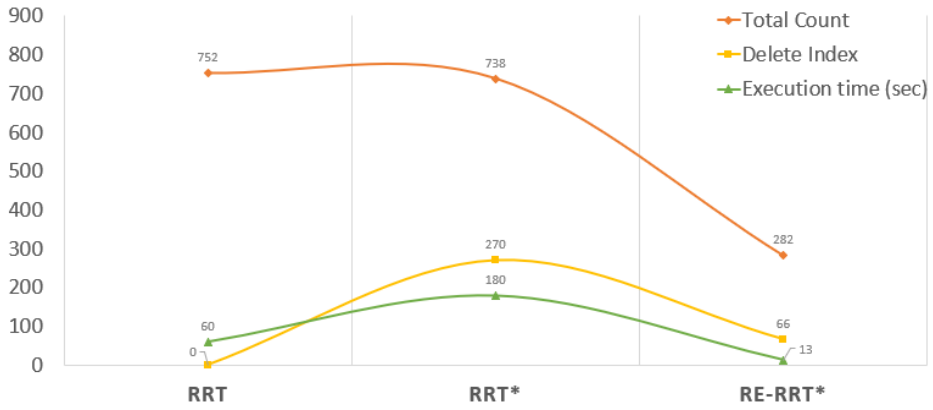
Scenario	Total Count/No of Sampling	Distance (m)	Delete Index	Angle (φ)	Execution time (Sec)
RRT*	193	98	20	1.04	The first result in 22 sec Pruned output takes 1-2 min
RE-RRT*	80	61	6	0.6	11sec

Table5. 4: The main evaluation indicators of RRT* and RE-RRT* path planning results of scenario 3

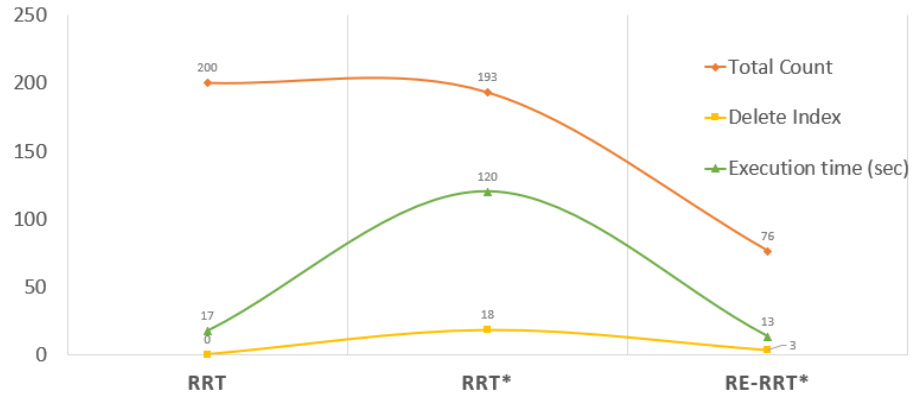
The basic RRT* takes 22 sec to converge toward the goal position, while our proposed RE-RRT* method takes only 11 sec to converge to the goal position using the same parameters

5.3 Graphical Representation of Scenarios I, II and III

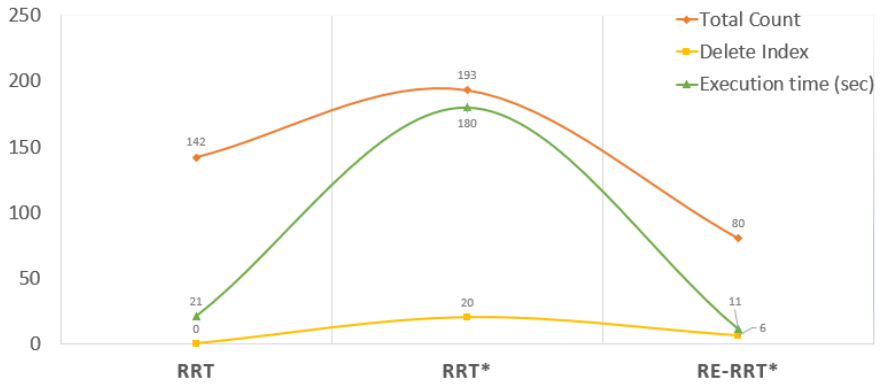
Scenario I



Scenario II



Scenario III


250
200
150
100
50
0
Total Count
Delete Index
Execution time (sec)
142
193
80
21
180
11
0
20
6
RRT
RRT*
RE-RRT*

Chapter # 6

Discussions and Limitations

Chapter # 6
Discussions and Limitations

A thorough explanation of the planned study project is provided in sub-Section 6.1, and the research's limitations are covered in sub-Section 6.2.

6.1 Discussion

This article describes a revolutionary framework for creating an automatic route free of obstacles from a beginning point to a desired position. The feasibility of the proposed framework is evaluated through the different map with unique solid obstacles. Previously, the work has been done on autonomous vehicles using the basic RRT and RRT* method which discover the optimal path and generate an infinite number of nodes. In our approach, we used the advance form of RRT* that is RE-RRT* (Robust and Efficient Rapidly Exploring Random Tree star). RE-RRT* limits the unlimited searches and comes up with the best optimal path in a few seconds. The findings, which are presented in Tables 5-2, 5-3 and 5.4, demonstrate that the suggested framework may produce an optimal path with fewer networks and convergence to the final stage in a short amount of time. We target different maps with solid obstacles for our results. Every scenario has different outputs according to the map and obstacles on the way.

The goal is to demonstrate the efficacy of the proposed framework, which is accomplished fairly through given scenario. In this regard, the RRT algorithm’s explanation is freely available for further evaluation using different scenario. The suggested framework's greatest benefit is its ability to address many types of barriers without any driver usage, it is fully automatic. Once you enter your map with stating and final position, our algorithm will work on finding the optimal path on back end within few seconds. You don’t need to wait a lot just like RRT*. Once it done in finding the optimal path, our vehicle will start moving from starting point to goal position without any interruption of any kind of obstacles. The RE-RRT* has indeed been found to provide more precision in comparison to RRT, RRT*, and improved Bi-RRT*.

The suggested approach is an important step towards automated vehicles in finding an obstacle free optimal path using even complex scenarios. The suggested architecture has many advantages for both commercial heavy vehicles and private use. In particularly, the suggested framework aids in developing the appropriate approach at the appropriate moment at a less

expensive price. It yields a positive result, and validation demonstrates that it will greatly assist drivers and passengers. The main advantage of the suggested structure is that it drastically cuts down on journey time of the path as it automatically generated feasible path in few seconds.
The suggested structure is quite expandable and can accommodate more improvements as needed.

6.2 Limitations

By concentrating on the various types of maps with only static obstacles, we merely offer the skeleton of the suggested framework in this article. In this regard, the proposed framework currently lacks with dynamic obstacles. We believe that such lacking distractors notions can be simply added into the proposed approach by using the methodological approach (Section 3) and practical technique offered (Section 3).

Chapter # 7

Conclusion and Future Work

Chapter # 7
Conclusion and Future Work

To increase the speed and stability of determining the best path, we proposed the RE-RRT* algorithm, a novel RRT-based path planning technique. As a result, enhancements were made to random sampling is limited, which helps to avoid searching over the entire space. To increase the effectiveness of path planning even further, we developed the collision detection algorithm. The suggested RE-RRT* method has significant speed and stability improvements over the RRT and RRT* algorithms. For instance, as compared to the RRT method, the RE-RRT* algorithm dramatically decreased the variation and overall searching duration for discovering a true path. At the same time, RE-RRT* is much faster than RRT* in terms of searching for a path that is close to optimum. As a result, our RE-RRT* method shows excellent promise in real-world motion planning applications. The simulation results in MATLAB 2022a showed that the algorithm has a fast convergence speed. Future efforts will concentrate on accelerating planning and adding more testing scenarios.

Thereby this method avoids a lot of unnecessary searches and speeds up the convergence rate. The basic RRT takes 26 sec to converge toward the goal position, while our proposed RE-RRT* method takes only 11 sec to converge to the goal position using the same parameters.

The findings lead to the following conclusions:

- The high precision RE-RRT* technique produced encouraging outcomes for the real-time environment.
- Proposed approach provides the fast and easiest way to generate optimal path automatically that benefit the driver in real-time environment. Vehicles can easily generate the path by taking only a few seconds using RE-RRT* algorithm.
- Not even industrial vehicles but everyone including paralyzed peoples can also get benefit by using these automatic vehicles to visit different location by their own.

The evaluation's findings demonstrate that the suggested approach is able to generate an optimally priced, obstacle-free path with excellent accuracy. As a result, the suggested system represents an important step towards automating path generation. This leads to several benefits for every citizen.

The suggested structure can be greatly expanded for future improvements. There's still an opportunity to enhance these criteria to provide the ideal path for moving cars, even though the RE-RRT* technique is effective in extracting the low-cost path. Future efforts will concentrate on accelerating planning and adding more testing scenarios.

The proposed framework currently lacks with dynamic obstacles. So, in future the researcher can do the same work of finding the optimal obstacle free path with the help of different scenarios for dynamic environment.

REFERENCE

[1] SAE, “J3016 Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles,” SAE International: Warrendale, Germany, 2018

[2] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, and M. Gittleman,‘‘Autonomous driving in urban environments: Boss and the urban challenge,’’ J. Field Robot., vol. 25, no. 8, pp. 425–466, Aug. 2008.

[3] D. Ferguson, T. M. Howard, and M. Likhachev,‘‘Motion planning in urban environments , ’’ J. Field Robot., vol. 25, nos. 11–12, pp. 939–960, Nov. 2008.

[4] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, ‘‘Path planning for autonomous vehicles in unknown semi-structured environments,’’ Int. J. Robot. Res., vol. 29, no. 5, pp. 485–501, Apr. 2010.

[5] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli,‘‘Autonomous parking using optimization -based collision avoidance,’’ in Proc. IEEE Conf. Decis. Control (CDC), Miami Beach, FL, USA, Dec. 2018, pp. 4327–4332.

[6] S. Karaman and E. Frazzoli, ‘‘Sampling-based algorithms for optimal motion planning,’’ Int. J. Robot. Res., vol. 30, no. 7, pp. 846–894, Jun. 2011.

[7] L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng, ‘‘Efficient sampling-based motion planning for on-road autonomous driving,’’ IEEE Trans. Intell. Transp. Syst., vol. 16, no. 4, pp. 1961–1976, Aug. 2015

[8] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, and K. Lau, ‘‘Stanley: The robot that won the DARPA grand challenge,’’ in DARPA Grand Challenge. Berlin, Germany: Springer-Verlag, 2007, pp. 1–43.

[9] K. Yang and S. Sukkarieh, ‘‘An analytical continuous-curvature path smoothing algorithm,’’ IEEE Trans. Robot., vol. 26, no. 3, pp. 561–568, Jun. 2010.

[10] Kwangjin, D. Jung, and S. Sukkarieh, ‘‘Continuous curvature pathsmoothing algorithm using cubic B zier spiral curves for non-holonomic robots,’’ Adv. Robot., vol. 27, no. 4, pp. 247–258, Mar. 2013.

[11] T. Maekawa, T. Noda, S. Tamura, T. Ozaki, and K.-I. Machida, ‘‘Curvature continuous path generation for autonomous vehicle using B-spline curves,’’ Comput.-Aided Des., vol. 42, no. 4, pp. 350–359, Apr. 2010.

[12] Y. Li, Y. Ming, Z. Zhang, W. Yan and K. Wang, "An Adaptive Ant Colony Algorithm for Autonomous Vehicles Global Path Planning," 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2021, pp. 1117-1122, doi: 10.1109/CSCWD49262.2021.9437682.

[13] Advantages of Autonomous Vehicles https://www.itsdigest.com/10-advantages-autonomous-vehicles

[14] H. Gao, B. Cheng, J. Wang et al., "Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment," IEEE Transactions on Industrial In- formatics, vol. 14, no. 9, pp. 4224–4231, 2018.

[15] C. Urmson, J. Anhalt, D. Bagnell et al., "Autonomous driving in urban environments: boss and the urban challenge," Journal of Field Robotics, vol. 25, no. 8, pp. 425–466, 2008.

[16] J. Leonard, J. How, S. Teller et al., "A perception-driven autonomous urban vehicle," Springer Tracts in Advanced Robotics, vol. 65, pp. 163–230, 2009.

[17] G. V. Raffo, G. K. Gomes, J. E. Normey-Rico, C. R. Kelber, and L. B. Becker, "A predictive controller for autonomous vehicle path tracking," IEEE Transactions on Intelligent Trans- portation Systems, vol. 10, no. 1, pp. 92–102, 2009.

[18] M. Kulich, V. Kozák, and L. Přeučil, "Comparison of local planning algorithms for mobile robots," Modelling and Simulation for Autonomous Systems, Springer International Publishing, New York, NY, USA, 2015.

[19] N. Ganganath and C. T. Cheng, "A 2-dimensional ACO-based path planner for off-line robot path planning," in Proceedings of the International Conference on Cyber-Enabled Distributed Computing & Knowledge Discovery, IEEE, Beijing, China, October 2013.

[20] R. Kala and K. Warwick, "Planning of multiple autonomous vehicles using RRT," in Proceedings of the IEEE International Conference on Cybernetic Intelligent Systems, IEEE, Las Vegas, NV, USA, August 2012.

[21] N.Ganganath, C.T. Cheng, and K. Chi Tse, "Rapidly replanning A*," in Proceedings of the International Conference on Cyber-Enabled Distributed Computing & Knowledge Dis- covery, IEEE Computer Society, Chengdu, China, October 2016.

[22] Harabor, D.; Grastien, A. Online graph pruning for pathfinding on grid maps. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA,

[23] 7–11 August 2011; AAAI Press: Palo Alto, CA, USA, 2011; Volume 25, pp. 1114–1119

[24] Liu, S.; Watterson, M.; Mohta, K.; Sun, K.; Bhattacharya, S.; Taylor, C.J.; Kumar, V. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. IEEE Robot. Autom. Lett. **2017**, 2, 1688–1695.

[25] Dolgov, D.; Thrun, S.; Montemerlo, M.; Diebel, J. Practical search techniques in path planning for autonomous driving. Ann Arbor **2008**, 1001, 18–80.

[26] A. Stentz, "Optimal and efficient path planning for partiallyknown environments," in Proceedings of the IEEE International Conference on Robotics & Automation, IEEE, Washington, DC, USA, May 2002

[27] Y. Kuwata, G. A. Fiore, J. Teo et al., "Motion planning for urban driving using RRT," in Proceedings of the IEEE/RSJ International Conference on Intelligent Robots & Systems, IEEE, Nice, France, September 2008

[28] D. Dolgov, S. +run, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," /e International Journal of Robotics Research, vol. 29, no. 5, pp. 485–501, 2010

[29] S. Suresh, J. Poornaselvan, and C. Divyapreya, "Optimal path planning approach to grid environment," Pollack Periodica, vol. 6, no. 1, pp. 131–140, 2011.

[30] K. Chu, M. Lee, and M. Sunwoo, "Local path planning for offroad autonomous driving with avoidance of static obstacles,"IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 4, pp. 1599–1616, 2012.

[31] L. Makarem and D. Gillet, "Decentralized coordination of autonomous vehicles at intersections," IFAC Proceedings Volumes, vol. 44, no. 1, pp. 13046–13051, 2011.

[32] A. Chebly, G. Tagne, R. Talj et al., "Local trajectory planning and tracking for autonomous vehicle navigation using clothoid tentacles method," in Proceedings of the IEEE International Symposium on Intelligent Vehicles (IV), IEEE, Seoul, South Korea, June 2015

[33] J. Moreau, P. Melchior, S. Victor et al., "Reactive path planning in intersection for autonomous vehicle," IFACPapersOnLine, vol. 52, no. 5, pp. 109–114, 2019

[34] M. L. Tazir, O. Azouaoui, M. Hazerchi et al., "Mobile robot path planning for complex dynamic environments," in Proceedings of the International Conference on Advanced Robotics, IEEE, Istanbul, Turkey, July 2015.

[35] X. Li, Z. Sun, D. Cao, D. Liu, and H. He, “Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles,” Mechanical Systems and Signal Processing, vol. 87, pp. 118–137, 2017

[36] Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Autom. **1996**, 12, 566–580

[37] Lavalle, S.M. Rapidly-Exploring Random Trees: A New Tool for Path Planning; Technical Report; Computer Science Department, Iowa State University: Ames, IA, USA, 1998.

[38] Lee, J.; Kwon, O.S.; Zhang, L.; Yoon, S.E. SR-RRT: Selective retraction-based RRT planner. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 2543–2550.

[39] Shi, Y.; Li, Q.; Bu, S.; Yang, J.; Zhu, L. Research on Intelligent Vehicle Path Planning Based on Rapidly-Exploring Random Tree. Math. Probl. Eng. **2020**, 2020,1-14

[40] Karaman, S.; Frazzoli, E. Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **2011**, 30, 846–894.

[41] Nasir, J.; Islam, F.; Malik, U.; Ayaz, Y.; Hasan, O.; Khan, M.; Muhammad, M.S. RRT*-SMART: A rapid convergence implementation of RRT. Int. J. Adv. Robot. Syst. **2013**, 10, 299

[42] Wang, J.; Chi, W.; Li, C.; Wang, C.; Meng, M.Q.H. Neural RRT*: Learning-based optimal path planning. IEEE Trans. Autom. Sci. Eng. **2020**, 17, 1748–1758

[43] Kuffner, J.J.; LaValle, S.M. RRT-connect: An efficient approach to single-query path planning. In Proceedings of the 2000 ICRA. Millennium Conference, IEEE International Conference on Robotics and Automation, Symposia Proceedings (Cat. No. 00CH37065), San Francisco, CA, USA, 24–28 April 2000; IEEE: Piscataway, NJ, USA, 2000; Volume 2, pp. 995–1001.

[44] Wang, J.; Chi, W.; Shao, M.; Meng, M.Q.H. Finding a high-quality initial solution for the RRTs algorithms in 2D environments. Robotica **2019**, 37, 1677–1694.

[45] Gammell, J.D.; Srinivasa, S.S.; Barfoot, T.D. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In Proceedings of the 2015 IEEE international conference on robotics and

automation (ICRA), IEEE 2015, Seattle, WA, USA, 26–30 May 2015; pp. 3067–3074.

[46] Qureshi, A.H.; Ayaz, Y. Potential functions based sampling heuristic for optimal path planning. Auton. Robots **2016**, 40, 1079–1093.

[47] (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 11, 2016.

[48] Wang, Kun, Chen, Jiajia, Zhang, Rui, Han, Wei, Jiang, Wuhua, Hu, Jinfang, Lu, Xiaoshan, Liu, Xingtao, Zhao, Pan, Path Planning for Autonomous Vehicle Based on a Two-Layered Planning Model in Complex Environment. Hindawi. Journal of Advanced Transportation 2020.

CERTIFICATE OF COMPLETENESS

It is hereby certified that the dissertation submitted by NS Kaynat Gul, Regn No. 00000330764 Titled: **Trajectory Optimization and Obstacle Avoidance of Autonomous Vehicles Using Robust and Efficient Rapidly Exploring Random Tree (RE-RRT*)**, has been checked/reviewed and its contents are complete in all respects.

Supervisor's Name: Dr. Naeem ul Islam

Signature: _____

Date: _____